Chapter 1
Multi-Objective Power Management for CMPs in the Dark Silicon Age

Amir M. Rahmani, Mohammad-Hashem Haghbayan, Pasi Liljeberg, Axel Jantsch, and Hannu Tenhunen

Abstract New power management challenges in networked many-core systems arise when limitations of the dark silicon era come into reality. The main goal in the power management process is to achieve optimal power-performance efficiency considering thermal design power budget. This necessitates i) monitoring several system characteristics including both communication and computation aspects, ii) categorizing, prioritizing, and processing the information in an intelligent way, iii) and controlling a rich set of actuators. More precisely, a comprehensive Observe-Decide-Act (ODA) loop based multi-objective control approach is needed, which has access to a rich set of sensors and actuators. In this chapter, we first identify a necessary set of system parameters for monitoring such as an upper limit on total power consumption, dynamic behaviour of workloads, utilization of processing elements, per-core power consumption, load on network-on-chip, etc. We also discuss essential actuators needed for the power management process together with a multi-objective and dark silicon aware power management policy that is able to simultaneously consider all the mentioned parameters. As actuator, fine-grained voltage and frequency scaling is utilized, including near-threshold operation, per-core power gating, as well as scheduler-level actuation to maximize the system throughput while honoring the power budget.

Amir M. Rahmani
University of Turku, Turku, Finland e-mail: amirah@utu.fi

Mohammad-Hashem Haghbayan
University of Turku, Turku, Finland, e-mail: mohhag@utu.fi

Pasi Lijeberg
University of Turku, Turku, Finland, e-mail: pakrli@utu.fi

Axel Jantsch
TU Wien, Vienna, Austria, e-mail: axel.jantsch@tuwien.ac.at

Hannu Tenhunen
KTH Royal Institute of Technology, Stockholm, Sweden, e-mail: hannu@kth.se

1.1 Introduction

Dim Silicon concept is a promising approach to increase the overall throughput of chip multiprocessors (CMPs), at the expense of much lower operating frequency [1]. It is considered as one of the most effective methods to mitigate the dark silicon phenomenon. However, implementing an efficient Dim Silicon based approach necessities a comprehensive multi-objective power management mechanism having access to a rich set of on-chip sensors and actuators to utilize several Observe-Decide-Act (ODA) loops (i.e., feedback control) for controlling different aspects of the system. Such a multi-objective power management activity becomes even more challenging when considering near future manycore systems accommodating tens to hundreds of cores interconnected via Network-on-Chip (NoC). On top of that, manycore systems often need to handle an extremely dynamic workload with an unpredictable sequence of different applications entering and leaving the system at a runtime. In addition, due to the need to honor an upper limit on power consumption, i.e. fixed thermal design power (TDP) or dynamic thermal safe power (TSP) [2], in the dark silicon era, a power capping mechanism is required to monitor the instantaneous total system power consumption and manage the power-performance requirements of the system.

The related work on closed-loop dynamic power management for chip multiprocessors can be classified into two main categories:

- **NoC-centric techniques** that utilize different communication related information such as queue length and injection rate as feedback to adjust voltage and frequency of processing elements, routers, or voltage-frequency islands (VFI) accordingly (e.g., [3] and [4]).
- **Power capping techniques** proposed for bus-based multiprocessor systems which utilize chip/per-core/per-cluster power measurement and per-core performance as sensory data to optimize system power-performance characteristics within a fixed power cap where there is no concern regarding network congestion and saturation (e.g., [5] and [6]).

Even though all the techniques in these categories efficiently control the power consumption for their target platforms, they are not comprehensive enough to consider several factors affecting the performance in many-core systems. Therefore, we first characterise different key parameters which should be taken into consideration to devise a proper power management approach for the dark silicon era. In the following, we list the parameters and discuss their significance:

- **Power Budget:** Due to thermal issues in the dark silicon era, there exist an upper limit on power consumption which is called thermal design power (TDP) if it is a fixed value or thermal safe power (TSP) [2] if it can change dynamically at a runtime depending on the number of active cores in a system. To guarantee the safety of the chip this limit should be strictly honored by the power manager.
- **Application Performance:** In order to monitor the impact of DVFS on application performance, a virtual or physical sensor to measure processors' utilization

such as performance counters are needed. The main idea is to monitor how much impact voltage-frequency (VF) upscaling has had in the last epoch to increasing performance, and similarly how much VF downscaling has had negative impact on performance in the previous monitoring time-window.

- **Network-on-Chip Congestion:** Congestion in communication medium can easily lead to a poor efficiency of DVFS process. Assume a pair of producer and consumer processing elements (PEs) where there is a congestion in one or multiple routers in their communication path. VF upscaling of such PEs will result in either zero or marginal performance gain, while a considerable amount of energy can be wasted due to a long waiting time of data transactions. Therefore, utilizing congestion meters in NoC routers can provide to the power manager a beneficial source of information.
- **Application's Network-Intensity:** A source-throttling congestion control mechanism will impact in a limited way performance if it is done only based on network-load [7]. Such a mechanism is not application-aware, but rather throttles all applications equally regardless of applications sensitivity to latency. Different applications impose different injection rates to the network and suffer differently from network congestion. As DVFS on PEs has also affect on application throttling, i.e. VF upscaling (downscaling) of a PE may result in increasing (decreasing) packet injection rate by the PE to the network, applications' characteristics in terms of their network-sensitivity should be also monitored and considered in power management.
- **Applications' Priorities:** There are different types of applications, for instance non-realtime, soft realtime, and hard realtime, where they demand different quality of service at a runtime. These requirements including the minimum required power-budget for each application need to be considered in the prioritization phase in the controller.
- **Disturbances Caused by Runtime Mapping:** 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 should be also considered and proactively managed.

Network centric techniques in the first category do not consider TDP or TSP, and therefore, they are not closed loop power budgeting techniques. Power capping techniques from the second category are on the other hand unable to address communication related issues in NoC-based manycore systems. In addition, both categories consider scenarios where the impact of dark silicon phenomena is not yet that significant, e.g. in 45nm CMOS technology. Furthermore, they often perform off-line application analysis and mapping, without support for Runtime Application Mapping (RTM) and disturbance rejection.

We argue, in the power management context, *dark silicon awareness necessitates an efficient multi-objective ODA control approach which considers workload characteristics, per-core power and performance measurements, network-load, disturbances caused by runtime mapping, and total chip power measurement all together*.

In this chapter, we present a comprehensive multi-objective dark silicon aware power management platform for NoC-based manycore systems which is based on our contribution presented in [8] and considers all the discussed parameters.

The rest of the chapter is organized as follows: In Section 1.2, related work is presented. Our proposed multi-objective power management platform is presented in Section 1.4. Experimental results are provided in Section 1.6, while Section 1.7 concludes the chapter.

1.2 Related Work

Over the past recent years, researchers have to mitigate the impact of dark silicon to some extent. In general, the major contributions can be classified into three main categories: 1) heterogeneous computing including asymmetric multicore and 3D architectures, 2) Dim Silicon and near-threshold computing including DVFS techniques, and 3) variable symmetric multiprocessing (vSMP) technology (i.e., device-level heterogeneity).

Goulding-Hotta *et al.* [9] present the GreenDroid architecture which uses specialized energy-efficient processors to execute frequently used portions of the application code. The authors claim that in their Android-based workload, the specialized cores cover about 95% of the execution time. However, the target of their proposed architecture and approach is general-purpose smartphone applications. Therefore, their specialized cores can be utilized to execute only popular frequently used applications (e.g., browser, gallery, maps). Esmaeilzadeh *et al.* [10] propose a neural network based approach to the acceleration of approximate programs. In their approach, the compiler finds code sections which can be replaced by an invocation of a low-power accelerator called a neural processing unit using a learning phase and interchanges these parts accordingly. This approach can be used for specific application domains where approximate computation can be accepted such as signal processing, data mining, and robotics.

In [5], a hierarchical power management framework for asymmetric multi-core architectures is demonstrated for ARM big.LITTLE [11] mobile platforms. In this architecture, cores have different size and processing power while having the same instruction-set-architecture (ISA). Ma *et al.* [6] 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 contrast, our approach is applied to NoC-based general-purpose manycore systems in excess of hundred cores. The NoC-based communication structure of manycore systems necessitates more advanced controllers capable of considering workload characteristics and network congestion, in addition to power-performance feedbacks. In addition, both platforms [5, 6] do not consider runtime application mapping making it trivial to control as no disturbance occurs when an application enters or leaves the system.

Wang *et al.* [1] attack dark silicon by using near-threshold computing capable of increasing the number of simultaneously active cores, at the expense of lower operating frequency (i.e. dim silicon). Even though the approach presents promising speedup with the same power budget, they do not present any solution or control mechanism on how it can be used to manage the power consumption of manycore systems. vSMP [12] is another energy-efficient methodology presented by NVIDIA where cores with the same architecture but fabricated by different silicon processes are integrated, some using a low power silicon process and others using a standard silicon process. This approach was tested for bus-based embedded mobile processors including five cores and optimized for key mobile use cases. However, the approach does not consider dark silicon related limitations where, for example, TDP should be taken into account.

There are several works dealing with management of dynamic workload in many-core systems. Early works in the supercomputer domain map applications only onto convex set of nodes [13]. While recent works focus on efficient processor allocation methods for many-core systems [14, 15, 16]. However, none of these works is dark silicon aware. More precisely, they do not control the system power consumption with respect to TDP.

In [17], a power management technique is presented for on-chip communication network. In this work the voltage and frequency of the interconnection network are adjusted to gain power efficiency when the network operates just below the saturation point. The approach is based on a feedback controller but focuses on the network. Its objective is to minimize power consumption while delivering all the data requested by the application. In contrast, we focus on the processing cores, which have considerably higher potential to save power, and we deal with a hard upper constraint on power consumption. In [3], 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. On the other hand, their approach is for VFI-based NoCs where VFIs are formed in design time and hence general-purpose runtime application mapping is not properly supported.

Haghbayan *et al.* [18] present a power management technique for many-core systems using power feedback from the system to meet the TDP bound. This technique is categorized to single objective control approach 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 [18], is designed for fixed TDP and does not benefit from a dedicated disturbance rejector to handle sudden overshoots when new applications commence execution.

In [19], Chen *et al.* present a power allocation technique for many-core system performance improvement under power constraints. They formulate a performance optimization problem and apply an optimal power allocation method using on-line distributed reinforcement learning. This contribution does not consider on-chip communication in the problem formulation. However, it should be noted that this work is orthogonal to our multi-objective control management and can complement and enhance our method by integrating the concept of on-line learning towards more efficient global power budget reallocation.

There have been some efforts to minimize the power consumption of on-chip communication network in the dark silicon era [20, 21]. However, we focus on the processing cores, which have considerably higher potential to save power. However, these techniques can also complement our platform to manage the power consumption of the interconnection network to further optimize the power.

Although such efforts have greatly expanded in recent years, there is relatively small improvement in mitigating the dark silicon issue due to highly dynamic nature of general-purpose target workloads. The other important aspect is the dynamicity of the dark/dim area as it grows and shrinks at runtime. This motivates us to provide a general framework that can handle any number of cores in a NoC-based environment running highly dynamic workloads, minimize energy by entering even near-threshold operation, and satisfy QoS and thermal constraints by considering network congestion and application characteristics.

1.3 Preliminaries

Each application in the system is represented by a directed graph denoted as a task graph $Ap = TG(T,E)$. Each vertex $t_i \in T$ represents one task of the application, while the edge $e_{i,j} \in E$ stands for a communication between the source task t_i, and the destination task t_j [22]. Task graph of an application extracted using TGG [23] is shown in Figure 1.1.

An architecture graph $AG(N,L)$ describes the communication infrastructure of the processing elements. We consider a 2D mesh NoC (Figure 1.1) with XY deterministic wormhole routing. The AG graph contains a set of nodes $n_{w,h} \in N$, connected together through unidirectional links $l_k \in L$. Each node is the combination of a PE connected to a router.

We define a non-real-time task as 3-tuple $tnr_i = \langle id_i, ex_i, pr_i \rangle$, and a real-time task as 5-tuple $tr_i = \langle p_i, id_i, ex_i, d_i, pr_i \rangle$, where: id_i stands for the task identification, p_i represents period of task i; ex_i represents the task execution time, d_i is its deadline, and pr_i denotes the task priority. We define an abstract time unit, called as tick (e.g. 1 ms) [22].

We define an application as a set of tasks having inter-dependencies. Therefore, application mapping is a one-to-many function. We use a simple mathematical model for representing applications running on the system. Hence, no multi-tasking is assumed in any node. We denote by Application Matrix (AM) the matrix whose

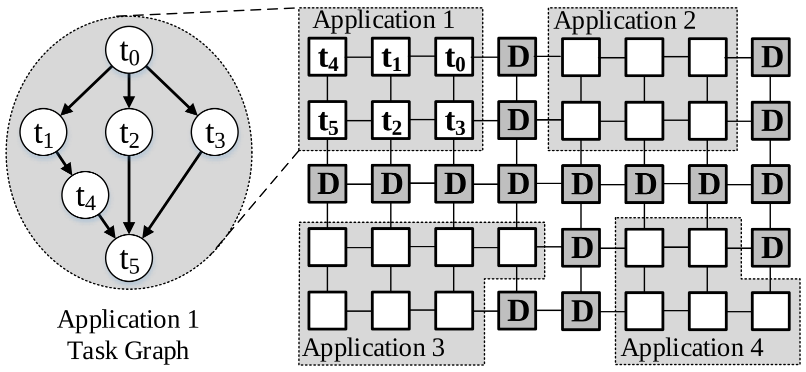

Fig. 1.1: Mesh-Based platform with an application mapped onto it (the highlighted region.) where some cores are dark (D)

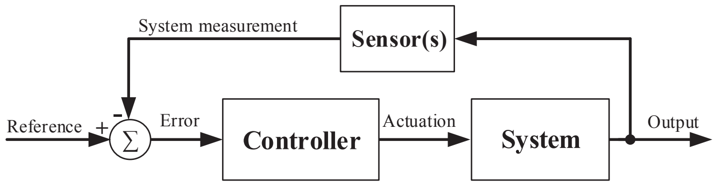

Fig. 1.2: structure of the feedback controller

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. For example, the following application matrix shows how four applications with IDs from 1 to 4 are mapped onto a 4×4 mesh-based NoC.

$$AM = \begin{bmatrix} 1 & 1 & 4 & 4 \\ 1 & 1 & 3 & 4 \\ 1 & 2 & 3 & 3 \\ 2 & 2 & 2 & 3 \end{bmatrix} \tag{1.1}$$

1.4 Power Management Platform

Structure of the proposed dark silicon aware power management platform is shown in Figure 1.2. As can be seen, a feedback controller that make use of system power measurement is incorporated. Similar to every other control systems, the controller compares the system output with a target value. The system output in our framework is the overall power consumption of the system and the target value can be TDP or TSP. After comparison, it manipulates the system actuators to minimize the error. The controller policy to tune the actuators strongly depends on the dynamic model of the target system and the system robustness against error disturbance. The dynamic model defines how the system reacts to the inputs including actuations and other inputs. The system robustness is defined as the system stability against overshooting of the output values from the target intended output.

In our framework, per-core DVFS, per-core power gating, and application termination are used as actuators. It should be noted that the power manager does not scale the voltage and frequency of the interconnection network components (e.g., routers, links), to ensure that there is no waiting time and gainless static power consumption of the consumer PEs.

Details of the multi-objective controller (MOC) is presented in Figure 1.3. The framework represents a general controlling strategy for many-core systems enabled with run-time mapping that can easily be applied to any NoC topologies such as 3D architectures. Run-time Mapping Unit (RMU) allocates cores in the NoC-based system to tasks of applications commenced for execution. Some information regarding the mapped applications is also provided by this unit to be passed to other controlling units. This is what we call Runtime Application Information (RAI). The priority of an application is proportional to the amount of expected *QoS* for that application. On a system, there might be different types of applications running with different priorities. For example soft realtime and non-realtime application might have different levels of priority in such systems.

As discussed before, an efficient ODA-based management strategy in this context requires several observation units to monitor different system characteristics at runtime. In the following, we list the observation units integrated in our platform. It should be mentioned that the observation can be enhanced by including other system characteristics such as thermal profile, aging profile, etc.

1.4.1 Application Power Calculator

Each tile in our platform is assumed to be equipped with a power sensor. Power sensors transfer the information about instantaneous power consumption of cores to the central manager which forms Tile Power Matrix. It is worth to mention that many of today's platforms support such power meter equipments, e.g., Versatile Express Development Platform [11] which supports per-cluster power meters.

There are techniques presented in the literature to measure per-core power consumption by reading out current and voltage. For instance, Bakker *et al.* [24] propose a power measurement algorithm for Intel SCC [25] and Ma *et al.* [6] in the *PGCapping* approach propose a hybrid technique to monitor power consumption of individual cores. The importance of power meters is getting more evident. For instance, Esmaeilzadeh *et al.* [26] state that "Just as hardware event counters provide a quantitative grounding for performance innovations, power meters are necessary for optimizing energy".

The power consumption of individual routers also vary dynamically due to primarily uneven traffic distribution in a network. When a set of system resources are

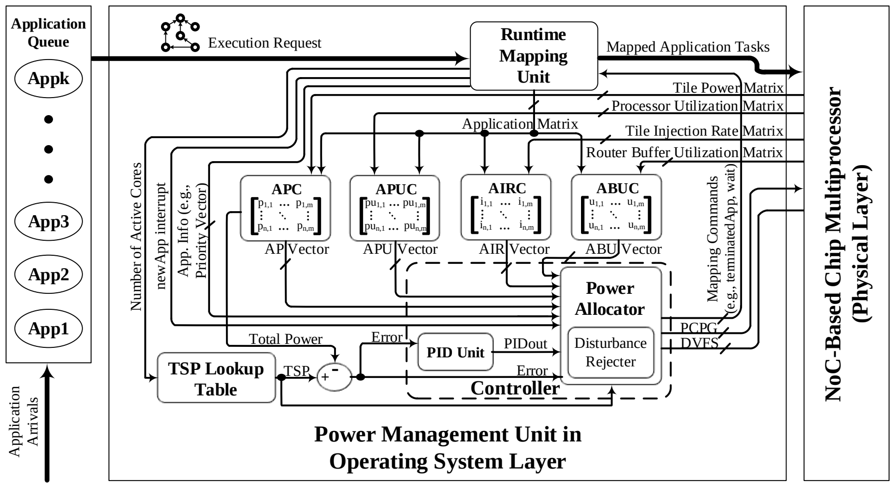

Fig. 1.3: Overview of the multi-objective dark silicon aware power management system (AIRC: Application Injection Rate Calculator, ABUC: Application Buffer Utilization Calculator, APUC: Application Processor Utilization Calculator, APC: Application Power Calculator)

allocated to a task, part of the network become active with packets flowing in different directions. The associated routers regulating the packet flow thus dissipate proportional power in order to manage such a traffic.

To accurately measure power dissipation in routers, a power meter is designed within the router micro-architecture [27]. The power meter reads the rate of packet flow at link level and sends its aggregate value to the central control. There are four directional links (South, West, East, North) and a local link connecting to a processing element. If there is no packet flow in all links, then only the leakage power is consumed. If every link is passing a packet per cycle, then the router is consuming 100% its dynamic power actively. This happens when the network traffic is congested. However, under optimal conditions for unsaturated network, the router level power reading is less than 100%.

The aggregate value of both core and router power consumption is sent to the central controller. Application Power Calculator (*APC*) unit in our platform calculates the current power consumption of each application based on the per tile power consumption values received from power meters and mapping information from DMU. This unit calculates Application Power Vector (*APV*), which contains the current power consumption value for each application. This is done by masking Application Matrix on the Tile Power Matrix.

1.4.2 Application Processor Utilization Calculator

To monitor the impact of power capping on performance, we equipped processors with performance counter through which the utilization of the processing element, in a specified time interval, is calculated and reported to the central controller. In the same manner as *APC*, Application Processor Utilization Calculator (*APUC*) unit calculates the aggregate processor utilization for each application based on the Processor Utilization Matrix resulting the Application Processor Utilization Vector (*APUV*).

1.4.3 Application Buffer Utilization Calculator

An ideal network topology is one with a scalable network configuration and traffic distribution where every packet is transmitted and received without delay and bandwidth limitations. For example, a network running a highly localized traffic where every node sends packets only to its immediate neighboring node, can be considered as an ideal one because it exploits its maximum performance. There is no traffic congestion in the network and each packet reaches its destination within a predictable latency. Nevertheless, in practice, network traffic distribution is non-uniform and due to interconnection complexity and intrinsic wire delays such an ideal topology is not feasible. Instead more practical configurations, such as generic 2-D mesh or 3-D cube topologies, are used. However, the scalability of such practical topologies is limited as the capacity of the networks do not grow proportionally to accommodate traffic generated with increasing number of cores [28].

For each added core, the network traffic gets more easily congested and the overall throughput per-core decreases. Hence the total network performance gives a diminishing return due to increased communication distance. This leads to a network performance gap where every core is not able to send or receive packets in every cycle. In such cases, there is no need for a core to operate at high frequencies or voltages. Thus, we find it imperative to take the network performance gap into account when designing dynamic power management for many-core systems.

In our platform, each router is equipped with a buffer utilization meter that measures router congestion level in a specified time interval. More precisely, it measures the traffic dynamically by calculating the moving average of packet flow in every link of a router as follows:

$$C_{Total} = \frac{1}{W} \sum_{cycle=i}^{i+w} (\theta_{South,i} + \theta_{North,i} + \theta_{East,i} + \theta_{West,i} + \theta_{Local,i}) \tag{1.2}$$

Where $\theta = \{0,1\}$ is the presence or absence of a packet in a link at any given cycle, W is the width of the moving window, and C_{Total} is the moving average congestion level. This buffer utilization of each router (Router Buffer Utilization Matrix in Figure 1.3) is sent to the Application Buffer Utilization Calculator (*ABUC*).

Then, by masking the Application Matrix (provided by the RMU) on the Router Buffer Utilization Matrix, *ABUC* calculates the average buffer utilization vector, i.e., *ABUV*, which is exploited in the controller unit.

1.4.4 Application Injection Rate Calculator

In the power management process, applications are categorized and managed based on their computation intensiveness as well as communication intensiveness. In [7], it is shown that a source-throttling congestion control mechanism will have a limited performance improvement if it is done only based on the network load. Such a mechanism is not application aware, but rather throttles all applications equally regardless of applications' sensitivity to latency. Inspired from [7], 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 the last epoch 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 put it on the use at the controller unit.

1.4.5 TSP Lookup

TDP calculation is performed at a design time regardless of runtime thermal distribution across the silicon area. Therefore, using a constant value as an upper bound for power (i.e., TDP) can result in large performance losses [2]. To optimize the power budget calculation, a new power budget concept called Thermal Safe Power (TSP) has been proposed which is a function of number of active, i.e. non-dark, cores in a system determined at runtime. In our platform, TSP_{worst} is used which is calculated for the worst-case mapping through which it is assumed that all active cores are physically packed and influencing the temperature of their adjacent cores. Other parameters needed to calculate TSP, e.g., floorplan, power consumption of an inactive core, etc., are generally available at a design time. The worst-case calculation function returns a uniform value of TSP per-core for all active cores.

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). The overhead of such a lookup table is negligible as it needs H entries where H is the number of cores in the system. The *number of active cores* can be used as the index for the array to avoid any search function. For example, $P_{TSP}^{worst}(9) = 12.5W$ indicates that the safe power budget is 12.5W when there are 9 active cores in the system. This function is called whenever an application enters or leaves the system at runtime. It should be noted that if a fixed TDP value is desired, the lookup table can be simply replaced with the fixed value.

1.5 Power Management Policy

In the previous sections, the multi-objective Observe-Decide-Act (ODA) loop were discussed. As shown in Figure 1.3, within the Controller Unit, a Proportional-Integral-Derivative (PID) controller monitors the error between the actual power consumed by the system and the reference thermal power (i.e., TDP or TSP). Then, the downstream Power Allocator Unit decides how to handle the power management features based on the output of PID controller and other system parameters. In this section, we present the decision making policy (i.e., Decide state comprising different controllers) in detail.

1.5.1 PID Controller Unit

The general expression for a PID controller is formulated as follows:

$$PID_{out}(t) = K_p e(t) + K_i \int e(t)\,\mathrm{d}t + K_d \frac{de(t)}{dt} \tag{1.3}$$

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.

Several Matlab simulations are performed to adjust the gains of the PID controller. The system stability and robustness are two essential aspects that need to be carefully considered when adjusting the gains. In a PID controller, each gain magnifies the importance of a specific system behaviour. The proportional gain directs a change in the controller output by magnifying the change of the error value. The integral gain responds to the accumulated errors that magnifies the effect of history on controller's output value to eliminate the residual steady-state error. The derivative gain predicts the system behaviour by magnifying the most recent changes in error value.

A high proportional gain leads to a large change in the output for a given change in the error value. Therefore, the proportional gain should have the foremost contribution when determining the controller's output. On the other hand, the integral term accelerates the movement of the process towards the target value. However, as the integral term responds to the accumulated errors from the past, it can affect the present value by overshooting the target value. Derivative action predicts system behavior and thus improves settling time and stability of the system.

In multiprocessor systems utilizing runtime task mapping, there are often three influential bearings 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. These three behaviors make the system workload often highly dynamic and unpredictable, resulting in sudden steep slopes in the power trace curve, i.e. disturbances. We address all the three behaviors. The PID controller can efficiently handle the second and third behaviour. However, when an application enters the system (the first behaviour), a high overshoot may 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, discussed in later sections. In summary, *Power Allocator handles the second and third behaviours using PID Controller unit while it manages the first behaviour with the help of Disturbance Rejector unit*.

1.5.2 Power Allocator

The main task of the power allocator unit is to manipulate voltage and frequency of the processing elements, i.e., (V_{PEs}, $Freq_{PEs}$), by utilizing the information obtained from the observation units and the directions provided by the PID controller. Algorithm 1 shows the process to obtain V_{PEs} and $Freq_{PEs}$. At the first step, each active core's power limit is determined by dividing the overall power budget i,e, TDP or TSP, by the number of active cores. After that, based on applications' injection rate (IR) information obtained form Application Injection Rate Vector (AIRV), all the applications are classified into two categories, intensive (I_{set}) and non-intensive (NI_{set}), by making use of *IRClassifier* function. Similarly, through *BUClassifier* function the applications are also classified into two categories, congested (C_{set}) and non-congested (NC_{set}) based on their buffer utilization (BU) information obtained from Application Buffer Utilization Vector (ABUV). An application is considered to be congested if its corresponding routers' buffer utilization value is larger than a predefine threshold, for example 75%. Figure 1.4 shows four possible application types after classification by *IRClassifier* and *BUClassifier* functions. 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. This provides appropriate target set of applications that can be upscaled or downscaled to maximize network throughput.

After clasification, based on the applications' type and the PID_{out} (the output of the PID controller), voltage/frequency of each processing elements is downscaled or upscaled. There can be two possible scenarios to deal with: overshoot (i.e., power consumption exceeding TSP/TDP) and undershoot (i.e., power consumption beneath the TSP/TDP). An overshoot violates TSP/TDP constraint, while an un-

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
```

dershoot represents resource under-utilization. $VF_{downscaler}$ 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. The power monitoring continues normally when there is no new application arrival. Pruning the application space in case of an overshoot and undershoot are explained in the following subsections.

Application Types

Non-Congested (*NC*) Non-Intensive (*NI*)	**Non-Congested (NC) Intensive (I)**
Congested (C) Non-Intensive (NI)	**Congested (C) Intensive (I)**

Fig. 1.4: Four possible applications types classified by *IRClassifier* and *BUClassifier* functions

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
```

1.5.2.1 Voltage-Frequency 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. RAI includes application priorities known from the application priority vector. 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 a low network throughput. As VF downscaling also affects on throttling of packet injection, it can alleviate 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 application ($targetApp$) is then downscaled by the $DVFS$ function as per PID_{out} and $PCPowerLimit$.

The DVFS function fails when it cannot throttle the target application any further according to the application type, which occurs when VF level cannot be reduced anymore. When the $failedDVFS$ flag is asserted, the application will be removed from the $availableApp$ and the algorithm will keep on searching until an alternative application is found.

1.5.2.2 Voltage-Frequency Upscaler

Voltage/Frequency 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 to 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 $PCPowerLimit$.

The DVFS function considers both normal and near-threshold operations. Voltage-to-frequency scalings are modeled by interpolating empirical results from circuit simulations [29]. Transistor switching speed scales exponentially with the threshold voltage while operating at near-threshold voltage. As a result, near-threshold operation region is highly sensitive to the threshold voltage [29]. More details regarding near-threshold frequency and voltage modeling can be found in [29]. As there are limited amount of voltage and frequency levels, the $DVFS$ function ignores marginal deviations of PID_{out} from its previous value for the sake of stability.

1.5.2.3 functions $highD_{prf-pwr}$ and $lowD_{prf-pwr}$

functions $highD_{prf-pwr}$ and $lowD_{prf-pwr}$ 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 [6], 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 as:

$$Perf_{current} = appUtil \times Freq_{current} \tag{1.4}$$

Then, the performance-power ratio is calculated as the following:

$$D_{prf-pwr} = \frac{Perf_{next} - Perf_{current}}{Power_{next} - Power_{current}} \tag{1.5}$$

$Power_{current}$ is the power consumption of the current application provided by the APC unit. $Power_{next}$ and $Perf_{next}$ are the estimated power consumption and performance of the application after the DVFS process. The next level of voltage and frequency (V_{dd_next} and $Freq_{next}$) are estimated for the candidate applications based

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 $DVFSList$;
```

on the magnitude of PID_{out} and application size. The $Perf_{next}$ and $Power_{next}$ are calculated as follows:

$$Perf_{next} = Perf_{current} \times \frac{Freq_{next}}{Freq_{current}} \tag{1.6}$$

$$Power_{next} = Power_{current} \times \frac{Freq_{next}}{Freq_{current}} \times (\frac{V_{dd\_next}}{V_{dd\_current}})^2 \tag{1.7}$$

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.

1.5.2.4 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 *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 av-

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
```

erage 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 that can be allocated to new application, it is mapped as it 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}. The new application is pushed onto the stack, annexed to the list of applications running with DVFS.

1.6 Experimental Results

In this section, we present the experimental results with 16nm Complementary metaloxidesemiconductor (CMOS) technology node for evaluating our proposed multi-objective dark silicon aware power management platform.

1.6.1 Experiment Setup

To experimentally evaluate the proposed approach, we implemented a system-level simulation platform for the described many-core architecture together with accompanying runtime management layer and testing procedures in SystemC on the basis of Noxim NoC simulator [30]. The basic core has been characterized by using the

Niagara2-like in-order core specifications obtained from McPAT [31]. Physical scaling parameters were extracted from the Lumos framework (by Wang and Skadron) [29]. 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 [32]. The physical scaling parameters have been calibrated by circuit simulations with a modified Predictive Technology Model [33]. 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 [23] where the communication and computation volumes are randomly distributed. We model MPEG4 and VOPD multimedia applications as soft realtime applications. The realtime requirements of these applications require the system to respond within certain deadlines for different priority levels. We pre-calculate the minimum VF level for soft realtime tasks for their worst-case contiguous mapping. Soft realtime here means that these applications can provide different quality of services for example by processing different frame rates per second depending on the availability of system resources.

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 [34] 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. This makes the area overhead of the proposed method so negligible. In other words, the congestion meters embedded in the NoC routers and the power sensors are the only extra hardware components needed to implement our idea. CM receives feedbacks from the whole network and sends actuation commands to each tile. These short control packets are synchronously sent along with the ordinary traffic using the same on-chip network. At first glance, the centralized approach seems unscalable and the control traffic overhead looks considerable. 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 and control packets have enough time to reach the destination even in large networks. 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.

In our platform, we assumed that the chip is equipped with power sensors. Therefore, we need to model the power sensors to estimate the power consumption in our simulations. A PE can be in three different states in our simulations: 1) "*Busy*" when all the required input packets have been received and the corresponding task is being executed, 2) "*Waiting*" when a PE is clock-gated as it is waiting (due to data dependencies) for upcoming packets to be completely stored in the input FIFO, and 3) "*Dark*" when a PE is idle and power-gated. We estimate the power consumption for these different states according to the following equations:

$$P_{Busy} = P_{static} + P_{dynamic} \tag{1.8}$$

$$P_{Waiting} = P_{static} \tag{1.9}$$

$$P_{Dark} \simeq 0 \tag{1.10}$$

where P_{static} and $P_{dynamic}$ are static power and dynamic power of each core in the system, respectively. The static and dynamic power consumption are calculated using the following equations [29]:

$$P_{dynamic} = \alpha.C_{eff}.V_{dd}^2.f \tag{1.11}$$

$$P_{static} = V_{dd}.N.k_{design}.\hat{I}_{leak} \tag{1.12}$$

where α denotes the switching activity factor of a PE while running a task, C_{eff} is the effective capacitance, V_{dd} is the supply voltage, f is the core's frequency, N is the number of transistors, k_{design} is a device-specic constant, and $\hat{I}_{leak}$ is the normalized per-transistor leakage current. To be consistent with the parameters used in the Lumos framework, we assume a constant activity factor and effective capacitance when the core frequency is scaled with various supply voltages. More details regarding the above equations can be found in [29].

For the DVFS purpose, we use 15 VF levels (similar to Intel SCC) including near-threshold operation extracted from the Lumos framework. The minimum and maximum VF levels are set to (0.456V, 300MHz) and (0.908V, 5.2GHz), respectively. Lumos models voltage-frequency scalings in an *optimistic way* - without considering reliabilities and also in *pessimistic way* - considering the process variations through an analytical model with rigid voltage downscaling. We chose the pessimistic option in our simulations to ensure a higher degree of reliability for same architecture, sacrificing possible energy gains.

We define different minimum voltage-frequency levels depending on the application type. For example, we use all the 15 levels to scale voltages and frequencies of PEs running non-realtime applications. The frequency of the on-chip communication network (e.g., routers) is set to the maximum level (i.e., 5.2GHz) 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 [2]. We set ambient temperature to 45°C, a threshold temperature that triggers thermal management to 80°C, maximum chip

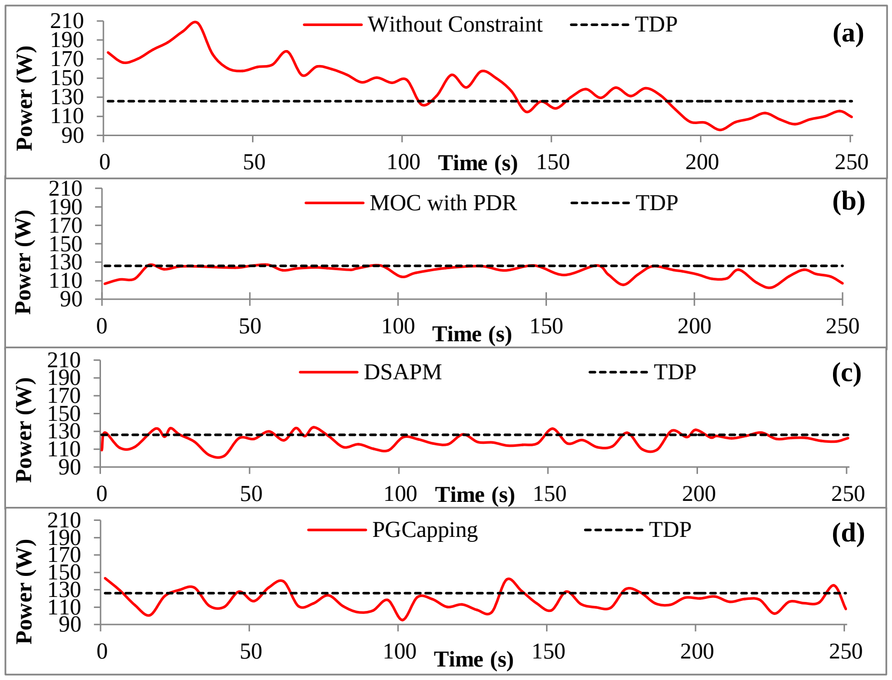

Fig. 1.5: The power consumption of the system using (a) without TSP/TDP constraint, (b) MOC with PDR, (c) DSAPM, and (d) PGCapping power management policies to honor TDP

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 [6] in which the power management technique only considers core's power-performance ratio as the feedback for the PCPG and per-core DVFS actuation, 3) DSAPM [18] in which PID controller is used to controll the system power consumption, however no information regarding performance and packet injection rate of PEs is considered in power allocation policy, and 4) without TSP/TDP constraint in which there is not any policy to controll the power. 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. We consider 10s warm up phase for the results. To perform a fair comparison, we run PGCapping and DSAPM techniques with the same 15 VF levels for per-core DVFS actuation.

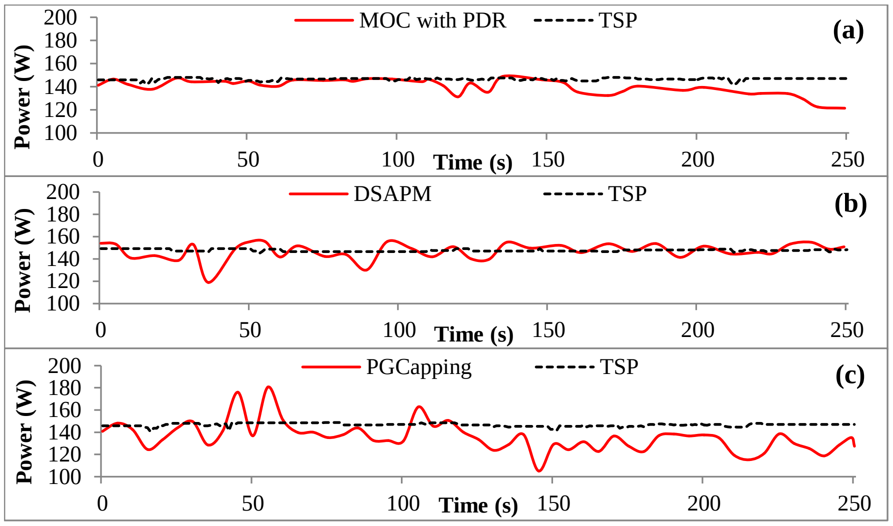

Fig. 1.6: The power consumption of the system using (a) MOC with PDR, (b) DSAPM, and (c) PGCapping power management policies to honor TSP

1.6.2 Results

Power consumption of the system under the aforementioned power management scenarios to honor constant TDP is presented in Figure 1.5. 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 1.6 demonstrates the aforementioned power management scenarios to honor dynamic TSP values. As can be observed from the figure, the conclusions we made for Figure 1.5 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 1.7. 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 1.8 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.

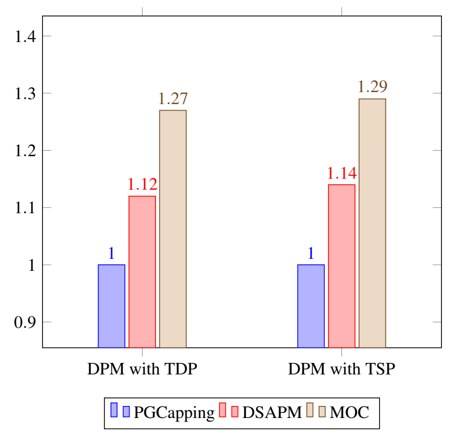

Fig. 1.7: Normalized throughput of MOC vs. PGCapping vs. DSAPM

1.7 Conclusions

The need to utilize controlling mechanisms in management of complex multiprocessor systems is becoming more evident particularly when number of cores in a chip increases. In this chapter, we introduced a multi-objective feedback based controller approach to protect many-core systems against exceeding the power con-

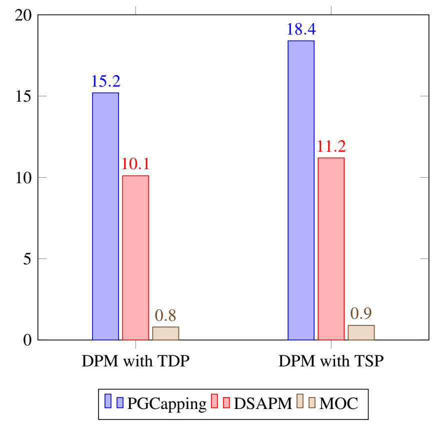

Fig. 1.8: TDP/TSP violation for different dynamic power managements

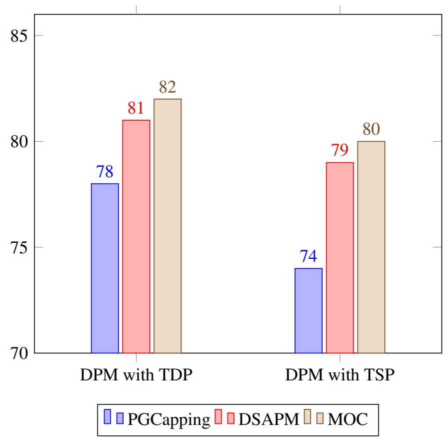

Fig. 1.9: Average of the core utilization (%) for different dynamic power managements

sumption from a certain limit while maximizing system utilization and throughput. The target system architecture was a Network-on-Chip-based multiprocessor system using dynamic application mapping where applications enter and leave the system at runtime. We utilized a variety of feedbacks such as processing elements' power-performance measurements, application workloads, and network congestion to monitor the system. Two different algorithms for down-scaling and up-scaling the voltage and frequencies of the cores, and a proactive strategy to avoid power consumption violations when the system encounters rapid power increases were discussed. It was shown that the controller efficiently changes voltage and frequency of appropriate processing elements, down to near threshold operation when needed. The results show improvements in system throughput as well as reductions in TDP/TSP violations, for the proposed platform when compared to state-of-the-art power management policies.

References

1. W. Liang and K. Skadron. Implications of the Power Wall: Dim Cores and Reconfigurable Logic. *IEEE Micro*, 33(5):40–48, 2013.
2. S. Pagani, H. Khdr, W. Munawar, J. Chen, M. Shafique, M. Li, and J. Henkel. TSP: Thermal Safe Power: Efficient Power Budgeting for Many-core Systems in Dark Silicon. In *Proc. of the 2014 International Conference on Hardware/Software Codesign and System Synthesis*, CODES '14, 2014.
3. P. Bogdan, R. Marculescu, and S. Jain. Dynamic Power Management for Multidomain System-on-chip Platforms: An Optimal Control Approach. *ACM Trans. Des. Autom. Electron. Syst.*, 18(4):46:1–46:20, 2013.
4. R. David, P. Bogdan, R. Marculescu, and U. Ogras. Dynamic Power Management of Voltage-Frequency Island Partitioned Networks-on-Chip Using Intel Sing-Chip Cloud Computer. In *Proc. of the Fifth ACM/IEEE International Symposium on Networks-on-Chip*, NOCS '11, pages 257–258, 2011.
5. T.S. Muthukaruppan, M. Pricopi, V. Venkataramani, T. Mitra, and S. Vishin. Hierarchical power management for asymmetric multi-core in dark silicon era. In *Proc. of Design Automation Conference*, pages 1–9, 2013.
6. K. Ma and X. Wang. PGCapping: Exploiting Power Gating for Power Capping and Core Lifetime Balancing in CMPs. In *Proc. of the 21st International Conference on Parallel Architectures and Compilation Techniques*, PACT '12, pages 13–22, 2012.
7. K.K. Chang, R. Ausavarungnirun, C. Fallin, and O. Mutlu. HAT: Heterogeneous Adaptive Throttling for On-Chip Networks. In *Proc. of IEEE 24th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*, pages 9–18, 2012.
8. A.-M. Rahmani, M.-H. Haghbayan, A. Kanduri, A.Y. Weldezion, P. Liljeberg, J. Plosila, A. Jantsch, and H. Tenhunen. Dynamic Power Management for Many-Core Platforms in the Dark Silicon Era: A Multi-Objective Control Approach. In *Proc. of Int. Symp. on Low Power Electronics and Design (ISLPED)*, pages 1–6, 2015.
9. N. Goulding-Hotta, J. Sampson, G. Venkatesh, S. Garcia, J. Auricchio, P. Huang, M. Arora, S. Nath, V. Bhatt, J. Babb, S. Swanson, and M.B. Taylor. The GreenDroid Mobile Application Processor: An Architecture for Silicon's Dark Future. *IEEE Micro*, 31(2):86–95, 2011.
10. H. Esmaeilzadeh, A. Sampson, L. Ceze, and D. Burger. Neural Acceleration for General-Purpose Approximate Programs. In *Proc. of IEEE/ACM International Symposium on Microarchitecture*, pages 449–460, 2012.
11. ARM Ltd., http://www.arm.com/products/tools/development-boards/versatile-express/index.php, 2011.
12. Variable SMP: A multi-core cpu architecture for low power and high performance. In *White paper, Nvidia*, 2011.
13. Michael A. Bender, David P. Bunde, Erik D. Demaine, Sandor P. Fekete, Vitus J. Leung, Henk Meijer, and Cynthia A. Phillips. Communication-Aware Processor Allocation for Supercomputers: Finding Point Sets of Small Average Distance. *Proc. of Algorithmica*, pages 279–298, 2008.
14. E. Carvalho, N. Calazans, and F. Moraes. Heuristics for Dynamic Task Mapping in NoC-based Heterogeneous MPSoCs. In *Proc. of 18th IEEE/IFIP International Workshop on Rapid System Prototyping*, pages 34–40, 2007.
15. M. Fattah, P. Liljeberg, J. Plosila, and H. Tenhunen. Adjustable contiguity of run-time task allocation in networked many-core systems. In *Proc. of Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 349–354, 2014.

16. M.-H. Haghbayan, A. Kanduri, A.-M. Rahmani, P. Liljeberg, A. Jantsch, and H. Tenhunen. MapPro: Proactive Runtime Mapping for Dynamic Workloads by Quantifying Ripple Effect of Applications on Networks-on-Chip. In *Proc. of Int. Symp. on Networks-on-Chip (NOCS)*, pages 1–8, 2015.
17. G. Liang and A. Jantsch. Adaptive Power Management for the On-Chip Communication Network. In *Proc. of the EUROMICRO Conference on Digital System Design*, pages 649–656, 2006.
18. M.-H. Haghbayan, A.-M. Rahmani, A.Y. Weldezion, P. Liljeberg, J. Plosila, A. Jantsch, and H. Tenhunen. Dark silicon aware power management for manycore systems under dynamic workloads. In *Proc. of Int. Conf. on Computer Design (ICCD)*, pages 509–512, 2014.
19. Z. Chen and D. Marculescu. Distributed Reinforcement Learning for Power Limited Many-core System Performance Optimization. In *Proc. of the 2015 Design, Automation & Test in Europe Conference & Exhibition*, pages 1521–1526, 2015.
20. H. Bokhari, H. Javaid, M. Shafique, J. Henkel, and S. Parameswaran. darkNoC: Designing energy-efficient network-on-chip with multi-Vt cells for dark silicon. In *Proc. of 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6, 2014.
21. J. Zhan, Y. Xie, and G. Sun. NoC-sprinting: Interconnect for fine-grained sprinting in the dark silicon era. In *Proc. of 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6, 2014.
22. M. Fattah, A.-M. Rahmani, T.C. Xu, A. Kanduri, P. Liljeberg, J. Plosila, and H. Tenhunen. Mixed-Criticality Run-Time Task Mapping for NoC-Based Many-Core Systems. In *Proc. of International Conference on Parallel, Distributed and Network-Based Processing*, pages 458–465, 2014.
23. TGG: Task Graph Generator. *URL: http://sourceforge.net/projects/taskgraphgen/*, 2010.
24. R. Bakker, M.W. van Tol, and A.D. Pimentel. Emulating Asymmetric MPSoCs on the Intel SCC Many-core Processor. In *Proc. of the Eurpmicro International Conference on Parallel, Distributed, and Network-Based Processing*, pages 520–527, 2014.
25. J. Howard, S. Dighe, Y. Hoskote, and Vangal. A 48-Core IA-32 message-passing processor with DVFS in 45nm CMOS. In *Proc. of Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pages 108 –109, feb. 2010.
26. H. Esmaeilzadeh, T. Cao, Y. Xi, S.M. Blackburn, and K.S. McKinley. Looking Back on the Language and Hardware Revolutions: Measured Power, Performance, and Scaling. In *Proc. of the Sixteenth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XVI, pages 319–332, 2011.
27. A.Y. Weldezion, M. Grange, D. Pamunuwa, A. Jantsch, and H. Tenhunen. A scalable multi-dimensional NoC simulation model for diverse spatio-temporal traffic patterns. In *Proc. of the IEEE International 3D Systems Integration Conference*, pages 1–5, 2013.
28. A.Y. Weldezion, M. Grange, D. Pamunuwa, Zhonghai Lu, A. Jantsch, R. Weerasekera, and H. Tenhunen. Scalability of network-on-chip communication architecture for 3-D meshes. In *Proc. of the International Symposium on Networks-on-Chip*, pages 114–123, 2009.
29. L. Wang and K. Skadron. Dark vs. Dim Silicon and Near-Threshold Computing Extended Results. In *University of Virginia Department of Computer Science Technical Report TR-2013-01*, 2012.
30. F. Fazzino, M. Palesi, and D. Patti. Noxim: Network-on-chip simulator. *URL: http://sourceforge.net/projects/noxim*, 2008.
31. S. Li, J.H. Ahn, R.D. Strong, J.B. Brockman, D.M. Tullsen, and N.P. Jouppi. McPAT: An Integrated Power, Area, and Timing Modeling Framework for Multicore and Manycore Architectures. In *Proc. of the 42Nd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 469–480, 2009.
32. Lumos Framework, http://liangwang.github.io/lumos/. Accessed: 2014-05-20.
33. B.H. Calhoun, S. Khanna, R. Mann, and Jiajing Wang. Sub-threshold circuit design with shrinking CMOS devices. In *Proc. of the International Symposium on Circuits and Systems*, pages 2541–2544, 2009.
34. M. Fattah, M. Daneshtalab, P. Liljeberg, and J. Plosila. Smart hill climbing for agile dynamic mapping in many-core systems. In *Proc. of Design Automation Conf (DAC)*, pages 1–6, 2013.