Power-Aware Online Testing of Manycore Systems in the Dark Silicon Era

Mohammad-Hashem Haghbayan[1], Amir-Mohammad Rahmani[1], Mohammad Fattah[1], Pasi Liljeberg[1], Juha Plosila[1], Zainalabedin Navabi[2], and Hannu Tenhunen[1,3]
[1]Department of Information Technology, University of Turku, Turku, Finland
[2]School of Electrical and Computer Engineering, College of Engineering, University of Tehran, Iran
[3]School of ICT, KTH Royal Institute of Technology, Stockholm, Sweden
Email: {mohhag, amirah, mofana, pakrli, juplos}@utu.fi, navabi@cad.ut.ac.ir, hannu@kth.se

***Abstract*—Online defect screening techniques to detect runtime faults are becoming a necessity in current and near future technologies. At the same time, due to aggressive technology scaling into the nanometer regime, power consumption is becoming a significant burden. Most of today's chips employ advanced power management features to monitor the power consumption and apply dynamic power budgeting (i.e., capping) accordingly to prevent over-heating of the chip. Given the notable power dissipation of existing testing methods, one needs to efficiently manage the power budget to cover test process of a many-core system in runtime. In this paper, we propose a power-aware online testing method for many-core systems benefiting from advanced power management capabilities. The proposed power-aware method uses non-intrusive online test scheduling strategy to functionally test the cores in their idle period. In addition, we propose a test-aware utilization-oriented runtime mapping technique that considers the utilization of cores and their test criticality in the mapping process. Our extensive experimental results reveal that the proposed power-aware online testing approach can efficiently utilize temporarily free resources and available power budget for the testing purposes, within less than 1% penalty on system throughput for the 16nm technology.**

***Keywords*—Online Testing; Functional Testing; Dark Silicon; Power Capping; Many-Core Systems**

I. Introduction

Continuous technology scaling has enabled the fabrication of massive parallel computing systems with many cores on the same chip. By reducing the transistors' size, susceptibility to internal defects has increased and designers have faced the effects of growing variability in device geometries and unpredictable changes in functional operations [1]. For instance, the rate of soft-errors at 16nm CMOS technology will increase by hundredfold compared to the failure rate at 180nm technology [2]. The complexity of large many-core system architectures, the high failure probability of modern chips, and the high stress to reduce time-to-market have made the in-field test and verification process increasingly challenging [3]. The limitations of in-field test and verification process as well as the increased probability of intermittent and permanent faults, due to environmental impact and wear-out effects, necessitate efficient online testing methods for many-core systems [4].

Apart from the high failure rate, modern nanometer technologies face another design challenge, power consumption. Power has become a first-class constraint in many-core system design due to thermal and power constraints and the emergence of the dark silicon era [5]. 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. According to predictions, designers will face more than 90% dark area within 5 years if this phenomenon is not properly mitigated [6]. Therefore, a paradigm shift from conventional online testing to power-aware online testing in the dark silicon era, especially for many-core systems, is inevitable [7].

Recently, a lot of efforts have been done to minimize the effect of dark silicon such as attempts toward near-threshold computing (i.e., Dim Silicon [8]) or emergence of asymmetric single-ISA (instruction-set architecture) many-core systems [9]. However, the dynamic nature (i.e., heterogeneity) of workloads makes the amount of dark area on the chip (i.e., total chip utilization) highly changeful. If suitable scenarios is intelligently detected (when there is enough room in power budget), such temporary dark areas can be favorable targets for online testing in order to improve the system reliability [7], [10].

In this paper, we propose a power-aware online testing strategy for many-core systems in the dark silicon era. Our power-aware test strategy benefits from a non-intrusive online test scheduling algorithm using software based self-test (SBST) to test idle cores in the system while respecting the system power budget. The key contributions of this work are as follows:

- Power-aware online test scheduling method with explicit consideration of limited power budgets in many-core systems using runtime application mapping.
- Test-aware runtime mapping algorithm that considers cores with high test criticality in the mapping process.
- Detecting suitable scenarios in many-core systems when online testing methods can be applied in a minimally intrusive (often non-intrusive) way.
- Feedback controller based power management mechanism considering power consumption of cores in the normal operation and test modes.
- Modeling a many-core system using current and future technology nodes down to 16*nm* for different die area budgets to demonstrate the efficiency of the proposed approach in the dark silicon era.

The rest of the paper is organized as follows: Section II covers the definition of our framework and the problem. Section III discusses the related work and motivations of this paper and presents suitable scenarios for online testing process. The proposed power-aware online testing framework is described in Section IV. Experimental results are provided and discussed in Section V. Finally, Section VI concludes the paper and discusses some potential future works.

II. Runtime Mapping (RTM) and Dynamic Power Management (DPM) in Many-core Systems

Before we start to discuss the related work and motivations of this work, we present general architecture of many-core systems and some common features often supported by such systems. As explained before, power budgeting plays a key role in many-core systems, specially in dynamic systems supporting parallel execution of multiple applications arriving and leaving the system at a runtime. In this section, we present how dynamic power management (DPM) and runtime mapping

978-3-9815370-4-8/DATE15/©2015 EDAA

(RTM) are typically performed in such systems. Figure 1 shows a 8×7 NoC-based many-core system with mesh topology supporting DPM and RTM. In such a system, applications request the RTM unit at runtime to be mapped and run on the system. The RTM unit is responsible for allocating proper available cores to the application's tasks. Each application in the system is represented by a directed graph denoted as a task graph. Each vertex in the task graph represents one task of the application, while each edge stands for a communication between the source task and the destination task. Task graph of one application (Gaussian Elimination application [11]) which is extracted using TGG [12] is shown in Figure 1 (i.e., App 4). Packet-based message passing is used to exchange data between cores in the system.

On the other hand, in recent multi-/many-core systems, power has become an important constraint because of limitations in circuit cooling, packaging, and power delivery. One of the today's most important challenges in such systems is to optimize the performance within a given power limit (i.e., power capping) [5]. Recently, there has been many efforts in utilizing on-chip power sensors as feedback for dynamic voltage and frequency scaling (DVFS) and power gating techniques in order to perform power capping. Even though the power consumption of the system in general has a straightforward relationship with the number of cores that are turned-on in the system, dynamic behavior of the system due to running different applications with different requirements and characteristics (e.g., non-realtime, soft realtime, etc.) and DVFS make it difficult to develop an accurate power model for all cases. Therefore, most of the recently published works adopt feedback-based control mechanism to manage the power using per-core DVFS or per-core power gating (PCPG) [13], [14], [9].

It is worth noting that many of today's platforms (e.g., Versatile Express Development Platform [15] which includes ARM big.LITTLE chip and Intel Single-chip Cloud Computer (SCC) [16]) have been equipped with sensors to measure frequency, voltage, power, and energy consumption of each core/cluster or at least the total chip. Figure 1 shows a typical many-core system using feedback controller mechanism to manage the overall power consumption of the system. The controller monitors the total power consumption of the system and compares it with a reference power budget. Whenever, the system power consumption violates the power budget, the controller reduces the power by applying DVFS and PCPG techniques [13], [14], [9], [16]. The free (i.e., unallocated) cores in the system are powered off by the controller (dark cores in Figure 1). The dynamic nature (i.e., heterogeneity) of workloads makes the amount of dark area on the chip (i.e., total chip utilization) highly changeful. For example, when applications running on the system do not demand high operating frequencies (e.g., non-realtime applications), the total chip utilization can reach up to 100% leaving no dark area on the chip. However, if the system is running computationally intensive applications demanding to meet certain deadlines, the total chip utilization can drop and some cores will stay dark for some time (i.e., temporarily) due to the lack of power budget.

III. Related Work, Motivations, and Suitable Scenarios for Online Testing

Online periodic error detection using functional testing techniques, has gained increasing acceptance for multi- and many-core system testing [17]. Several works have been presented in the literature studying the impact of online error detection on the performance of multi- and many-core systems [18], [19], [20], [3], [1]. However, non of these state-of-the-art online error detection techniques consider power budgeting in runtime. More precisely, these techniques are not power-aware. It should be noted that power-aware testing is not to be confused with power-constrained testing. Power-constrained testing is an offline process focusing on power consumption management and minimization during test process. There are several studies on power-constrained and thermal-aware test scheduling techniques for test application time (TAT) minimization in multi- and many-core systems [21], [22]. The vital fact in power-constrained testing is that power consumption of a chip during test process is generally greater than the chip power consumption in normal operation mode, and that can negatively affect the reliability of testing. However, in contrast with power-constrained testing, in power-aware testing the current power consumption feedback of the system is used to manage the online test application in runtime and ensure that the total system power budget is not violated.

Online testing of many-core systems can be generally performed using two different ways: intrusive and non-intrusive [1]. In intrusive testing, the normal operation of the system is interrupted and a group of cores or all the cores are reconfigured to test mode at runtime and then run the test program. In non-intrusive testing, each core executes the test program individually whenever the core is in idle state. As mentioned before, the power consumption needed for the test purpose is considerably higher than the power consumption of the system in the normal operation mode. As the power budget of the system is limited specially in the dark silicon era, we cannot expect to have a fully parallel intrusive testing as the test power consumption can easily exceed the power budget and endanger the chip reliability. Furthermore, as the system is running multiple independent applications with different requirements in parallel, interrupting all or some of the applications might lead to deadline miss for some realtime applications. Due to these facts, in this paper, we focus on non-intrusive testing while honoring power budget.

Our main motivation for this work is that we believe *online testing is gradually reshaping to power-aware online testing in the dark silicon era, especially for many-core systems*. The main ground for this statement is that due to thermal and power constraints, the fraction of transistors that can operate at full frequency is decreasing with each technology generation. This highlights the fact that power budget is an extremely crucial resource in those technologies where the dark silicon phenomenon is more challenging to address (e.g., 22nm or 16nm). In such technologies, a many-core system demands an efficient power-aware online testing method capable of minimizing the usage of power for the online testing purpose. In other words, the online testing method should have the lowest negative impact on the system performance by efficiently using the power budget.

We performed an extensive investigation on many-core systems using DPM and RTM. We analysed frequent scenarios in these systems to find the most appropriate moments when the online testing can have the minimum negative (often zero) impact on the system performance. Figure 2 shows a many-core system running dynamic workloads under six frequent scenarios. In each scenario, the state of the system (i.e., current power consumption of the system, allocated cores to different applications, size of the application requested to be mapped if there is any) in a specific time is shown. In each power consumption graph, the power budget is shown by a dotted line named as *budget*. It should be noted that in order to run a non-intrusive online testing application on the system, there should be available resource(s) (i.e., unallocated core(s)) and available power consumption for the test purpose ensuring the upper bound on power consumption will not be violated. The frequent scenarios are as follows:

- In the first scenario (at the time t_1 shown in Figure 2(a)), eight applications are running on the system and all the cores in the system are busy. Therefore, even though we have enough available power budget for non-intrusive online testing, none of the cores in the

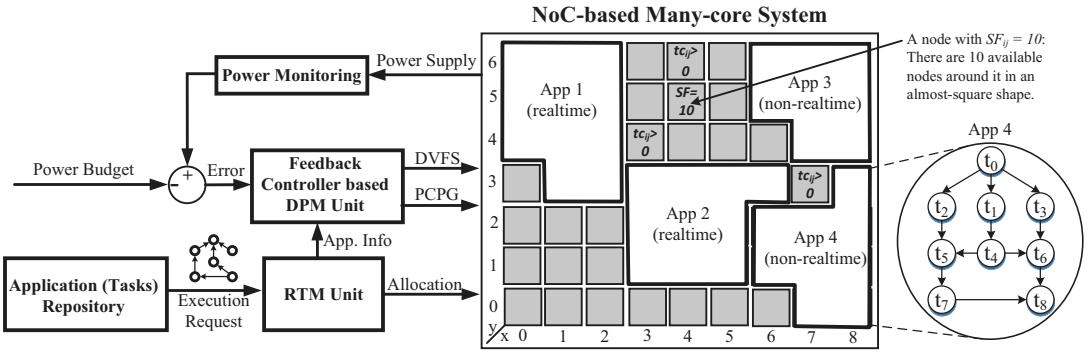

Fig. 1: A NoC-based many-core system with mesh topology supporting dynamic power management and runtime mapping

system can be tested as there is no available core to be tested.

- In the second scenario (at the time t_2 shown in Figure 2(b)), five applications are running on the system and the overall power consumption is too close to the upper bound on power consumption. In this case, although we have available resources for the test purpose, there is not enough available power budget to be dedicated to non-intrusive online testing.
- In the third scenario (at the time t_3 shown in Figure 2(c)), as the system is not heavily loaded and the application repository is empty, there is enough available power and resources on the system to be used for online testing process. As can be observed from the figure, online testing in such scenarios does not degrade the system performance because the available power budget is efficiently used.
- In the fourth scenario (at the time t_4 shown in Figure 2(d)), an application including 8 tasks sends a request to the RTM unit to be mapped on the system and there are more than 8 available cores in the system. However, it has been shown in [23] that due to resource dispersion and high penalty of on-chip communication when non-contiguous mapping is applied, it will be more efficient in terms of system performance and energy if the application waits until one of the applications running on the system leaves and a contiguous area is available for the waiting application. In this case, there is enough available power and resources in the system to be used for the test process. Dispersed cores in such scenarios are the best candidates for being non-intrusively tested without any degradation of the system performance.
- In the fifth scenario (at the time t_5 shown in Figure 2(e)), one application with the size of 9 sends a request to the RTM unit to be mapped on the system. However, despite there is enough power budget to be allocated to the waiting application, there are not enough cores available in the system to accommodate the waiting application. Once again, the available power budget can be used for non-intrusive online testing of the available cores.
- In the sixth scenario (at the time t_6 shown in Figure 2(f)), there are enough available cores to be allocated to the waiting application with the size of 9. However, on one hand, according to the pre-mapping power estimation result, the available power budget is not large enough to support the waiting application, and on the other hand, the DPM unit is not able to reduce the power consumption of the applications currently running on the system due to their high criticality levels. This is also another scenario where the available power budget can be used to test a number of unallocated cores. It should be noted that there are pre-mapping power estimation techniques in the literature such as the light-weight technique presented in [13].

In the following section, we present a test scheduling policy which starts testing at least one unallocated core in the system at a time (e.g., t_i) when there are enough resources and power budget for test. We then gradually increase the number of cores under test depending on the number of available resources and amount of existing power budget. One may ask how it can be assured that the available power budget at the time t_i will stay available during the whole online testing precess of a core. In other words, how it can be guaranteed the power consumption used for test will not result is violation of upper bound on power consumption. According to our extensive experimental results, in such scenarios, the violation of the upper bound on power consumption can be properly managed with a negligible overhead on overall system performance, thanks to the feedback controller used in the DPM module.

A promising yet hidden finding can be also concluded from the presented scenarios is the efficiency of non-intrusive online testing in such systems. The DPM unit often deactivates some cores in the system due to power limitations. However, an opportunistic online test scheduling method can take advantage of such situations and test the dark cores when there is enough room in the power budget.

IV. Power Aware Online Testing Framework

An overview of our power-aware online testing framework for a NoC-based many-core system with mesh topology is presented in Figure 3. As mentioned earlier, the RTM unit is responsible for allocating system cores to the new incoming application and the DPM unit manipulates the actuators (i.e., PCPG using the V-Gate module and per core DVFS) to optimize the system performance within a power budget (i.e., cap). Our proposed online testing framework consists of a Test Scheduling Unit (TSU) which is completely implemented in software and its objective is to manage the online testing process for the available cores on the system in a minimally intrusive (often non-intrusive) way. More precisely, the TSU monitors the system in terms of resource and power budget availability for test and applies online testing at suitable moments (i.e, scenarios 3 to 6 in Figure 2).

The TSU also tracks utilization information for every core in the system. The term *"utilization"* refers to the number of executed instructions in each core from the last time the core has been tested. Using the utilization information, TSU can considerably reduce the test time by avoiding over-testing the under-utilized cores in the system. For example, if a core has been rarely allocated by the RTM Unit to applications, it will not be necessary to be tested in the near future. In our system, each core is equipped with a Utilization Meter (UM) module which is a simple instruction counter (Figure 3). This counter increments whenever an instruction is executed

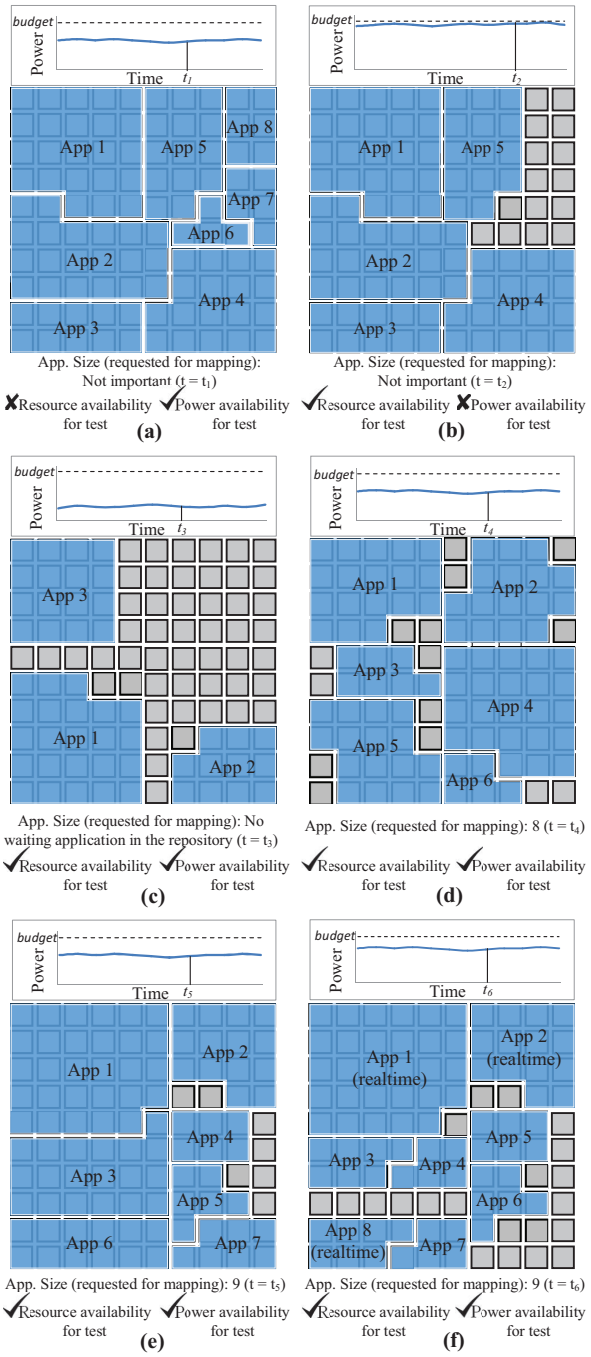

Fig. 2: Frequent scenarios regarding resource and power availability for test

in the core. Using a simple counter to count the number of executed instructions is a common and useful way to measure the utilization of a core. For example, in [17], the same technique is used to count the number of executed instructions and a predefined threshold is determined for the checkpoint interval after which the normal operation is suspended and the test procedure is performed. In order to present the trade-off between performance and reliability, three predefined threshold values (i.e., 10M, 100M and 1B instructions) are used for comparison [17].

The value of the UM module for a core located in the row *i* and column *j* in the system (i.e., α_{ij}) is then mapped

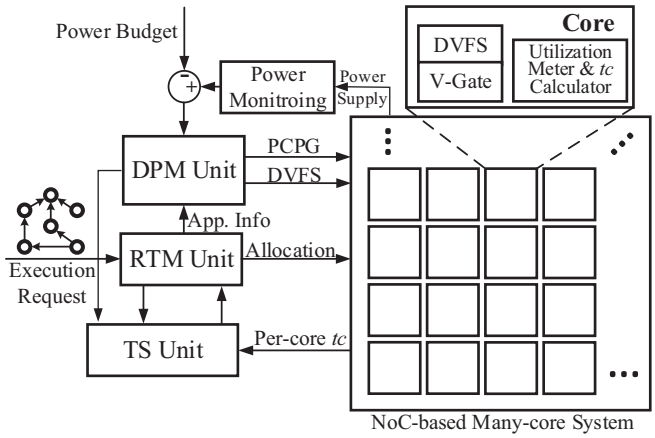

Fig. 3: The system architecture including the online testing framework

to a parameter called *test criticality* (i.e., tc_{ij}) and sent to the TSU. We apply the mapping process to normalize the utilization parameter. tc_{ij} which can have a value (i.e., a real number) between -1 and $+\infty$, is used as a metric for the TSU to determine which core is more critical to be tested at the next suitable time.

We use the following formula to map a utilization parameter to a test criticality for a core located at row *i* and column *j*:

$$tc_{ij} = \frac{\alpha_{ij}}{\delta} - 1 \qquad (1)$$

Where δ is a predefined threshold which is typically determined by the technology and criticality of the system. When the predefined threshold of δ instructions is met, online testing is needed for the corresponding core. Therefore, in the test criticality domain, whenever tc_{ij} exceeds 0.0, it means the corresponding core needs to be tested at the earliest possible moment. tc_{ij} is sent to the TSU at a fixed timing interval (e.g., every 0.1) as long as it holds a positive value. In this way, the test request will be interrupt-based rather than polling-based and redundant packet generation on the system will be avoided.

In order to keep track of the test criticality of each core, TSU forms a test criticality matrix (TCM) and updates the respective element of the TCM whenever it receives a *tc* from a core. Therefore, for a M×N NoC-based system, TCM can be defined as a M×N matrix whose entry $(i, j) \in [M] \times [N]$ corresponds to test criticality value (tc_{ij}) of the core located at row *i* and column *j* in the system. As mentioned before, the test criticality value of each core is a number between -1 to $+\infty$ indicating criticality to be tested according to utilization. As long as the number is in the range of [-1, 0), the corresponding core does not need the testing process. The *tc* is reset to '-1' when the testing process of the corresponding core is initiated.

It should be noted that, as our test methodology is proposed for DPM-based many-core systems, it can be straightforwardly extended to support the online testing for faults appearing at different voltage/frequency levels. However, this extension has been planned for future work.

We aim to maintain system performance, while avoiding cores to reach high values of test criticality. In the following, we explain our changes applied to RTM unit in order to consider test criticality of cores while allocating them to applications. Moreover, we describe our test scheduling algorithm that select appropriate cores for testing. As mentioned, the main goal of these two tightly connected units is maintaining system performance without leaving cores untested.

A. The Proposed Test-aware Mapping Method

As mentioned above, it is desired to test a core once its tc value becomes greater than zero. However, as discussed in Section III, such might not be always possible due to resource and/or power unavailability. As such, we prevent RTM unit from allocating cores with $tc > 0.0$, to be later scheduled by our test scheduling unit in an appropriate time.

In order to achieve high run-time performance, it is desired to map each application onto a contiguous set of available cores [24][25]. That is to place communicating tasks on neighboring nodes, which leads to less power consumption and improved execution time of applications. Nevertheless, a planned contiguous allocation might get dispersed as we prohibit allocation of cores with $tc > 0.0$. For instance, given the system configuration of Fig. 1, assume that an application with 10 tasks is requested for execution. As the square factor (SF_{ij})[1] of the denoted node is 10, it will be selected to map the application onto its surrounding nodes. Assuming 2 of these cores having $tc_{ij} > 0.0$, however, RTM units has to allocate available nodes from south-east region of the system which leads to high dispersion.

In order to prevent such performance crippling dispersions, we recalculate the SF value by subtracting the number of cores with $tc > 0.0$ from the original SF value. As a result, the new SF value shows the number of available cores around a given core that do not have high criticality for testing. For instance, the new SF value of the core in Fig. 1 will be $newSF_{ij} = 10 - 2 = 8$. This means that the core will not be selected for mapping an application with 10 tasks, but with 8 tasks instead.

B. The Proposed Test Scheduling Method

As mentioned before, cores with $tc_{ij} > 0.0$ should be scheduled for testing in the system. However, we might not have enough power budget to test all of them. Hence, we need to rank cores based on their tc values and test those of top ranks as long as we have power budget for testing.

On the other hand, testing arbitrary cores *only* based on their tc values can lead to high dispersion of running applications. That is, we prefer to clean up regions of available cores (e.g., the dark region in Fig. 2 (c)) from cores with $tc_{ij} > 0.0$. In other words, when having two cores with similar tc values, we prefer to test the one which is more likely to be used in near future allocations. As mentioned before, SF_{ij} estimates the number of available contiguous cores around a given core; i.e. the larger the SF_{ij} value is, the more likely to be selected for future allocation. As such, we rank cores for testing, tr_{ij}, using the following equation:

$$tr_{ij} = tc_{ij} + \frac{\sqrt{SF_{ij}}}{total\ number\ of\ cores} \quad (2)$$

In addition to tc value, as can be seen in Equation 2, the normalized SF value of a core contributes to its rank. For instance, in case of equal tc_{ij} values in Fig. 1, we rank cores $(3, 4)$ and $(4, 6)$ higher than the core $(7, 3)$. Our pseudo-code in selecting a core for test scheduling is shown in Algorithm 1. Although, we might have enough power at the moment to test more cores, this might change in near future. New applications might enter the system, and/or the behavior of currently running applications might change and demand more power. Thus, as can be seen (line 3), we limit the maximum number of cores that can be simultaneously scheduled for testing ($\tau_{\#Test}$). As a result, we further control the negative impact of testing on system performance.

V. Experimental results

To demonstrate the efficiency of our power-aware online testing methodology on many-core systems, we run our method

[1]Square factor is a metric, introduced in literature [24], to estimate the number of almost-contiguous available nodes around a given node.

Algorithm 1 Selecting Cores for Test Scheduling

In predefined intervals:

1: **if** there is available resource and power for test **then** {// One of the suitable scenarios shown in Fig. 2}
2: Sort available cores based on their tr_{ij} values;
3: **while** there is enough power budget for test **and** # of (cores under test) $< \tau_{\#Test}$ **do**
4: Schedule the first core in the ranked list for testing;
5: **end while**
6: **end if**

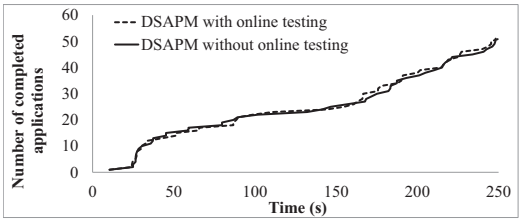

Fig. 4: The number of completed applications vs. time

for a 12×12 NoC-based many-core system in 16nm technology. We use the runtime mapping algorithm presented in [23]. For processing element (PE) baseline design, we use Niagara2-like in-order core specifications obtained from McPAT [26]. *NetlistGen.exe* is used for generating netlists of the synthesized cores and fault simulation with PLI library in HDL environment [27]. Physical scaling parameters were extracted from the Lumos framework [28]. Lumos is a framework to analytically quantify the power-performance characteristics of many-core systems especially in near-threshold operation. The physical scaling parameters have been calibrated by circuit simulations with a modified Predictive Technology Model [29]. Moreover, we have imported other models and specifications such as power model, voltage-frequency scaling model, thermal design power (TDP) calculation, and near threshold computing parameters from the Lumos framework. Our many-core platform was enforced to support runtime application mapping by implementing a central manager (CM) residing in the node $n_{0,0}$. We set the maximum number of cores that can be simultaneously under test to 4.

We select the dark silicon aware power management (DSAPM) technique presented in [13]. In this power management strategy, a PID (Proportional Integral Derivative) controller is used for dynamic power management that considers an upper bound on power consumption (i.e., TDP). To avoid violation of the TDP constraint, the controller provides fine-grained DVFS including near-threshold operation and PCPG. Functional testing is used to test PEs. The test result of each step of the software is compressed in a local memory in the central manager (i.e. $n_{0,0}$). To prepare the test program, we first generate deterministic test patterns from the netlist of HDL implementation of Niagara2-like cores using the technique proposed in [30]. Then, we develop test macros based on the generated deterministic test patterns. The overall coverage for the cores' datapath and controller are 79% and 63%, respectively. The test application duration is 9000 cycles for each core. The models presented in [28] are also used to calculate dynamic and static power consumption of the test process. We run the simulations for different utilization threshold values (δ=10M, 100M, and 1B instructions).

Table I shows the throughput penalty of our proposed testing method when δ is set to 10M, 100M, and 1B instructions. As can be seen, our proposed online testing method

TABLE I: Throughput penalty for different δ values

δ value	10M	100M	1B
Throughput penalty	1.3%	0.31%	0.14%

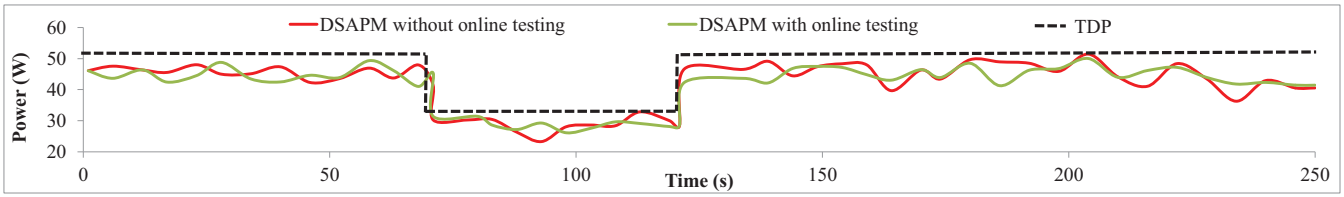

Fig. 5: The power consumption of the system with and without online testing approach

has a negligible throughput penalty compared with many-core systems without any online testing method. It should be noted that the throughput penalty of our online testing method is considerably lower than in the existing online testing methods such as [17], [18]. The reason of such improvement is that our method takes advantage of non-intrusive testing of the cores which are temporarily located in the dark area. Figure 4 shows the system throughput over time (250 seconds) for different experiment setups in more details when δ is set to 10M (i.e., the worst case scenario). It can be seen from the figure that applications are being completed and leaving the system in almost the same trend for both with online testing and without online testing scenarios.

Figure 5 shows the power consumption of the system while running a group of random applications on the implemented framework with and without our online testing technique when δ is set to 10M. We run the system for 250 seconds while applications enter and leave the system in runtime. As it can be observed from the power curves, the total power consumption does not violate the TDP for both approaches. This shows that even though, a dedicated power budget was *not* allocated to the test purpose, the DPM unit has efficiently honored the TDP bound even when the TDP is changed at runtime. The power curves show that the small throughput penalties are experienced when the system is frequently busy and the total chip power consumption is most of the time close to the upper bound.

VI. Conclusion and Future Work

Power consumption has become a first-class constraint in many-core system design due to thermal and power constraints and the emergence of the dark silicon era. Modern many-core systems often support a power capping mechanism which utilizes feedback controller to protect the system against overshooting the power consumption from a certain power limit. In this paper, we propose that power limitations should be considered in online testing process as well. More precisely, we claim that a paradigm shift from conventional online testing to power-aware online testing in the dark silicon era, especially for many-core systems, is inevitable. To this end, we identified suitable scenarios in many-core systems when online testing methods can be applied in a minimally intrusive way. In the future, our technique will be extended to support online testing for faults appearing at different voltage/frequency levels.

Acknowledgment

The authors acknowledge the financial support by the Academy of Finland project entitled "MANAGE: Data Management of 3D Systems for the Dark Silicon Age", University of Turku graduate school (UTUGS), EU COST Actions IC1103: Manufacturable and Dependable Multicore Architectures at Nanoscale (MEDIAN) and IC1202: Timing Analysis on Code-Level (TACLe). 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] M. Kaliorakis et al. Accelerated online error detection in many-core microprocessor architectures. In *VTS*, 2014.

[2] S. Borkar. Designing Reliable Systems from Unreliable Components: The Challenges of Transistor Variability and Degradation. *IEEE Micro*, 25(6), 2005.

[3] M. Kaliorakis et al. Online error detection in multiprocessor chips: A test scheduling study. In *IOLTS*, 2013.

[4] D. Gizopoulos et al. Architectures for Online Error Detection and Recovery in Multicore Processors. In *DATE*, 2011.

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

[6] M.B. Taylor. Is dark silicon useful? Harnessing the four horsemen of the coming dark silicon apocalypse. In *DAC*, 2012.

[7] M.H. Haghbayan et al. Energy-Efficient Concurrent Testing Approach for Many-Core Systems in the Dark Silicon Age. In *DFT*, 2014.

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

[9] T.S. Muthukaruppan et al. Hierarchical Power Management for Asymmetric Multi-core in Dark Silicon Era. In *DAC*, 2013.

[10] M.H. Haghbayan et al. Online Testing of Many-Core Systems in the Dark Silicon Era. In *DDECS*, 2014.

[11] A.K. Amoura et al. Scheduling Algorithms for Parallel Gaussian Elimination with Communication Costs. *IEEE Trans. on Parallel and Distributed Systems*, 1998.

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

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

[14] Kai Ma et al. PGCapping: Exploiting Power Gating for Power Capping and Core Lifetime Balancing in CMPs. In *PACT*, 2012.

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

[16] J. Howard et al. A 48-Core IA-32 message-passing processor with DVFS in 45nm CMOS. In *ISSCC*, 2010.

[17] K. Constantinides et al. Software-Based Online Detection of Hardware Defects Mechanisms, Architectural Support, and Evaluation. In *MICRO*, 2007.

[18] S. Nomura et al. Sampling + DMR: Practical and low-overhead permanent fault detection. In *ISCA*, 2011.

[19] A Apostolakis et al. Software-based self-testing of symmetric shared-memory multiprocessors. *IEEE Trans. on Computers*, 2009.

[20] N. Foutris et al. Mt-sbst: Self-test optimization in multithreaded multicore architectures. In *ITC*, 2010.

[21] Yu Xia et al. Using a distributed rectangle bin-packing approach for core-based soc test scheduling with power constraints. In *ICCAD*, 2003.

[22] T. Yoneda et al. Power-Constrained Test Scheduling for Multi-Clock Domain SoCs. In *DATE*, 2006.

[23] M. Fattah et al. Adjustable Contiguity of Run-Time Task Allocation in Networked Many-Core Systems. In *ASP-DAC*, 2014.

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

[25] C.-L. Chou et al. Energy- and Performance-Aware Incremental Mapping for Networks on Chip With Multiple Voltage Levels. *Trans. Comp.-Aided Des. Integ. Cir. Sys.*, 27(10), 2008.

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

[27] Z. Navabi. *Digital System Test and Testable Design: Using HDL Models and Architectures*. Springer, 2010.

[28] L. Wang et al. Dark vs. Dim Silicon and Near-Threshold Computing Extended Results. In *University of Virginia Department of Computer Science Technical Report TR-2013-01*, 2012.

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

[30] M. H. Haghbayan et al. Test Pattern Selection and Compaction for Sequential Circuits in an HDL Environment. In *ATS*, 2010.