A Power-Aware Approach for Online Test Scheduling in Many-Core Architectures

Mohammad-Hashem Haghbayan, *Student Member, IEEE*, Amir-Mohammad Rahmani, *Member, IEEE*, Antonio Miele, *Member, IEEE*, Mohammad Fattah, *Student Member, IEEE*, Juha Plosila, *Member, IEEE*, Pasi Liljeberg, *Member, IEEE*, and Hannu Tenhunen, *Member, IEEE*

Abstract—Aggressive technology scaling triggers novel challenges to the design of multi-/many-core systems, such as limited power budget and increased reliability issues. Today's many-core systems employ dynamic power management and runtime mapping strategies trying to offer optimal performance while fulfilling power constraints. On the other hand, due to the reliability challenges, online testing techniques are becoming a necessity in current and near future technologies. However, state-of-the-art techniques are not aware of the other power/performance requirements. This paper proposes a power-aware non-intrusive online testing approach for many-core systems. The approach schedules software based self-test routines on the various cores during their idle periods, while honoring the power budget and limiting delays in the workload execution. A test criticality metric, based on a device aging model, is used to select cores to be tested at a time. Moreover, power and reliability issues related to the testing at different voltage and frequency levels are also handled. Extensive experimental results reveal that the proposed approach can i) efficiently test the cores within the available power budget causing a negligible performance penalty, ii) adapt the test frequency to the current cores' aging status, and iii) cover available voltage and frequency levels during the testing.

Index Terms—Online testing, functional testing, dark silicon, power capping, many-core systems, aging, lifetime reliability

1 Introduction

Recent advances in technology, fabrication processes and computing architecture have enabled the integration of some hundreds of cores in the same chip, thus leading to the design of many-core architecture. However, such aggressive trend presents a relevant drawback related to the increasing power consumption issues. In fact, power has become a first-class constraint in many-core system design due to thermal issues and the emergence of the dark silicon era [1]. 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 percent dark area within five years if this phenomenon is not properly mitigated [2].

On the other hand, the technology scaling is causing also a reliability challenge. Indeed, the transistor shrinking is leading to devices that are more susceptible to internal defects, variability phenomena, and wear-out and aging process [3]. The complexity of large many-core 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 [4]. In such a scenario, in order to detect the occurrence of permanent failures, especially due to the aging effects, online testing represents a viable solution to cope with the limitations of in-field test and verification process. Moreover, since such architectures are usually not provided with specific Built-in-Self-Testing (BIST) circuitry (e.g., [5]), software based self-test (SBST) has been selected as a promising solution (e.g., [3], [6]). However, two different issues have to be taken into account: 1) the considerable workload to be executed which requires strict performance levels, thus leading to the necessity of a transparent *test scheduling*, and 2) the necessity to consider the power consumption in testing activities, thus leading to a *power-aware testing approach* [6], [7]. As a conclusion, we claim that in the scenario of the dark silicon era there is the quest for a *power-aware test scheduling approach* to detect at runtime permanent faults occurring in many-core architectures while not degrading the overall system performance.

An interesting aspect of modern many-core systems is that they are generally characterized by a highly variable and heterogeneous workload which makes the amount of dark area on the chip (i.e., total chip utilization) highly valuable. Furthermore, due to the emergence of dim silicon [8] as a way to minimize dark areas and increase the number of active cores, the system might reach up to 100 percent utilization of its resources (if the majority of running application are not performance-demanding) by making use of advanced power management features such as Dynamic

- *M.-H. Haghbayan, M. Fattah, J. Plosila, and P. Liljeberg are with the Embedded Computer and Electronic Systems Laboratory, Department of Information Technology, University of Turku, Turku 20520, Finland. E-mail: {mohhag, mofana, juplos, pakrli}@utu.fi.*
- *A.-M. Rahmani and H. Tenhunenis are with the Embedded Computer and Electronic Systems Laboratory, Department of Information Technology, University of Turku, Turku 20520, Finland, and with the Department of Industrial and Medical Electronics, Royal Institute of Technology (KTH), Kista 16440, Sweden. E-mail: amirah@utu.fi, hannu@kth.se.*
- *A. Miele is with the Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Milano 20133, Italy. E-mail: antonio.miele@polimi.it.*

Manuscript received 1 Feb. 2015; revised 20 July 2015; accepted 27 Aug. 2015. Date of publication 22 Sept. 2015; date of current version 10 Feb. 2016. Recommended for acceptance by C. Bolchini, S. Kundu, and S. Pontarelli.

© 2016 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. To access the final edited and published work see http://dx.doi.org/10.1109%2FTC.2015.2481411

Voltage and Frequency Scaling (DVFS) [9]. This makes the behavior of such systems to be highly related to the characteristics of the workload where at different moments it is possible to have wide dark areas with small resource utilization, due to the fact that some other group of cores are set on a high voltage-frequency (VF) level thus reserving the overall power budget, or small dark areas with large resource utilization by globally setting a very low voltage-frequency level. Therefore, if suitable scenarios are 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 [10], [11].

Recent studies show also that the reliability of systems equipped with DVFS knobs should be ensured at multiple voltage-frequency settings as faults are manifested in different ways in the various configurations [12]. Therefore, the test scheduling method needs to be adapted to test the cores in different voltage-frequency levels.

Finally, there are several studies (e.g., [13], [14]) that propose statistical models to estimate the aging trend and the lifetime reliability during the system lifetime. Such models are used, for instance, in Dynamic Reliability Management (DRM) approaches [15] to prolong the overall lifetime of the system. The basic idea is to drive the runtime decisions on architectural parameters tuning (such as DVFS) and workload distribution to avoid an excessive stress of the cores and consequently slowing down the aging process. In fact, the estimated aging trend is also an effective metric to tune at runtime how frequent an online testing routine should be executed to avoid overtesting of various cores.

Given these motivations, in this paper, we propose a power-aware online testing strategy for many-core systems in the dark silicon era. Our dark silicon aware test strategy benefits from a non-intrusive online test scheduling algorithm using SBST techniques to test idle cores in the system while respecting the system's power budget. We define a reliability metric called test criticality factor that represents the priority to test the cores in many-core systems; moreover, we compute such metric also on the basis of the cores' aging status. Further, we propose an efficient test scheduling algorithm that chooses near optimal possible voltage-frequency settings considering the system's power budget.

The main contributions of this paper, which is major extension of our recent work published in [6], are:

- An enhanced power-aware online test scheduling method with explicit consideration of limited power budgets in many-core systems using runtime application mapping.
- An efficient test scheduling method for testing the cores in different voltage-frequency settings.
- Extending the test criticality metric used in [6] with a lifetime reliability estimation to achieve accurate fault occurrence probability as a priority to test the cores, and to balance the regularity of the test according to the aging status of the cores.
- Modeling and evaluation of a many-core system using various current and future technology nodes (32, 22, and 16 nm) for different die area budgets to demonstrate the efficiency of the proposed approach in the dark silicon era.

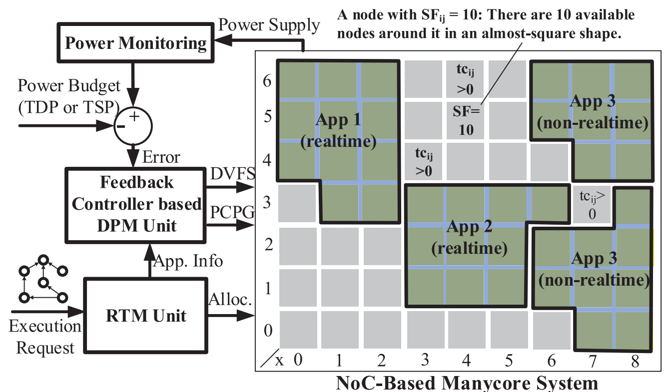


Fig. 1. A NoC-based many-core system with mesh topology supporting DPM and RTM.

The rest of the paper is organized as follows: Section 2 presents the background on the considered system architecture. Section 3 discusses the related work and motivations of this paper, while Section 4 describes suitable scenarios for online testing by means of a running example. The proposed dark silicon aware online testing framework is presented in Section 5, while Section 6 proposes an enhancement to handle testing at different voltage-frequency levels. Section 7 discusses the experimental results demonstrating the efficiency of the proposed approach, and, Section 8 draws the conclusions.

2 Background on the Adopted Manycore Architecture

The general model of the architectural platform targeted in this paper consists of a 2D-mesh-based grid of processing cores whose activity is managed by a controller running on the host machine. Each node within the architecture is an independent processing core, and different cores communicate by exchanging data through a packet-based message communication protocol. The system executes applications that are generally organized in a set of interdependent tasks. Therefore, each application is modeled in terms of a task graph, where nodes represent the various activities, and directed edges represent control and data dependencies between pairs of tasks. In order to execute an application, each task is dispatched (or *mapped*) on a specific core that will execute it; in other words, we may also say that a core has to be *allocated* for the execution of the task. Finally, to satisfy the specified dependencies, the task can be executed only when all input data is available on the core. Fig. 1 shows an example of 9×7 architecture running four different applications.

The working scenario where such systems are commonly employed is generally characterized by a highly variable workload. Indeed, applications arrive with an unknown trend (depending on the requests of the various users), and may have different characteristics (in terms of the size and shape of the task graph), and requirements (for instance on the performance or on the amount of the processed data). As an example in Fig. 1, applications are annotated with realtime/non-realtime requirements. In order to deal with this evolving scenario, runtime mapping (RTM) is generally performed by means of a specific software unit running on the host machine. The RTM unit is in charge of receiving the

request of execution arriving from the users, and dispatching the application tasks on the available cores to satisfy the specified performance requirements.

On the other side, modern multi-/many-core systems present non negligible power issues due to physical limits in circuit cooling, packaging, and power delivery. In particular, during its activity the system is subject to a given power budget, known as *power capping*, which may prevent the system to use all available cores at the same time. For instance, Fig. 1 shows that some cores are switched off (i.e., dark) shown by gray color. As shown in Fig. 1, such budget may be defined at design time (i.e., Thermal Design Power, TDP, [1]), or dynamically managed at run-time with another feedback loop, such as Thermal Safe Power (TSP [16], [17]). Moreover, during the system activity, it may happen that the set of applications to be executed does not demand high operating frequencies, and therefore it is possible to use the total chip utilization at a low voltage-frequency level, leaving no dark area on the chip. However, when computationally intensive applications are demanding to meet certain deadlines for execution, the active cores need to work at a high voltage-frequency level and, therefore, may consume larger power; as a consequence, the other cores will have to stay dark for some time (i.e., temporarily) due to the lack of power budget. This example shows that the dynamic and heterogeneous nature of the workload makes the amount of dark area on the chip highly variable, and, therefore, shows the necessity of a dynamic power management (DPM).

The power budget is generally managed by exploiting on-chip power sensors as feedback for DVFS and power gating techniques. Fig. 1 shows that a DPM Unit is placed side by side to the RTM Unit to optimize at runtime applications requirements (in particular, on the performance) while fulfilling the power limit. As proposed in the literature [18], [19], [20] and shown in the figure, the DPM Unit features a feedback-based control mechanism to manage the power using per-core DVFS or per-core power gating (PCPG). The unit monitors the total power consumption of the system; whenever, the system power consumption violates the power budget, the controller reduces the power by acting on the two available knobs [18], [19], [21]. To support this solution, let us mention that various modern devices (such as the Intel Single-chip Cloud Computer (SCC) [21] and the Haswell Xeon 5 [22]) have been equipped with sensors for measuring current intensity, power and energy consumption, and knobs for DVFS and power gating of each core/cluster or at least the total chip.

3 Related Work and Motivations

Online periodic error detection using functional testing techniques has received an increasing attention in multi-/many-core system testing [23]. Indeed, since such architectures are generally employed for the acceleration of computational intensive applications, the relevant aspect is the minimization of the overhead of online error detection activities on the overall system performance [24]. However, none of the state-of-the-art online error detection techniques consider power budgeting in runtime. *They are actually 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 off-line process focusing on power consumption 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-/many-core systems [25]. Various methods, such as 2D/3D rectangular packing models [26] or test scheduling graphs [27], are used to achieve optimal TAT. The vital fact in power-constrained testing is that power consumption of the single core during test process is generally greater than in normal operation mode [28]. Indeed, core testing process typically results in considerably higher circuit activity rate, compared to normal mode operation, hence, causing above normal power dissipation. This high power consumption generates extra temperature that can negatively affect the reliability of testing [28]. 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 at runtime and to ensure that the total system power budget is not violated.

Recently, there have been some contributions in testing multi-core and many-core systems with advanced static and dynamic power management capabilities such as PCPG and DVFS. These efforts can be divided into two main categories 1) techniques which test the system including power switches to ensure that it properly works at different voltage-frequency settings [29] and 2) contributions in which associated hardware for power management is made use of to control power dissipation during the test process while reducing the TAT [30]. Most of the techniques in these two categories have been proposed for the offline testing process of DPM-based systems. Even though we can find a limited number of online testing methods in these two categories, they do not yet consider any power feedback from the system at runtime and instead use a pre-defined dedicated power budget for testing. An example of power-aware optimization of the SBST has been presented in [31], where the authors propose an optimal approach to test the L1 cache in microprocessors considering power profile. However, this work is not either fully power-aware as the authors use a pre-defined power model of the microprocessors for different applications, which lacks an online power feedback from the system.

There are two different ways to perform online testing in many-core systems: intrusive and non-intrusive [3]. In intrusive testing, the normal system operation is interrupted and the cores, or a subset of them, 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, it is not possible to perform a fully parallel intrusive testing as the power consumption can easily exceed the available budget and endanger the chip reliability. Furthermore, as the system is concurrently running multiple independent applications with different requirements, interrupting all or some of these might lead to deadline miss for some applications. Due to these facts, in this paper, we focus on non-intrusive testing while honoring power budget.

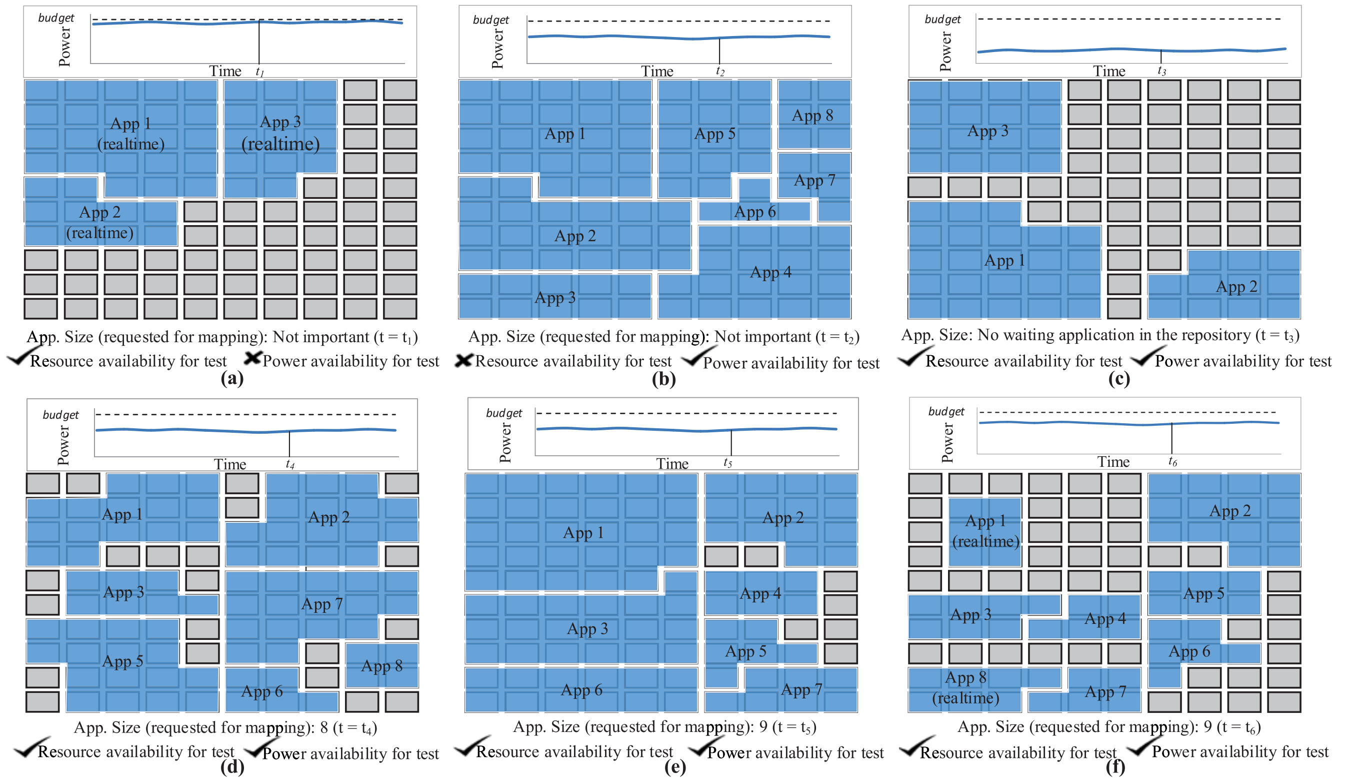


Fig. 2. Frequent scenarios regarding resource and power availability for test.

Due to these motivations, we claim that *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., 22 or 16 nm). 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.

4 Online Testing Suitable Scenarios

We performed an extensive investigation on a many-core system using DPM and RTM to identify the most appropriate moments when the online testing can be executed with a minimum negative (often zero) impact on the overall system performance. The outcome of this analysis has been a taxonomy of the frequent situations of the many-core system running dynamic workloads presented in Fig. 2. In each scenario, the state of the system (i.e., current power consumption of the system, allocated cores to different applications and idle cores, size of the application requested to be mapped if there is any) in a specific time is shown. Moreover, the related power consumption graph is shown, also specifying the given power budget by means of a dotted line. In each situation, in order to run a non-intrusive online testing application on the system, the candidate core to be tested should be unallocated (i.e., idle/dark), and the power consumption required for the test purpose should not cause a violation of the available power budget. These scenarios are commented in the following paragraphs.

Scenario (a). At time t_1, the system is executing three applications with strict performance requirements. In this scenario, the active cores are set to a high frequency-voltage level thus leading the overall power consumption to be too close to the available budget and forcing the other cores to be dark. In this case, although there are idle cores that can be tested without affecting the nominal activities of the system, there is not enough available power budget to be dedicated to online testing.

Scenario (b). At time t_2 all the cores have been allocated for the execution of the eight running applications without performance requirements. In this case it is possible to set the dim area to the 100 percent of the available resources. Consequently, even though there is an available amount of power budget for the testing activity, it cannot be executed in a non intrusive way since all cores are busy. This scenario can happen when the running applications are not performance-demanding.

Scenario (c). At time t_3, the system is almost unloaded, since few applications are running and the power consumption is quite low. This is the best scenario, since the online testing activities can be executed in a transparent way. In fact, we can allocate idle cores to the online testing application; at the same time the system performance is not degraded by the testing activities because the available power budget is used in an efficient way.

Scenario (d). At time t_4, the RTM Unit receives the request to execute an application composed of eight tasks. However, the dynamic mapping strategy [32] decides that it is

not convenient to immediately execute the applications; in fact, even if there is an available power budget, the system presents a high dispersion of the idle cores that would imply a inefficient choice in terms of performance and energy consumption due to the communication costs. Therefore, since the RTM Unit will wait for a contiguous region composed of at least eight cores, the system can employ the available power budget to run the test process on the idle cores. Dispersed cores in such scenarios are the best candidates for being non-intrusively tested without any degradation of the system performance.

Scenario (e). At the time t_5, the RTM Unit receives the request to execute an application composed of nine tasks. In this scenario, even if the available power budget suffices for the execution of the application, there are not enough cores available in the system to accommodate the arrived application. Once again, the available power budget can be exploited for non-intrusive online testing of the available cores.

Scenario (f). At time t_6, the RTM Unit receives the request to execute an application composed of nine tasks, which can be potentially executed in that moment due to the availability of more than nine idle cores. Unfortunately, according to the pre-mapping power estimation result (performed with specific techniques, such as [18]), the available power budget is not large enough to support the arrived application, and on the other hand, the DPM unit is not able to reduce the power consumption of the other applications currently running on the system due to their performance requirements. This is also another situation in which the available power budget can be used to test a number of unallocated cores.

There are two other additional consideration that can be drawn. When it is possible to execute online testing (as in scenarios from (c) to (f)), it is also necessary to choose the candidate cores to be tested; in fact, the concurrent test of all the idle cores generally overcomes the available power budget. Moreover, as discussed above, testing activities have to be performed with several voltage-frequency levels in order to ensure the correct behavior of the system with the various settings [12], [29]. Therefore, the testing application needs to be executed with various voltage-frequency levels, each one having a different power consumption/execution time trade-off. As a result, in the scenarios (d), (e), (f), it is also possible to exploit such trade-off to accommodate the test application with a low voltage-frequency setting on several cores at the same time, or, when it is required, to run a single test with a high voltage-frequency setting on a specific core.

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 since the available power budget does not suffice for the execution of the newly arrived applications having strict performance requirements. However, an opportunistic online test scheduling method can take advantage of such situations and test the dark cores as long as there is enough room in the remaining power budget.

5 Dark Silicon Aware Online Testing Framework

The proposed dark silicon aware online testing framework is presented in Fig. 3. It actually enhances the standard runtime controller discussed in Section 2, with some additional components devoted to the execution of the test-related activities.

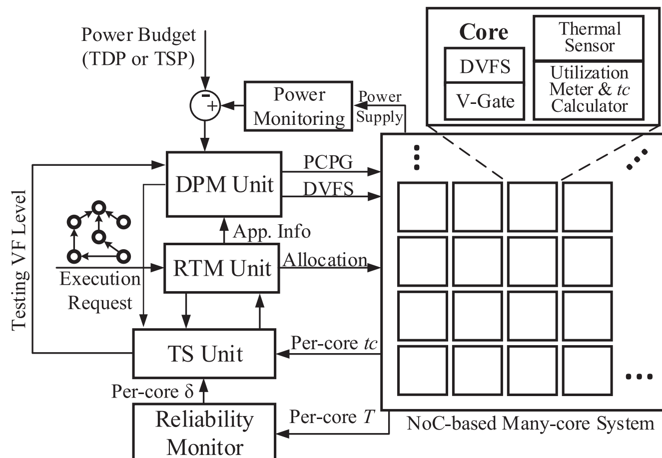


Fig. 3. The system architecture including the online testing framework.

The goal of the framework is to integrate the test execution in a transparent way with respect to the nominal activities of the architecture; in this way, it is possible to ensure the reliability of the system by testing the cores when strictly necessary, while limiting the performance overhead of the executed applications. The basic idea of the proposed approach is to test each core with a frequency proportional to the stress it has been affected. If a core is frequently used by the RTM Unit for the execution of the applications, it is highly stressed and therefore needs frequent tests; on the other side, if the core has been rarely allocated, it does not require urgent testing in the near future. Therefore, using information on the utilization of each core, a test scheduler can considerably reduce the test time by avoiding over-testing the under-utilized cores in the system. This aspect is highly relevant in the considered working scenario because of the discussed power and performance issues.

In this perspective, the proposed framework contains a novel software component, running on the host machine, that is called Test Scheduling Unit (TSU) and is in charge of the selection of the cores that needs to be tested and the scheduling of the testing task on such cores. This unit will work in a tightly-coupled way with the RTM Unit and the DPM one to define the proper test scheduling. Moreover, a utility module, called Reliability Monitor (RM), will provide to the TSU information on the aging status of the processors in order to drive the selection of the cores to be tested. In the following sections, the various activities of the TSU and of the RM will be discussed in details together with the internal modifications to the RTM Unit necessary to handle the test information received by the TSU.

5.1 Monitoring of the Cores' Utilization

A classical approach used in literature on online testing to estimate the stress suffered by each core is to simply count the number of instructions executed on a specific core. For instance, the approach proposed in [23] schedules a test of a core within the system every time the instruction counter reaches a specified threshold (i.e., 10 M, 100 M, and 1 B instructions); then after the test, the counter is reset.

In literature, such count is called *utilization metric*, since it is used to quickly estimate how much the core has been utilized. It is worth mentioning that some of today's platforms are equipped with such utilization meter (UM) feature. As a real platform in Intel SCC platform there exists hardware performance counter to report the core's utilization.

Therefore, in the considered framework, each core is equipped with an Utilization Meter, that is a hardware counter of the executed instructions; the counter is incremented every time an instruction is executed and is reset when the core is tested. The UM values of the cores are organized in an M×N matrix, with the same dimension of the architecture, and each value is identified as $\alpha_{i,j}$. Based on the $\alpha_{i,j}$, the UM computes a *test criticality* parameter tc_{ij}, also organized in a matrix with the same dimension, that point out which is the core that is more urgent to be tested at the next suitable time. The test criticality parameter is defined according to the following formula:

$$tc_{ij} = \frac{\alpha_{ij}}{\delta} - 1, \tag{1}$$

where δ is the threshold stating the number of executed instructions that triggers the test execution. The value of the threshold is tuned at run-time according to an estimation of the current lifetime reliability of the core; the approach will be presented later in Section 5.4.

According to this definition, tc_{ij} assumes a value (i.e., a real number) between -1 and $+\infty$. In this range, as long as tc_{ij} is lower than 0.0, the corresponding core does not need to be tested, since the α_{ij} value is still lower than the specified threshold δ. Then, whenever tc_{ij} exceeds 0.0, it means the corresponding core needs to be tested at the earliest possible moment. The UMs send tc_{ij} matrix to the TSU at a fixed timing interval. In this way, the test request will be interrupt-based rather than polling-based and redundant packet generation on the system will be avoided.

Finally, when the testing process of the corresponding core is started, the corresponding α_{ij} counter is reset to 0, and consequently, the tc_{ij} value is re-updated to -1.

5.2 The Proposed Test-Aware Mapping Method

As discussed above, the TSU requires to test a core every time its test criticality tc_{ij} becomes greater than 0.0. However, to run the test, the candidate core must be idle and there should be enough power budget for the test application. Therefore, to prepare the testing activities, the TSU prevents the RTM Unit from allocating cores with $tc_{ij} > 0.0$; in this way, unallocated cores can be later scheduled for testing in an appropriate time, when there is enough available power budget.

At the same time, to achieve a high runtime performance, the RTM Unit uses a strategy which maps the tasks of the same application on a contiguous set of cores [33], [34]. In this way, communication power and latencies are considerably reduced. In our RTM Unit, such region of neighbor cores is identified through the Squared Factor SF_{ij} (introduced in [33]), a metric which counts the number of almost-contiguous available nodes around a given node. However, disabling the cores having $tc_{ij} > 0.0$ may cause a dispersion of the planned contiguous allocations. For instance, let us assume that an application with 10 tasks has to be executed on the system depicted in Fig. 1. As the SF_{ij} of the node (4, 5) is 10, it will be selected to map the application onto its surrounding nodes. However, if two cores of this region have $tc_{ij} > 0.0$, the RTM Unit has to allocate some available nodes from south-west region of the system which leads to high dispersion.

To avoid such performance crippling dispersions, the RTM Unit calculates a new SF_{ij} value by subtracting the number of cores with $tc_{ij} > 0.0$ from the original SF value. As a result, the new SF_{ij} value shows the number of available cores around a given core that are not candidate for testing. For instance, the new SF value of the core in Fig. 1 will be $newSF_{ij} = 10 - 2 = 8$. Thus the core will not be selected as the first node for mapping an application with 10 tasks, but with eight tasks instead.

5.3 The Proposed Test Scheduling Method

The test scheduling algorithm determines the cores to be tested among the list of candidates. In practice, due to limits on the available power budget, it may not be possible to test all the candidate cores at the same time. Hence, it is necessary to define a ranking to assign a priority to the testing activities. Indeed, the ranking should follow the test criticality values. The higher the tc_{ij} is, the more critical to schedule a testing application on that core. However, our analysis shows that in the cases with close tc values, the cores with vacant vicinity should be ranked higher for testing.

First, cleaning up regions of idle cores facilitates future application mappings. Isolated cores with or without $tc_{ij} > 0.0$ are, nevertheless, not suitable for being allocated. Let us assume we test a core with busy neighbors instead of the one with idle cores around. The allocation of isolated (just tested) core leads to high dispersion of applications (degrading the system performance). Whereas, the idle area of cores is not cleaned up from cores with $tc_{ij} > 0.0$. Hence, utilization of its cores leads to a dispersed mapping. Moreover, testing applications are power-hungry. We should avoid placing them in proximity to each other or to other running applications. Otherwise, testing several adjacent cores together can cause high power densities, and consequently thermal hotspots.

Based on the above, we tend to clean up regions of idle cores (e.g., the dark region in Fig. 2c) from cores with $tc_{ij} > 0.0$ when scheduling test routines. In other words, when having two cores with similar tc_{ij} values, we prefer to test the one which is more likely to be used in near future allocations and is far from other running cores. As a metric, SF_{ij} estimates the number of vacant cores around a given core; i.e., the larger the SF_{ij} value of a core is, the higher the number of idle cores around it. In summary, we define our ranking parameter tr_{ij} for each core:

$$tr_{ij} = tc_{ij} + \frac{\sqrt{SF_{ij}}}{total\ number\ of\ cores}, \tag{2}$$

where SF_{ij} value is normalized to the total number of cores in the system. Moreover, we used a square root of SF_{ij} value to limit its impact to the cases where tc_{ij} values are too close

together. For instance, in case of equal tc_{ij} values in Fig. 1, we rank cores $(3, 4)$ and $(4, 6)$ higher than the core $(7, 3)$.

Algorithm 1 shows the pseudo-code for the selection of the cores for the test scheduling. Do note that we further control the negative impact of testing on system performance. That is, the maximum number of cores that can be simultaneously tested is limited by a given threshold, $\tau_{\#Test}$. The motivation is that we have to cope with a highly evolving scenario: while there might be enough power at the moment to test even more cores, this can change in near future. Other applications might enter the system, or the power demand and behavior of running applications might change.

Algorithm 1. Selecting Cores for Test Scheduling

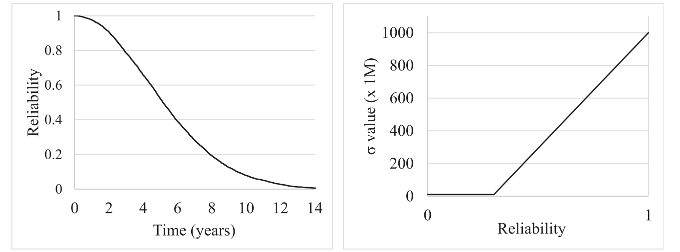
In predefined intervals:
1: **if** there is available resource and power for test **then** {// One of the suitable scenarios shown in Fig. 2}
2: Sort available cores based on their tr_{ij} values;
3: **while** there is enough power budget for test **and** # of (cores under test) $< \tau_{\#Test}$ **do**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

Online-testing overhead. In general, test application time is proportionally short compared to applications' execution time. However, regardless of the application types, the overhead of the online testing is almost independent of the applications' execution time. The test criticality value of a core (tc_{ij}) depends on the number of instructions executed over time. In case of short applications, tc_{ij} becomes greater than 0 only *after* execution of several applications. While, in case of long applications, the allocated core might need to go under test after the application execution. In this case, the overhead would be again negligible compared to execution time of the application. Finally, it is worth noting that the tc_{ij} value increases significantly in case of executing very long applications. We leave it for future work to better analyze this case and develop methods such as task migration to overcome the issue.

5.4 Aging-Aware Tuning of the Test Scheduling

The goal of the online testing approach is to detect at runtime the possible occurrence of permanent faults mainly due to device aging and wear-out phenomena, in order to take some recovery or containment actions. It is worth noting that, at the beginning of the lifetime of the system, when the core is new, it is presumable that it will have a low failure probability, and, therefore, it is not strictly necessary to test the core very frequently; however, as the core ages, this necessity increases proportionally to the failure probability. For this reason, we propose to test a core with a frequency proportional to its aging status. To implement such idea, we have defined an approach to dynamically tune the δ threshold, used in the test criticality computation, on the basis of the reliability status of the core itself.

As shown in Fig. 3, the system contains a Reliability Monitor, devoted to the computation of the aging status of each core. The RM can be implemented in two different ways: in hardware, by means of wear-out sensors, or in software, by using a statistical lifetime reliability model relying on the existence of per-core thermal sensors within the platform. However, even if there is a large effort on the study and the design of wear-out sensors (e.g., [35]), they are not currently integrated in commercial devices, while per-core thermal sensors are commonly available (e.g., Intel Single-chip Cloud Computer [21]). Therefore, Dynamic Reliability Management approaches proposed in the literature (e.g., [15]) generally implements the statistical lifetime reliability model as a software process, and we acted in the same direction. Indeed, we envision that the proposed testing framework is used in conjunction with a DRM approach for the application mapping; in this situation, the RM would be already available in the system and shared with the DRM Unit.



(a) Lifetime reliability curve (b) Tuning of the δ parameter

Fig. 4. Relationship between reliability and δ parameter.

The RM has been implemented similarly to the one proposed in [15], and based on the reliability model presented in [13]. The internals of the monitor are here briefly presented; further details can be found in [13], [15]. The RM implements the standard lifetime reliability model based on a Weibull distribution [14]:

$$R(t) = e^{-\left(\frac{t}{\alpha(T)}\right)^{\beta}} \tag{3}$$

being t the current instant of time (generally measured in hours), T the constant worst-case processor temperature (Kelvin degrees), β the Weibull slope parameter, and $\alpha(T)$ the scale parameter, or aging rate. Fig. 4a shows an example of reliability curve of a core according to the technological characterization presented in [13]; in such a situation the Mean Time To Failure (MTTF) of the system (measured as $\int_0^{\infty} R(t)\,\mathrm{d}t$) would be around five years. The $\alpha(T)$ parameter formulation depends on the considered wear-out mechanisms, that are for instance the electromigration, the hot carrier injection (HCI), or the thermal cycling. Moreover, several failure mechanisms can be combined in a single $\alpha(T)$ formula by means of the Sum Of Failure Rate (SOFR) approach. For simplicity, Equation (3) considers only a constant temperature; this aspect may cause pessimistic non-accurate evaluation of the reliability especially when the designer aims at analyzing the reliability with respect to a variable usage of the core. Therefore, as shown in [13], to consider a varying temperature caused by changing in the core utilization, Equation (3) has been enhanced in the RM internal implementation as following:

$$R(t) = e^{-\left(\sum_{j=1}^{i} \frac{\tau_j}{\alpha_j(T)}\right)^{\beta}}, \tag{4}$$

where τ_j represents the duration of each period of time with constant steady-state temperature T_j up to time t (i.e., $t = \sum_{j=1}^{i} \tau_i$).

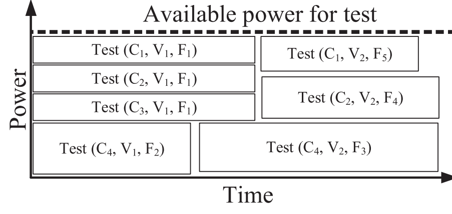


Fig. 5. An example of rectangular packing model for power aware testing.

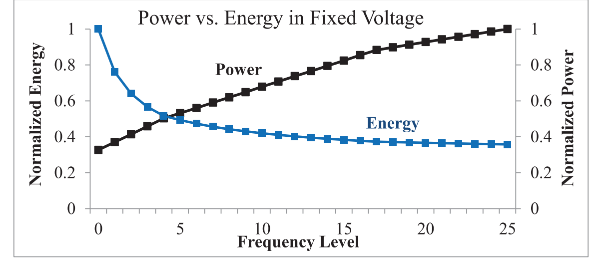


Fig. 6. Power versus energy in a fixed voltage level.

The RM periodically analyzes the aging status of each core to tune the related δ value; similarly to the other informations, the reliability values of the various cores $R_{ij}(t)$ and the corresponding $\delta_{ij}(t)$ are organized in a matrix that are periodically transmitted to the RTU. To compute at runtime the δ threshold in Equation (1), to be used at time t, the RM uses the following formula:

$$\delta(t) = \frac{\delta_{R_{max}} - \delta_{R_{min}}}{R_{max} - R_{min}} \cdot (R_{max} - R(t)) + \delta_{R_{max}}. \tag{5}$$

The above formula has been obtained by means of the standard equation of the line connecting two points. $\delta_{R_{max}}$ and $\delta_{R_{min}}$ are the two threshold set by the designer for the scenario in which the system is new (typically $R_{max} = 1$), and the system is assumed to be at the end of the lifetime (i.e., $R = 0.3$ in Fig. 4b), respectively. If the reliability of the core overcomes the R_{min} value, the δ value will remain constant since it is non possible to decrease it below a given value (it would mean to continue testing the core without actually executing anything). As an example, Fig. 4b reports the curve for the tuning of the δ value specifying $\delta_{R_{max}} = 1\text{B}$ when the core is new ($R = 1$), and $\delta_{R_{min}} = 10$ M when the core has a reliability value $R = 0.3$.

6 Test Scheduling for Different Voltage-Frequency Settings

Recent studies have shown that some specific faults manifest themselves in a particular voltage-frequency settings. For instance, in [36], it is shown that certain resistive bridging faults can occur at specific voltage levels. These studies have concluded that multi-/many-core systems equipped with DVFS feature should be tested at multiple voltage levels to ensure that cores can operate reliably at different conditions. However, testing at multiple voltage levels is more challenging compared to single voltage level testing due to repetitive test process, high switching time to change the voltage of the core under test, and limited upper bound for the maximum possible operating frequency for each voltage level [37]. The trivial and straightforward solution to repetitively run a test process for every voltage level drastically increases the overall test application time, having a direct impact on the overall system performance. Furthermore, at low voltage levels, test process becomes slower resulting in a longer TAT. In this section, we propose an efficient technique to test cores at different voltage levels with the aim of providing a uniform testing probability for all the levels while minimizing the performance overhead.

To apply online testing on cores running at different voltage levels, it is essential to use a test scheduling policy with the minimum negative impact on system performance. To this end, allocated cores(s) need to be detected and enough power budget need to be available for the test purpose so that the upper power consumption bound will not be violated. However, as test power consumption at different voltage levels considerably varies, the suitable frequency level in each voltage level should be properly determined at runtime.

In multi-/many-core systems equipped with DVFS feature, usually for each voltage level, an upper bound for the maximum frequency that can operate at that voltage level is defined. For example, in Intel SCC platform, seven voltage levels for each island are defined where each voltage level has a maximum possible frequency, thus forming more than 15 VF levels per island which can be changed at runtime. In each particular voltage level, different operating frequencies used for testing result in different test power/energy. As the system is tested at runtime with functional methods, and a test at a certain voltage level can be performed at different frequencies (i.e., equal or lower than the maximum frequency at the respective voltage [37]), we define a VF set as the set of different available frequencies (i.e., VF levels) that can be selected for testing at a given voltage level. At low VF levels, power consumption is lower at the cost of longer TAT, compared to high VF levels where higher power consumption is needed to achieve a shorter TAT. This raises a question whether it is more efficient to use a low VF level and save the power to have parallel testing of multiple cores or to use a higher VF level and reduce the TAT for individual cores. Our solution to address this issue is inspired by the traditional 2D rectangular packing model used in power-constrained testing. Fig. 5 shows an example of using 2D rectangular model when three cores are tested over time at different VF levels. Each rectangle depicts a test process as a triplet (C_i, V_j, F_k) where C_i is the core to be tested, V_j is the voltage of test, and F_k is the frequency of test. The width and length of the rectangle correspond to the test power consumption and TAT, respectively, where the total summation of test power at each moment of time should not exceed the maximum available power budget for test. It can be observed that when power budget is limited and an optimal test scheduling algorithm is used, the total areas of the all test rectangles determine the overall test time. This area is the TAT-test power product which can be called as energy consumption for test. We use the energy consumption for test as a metric to choose the proper VF level for test when there is an option to select one VF level among the available VF levels in a particular VF set.

In Fig. 6, we have compared the normalized energy with different frequency levels when the voltage is fixed. As can be seen, by increasing the frequency up to the maximum possible frequency, the energy consumption exponentially decreases. That is because of the fact that for a constant

voltage, the static power remains constant, and by decreasing the frequency, the penalty of unchanged high static power superimposes the overall power and energy accordingly. From these two observations, we propose a general rule for our test scheduling algorithm that for a given voltage level, the test frequency should be increased as much as possible while monitoring and honoring the total power budget.

Algorithm 2. Selecting Cores for Test Scheduling with VF Selection Algorithm

Inputs: P_c: Instantaneous power measurement from the sensor;
P_{max}: The maximum power budget (i.e., TDP or TSP);
Output: CUT'_{set}: The target core(s) to be tested at specified VF level(s) (i.e., set of (C_i, V_j, F_k));

Variables: $availablePower$: Available power for test;
CUT_{set}: Temporary variable for the target core(s) and their VF level(s) for test;
CUT_{power}: Core under test power consumption at a given VF level;
Constant: $\tau_{\#Test}$: Maximum number of core(s) under test;

Body:
1: $availablePower \leftarrow P_c$ - P_{max};
2: **while** $availablePower > 0$ and # of (cores under test) $< \tau_{\#Test}$ **do**
3: Sort available cores based on their tr_{ij} values;
4: $C_i \leftarrow$ The first core in the ranked list for testing;
5: **if** C_i is not tested in $VFset_j$ **then**
6: $(CUT_{power}, V_j, F_k) \leftarrow minPower(VFset_j)$;
7: **if** $CUT_{power} < availablePower$ **then**
8: $CUT_{set} \leftarrow (C_i, V_j, F_k)$;
9: Update tr for all cores;
10: $availablePower \leftarrow availablePower$ - CUT_{power};
11: **while** $availablePower > 0$ and there is unselected core(s) in CUT_{set} **do**
12: select core (C_i, V_j, F_k) from CUT_{set};
13: (CUT_{power}, V_j, F'_k) $maxVF((C_i, V_j, F_k)$, $availablePower)$;
14: $CUT'_{set} \leftarrow (C_i, V_j, F'_k)$;
15: update $(availablePower)$;

Algorithm 2 shows in more details the proposed test scheduling strategy for testing the cores at different VF levels. Algorithm 2 is the extension of Algorithm 1 to consider VF levels in test scheduling, thus offering the system manager the option to choose two different test scheduling policies with contrasting reliability-complexity trade-offs. The input of the test scheduler is the instantaneous power consumption of the system (i.e., P_c) which is provided by the chip power sensor and the output of the test scheduler is the core(s) targeted for being tested at specified VF level(s) (i.e., set of (C_i, V_j, F_k) where V_k and F_k are the voltage and frequency of the core C_i during the test process).

First the amount of available power budget (i.e., $availablePower$) is calculated which is the available portion of power budget that can be used for test purpose (Line 1 in Algorithm 2). If it is less than or equal to zero, it means there is no available power for the test purpose. If $availablePower$ is greater than zero (i.e., there exists available power for the test purpose) and the number of cores under test is smaller than the maximum threshold (i.e., $\tau_{\#Test}$), the algorithm will find the first core in the list of cores sorted based on their tr_{ij} values as the target for test (Line 3-4). If such a core exists in the system, for each VF set (i.e., $VFset$) at which the core has not been tested yet, the algorithm will check if the available power can be used to test the core at that VF set or not (Line 5-12). Based on a pessimistically pre-calculated test power for each VF level, the function $minPower$ returns the minimum required test power (i.e., CUT_{power}) and the corresponding voltage and frequency (i.e., (V_j, F_k)) to test the core at one of the VF levels at that specific VF set (i.e., $VFset_j$). If the test power is less than the available power, then the core and voltage-frequency for test will be added to the set of target cores for test (i.e., CUT_{set}) and $availablepower$ will be updated accordingly (Line 7-11). Whenever a core is selected for test, the t_r value for other cores will be updated based on the consideration of the selected core as an occupied node. This causes the next core for test to be selected in other vacant areas.

The process of searching for cores to be tested at the minimum possible VF level continues until either the $\tau_{\#Test}$ threshold is reached or $availablePower$ is less than zero. After that, if $availablePower$ is larger than zero meaning that extra power budget is available for test, the algorithm will attempt to use $availablePower$ to minimize the TAT by selecting the highest possible VF level for the core(s) under test (Line 14-19). The function $maxVF$ returns the highest possible VF level from the VF set considering the $availablePower$. If such a level exists, then the new voltage-frequency for test will be added to the updated set of target cores for test (i.e., CUT'_{set}) and $availablepower$ will be updated. This process continues until either $availablePower$ is larger than zero or all the cores in CUT_{set} are selected. The proposed algorithm for determining the appropriate VF level for test is a simple greedy approach inspired from solution proposed and validated in state-of-the-art power constrained testing in multi-clock domain SoCs [38]. The difference is that, the available power budget in power constrained testing is a fixed value while in online testing this value might change during the test process. However, if the test time is short enough (which is reasonable assumption as discussed in Section 5.3), we can assume that the power budget for test does not change and two problems are the same. More details regarding the efficiency of this method can be found in [38]. It is worth noting that the proposed algorithm for test scheduling is targeted for platforms featuring per-core DVFS. The extension to also consider per-cluster DVFS is left as a future work.

TABLE 1
The System Settings for Different Experiment Setups

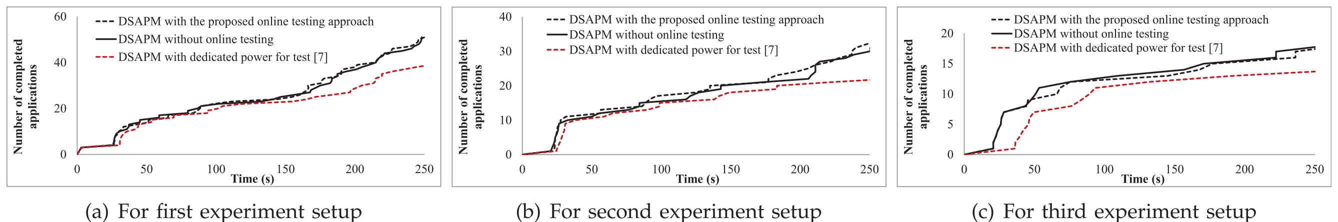
	Technology Node	System Type	Area (mm^2)	NoC Size
First Setup	16 nm	medium	138	12×12
Second Setup	22 nm	large	232	11×11
Third Setup	32 nm	large	254	8×8

7 Experimental Results

To demonstrate the efficiency of our dark silicon aware online testing approach on many-core systems, we run three set of experiments using different technology nodes as shown in Table 1. The simulation platform implemented

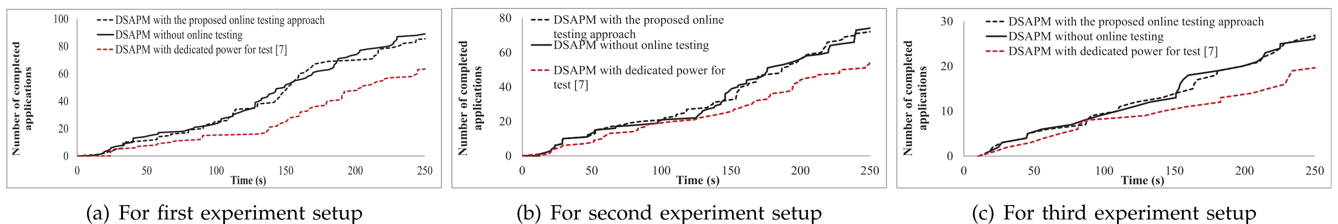
TABLE 2
Throughput Penalty for Different Experiment Setups and δ Values While Using TDP and TSP

	First experiment setup			Second experiment setup			Third experiment setup		
δ value	10 M	100 M	1 B	10 M	100 M	1 B	10 M	100 M	1 B
TDP-based(%)	1.31	0.29	0.13	2.9	0.58	0.21	5.9	1.3	1.06
TSP-based(%)	1.38	0.32	0.16	3.2	0.63	0.21	5.86	1.3	1.12



(a) For first experiment setup (b) For second experiment setup (c) For third experiment setup

Fig. 7. The number of completed applications versus time (TDP-based approach).



(a) For first experiment setup (b) For second experiment setup (c) For third experiment setup

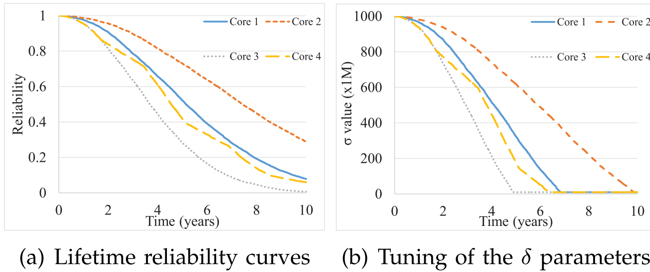
Fig. 8. The number of completed applications versus time (TSP-based approach).

for [6] has been adapted to integrate the new online testing framework. As a processing core in baseline design, we use Niagara2-like in-order core specifications obtained from McPAT [39]. *NetlistGen.exe* is used for generating netlists of the synthesized cores and fault simulation with PLI library in HDL environment [40]. Physical scaling parameters, power model, voltage-frequency scaling model, and TDP calculation were extracted from the Lumos [8], a framework to analytically quantify power-performance characteristics of many-cores in near-threshold operation. The physical scaling parameters have been calibrated by circuit simulations with a modified Predictive Technology Model [41]. We integrated HotSpot 5.0 [42], and Caliper [13] for modeling the thermal behavior and the lifetime reliability of the system, respectively. We also integrated TSP calculation tool from [16] to evaluate the dynamic power budget based on the number of active cores at the moment. We generate several set of applications with four to 35 tasks using TGG [43], and we consider three real applications, namely MPEG-4, UAV and VOPD, from [44].

We use the run-time mapping algorithm presented in [32], and the dark silicon aware power management (DSAPM) technique presented in [18]. In this power management strategy, a Proportional Integral Derivative (PID) controller is used for dynamic power management that considers a fixed power budget (i.e., TDP). Functional testing is used to test cores. 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 [45]. 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 percent, respectively. The test application duration is 9,000 cycles for each core. The models presented in [8] are also used to calculate dynamic and static power consumption of the test process. Finally, we set $\tau_{\#Test}$, to 4.

Table 2 shows the throughput penalty of our proposed testing method when a constant δ is set to 10 M, 100 M, and 1 B instructions while using two different values as the maximum power limit viz. TDP and TSP. As can be seen from the table, our proposed online testing method has a negligible throughput penalty compared with many-core systems without any online testing method for both TSP and TDP based budgets. For all the δ values, the minimum throughput penalty is observed at the first experiment setup where power limitation is more challenging and the system size (i.e., the total number of cores) is larger than in the other experiments. This will give more opportunities to the TSU to find suitable scenarios for online testing. The penalty also increases when δ is set to smaller values because cores enter the test critical range more rapidly. It can be noticed that the penalty while using TSP as the maximum power limit is so close to the penalty of using TDP. 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 [23], [24]. 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.

In a second experimental session, the performance overhead of the proposed approach is compared more in details against one of the most relevant state-of-the-art methods [7] in which fixed amount of power budget is dedicated to the test process. Figs. 7 and 8 show the system throughput over time (250 seconds) for the both TDP-based and TSP-based



(a) Lifetime reliability curves (b) Tuning of the δ parameters

Fig. 9. Adaptive tuning of the δ parameter for four cores characterized by different aging trends. In the two figures, corresponding curves, related to the same core, are represented with the same line pattern.

approaches, and different experiment setups when δ is set to 10 M (i.e., the worst case scenario). It can be seen that the proposed online testing approach achieves a better performance over time compared to the DSAPM approach with dedicated power for test. The throughput penalties while $\delta = 10$ M for DSAPM with dedicated power for test in 16, 22, and 32 nm technologies are 23, 20, and 16 percent, respectively for the TDP-based approach, and 25, 23, and 22 percent for the TSP-based approach which is larger compared to the corresponding results reported in Table 2 for the proposed method in this paper. Furthermore, it can be noted that applications are being completed and leave the system with almost the same trend for the both DSAPM with online testing and DSAPM without online testing. This shows the capability of the proposed method to make use of available power budget and resources at runtime for testing the cores with a negligible penalty on the system performance.

The performed analysis for constant δ values offers the possibility to evaluate the throughput penalty in specific instants over the lifetime of the considered many-core architecture. We also analyzed how the δ value of various cores changes over time according to its aging trend. To this end, on the basis of an envisioned average workload, we computed the cores of the system to have an average aging trend shown in Fig. 4a, and, therefore, an average MTTF of the cores approximately equal to five years. Thus, we set the $\delta_{R_{max}=1} = 1$B, at the beginning of the lifetime of the system, while we set $R_{min} = 0.3$ (from the graph it is possible to see

TABLE 4
The Share of Each VF Set from the Total Number of Tested Cores (Percent) for 16, 22, and 32 nm Technologies

Tech.	VF level					
	Set 1	Set 2	Set 3	Set 4	Set 5	Set 6
16 nm	15%	15%	16%	17%	18%	19%
22 nm	16%	16%	16%	17%	17%	18%
32 nm	16%	16%	16%	17%	17%	18%

that it is a reasonable reliability value after the MTTF) and $\delta_{R_{min}=0.3} = 10$ M. Fig. 9a shows the reliability curves for four different cores within the architecture, each one aging with a different trend according to the actual specific stress it is affected during the lifetime; in fact, according to the mapping strategy and DVFS tuning, each core is subject to a different load over the time. For this reason, to show the adaptive capabilities of the technique, we represent in the figure four highly different situations: in particular, Core 1 is characterized by an aging trend similar to the envisioned one, while the other ones diverges from that trend due to a load different from the expected one; finally, Core 4 has an oscillating trend since it is subject to a highly variable load over the time. Fig. 9b shows how the system adapts at runtime the δ value according to the actual aging of each core.

To apply online testing on cores running at different voltages/frequency sets, six VF sets (i.e., voltage levels) are defined for our system which have totally 29 VF levels. Table 3 shows these different VF sets, corresponding voltage, and available frequencies in each set. The target VF level to be assigned to the core under test is chosen among all the options in each VF set. Table 4 show the share of each VF set from the total number of tested cores at the end of the simulations for 16, 22, and 32 nm technology, respectively. As can be noticed from the table, the share of each VF set is almost equal and fair for all the three experiment setups. As the sets with higher VF levels consume more power, their shares are a bit lower than the sets with lower VF levels; as a conclusion the sets with lower VF levels have a better chance to use the available power than the other sets.

TABLE 3
Voltage-Frequency Sets for Test

Technology	16 nm					
VF set for test	Set 1	Set 2	Set 3	Set 4	Set 5	Set 6
VF Level	1-5	6-10	11-15	16-20	21-25	26-29
Voltage(V)	0.47	0.51	0.56	0.59	0.63	0.68
Frequency(GHz)	0.4-0.64	0.4-1	0.4-1.54	0.4-2	0.4-2.6	0.4-3.1
Technology	22 nm					
VF set for test	Set 1	Set 2	Set 3	Set 4	Set 5	Set 6
VF Level	1-5	6-10	11-15	16-20	21-25	26-29
Voltage(V)	0.49	0.54	0.6	0.65	0.7	0.74
Frequency(GHz)	0.4-0.67	0.4-1.1	0.4-1.6	0.4-2.1	0.4-2.8	0.4-3.2
Technology	32 nm					
VF set for test	Set 1	Set 2	Set 3	Set 4	Set 5	Set 6
VF Level	1-5	6-10	11-15	16-20	21-25	26-29
Voltage(V)	0.52	0.58	0.63	0.69	0.75	0.8
Frequency(GHz)	0.4-0.68	0.4-1.13	0.4-1.6	0.4-2.2	0.4-2.8	0.4-3.2

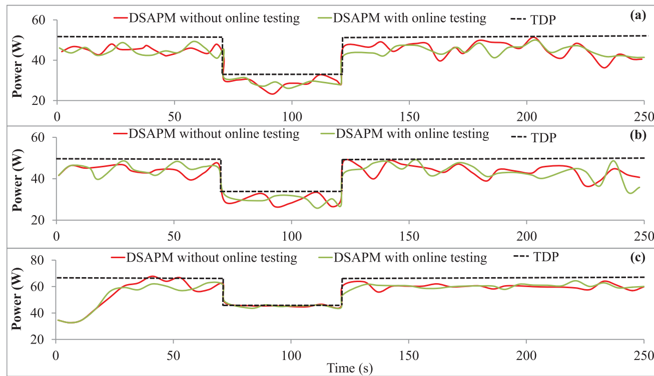


Fig. 10. The power consumption of the system with and without online testing approach for different experiment setups, (a) 16 nm, (b) 22 nm, and (c) 32 nm.

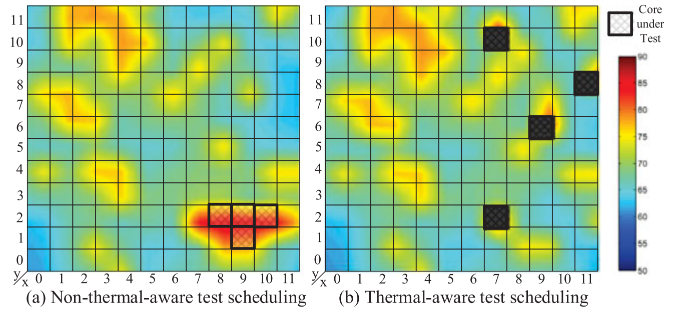


Fig. 11. Heat maps while running non-thermal-aware and thermal-aware test scheduling.

Fig. 10 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 for different application setups when δ is set to 10 M (i.e., the worst case scenario). We run the system for 250 seconds while applications enter and leave the system at 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 in scenarios when the system is frequently busy and the total chip power consumption is most of the time close to the upper bound.

Finally, to demonstrate how our test scheduling in a thermal-aware manner avoids hotspots, we study the effect of considering square factor (SF) in Equation (2). For this we monitored several thermal snapshots during system runtime and we compared it against a modified version in which such parameter is not considered in Equation (2) (dubbed as non-thermal-aware scheduling). Fig. 11 shows the heat maps while running non-thermal-aware and thermal-aware test scheduling in a thermal snapshot while the maximum number of cores under test is set to 4 (i.e., $\tau_{\#Test} = 4$). As can be seen, the non-thermal-aware scheduling selects four neighboring cores which results in hotspot, but the thermal-aware strategy selects cores which are far from each other to avoid thermal hotspots.

8 Conclusions

This paper proposed a new power-aware online testing strategy for many-core systems in the dark silicon era. The approach consists of a dynamic test scheduling strategy for software based self test applications able to perform testing activities in a transparent way with respect to the execution of the nominal workload. The goal of the approach is to guarantee the reliability of the system by detecting the occurred permanent faults, while minimizing the overhead on the system's performance and satisfying the limited available power budget. Our experimental results show that the proposed power-aware online testing approach can 1) efficiently utilize temporarily free cores and available power budget for the testing purposes, within less than 1 percent penalty on system throughput, 2) adapt to the current aging status of each core, and 3) cover and balance all the VF levels during the various test.

References

[1] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger, "Dark silicon and the end of multicore scaling," *IEEE Micro*, vol. 32, no. 3, pp. 122–134, May/Jun. 2012.
[2] M. Taylor, "Is dark silicon useful? harnessing the four horsemen of the coming dark silicon apocalypse," in *Proc. Des. Autom. Conf.*, 2012, pp. 1131–1136.
[3] M. Kaliorakis, M. Psarakis, N. Foutris, and D. Gizopoulos, "Accelerated online error detection in many-core microprocessor architectures," in *Proc. VLSI Test Symp.*, 2014, pp. 1–6.
[4] M. Kaliorakis, N. Foutris, D. Gizopoulos, M. Psarakis, and A. Paschalis, "Online error detection in multiprocessor chips: A test scheduling study," in *Proc. Int. On-Line Testing Symp.*, 2013, pp. 169–172.
[5] P. Parvathala, K. Maneparambil, and W. Lindsay, "FRITS—a microprocessor functional BIST method," in *Proc. Int. Test Conf.*, 2002, pp. 590–598.
[6] M.-H. Haghbayan, A.-M. Rahmani, M. Fattah, P. Liljeberg, J. Plosila, Z. Navabi, and H. Tenhunen, "Power-aware online testing of manycore systems in the dark silicon era," in *Proc. Conf. Des., Autom. Test Eur.*, 2015, pp. 435–440.
[7] M.-H. Haghbayan, A.-M. Rahmani, P. Liljeberg, J. Plosila, and H. Tenhunen, "Energy-efficient concurrent testing approach for many-core systems in the dark silicon age," in *Proc. Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst.*, 2014, pp. 270–275.
[8] L. Wang, et al., "Dark vs. dim silicon and near-threshold computing extended results," Dept. Comput. Sci., Univ. Virginia, Charlottesville, VA, USA, Tech. Rep. TR-2013-01, 2012.
[9] A.-M. Rahmani, M.-H. Haghbayan, A. Kanduri, A. 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. Int. Symp. Low Power Electron. Des.*, 2015, pp. 1–6.
[10] M. Shafique, S. Garg, J. Henkel, and D. Marculescu, "The EDA challenges in the dark silicon era: Temperature, reliability, and variability perspectives," in *Proc. Des. Autom. Conf.*, 2014, pp. 185:1–185:6.
[11] M. Haghbayan, A. Rahmani, P. Liljeberg, J. Plosila, and H. Tenhunen, "Online testing of many-core systems in the dark silicon era," in *Proc. Int. Symp. Des. Diagnostics Electron. Circuits Syst.*, 2014, pp. 141–146.
[12] X. Kavousianos and K. Chakrabarty, "Testing for SoCs with advanced static and dynamic power-management capabilities," in *Proc. Conf. Des., Autom. Test Eur.*, 2013, pp. 737–742.
[13] C. Bolchini, M. Carminati, M. Gribaudo, and A. Miele, "A lightweight and open-source framework for the lifetime estimation of multicore systems," in *Proc. IEEE Int. Conf. Comput. Des.*, 2014, pp. 166–172.
[14] JEDEC Solid State Tech. Association, "Failure mechanisms and models for semiconductor devices," JEP122G, 2010.
[15] P. Mercati, A. Bartolini, F. Paterna, T. Rosing, and L. Benini, "A Linux-governor based dynamic reliability manager for Android mobile devices," in *Proc. Conf. Des., Autom. Test Eur.*, 2014, pp. 1–4.

[16] S. Pagani, H. Khdr, W. Munawar, J.-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. Int. Conf. Hardware/Softw. Codes. Syst. Synthesis*, 2014, pp. 10:1–10:10.
[17] M. Shafique, S. Garg, T. Mitra, S. Parameswaran, and J. Henkel, "Dark silicon as a challenge for hardware/software co-design," in *Proc. Int. Conf. Hardware/Softw. Codes. Syst. Synthesis*, 2014, pp. 13:1–13:10.
[18] M.-H. Haghbayan, A.-M. Rahmani, A. Weldezion, P. Liljeberg, J. Plosila, A. Jantsch, and H. Tenhunen, "Dark silicon aware power management for manycore systems under dynamic workloads," in *Proc. Int. Conf. Comput. Des.*, 2014, pp. 509–512.
[19] K. Ma and X. Wang, "PGCapping: Exploiting power gating for power capping and core lifetime balancing in CMPs," in *Proc. Int. Conf. Parallel Archit. Compilation Techn.*, 2012, pp. 13–22.
[20] Z. Chen and D. Marculescu, "Distributed reinforcement learning for power limited many-core system performance optimization," in *Proc. Des. Autom. Test Eur. Conf. Exhib.*, 2015, pp. 1521–1526.
[21] J. Howard, et al., "A 48-Core IA-32 message-passing processor with DVFS in 45nm CMOS," in *Proc. Int. Solid-State Circuits Conf.*, 2010, pp. 108–109.
[22] Intel. (2014). Haswell Xeon 5 [Online]. Available: http://www.intel.com/
[23] K. Constantinides, O. Mutlu, T. Austin, and V. Bertacco, "Software-based online detection of hardware defects mechanisms, architectural support, and evaluation," in *Proc. Int. Symp. Microarchit.*, Dec. 2007, pp. 97–108.
[24] S. Nomura, M. Sinclair, C.-H. Ho, V. Govindaraju, M. de Kruijf, and K. Sankaralingam, "Sampling + DMR: Practical and low-overhead permanent fault detection," in *Proc. Int. Symp. Comput. Archit.*, 2011, pp. 201–212.
[25] Y. Xia, M. Chrzanowska-Jeske, B. Wang, and M. Jeske, "Using a distributed rectangle bin-packing approach for core-based SoC test scheduling with power constraints," in *Proc. Int. Conf. Comput. Aided Des.*, 2003, pp. 100–105.
[26] H. Yu, S. Reddy, C. Wu-Tung, P. Reuter, N. Mukherjee, T. Chien-Chung, O. Samman, and Y. Zaidan, "Optimal core wrapper width selection and SOC test scheduling based on 3-D bin packing algorithm," in *Proc. Int. Test Conf.*, 2002, pp. 74–82.
[27] M. H. Haghbayan, S. Safari, and Z. Navabi, "Power constraint testing for multi-clock domain SoCs using concurrent hybrid BIST," in *Proc. Int. Symp. Des. Diagnostics Electron. Circuits Syst.*, 2012, pp. 42–45.
[28] R. Chou, K. Saluja, and V. Agrawal, "Scheduling tests for VLSI systems under power constraints," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 5, no. 2, pp. 175–185, Jun. 1997.
[29] F. Vartziotis, X. Kavousianos, K. Chakrabarty, R. Parekhji, and A. Jain, "Multi-site test optimization for multi-Vdd SoCs using space- and time- division multiplexing," in *Proc. Conf. Des. Autom. Test Eur.*, 2014, pp. 1–6.
[30] P. Venkataramani and V. Agrawal, "ATE test time reduction using asynchronous clock period," in *Proc. Int. Test Conf.*, 2013, pp. 1–10.
[31] G. Theodorou, N. Kranitis, A. Paschalis, and D. Gizopoulos, "Power-aware optimization of software-based self-test for L1 caches in microprocessors," in *Proc. Int. On-Line Testing Symp.*, 2014, pp. 154–159.
[32] M. Fattah, P. Liljeberg, J. Plosila, and H. Tenhunen, "Adjustable contiguity of run-time task allocation in networked many-core systems," in *Proc. Asia South Pacific Des. Autom. Conf.*, 2014, pp. 349–354.
[33] M. Fattah, M. Daneshtalab, P. Liljeberg, and J. Plosila, "Smart hill climbing for agile dynamic mapping in many-core systems," in *Proc. Des. Autom. Conf.*, 2013, pp. 1–6.
[34] C.-L. Chou, U. Ogras, and R. Marculescu, "Energy- and performance-aware incremental mapping for networks on chip with multiple voltage levels," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 27, no. 10, pp. 1866–1879, Oct. 2008.
[35] J. Blome, S. Feng, S. Gupta, and S. Mahlke, "Self-calibrating online wearout detection," in *Proc. Int. Symp. Microarchit.*, 2007, pp. 109–122.
[36] N. Ali, M. Zwolinski, B. Al-Hashimi, and P. Harrod, "Dynamic voltage scaling aware delay fault testing," in *Proc. Eur. Test Symp.*, 2006, pp. 15–20.
[37] X. Kavousianos, K. Chakrabarty, A. Jain, and R. Parekhji, "Test schedule optimization for multicore SoCs: Handling dynamic voltage scaling and multiple voltage islands," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 31, no. 11, pp. 1754–1766, Nov. 2012.
[38] T. Yoneda, K. Masuda, and H. Fujiwara, "Power-constrained test scheduling for multi-clock domain SoCs," in *Proc. Des., Autom. Test Eur.*, 2006, pp. 1–6.
[39] S. Li, J. H. Ahn, R. Strong, J. Brockman, D. Tullsen, and N. Jouppi, "McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures," in *Proc. Int. Symp. Microarchit.*, 2009, pp. 469–480.
[40] Z. Navabi, *Digital System Test and Testable Design: Using HDL Models and Architectures*. New York, NY, USA: Springer, 2010.
[41] B. Calhoun, S. Khanna, R. Mann, and J. Wang, "Sub-threshold circuit design with shrinking CMOS devices," in *Proc. Int. Symp. Circuits Syst.*, 2009, pp. 2541–2544.
[42] K. Skadron, M. Stan, K. Sankaranarayanan, W. Huang, S. Velusamy, and D. Tarjan, "Temperature-aware microarchitecture: Modeling and implementation," *ACM Trans. Archit. Code Optim.*, vol. 1, pp. 94–125, Mar 2004.
[43] (2010). TGG: Task Graph Generator [Online]. Available: http://sourceforge.net/projects/taskgraphgen/, last Update: 2013-04-11.
[44] M. Fattah, A.-M. Rahmani, T. Xu, A. Kanduri, P. Liljeberg, J. Plosila, and H. Tenhunen, "Mixed-criticality run-time task mapping for NoC-based many-core systems," in *Proc. Int. Conf. Parallel, Distrib. Netw.-Based Process.*, 2014, pp. 458–465.
[45] M. Haghbayan, S. Karamati, F. Javaheri, and Z. Navabi, "Test pattern selection and compaction for sequential circuits in an HDL environment," in *Proc. Asian Test Symp.*, 2010, pp. 53–56.