Dynamic Power Management for Many-Core Platforms in the Dark Silicon Era: A Multi-Objective Control Approach

Amir-Mohammad Rahmani[1,2], Mohammad-Hashem Haghbayan[1], Anil Kanduri[1], Awet Yemane Weldezion[2], Pasi Liljeberg[1], Juha Plosila[1], Axel Jantsch[3], and Hannu Tenhunen[1,2]
[1]Department of Information Technology, University of Turku, Finland
[2]Department of Industrial and Medical Electronics, KTH Royal Institute of Technology, Sweden
[3]Institute of Computer Technology, Vienna University of Technology, Austria
E-mail: {amirah, mohhag, spakan, pakrli, juplos}@utu.fi, {aywe, hannu}@kth.se, axel.jantsch@tuwien.ac.at

Abstract— Power management of NoC-based many-core systems with runtime application mapping becomes more challenging in the dark silicon era. It necessitates a multi-objective control approach to consider an upper limit on total power consumption, dynamic behaviour of workloads, processing elements utilization, per-core power consumption, and load on network-on-chip. In this paper, we propose a multi-objective dynamic power management method that simultaneously considers all of these parameters. Fine-grained voltage and frequency scaling, including near-threshold operation, and per-core power gating are utilized to optimize the performance. In addition, a disturbance rejecter is designed that proactively scales down activity in running applications when a new application commences execution, to prevent sharp power budget violations. Simulations of dynamic workloads and mixed time-critical application profiles show that our method is effective in honoring the power budget while considerably boosting the system throughput and reducing power budget violation, compared to the state-of-the-art power management policies.

Keywords— Dark Silicon, Power Management, NoC-based Manycore Systems, Runtime Mapping

I. Introduction

Number of transistors on a chip is still scaling up steadily (about 2.8 times) for every technology node generation. However, power budgets have not increased on par with technology scaling, thus limiting the number of usable transistors on a chip (only 1.4 times) [1]. This leaves a section of chip area inactive, termed as Dark Silicon [2], while the rest can operate at full throttle (voltage and energy).

Recently, efforts have been made to minimize the effect of dark silicon by utilizing near-threshold computing (NTC) (i.e., Dim Silicon [3]). NTC increases the number of simultaneously active cores, at the expense of much lower operating frequency [3]. In order to implement an efficient NTC-based approach, an intelligent and stable power management mechanism using feedback control is required. The previous work on feedback-based dynamic power management for multi-core and many-core systems can be classified into two main categories: i) the techniques which use workload and network characteristics as feedback (e.g., queue utilization and injection rate), and then adjust voltage/frequency of processing elements, routers, or voltage/frequency islands (VFI) accordingly (e.g., the techniques presented in [4] and [5]), and ii) power budgeting (i.e., capping) techniques which utilize chip/per-core power measurement and per-core performance counters (i.e., core utilization) as feedback, and then apply DVFS or per-core power gating (PCPG) techniques to optimize the system performance within a fixed power cap (i.e., TDP). The approaches presented in [6] and [7] are two examples of power capping techniques which fall into the second category. However, both of them are proposed in the context of bus-based multi-core architectures where there is no concern regarding network congestion and saturation.

Even though all the techniques in these categories efficiently save and control the power consumption for their target platforms, they are not comprehensive and multi-objective enough for the dark-silicon era. The reason is that the techniques in the first category do not consider any safe upper bound on the total system power consumption (i.e., TDP) at runtime, and therefore, they do not feed any power metric back to the management unit. The power capping techniques from the second category are also unable to address the power management issues in the dark silicon era where many-core systems are typically NoC-based and multiple applications are running simultaneously. We believe, in this context, *dark silicon awareness necessitates an efficient multi-objective feedback-based control approach which considers workload characteristics, per-core power and performance measurements, network-load, and total chip power consumption all together.*

In this paper, we provide a comprehensive dark silicon aware power management platform for NoC-based many-core systems under limited power budget and running dynamic workloads (i.e., supporting runtime mapping). This platform benefits from a multi-objective feedback controller providing PCPG and per-core DVFS considering workload characteristics, network congestion, and power-performance characteristics of processing elements. It also provides a proactive runtime application mapping (RTM) technique to reject the disturbance which happens when a new application is mapped onto the system in runtime.

The rest of the paper is organized as follows: In Section II, related work is presented. Our proposed power management platform for NoC-based manycore systems is presented in Section III. Experimental results are provided in Section IV. Finally, Section V concludes the paper and discusses potential future work.

II. Related Work

In [6], a hierarchical power management framework for asymmetric multi-core architectures is demonstrated for

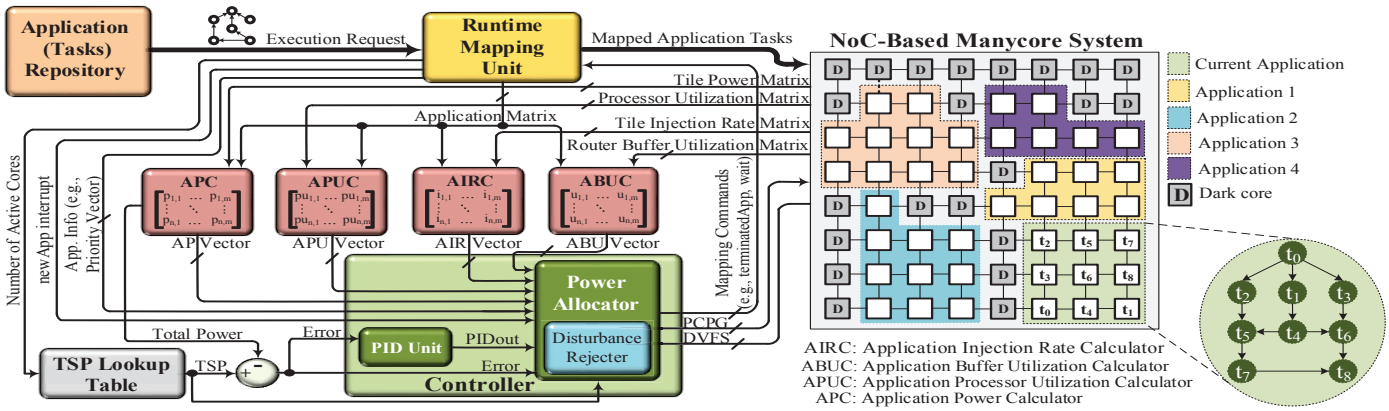

Fig. 1: Overview of our multi-objective dark silicon aware power management system

ARM big.LITTLE [8] mobile platforms. In this architecture, cores have different size and processing power while having the same instruction set architecture (ISA). Ma *et al.* [7] have done a similar attempt to exploit power gating and DVFS for power capping in symmetric multi-core processors. Their technique is demonstrated on the AMD Opteron 6168 processor and is called *PGCapping*. These platforms are energy efficient, yet they suffer from the lack of scalability as both the ARM big.LITTLE and AMD Opteron platforms are bus-based and are limited to a fewer number of cores (i.e., multicore).

In [3], the advantages of near-threshold computing in mitigating the dark silicon is presented. In [4], a control based approach is proposed to minimize dynamic power in MPSoC made of multiple, voltage frequency islands (VFIs). Their goal is to determine optimal operating frequencies for both PEs and routers. This work is not dark silicon aware either, as they do not utilise feedback from power sensors to avoid violating the TSP/TDP.

Haghbayan *et al.* [9] present a power management technique for many-core systems using power feedback from the system to meet the TDP bound. This technique is also categorized into the class of single objective control approaches as it lacks feedbacks from workload characteristics and per-core performance measurements from the system during DVFS process. Lack of information regarding performance and packet injection rate of PEs, can easily lead to inefficient core selection for DVFS purpose, as applying DVFS to an under-utilized PE results in a totally different power-performance behaviour compared when it is applied to a busy PE. In addition, the technique presented in [9], is designed for fixed TDP and does not benefit from a dedicated disturbance rejector to handle sudden overshoots when new applications commence execution.

III. Power Management Platform

Each application in the system is represented by a directed graph with inter-dependent tasks. Mapping of an application onto the system is defined as a one-to-one function from the set of application tasks to the set of tiles. Sequential tasks are assumed as a single task which can be mapped to a same core to reduce inter-core traffic. We also use a simple mathematical model for representing applications running on the system. We denote by Application Matrix the matrix whose entry $(i, j) \in [M] \times [N]$ corresponds to the task's application ID running on the tile located in row i and column j in a mesh-based NoC topology.

A many-core system using our proposed multi-objective power management approach is shown in Figure 1. It is a framework supporting parallel execution of multiple applications dynamically entering and leaving the system at runtime. The Runtime Mapping Unit (RMU) allocates system resources connected through the network to incoming application tasks in an efficient way. It also provides information of the existing application(s) running on the system (RAI) to efficiently manipulate the actuators (e.g., priority vector, application matrix, number of active cores). In order to notify the central power manager (i.e., Controller), RMU asserts an interrupt signal indicating that an application is about to be mapped onto the system. The priority of an application correlates to the level of expected QoS.

Our power management provides DVFS and power gating on a per-core basis. The power manager does not scale the voltage and frequency (VF) of on-chip interconnection network components (e.g., routers, links), to ensure that there is no waiting time and gainless static power consumption of the consumer PEs.

Application Power Calculator: We assume that each tile is equipped with a power sensor to report the current power consumption of the core to the central manager to form the Tile Power Matrix. It should be noted that many of today's platforms are equipped with power meters [6].

We read the rate of packet flow at link level and send its aggregate value to the central control, using a lightweight power meter within the router micro-architecture presented in [10]. In our power management platform, the Application Power Calculator (APC) unit calculates the current power consumption of each application based on the Application Matrix provided by DMU and Tile Power Matrix, measured by the core and router power meters. By masking the Application Matrix on the Tile Power Matrix, the APC block calculates the current power consumption of each application, forms the Application Power Vector (APV), and passes it to the Controller Unit.

Application Processor Utilization Calculator: Each PE is also equipped with a simple performance counter which reports the utilization of the corresponding PE during the previous timing window [7]. Likewise APC, Application Processor Utilization Calculator ($APUC$) unit calculates the aggregate processor utilization for individual applications based on the Processor Utilization Matrix, forms the Application Processor Utilization Vector ($APUV$), and passes it to the Controller Unit.

Application Buffer Utilization Calculator:
In our platform, each router is equipped with a buffer

utilization meter. The buffer utilization meter measures router congestion levels in its recent history. More precisely, it measures the traffic dynamically by calculating the moving average of packet flow in every link of a router. The buffer utilization level of each router (Router Buffer Utilization Matrix in Figure 1) is transferred to the Application Buffer Utilization Calculator ($ABUC$). By masking the Application Matrix (provided by the RMU) on the Router Buffer Utilization Matrix, $ABUC$ calculates the average buffer utilization level for each application and sends it to the controller unit.

Application Injection Rate Calculator: Inspired from [11], we also consider applications' network intensity in order to classify them into intensive and non-intensive categories in the power management process. We use application injection rate as a metric that closely correlates to network intensity. It should be noted that DVFS has a throttling effect on the system as voltage and frequency (VF) upscaling results in increasing packet injection rate, and likewise, VF downscaling leads to decreasing packet injection rate. The injection rate of each task running on a tile is measured at the tile's network interface for its recent history and transferred to the Application Injection Rate Calculator ($AIRC$). By masking the Application Matrix (provided by the RMU) on the Tile Injection Rate Matrix, $AIRC$ calculates the average injection rate for each application and sends it to the controller unit.

TSP Lookup Table: Pagani *et al.* [12] reason that using a single and constant value as a power constraint (i.e., TDP) can result in large performance losses. They present a new power budget concept called Thermal Safe Power (TSP) which is a function of number of active (i.e., non-dark) cores in a system. We use TSP_{worst} which is calculated for the worst-case mapping (i.e., assuming all the active cores are physically packed and influencing the temperature of their adjacent cores) and thus it is safe. In our system, TSP_{worst} values for different number of active cores are pre-calculated and stored in a small lookup table (a one-dimensional array).

A. PID Controller Unit

We employ Proportional Integral Derivative (PID) controller for actuator manipulation. The general formula for the PID controller is as follows:

$$PID_{out}(t) = K_p e(t) + K_i \int e(t)\,\mathrm{dt} + \mathrm{K_d}\frac{\mathrm{de(t)}}{\mathrm{dt}} \qquad (1)$$

Where $PID_{out}(t)$, $e(t)$, K_p, K_i, and K_d are the controller output, error, proportional gain, integral gain, and derivative gain, respectively. The gains of the PID controller are appropriately adjust after several Matlab simulations. Two key factors were considered in our simulations viz., the system stability and the system robustness against power disturbance.

In manycore systems using runtime task mapping, there are often three influential behaviors in the total power trace curve: 1) when an application enters the system, 2) when an application leaves the system and 3) when there is no incoming or outgoing application to/from the system yet power consumption changes due to different changes in intra-application task behaviours such as task dependencies and varying switching activities. We address all the three behaviors. The PID controller can efficiently handle the second and third behaviours. However, when an application enters the system (i.e., the first behaviour), a high overshoot may easily happen especially if the application size is large and demands several dark cores to be activated. This situation is separately handled by the disturbance rejector unit in a proactive way which is discussed in the later sections.

B. Power Allocator

Our power allocator attempts to scale the voltage and frequency of processing elements (V_{PEs}, $Freq_{PEs}$) dynamically by monitoring different feedbacks from the system, see Algorithm 1. As the first step, the TSP is divided by the number of active cores in the system ($\#activeCores$), extracted from RAI, and the result is assigned to the per-core power limit variable ($PCPowerLimit$). If TDP is chosen over TSP, this line can be ignored as TDP calculation is conservative enough to consider worst-case per-core power limit as well. Next, the *IRClassifier* function classifies all the applications into intensive (I_{set}) and non-intensive (NI_{set}) sets in terms of network intensity, based on Application Injection Rate Vector ($AIRV$). The detailed classification algorithm can be found in [11]. Similarly, *BUClassifier* function classifies all the applications into congested (C_{set}) and non-congested (NC_{set}) sets. An application is tagged as congested if the average buffer utilization of its associated routers is larger than a predefined threshold (e.g., 75%). In this way, every application is tagged at runtime with a 2-bit label which can get one of these values: *NI_NC* (non-intensive, non-congested), *NI_C* (non-intensive, congested), *I_NC* (intensive, non-congested), and *I_C* (intensive, congested). These tags are variable and updated in every iteration.

Difference between instantaneous power consumption and TSP/TDP is $Error$ value, transferred to the PID controller. After classifications, voltage and frequency downscaled or upscaled is performed, based on magnitude of PID_{out} such that system's power consumption is closer to TSP/TDP. $VF_{downscaler}$ and $VF_{upscaler}$ functions are used to scale the voltage and frequency of target applications. When a new application to be mapped arrives, Power Allocator receives a *newApp interrupt* from the mapping unit. This interrupt is serviced by the *proactiveDistRej* function implemented in the Disturbance Rejecter module which proactively scales currently running applications.

Downscaler: VF downscaling process of PEs is explained in Algorithm 2. We consider the entire application space ($C_{set} \cup NC_{set}$) to choose the target application set to be downscaled. When there is an overshoot, applications with the lowest priority are chosen by the LP_{apps} function, letting the high priority applications run at a higher QoS level. Among them, applications that are tagged as congested (C_{set}) are chosen to minimize congestion and improve network throughput. PEs residing in a congested area can dissipate unnecessary power (particulary static) due to low network throughput. As VF downscaling has also the effect on throttling of packet injection, alleviating the network congestion for such applications and save power. In case of unavailability, congested set is replaced by non-congested set (NC_{set}). These are further narrowed down to the application with the lowest performance loss to power reduction ratio by the $lowD_{prf-pwr}$ function which is presented in detail in the following. Finalized target ap-

Algorithm 1 Power Allocation Algorithm

Inputs: PID_{out}, RAI, $ABUV$, $AIRV$, APV, $APUV$, TSP, $newApp\_interrupt$, $Error$
Output: V_{PEs}, $Freq_{PEs}$, $terminatedApp$
Global Variables: $DVFSList$, I_{set}, NI_{set}, C_{set}, NC_{set}, $PCPowerLimit$
Constant values: $bufferUtilizationLimit$
Body:
1: $PCPowerLimit \leftarrow \frac{TSP}{\#activeCores}$; // calculating per-core power limit
2: $(I_{set}, NI_{set}) \leftarrow IRClassifier\ (AIRV, RAI)$; // classify I and NI
3: $(C_{set}, NC_{set}) \leftarrow BUCassifier\ (ABUV, RAI)$; // classify C and NC
4: **if** $newApp\_interrupt$ **then** {// interrupt - new application to be mapped}
5: $(V_{PEs}, Freq_{PEs}, terminatedApp) \leftarrow proactiveDistRej\ (Error, RAI, ABUV, APV, APUV)$;
6: **else**
7: **if** $PID_{out} < 0$ **then**
8: $(V_{PEs}, Freq_{PEs}, terminatedApp) \leftarrow VF_{downscaler}\ (RAI, ABUV, APV, APUV, PID_{out}, PCPowerLimit)$;
9: **else**
10: $(V_{PEs}, Freq_{PEs}, terminatedApp) \leftarrow VF_{upscaler}\ (RAI, ABUV, APV, APUV, PID_{out}, PCPowerLimit)$;
11: **end if**
12: **end if**

Algorithm 2 Voltage and Frequency Downscaling Function

Inputs: RAI, $ABUV$, APV, $APUV$, PID_{out}, $PCPowerLimit$
Outputs: V_{PEs}, $Freq_{PEs}$, $terminatedApp$
Variables: $availableApps$, $targetApp$, $failedDVFS$, $appSet$
Body:
1: $availableApps \leftarrow C_{set} \cup NC_{set}$; // application space
2: **while** true **do**
3: $targetApp \leftarrow \emptyset$; // the application targeted for DVFS
4: $appSet \leftarrow LP_{apps}\ (availableApps, RAI)$; // low priority apps
5: $appSet \leftarrow appSet \cap C_{set}$;
6: **if** $appSet = \emptyset$ **then** {// consider congested apps}
7: $appSet \leftarrow appSet \cap NC_{set}$; // consider non-congested apps
8: **end if**
9: $targetApp \leftarrow lowD_{prf-pwr}\ (appSet, APV, APUV, PID_{out})$;
10: $(V_{PEs}, Freq_{PEs}, failedDVFS) \leftarrow DVFS(targetApp, PID_{out}, PCPowerLimit)$;
11: **if** $failedDVFS$ **then**
12: remove $targetApp$ from $availableApps$; continue;
13: **if** $availableApps$ is empty **then**
14: $terminatedApp \leftarrow targetApp$; break;
15: **end if**
16: **else**
17: $DVFSList \leftarrow targetApp$; break;
18: **end if**
19: **end while**

plication ($targetApp$) is then downscaled by the $DVFS$ function as per PID_{out} and $PCPowerLimit$.

Upscaler: VF Upscaling of processing elements is presented in Algorithm 3. When there is an undershoot, first, the set of applications that are already downscaled and applications that are non-intensive and non-congested ($availableApps \cap NI_{set} \cap NC_{set}$) are chosen. The ground for this selection is that upscaling voltage and frequency of a PE residing in a congested area and having a high injection rate may result in zero performance gain if on-chip communication network is the bottleneck. That is the reason why in VF upscaling process, in contrast with downscaling, a higher priority is given to congestion than application priority in the algorithm. If there is no *NI_NC* application in the system, *I_NC* applications will be the next candidates set ($DVFSList \cap I_{set} \cap NC_{set}$). Among these, applications with the highest priority are picked by the HP_{apps} function to meet system's QoS demands. These are further narrowed down to the application with the highest performance gain to power increase ratio ($HighD_{prf-pwr}$). The chosen target application is then upscaled by the $DVFS$ function as per PID_{out} and $PCPowerLimit$.

***highD*$_{prf-pwr}$ and *lowD*$_{prf-pwr}$ functions:** These functions search for an application with the highest or lowest performance-power ratio (i.e., $D_{prf-pwr}$) in a given set to be the target of VF upscaling or downscaling. In [7], product of core utilization ($Util$) and aggregated frequency ($Freq$) is used as a high-level computational capacity metric. In this metric, the frequency is weighted to deduct the idling cycles. We extend this metric by aggregating core utilization in an application ($appUtil$), provided by $APUC$, to calculate the performance of an application. After calculating $D_{prf-pwr}$ for all the applications in $appSet$, $lowD_{prf-pwr}$ and $highD_{prf-pwr}$ functions use a simple *quicksearch* algorithm to find the application with the lowest and highest $D_{prf-pwr}$ value as the target application for DVFS, respectively.

Algorithm 3 Voltage and Frequency Upscaling Function

Inputs: RAI, $ABUV$, APV, $APUV$, PID_{out}, $PCPowerLimit$
Outputs: V_{PEs}, $Freq_{PEs}$, $terminatedApp$
Variables: $availableApps$, $targetApp$, $failedDVFS$, $appSet$
Body:
1: $targetApp \leftarrow \emptyset$;
2: $availableApps \leftarrow DVFSList$;
3: **while** $targetApp = \emptyset$ **do**
4: $appSet \leftarrow availableApps \cap NC_{set} \cap NI_{set}$; // non-congested/non-intensive apps
5: **if** $appSet = \emptyset$ **then**
6: $appSet \leftarrow availableApps \cap NC_{set} \cap I_{set}$; // non-congested/intensive apps
7: **if** $appSet = \emptyset$ **then**
8: $appSet \leftarrow availableApps$;
9: **end if**
10: **end if**
11: $appSet \leftarrow HP_{apps}(appSet, RAI)$; // high priority apps
12: $targetApp \leftarrow highD_{prf-pwr}\ (appSet, APV, APUV, PID_{out})$;
13: $(V_{PEs}, Freq_{PEs}, failedDVFS) \leftarrow DVFS\ (targetApp, PID_{out}, PCPowerLimit)$;
14: **if** $failedDVFS$ **then**
15: remove $targetApp$ from $availableApps$;
16: $targetApp \leftarrow \emptyset$; continue;
17: **end if**
18: **end while**
19: remove $targetApp$ from $DVFSList$;

Proactive Disturbance Rejection (PDR): Whenever a new application is mapped onto the system, it is likely to cause a sudden change in overall power consumption that shoots above the TSP/TDP. Such sporadic rises in power consumption can be minimized by proactively scaling down applications that are currently running on the system. Algorithm 4 details the PDR function. If the $Error$ is positive, indicating that new application can be accommodated, the predicted power consumption ($appPredictedPower$) is calculated based on number of tasks (N extracted from RAI) of the new application and average power consumed by actively running cores (P_{avg}). The difference between $Error$ and $appPredictedPower$ is the $proactiveError$, which is fed back to a proportional controller with gain K'_p. Here, the integral and derivative terms are removed because when such sporadic rises occur, history-based (i.e., integral term) or prediction-based (i.e., derivative term) decision making will most likely affect the controller's response. Output of the controller (P_{out}) determines the extent by which currently running applications are to be scaled so that the new application can be mapped without violating TSP/TDP. If the ($Error > 0$) and ($proactiveError > 0$), indicating availability of power budget for the new application, it is mapped as is without any further scaling. If ($Error > 0$) and ($proactiveError < 0$), indicating that power allocation to new application would violate TSP/TDP, currently running applications are downscaled by $VF_{downscaler}$ based on P_{out}.

Algorithm 4 Proactive Disturbance Rejection (*proactiveDistRej()*).

Inputs: $Error$, RAI, $ABUV$, APV, $APUV$, $PCPowerLimit$
Outputs: V_{PEs}, $Freq_{PEs}$, $terminatedApp$
Variables: $failedDVFS$, $appPredictedPower$, $proactiveError$, P_{out}, P_{avg}
Constant values: K'_p
Body:

1: $appPredictedPower \leftarrow N \times P_{avg}$;
2: $proactiveError \leftarrow Error$ - $appPredictedPower$;
3: **if** $proactiveError < 0$ **then**
4: $P_{out} \leftarrow K'_p \times proactiveError$;
5: $(V(PEs), Freq(PEs), terminatedApp) \leftarrow VF_{downscaler}$ $(RAI, ABUV, APV, APUV, P_{out}, PCPowerLimit)$;
6: **end if**

IV. Experimental Evaluation

We perform the experiments on our in-house cycle-accurate many-core platform implemented in SystemC using Noxim [13] as communication architecture. As PE baseline design, we use Niagara2-like in-order core specifications obtained from McPAT [14]. Physical scaling parameters were extracted from the Lumos framework (by Wang and Skadron) [15]. Lumos is a framework to analytically quantify the power-performance characteristics of many-core systems especially in near-threshold operation. Lumos is open source and publicly available [16]. The physical scaling parameters have been calibrated by circuit simulations with a modified Predictive Technology Model [17]. Moreover, we have imported other models and specifications such as power modeling, voltage-frequency scaling, thermal design power (TDP) calculation, and near threshold computing parameters from the Lumos framework. Our manycore platform was reinforced to support runtime application mapping by implementing a central manager (CM) residing in the node $n_{0,0}$. The network size is 12×12 and the the chip area is $138mm^2$.

We model two application categories – non-realtime (lowest priority) and soft realtime (highest priority). Several sets of non-realtime applications with 4 to 35 tasks are generated using TGG [18] where the communication and computation volumes are randomly distributed. We model MPEG4 and VOPD multimedia applications as soft realtime applications.

In our multi-application manycore system, a random sequence of applications enter the scheduler FIFO. This sequence is kept fixed in all experiments for the sake of fair comparison. The probabilities of selecting soft realtime and non-realtime applications from the application repository are 30% and 70%, respectively. CM selects the *first node* using SHiC [19] method, and maps the application based on its real-time attributes. The soft realtime applications are mapped contiguously. In addition to the runtime mapping unit, our multi-objective power management platform (including the controller, AIRC, ABUC, etc.) is also implemented in software (i.e., soft coded) as a part of the CM. As the control interval can be long (i.e., millisecond scale) compared to the system clock period (i.e., nanosecond scale), the control traffic overhead is negligible. For example, a NoC system running at 750MHz can be as large as 75000 cores, while the time for control packet collection is <1% of sampling interval of 10ms.

For the DVFS purpose, we use 15 VF levels (similar to Intel SCC) including near-threshold operation extracted from the Lumos framework. The frequency of the on-chip communication network (e.g., routers) is set to the maximum level to demonstrate that even at the maximum NoC speed, the network can get congested and should be taken into account in power management along with the other parameters. For the TSP calculation, we follow the same floorplan style, chip thickness, silicon thermal conductivity, and heat sink model as [12]. We set ambient temperature to 45°C, a threshold temperature that triggers thermal management to 80°C, maximum chip power consumption from the power supply to 300W, and the power consumption of an inactive core to 0.3W.

We compare different characteristics of the manycore system under four different management scenarios: 1) our multi-objective controller (MOC) with proactive disturbance rejection (PDR), 2) PGCapping [7] where only core's power-performance ratio is considered as feedback for the PCPG and per-core DVFS actuation, 3) DSAPM [9] where no information regarding performance and packet injection rate of PEs is used as feedback, and 4) without TSP/TDP constraint. Without TSP/TDP constraint is the scenario where the system is not limited in terms of maximum power consumption. This is the situation when, in reality, the chip is damaged due to overheating. To perform a fair comparison, we enable PGCapping and DSAPM techniques to use the same 15 VF levels for per-core DVFS actuation.

Power consumption of the system under the aforementioned power management scenarios to honor constant

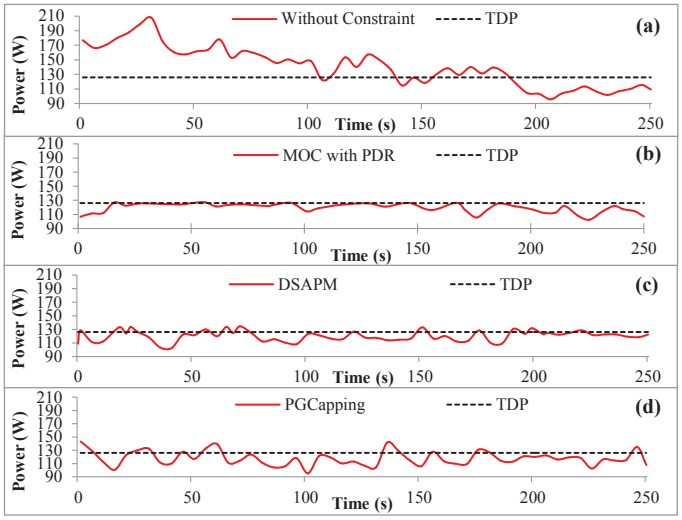

Fig. 2: The power consumption of the system using (a) without TSP/TDP constraint, (b) MOC with PDR, (c) DSAPM, and (d) PGCapping power management policies to honor TDP

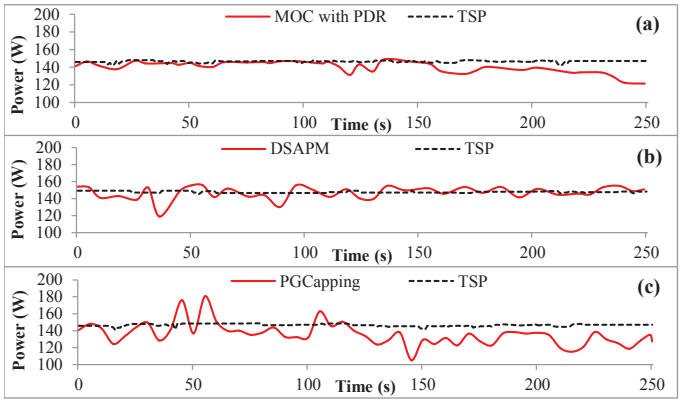

Fig. 3: The power consumption of the system using (a) MOC with PDR, (b) DSAPM, and (c) PGCapping power management policies to honor TSP

TDP is presented in Figure 2. The dashed black curve represents maximum power budget for the system (i.e., TDP). The TDP value is set to 126W which is calculated based on the chip power density. Deviation of power consumption from the baseline reflects either violation or under-utilization of power budget. Power consumption in case of the PGCapping, DSAPM and without-constraint power managements mostly tend to overshoot or undershoot from TDP. The without-constraint power management does not consider any upper bound on power consumption, subsequently it violates the TDP constraint right through.

PGCapping benefits from the cores' power-performance values, fed back by the controller and thus increases the system throughput to some extent. However, it suffers from the under-utilization issue as it does not consider the network congestion and applications injection rates. DSAPM considers network congestion, however it also suffers from the under-utilization issue as it is agnostic of cores' performance value and applications' injection rates. Moreover, both PGCapping and DSAPM techniques refuse to properly handle occasional overshoots due to new application arrivals. Evidently, MOC with PDR stays in close proximity with TDP and hence has the best power management mechanism in comparison with the others. In cases where power consumption exceeds TDP, the MOC controller rapidly reduces the power consumption by a proper voltage and frequency scaling. The control system is stable even for large fluctuations in power consumption that occur with arrival of intense applications. Figure 3 demonstrates the aforementioned power management scenarios to honor dynamic TSP values. As can be observed from the figure, the conclusions we made for Figure 2 are also valid for dynamic TSP, the MOC-based system is stable even when budget is changed at runtime. In the figure, TSP does not radically change (often between 141W and 149W) as the system is mostly busy and the majority of cores are active.

To assess the efficiency of our platform, we compare the normalized throughput for the set of applications under MOC (with PDR), PGCapping, and DSAPM policies, as shown in Figure 4(a). The results reveal that our proposed method can significantly improve the overall system throughput for different power budget types (up to 29% compared with PGCapping and up to 15% compared with DSAPM). The results reveal the advantage of our proposed multi-objective controller which considers both the computation and communication aspects in power management. Figure 4(b) shows TDP/TSP violation for different power management policies over time. We measure violation as the ratio of time for which power consumption exceeded TDP/TSP (resulting in a violation) to the entire simulation time. It can be observed that the proposed disturbance rejection technique honors the TDP/TSP constraints for more than 99% of the simulation time.

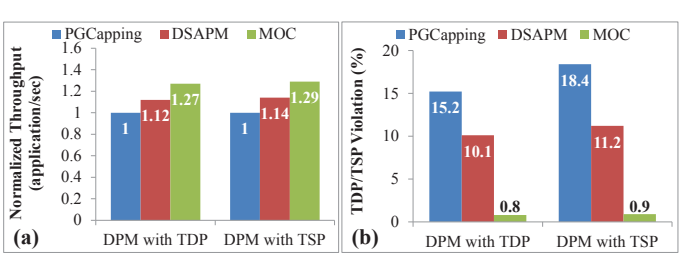

Fig. 4: MOC vs. PGCapping vs. DSAPM, (a) normalized throughput, (b) TDP/TSP violation for different dynamic power managements

V. Conclusions

In this paper, a multi-objective feedback controller system was proposed to protect many-core systems against overshooting of power consumption from a certain limit. The target framework is a NoC-based manycore system using runtime application mapping where applications enter and leave the system at runtime. The feedbacks to the controller are the processing elements' power-performance measurements, application workloads, and network congestion. Comparing the total system power with the maximum power budget, the controller efficiently changes voltage and frequency of appropriate processing elements, down to near threshold operation. The results showed improved system throughput and TDP/TSP violation, for the proposed platform when compared to state-of-the-art power management policies.

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" and EU COST Actions IC1103: Manufacturable and Dependable Multicore Architectures at Nanoscale (MEDIAN).

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

[3] Wang L. and K. Skadron. Implications of the Power Wall: Dim Cores and Reconfigurable Logic. *IEEE Micro*, 33(5), 2013.

[4] P. Bogdan et al. Dynamic Power Management for Multidomain System-on-chip Platforms: An Optimal Control Approach. *ACM Trans. Des. Autom. Electron. Syst.*, 18(4), 2013.

[5] R. David et al. Dynamic Power Management of Voltage-Frequency Island Partitioned Networks-on-Chip Using Intel Sing-Chip Cloud Computer. In *NOCS*, 2011.

[6] T.S. Muthukaruppan et al. Hierarchical power management for asymmetric multi-core in dark silicon era. In *DAC*, 2013.

[7] K. Ma and X. Wang. PGCapping: Exploiting Power Gating for Power Capping and Core Lifetime Balancing in CMPs. In *PACT*, 2012.

[8] ARM Ltd., http://www.arm.com/products/tools/development-boards/versatile-express/index.php, 2011.

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

[10] A.Y. Weldezion et al. A scalable multi-dimensional NoC simulation model for diverse spatio-temporal traffic patterns. In *3D-IC*, 2013.

[11] K. Chang et al. HAT: Heterogeneous Adaptive Throttling for On-Chip Networks. In *SBAC-PAD*, 2012.

[12] S. Pagani et al. TSP: Thermal Safe Power: Efficient Power Budgeting for Many-core Systems in Dark Silicon. In *CODES*, 2014.

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

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

[15] 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.

[16] Lumos Framework, http://liangwang.github.io/lumos/. Accessed: 2014-05-20.

[17] B.H. Calhoun et al. Sub-threshold circuit design with shrinking CMOS devices. In *ISCAS*, 2009.

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

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