Dark Silicon Aware Power Management for Manycore Systems under Dynamic Workloads

Mohammad-Hashem Haghbayan[1], Amir-Mohammad Rahmani[1], Awet Yemane Weldezion[2], Pasi Liljeberg[1], Juha Plosila[1], Axel Jantsch[2], and Hannu Tenhunen[1,2]
[1]Department of Information Technology, University of Turku, Turku, Finland
[2]School of ICT, KTH Royal Institute of Technology, Stockholm, Sweden
Email: {*mohhag, amirah, pakrli, juplos*}*@utu.fi*, {*aywe, axel, hannu*}*@kth.se*

Abstract—**Dark Silicon denotes the phenomenon that, due to thermal and power constraints, the fraction of transistors that can operate at full frequency is decreasing with each technology generation. We propose a PID (Proportional Integral Derivative) controller based dynamic power management method that considers an upper bound on power consumption (called the Thermal Design Power (TDP)). To avoid violation of the TDP constraint for manycore systems running highly dynamic workloads, it provides fine-grained DVFS (Dynamic Voltage and Frequency Scaling) including near-threshold operation. In addition, the method distinguishes applications with hard Real-Time, soft Real-Time and no Real-Time constraints and treats them with appropriate priorities. In simulations with dynamic workloads mixed-critical application profiles, we show that the method is effective in honoring the TDP bound and it can boost system throughput by over 43% compared to a naive TDP scheduling policy.**

Keywords—Dark Silicon; Power Management; Feedback Controller; Networks-on-Chip; Dynamic Manycore Systems

I. Introduction

Processing power in terms of number of transistors in a chip is still scaling considerably (about 2.8 times) per technology node generation. However there exists a utilization wall limiting the number of usable transistors on a chip (only 1.4 times) [1]. Therefore, only a limited chip area is able to operate full throttle (voltage and energy) and a considerable silicon area will be unused (i.e., Dark Silicon [2]). In the Dennard Scaling Era [3], it was possible to scale down the threshold and the supply voltage at the same time. Though, exponential rise in leakage or transistor delay prevents simple tuning down the threshold and supply voltage after the year 2005 which is also called Post Dennard Scaling Era. In this era, the excessive leakage leads to thermal issues violating the chip safe working temperature and hence threatening its functionality [4].

Recently, a lot of efforts have been done to minimize the effect of dark silicon such as attempts toward heterogenous computing to maximize the efficiency of domain specific systems. Some techniques also have been studied for near-threshold computing (i.e., Dim Silicon [5]) to reduce the power consumption of both domain/application specific heterogenous and general-purpose homogenous many-core systems. Near-threshold computing (NTC) increases the number of simultaneously active cores, at the expense of much lower operating frequency [5]. In order to implement an efficient NTC-based approach, an intelligent and stable power management mechanism using feedback control is required. Proposing such a mechanism becomes more challenging when we consider current and future manycore systems having tens or hundreds of resources connected together. Such systems often feature an extremely dynamic workload where an unpredictable sequence of different applications enter and leave the system at run-time. In order to handle the featured dynamic nature for the dark silicon era when thermal design power (TDP) should also be considered, a run-time system manager is required to efficiently monitor and manage the total system power consumption especially when an incoming application is mapped onto the system resources or leaves the system.

The previous work on dynamic power management for NoC-based multi-core and many-core systems, including those which have feedback-based approaches, are not dark silicon aware. More precisely, the ultimate objective has been to provide efficient runtime control mechanisms that can exploit the workload characteristics in order to save power. The aim of the previous work is to first accurately capture some important workload characteristics as a feedback, and then adjust voltage/frequency of processing elements, routers, or voltage/frequency islands (VFI) accordingly. As these methods have not been designed for the dark silicon era, they do not consider any safe upper bound on the total system power consumption (i.e. TDP) in runtime, and therefore, they do not include any power feedback in their power management unit.

Feedback-based power management framework for symmetric and asymmetric multi-core architectures are demonstrated in [6] and [7], respectively. However, in these platforms dynamic application mapping is not considered making the systems easier to control. Moreover, even though these architectures are energy efficient, they suffer from the lack of scalability as both platforms are bus-based and consist of only a few cores (i.e., multicore).

To the best of our knowledge, ours is the first work to provide a comprehensive dark silicon aware power management platform for NoC-based manycore systems under limited power budget and running dynamic workloads (i.e. supporting dynamic mapping). This platform benefits from a feedback controller providing dynamic voltage and frequency scaling including both normal and near-threshold operations considering application priorities and network congestion. It also selectively penalizes the cores and the tasks under thermal emergency.

II. Power Management Platform

The proposed dark silicon aware NoC-based system is shown in Figure 1. The target system is a $M \times N$ 2D-Mesh NoC. It is a dynamic framework supporting parallel execution of multiple applications entering and leaving the system at runtime. Each application in the system is represented by a directed graph denoted as a task graph in which each vertex represents one task of the application, while each edge of the task graph stands for a communication between the source

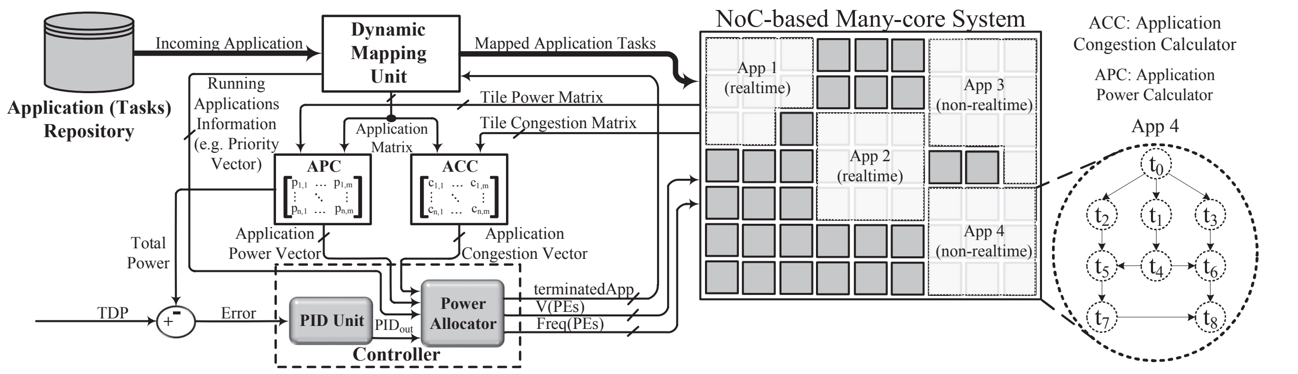

Fig. 1: Overview of the dark silicon aware power management system

task and the destination task. The Dynamic Mapping Unit (DMU) tries to allocate 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 properly manipulate the actuators (e.g., priority vector and application matrix).

We assume that each tile has been equipped with a power sensor to report the current power consumption of the core to the central power manager to form the Tile Power Matrix. It should be noted that many of today's platforms (e.g., Versatile Express Development Platform [8] which includes ARM big.LITTLE chip) have been equipped with sensors to measure frequency, voltage, power, and energy consumption of each core or cluster [7]. Other efforts to measure power consumption read out both current and voltage. For instance, Bakker *et al.* [9] propose a power measurement algorithm for Intel Single-chip Cloud Computer (SCC) [10].

The power consumption of individual routers also vary dynamically due to primarily the uneven traffic distribution in a network. When a set of system resources are 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 traffic. To accurately measure this power dissipation, a power meter is designed within the router micro-architecture [11].

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 the 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, and passes it to the Controller Unit.

It should be noted that if the fine-grained power measurement is not supported by a manycore platform, our power management approach still works fine even in the absence of the APC unit. The only feedback being necessary for our approach is the total chip power consumption. However, the APC unit improves the power allocator's decision by providing extra information regarding the contribution of each application in the total power consumption.

In practice, network traffic distribution is non-uniform and due to interconnection complexity and intrinsic wire delays an ideal topology is not feasible. For each additional core, the network traffic gets more easily congested and the overall throughput per core decreases and hence the total network performance gives a diminishing return due to increased communication distance. In such cases, there is no need for a core to be actively consuming power at high frequencies or voltages. Thus, we find it imperative to take the network performance gap into account when designing dynamic power management many-core systems.

In our platform, each router is equipped with a congestion meter. The congestion 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 congestion level of each router is transferred to the Application Congestion Calculator (ACC). By masking the Application Matrix on the Tile Congestion Matrix provided by the DMU, ACC calculates the average congestion level for each application and sends it to the controller unit.

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

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

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. Several simulations in Matlab and real environment were performed to appropriately adjust the PID gains. Two key factors were considered in our simulations: the system stability and the system robustness against power disturbance.

III. Power Allocation Algorithm

Algorithm 1 shows the proposed power allocation strategy. The algorithm assumes that the system is activated with its highest potential performance. In our system, the Power Allocator (PA) has four main inputs: output of the PID controller (PID_{out}), information of the existing application(s) running on the system (RAI) including application priority vector, application congestion vector from the ACC unit (ACV), and application power vector from the APC unit (APV). The PA has also three outputs: voltage and frequency of the PEs ($V(PEs)$ and $Freq(PEs)$) and the application that should be eliminated from the system, if needed ($terminatedApp$). The $terminatedApp$ is empty in case there is no application to be terminated from the system.

First RAI is passed to the available application(s) set ($availableApp$) as shown in the first line of Algorithm 1.

If the PID_{out} is smaller than zero meaning the total power consumption of the system exceeds TDP, the algorithm will try to find the most efficient manipulation of the actuators considering the output of the PID controller, network congestion, and application priorities (i.e. line 2). If there is no application running on the system (i.e. RAI is empty), the $lowest\_priority$ function will return *null* to $terminatedApp$ and the algorithm ends (i.e. lines 4-7). If there are some applications running on the system, the algorithm will find the set of running application(s) with the lowest priority among $availableApp$ (i.e. line 8). The priority level can be defined as hard real-time, soft real-time, and non real-time or any other application types.

Among the application(s) with the lowest priority ($appSet$), the application having the lowest congestion in its region will be selected as the target application to apply $DVFS$. The $DVFS$ function determines the voltage and frequency of the target application considering task specific data such as priority and current power consumption (i.e. line 10).

In two conditions the $DVFS$ function asserts the $saturated$ variable, 1) when the function cannot throttle the target application any further according to the application type, which occurs when the actuators are saturated for the target application (i.e. voltage and frequency cannot be reduced anymore) and 2) when the application power consumption is not large enough to offer adequate power reduction. When the $saturated$ variable is asserted the application will be removed from the $availableApp$ and the algorithm keeps on finding an alternative application (i.e. lines 11-12). If the target application is not saturated, it will be pushed into the $stackApp$ and the algorithm will be terminated (i.e. lines 14-15).

If all the applications are removed from the $availableApp$, the PA will find an application with the lowest priority to be eliminated from the system (i.e. lines 4-7). When PID_{out} is larger than zero, meaning the thermal emergency is over, the last pushed application will be chosen as the target application to be passed to the $DVFS$ function (i.e. lines 19-20). This ensures the application with the highest priority will get a more appropriate service. As all the stacked applications should be handled, the saturation factor is not checked in this phase. An empty stack means that all the running applications are in the full performance mode. As there are restricted levels of voltage and frequency, the $DVFS$ function ignores marginal deviations of PID_{out} from its previous value for the sake of stability.

Algorithm 1 Power allocation algorithm.

Inputs: PID_{out} : The output of the PID controller; RAI : Existing Application Set; $APRV$: Application Priority Vector; ACV : Application Congestion Vector; APV : Application Power Vector;
Output: $V(PEs)$: Voltage vector of Processing Elements; $Freq(PEs)$: Frequency vector of Processing Elements; $terminatedApp$: Application to be terminated (if needed);

Body:

```
1:  $availableApps \leftarrow RAI$;
2:  if $PID_{out} < 0$ then {//overshooting $TDP$}
3:      while true do
4:          if $availableApps$ is empty then
5:              $terminatedApp \leftarrow lowest\_priority(RAI,\ APRV)$; //Application with lowest priority
6:              break;
7:          end if
8:          $appSet \leftarrow$ lowest_priority ($availableApps$, $APRV$); //Application(s) with lowest priority
9:          $targetApp \leftarrow$ lowest_congestion ($appSet$, $ACV$); //Application with lowest average congestion
10:         ($V(PEs)$, $Freq(PEs)$, $saturated$) $\leftarrow$ $DVFS$ ($targetApp$, $PID_{out}$, $APV$);
11:         if $saturated$ is true then
12:             remove $targetApp$ from $availableApps$;
13:         else
14:             $stackApp \leftarrow$ push ($targetApp$);
15:             break;
16:         end if
17:     end while
18: else
19:     $targetApp \leftarrow$ pop ($stackApp$);
20:     ($V(PEs)$, $Freq(PEs)$) $\leftarrow$ $DVFS$ ($targetApp$, $PID_{out}$);
21: end if
```

IV. Experimental Evaluation

We perform the experiments on our in-house cycle-accurate many-core platform implemented in SystemC. A pruned version of Noxim [12] as its communication architecture was utilised. As PE baseline design, we use Niagara2-like in-order core specifications obtained from McPAT [13]. Physical scaling parameters were extracted from the Lumos framework [14]. We model three application categories, namely, non-realtime (lowest priority), soft realtime, and hard realtime (highest priority). Several sets of non-realtime applications with 4 to 35 tasks are generated using TGG [15] where the communication volumes are randomly distributed.

The probabilities of selecting hard realtime, soft realtime, and non-realtime applications from the application repository are 10%, 20%, and 70%, respectively. An allocation request for the application to be scheduled is sent to the Central Manager of the platform (CM). CM selects the *first node* using SHiC [16] method, and maps the application based on its real-time attributes. In addition to the dynamic mapping unit, our dark silicon aware power management (DSAPM) platform (including the controller, ACC, and APC) is also implemented (i.e. soft coded) as a part of the CM.

To demonstrate the efficiency of our power management platform for future technology generations, we run our method for a 12×12 NoC-based many-core system in 16nm technology. We use 29 voltage-frequency levels including near-threshold computing for the DVFS purpose. We define different minimum voltage-frequency levels depending on the application type. For example, we use all the 29 levels to scale voltages and frequencies of PEs running non-realtime applications. In this work, we do not scale voltage and frequency of the interconnection network to avoid the network performance gap.

As explained before, our DSAPM platform is based on the concept of dim silicon. To be able to conduct a fair comparison with a scenario where we have the dark silicon phenomenon, we proposed the PAM strategy. PAM was proposed as to the best of our knowledge there is no related work on dark silicon aware NoC-based system management for comparison in the literature. PAM does not utilize DVFS, however, it is also feedback control based meaning that it takes into account the current power consumption of the system. In the PAM

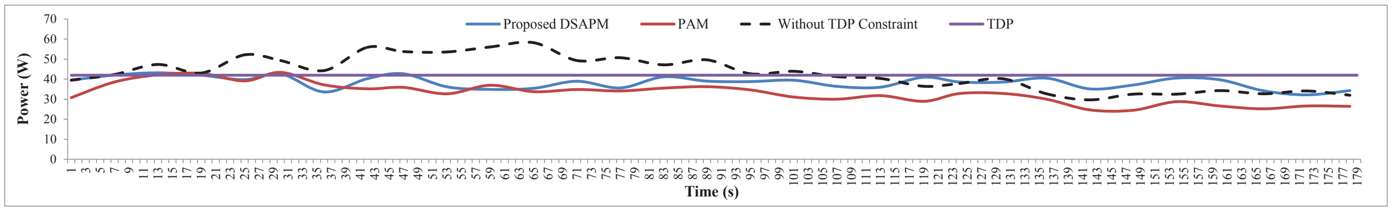

Fig. 2: The power consumption of the system using DSAPM, PAM, and without TDP constraint policies under TDP constraint

algorithm, based on the power estimation (i.e. prediction) of the new application and the current total power consumption of the system, the mapper decides whether to map the new incoming application or wait until an application leaves the system. More precisely, the total system power consumption is predicted before mapping the current application. If the prediction result indicates that "the TDP won't be exceeded", the application will be mapped, otherwise, it will be delayed.

Figure 2 plots the power consumption of the system for the experiment. In this environment, a group of applications are running that are selected randomly from the three mentioned application categories (i.e. hard realtime, soft realtime, and non-realtime). The dashed black curve in the graph shows the system power consumption without any power constraints. The purple curve illustrates the maximum power budget for the system (i.e. TDP). The TDP value is calculated based on the chip power density, reflecting the maximum power allowed in a unit area. As can be observed, the power consumption in some cases exceeds the TDP when no management policy is used. The blue curve shows the system power consumption while applying the proposed DSAPM policy. As can be seen, in most cases the power is kept below the TDP. It can be also observed that the control system is always stable even for large power consumption fluctuations which occur for example when a large application enters the system. In cases when the power consumption exceeds the TDP, the controller rapidly reduces the power consumption by a proper voltage and frequency scaling. As the chip temperature is the main constraint here, a short deviation from the TDP does not affect the overall chip temperature [17].

To assess the efficiency of our DSAPM platform, we compare the normalized throughput for the set of applications under DSAPM and PAM policies. The results reveal that our proposed method can significantly improve the overall system throughput by 43% for 16nm technology. System utilization is another important factor to be analysed for the DSAPM and PAM policies. The DSAPM policy increases the system utilization by 24% compared to the PAM policy for this technology. The main reason for the improvements in terms of system throughput and utilization is that in contrast with the PAM policy which follows the dark silicon phenomenon, the proposed DSAPM policy attacks the dark silicon by utilizing near-threshold computing (i.e., dim silicon).

V. Conclusion and Future Work

In this paper, a feedback based controller system was proposed to protect many-core systems against overshooting the power consumption from a certain power limit (i.e. Thermal Design Power). The target framework is a NoC-based manycore system using dynamic application mapping where applications enter and leave the system at runtime. Comparing the total current system power with the maximum power budget, the controller efficiently changes voltage and frequency of appropriate processing elements, down to near threshold operation. The future work would include improving the controller unit to achieve more efficient actuation manipulation and extending our platform to manage the power consumption of the interconnection network.

Acknowledgment

The authors acknowledge the financial support by the Academy of Finland project. The authors want to thank Mr. Mohammad Fattah for his help to establish the dynamic mapping tool for NoC based systems. We also mention that we used the source code of Lumos from University of Virginia to extract the power data for different technologies.

References

[1] N. Goulding-Hotta et al. The GreenDroid Mobile Application Processor: An Architecture for Silicon's Dark Future. *IEEE Micro*, 31(2):86–95, 2011.

[2] H. Esmaeilzadeh et al. Dark Silicon and the End of Multicore Scaling. *IEEE Micro*, 32(3):122–134, 2012.

[3] R.H. Dennard et al. Design of ion-implanted MOSFET's with very small physical dimensions. *IEEE Journal of Solid-State Circuits*, 9(5):256–268, 1974.

[4] M.B. Taylor. Is dark silicon useful? Harnessing the four horsemen of the coming dark silicon apocalypse. In *Proc. of DAC*, pages 1131–1136, 2012.

[5] Wang L. and K. Skadron. Implications of the Power Wall: Dim Cores and Reconfigurable Logic. *IEEE Micro*, 33(5):40–48, 2013.

[6] K. Ma and X. Wang. PGCapping: Exploiting Power Gating for Power Capping and Core Lifetime Balancing in CMPs. In *Proc. of PACT*, pages 13–22, 2012.

[7] T.S. Muthukaruppan et al. Hierarchical power management for asymmetric multi-core in dark silicon era. In *Proc. of DAC*, pages 1–9, 2013.

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

[9] R. Bakker et al. Emulating Asymmetric MPSoCs on the Intel SCC Many-core Processor. In *Proc. of PDP*, pages 520–527, 2014.

[10] J. Howard et al. A 48-Core IA-32 message-passing processor with DVFS in 45nm CMOS. In *Proc. of ISSCC*, pages 108 –109, 2010.

[11] A.Y. Weldezion et al. A scalable multi-dimensional NoC simulation model for diverse spatio-temporal traffic patterns. In *Proc. of 3D-IC*, pages 1–5, 2013.

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

[13] S. Li et al. McPAT: An Integrated Power, Area, and Timing Modeling Framework for Multicore and Manycore Architectures. In *Proc. of MICRO*, pages 469–480, 2009.

[14] 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.

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

[16] M. Fattah et al. Smart hill climbing for agile dynamic mapping in many-core systems. In *Proc. of DAC*, pages 1–6, 2013.

[17] A. Raghavan et al. Utilizing Dark Silicon to Save Energy with Computational Sprinting. *IEEE Micro*, 33(5):20–28, 2013.