Process-Variation Aware Multi-Temperature Test Scheduling

Nima Aghaee, Zebo Peng, and Petru Eles
Embedded Systems Laboratory (ESLAB), Linkoping University, Sweden
{nima.aghaee, zebo.peng, petru.eles}@liu.se

***Abstract*—Chips manufactured with deep submicron technologies are prone to large process variation and temperature-dependent defects. In order to provide high test efficiency, the tests for temperature-dependent defects should be applied at appropriate temperature ranges. Existing static scheduling techniques achieve these specified temperatures by scheduling the tests, specially developed heating sequences, and cooling intervals together. Because of the temperature uncertainty induced by process variation, a static test schedule is not capable of applying the tests at intended temperatures in an efficient manner. As a result the test cost will be very high. In this paper, an adaptive test scheduling method is introduced that utilizes on-chip temperature sensors in order to adapt the test schedule to the actual temperatures. The proposed method generates a low cost schedule tree based on the variation statistics and thermal simulations in the design phase. During the test, a chip selects an appropriate schedule dynamically based on temperature sensor readings. A 23% decrease in the likelihood that tests are not applied at the intended temperatures is observed in the experimental studies in addition to 20% reduction in test application time.**

I. Introduction

Temperature-dependent defects are a challenge for achieving high test quality for chips manufactured with modern technologies [1]. This entails the need to apply tests within specified temperature ranges and also the necessity of having a variety of tests applied at different temperatures in order to achieve high defect coverage. Therefore, it is important to develop efficient methods to apply tests at the specified temperatures with a minimal cost [2].

Tests could be performed at specified temperatures using a temperature-aware schedule that adjusts the temperature by introducing cooling and heating intervals [2, 3]. A heating interval is a period when the chip under test is consuming large amount of power that is achieved by test controls. A cooling interval, on the other hand, corresponds to a period with very small power consumption. Heating could be achieved by applying a section of the normally generated test pattern that has the maximal power or a sequence of patterns that is especially generated to heat up the chip rapidly; while cooling can be simply done by not applying any patterns. This way, multi-temperature tests are performed without costly extra test equipment (such as external heating mechanisms). The challenge is that the test application time (which is already long) could become excessively long, resulting in an extremely high cost of test. Therefore, it is necessary to find a test schedule with a short test application time.

The existing multi-temperature test scheduling methods (for instance [2] and [3]) optimize the test schedule for the shortest test application time while making sure that the tests are applied in the specified temperature ranges. These methods neglect the temperature deviations that are mainly caused by process variation. Therefore, a large process variation implies a decreased number of chips that are tested within the specified temperature ranges, which will reduce the effectiveness of the tests and, in the worst case, may lead to damage of the chips due to overheating.

In order to maximize the chances that the tests are applied within the intended temperature ranges, static schedules should be designed pessimistically. In this case, a large process variation implies a very long test application time due to the intensive use of the heating and cooling intervals. This means that the chips under test are heating up/cooling down more than actually needed in order to make sure that it is warm/cold enough for the majority of the chips (this situation is detailed in section III).

In this paper, an adaptive method that utilizes on-chip temperature sensors in order to adapt the test schedule to the thermal situation of individual chips is proposed. In the design phase, based on the cores' temperature deviation statistics, a schedule tree is generated. The schedule tree is designed to offer a short expected test application time and a large likelihood that the tests are applied at the correct temperatures. During the test, based on the actual temperatures of its cores, the schedule that is best suited for the particular chip under test is used.

The rest of the paper is organized as follows. Section II discusses the necessity of multi-temperature testing and then provides an overview of the thermal issues concerning SoC test scheduling as well as related work. Section III gives a motivational example focusing on thermal consequences of process variation and exemplifying the proposed approach. Section IV explains the proposed method. Section V presents the experimental results and section VI concludes the paper.

II. Background and Related Work

A. Temperature-dependent Defects

Temperature-dependent defects are a subclass of well-studied environment-sensitive defects. Environment-sensitive defects are an important contributor to parametric failures, especially for deep submicron technologies. Temperature is an important environmental parameter along with some other parameters like supply voltage and frequency [1–3].

An example for such an environment-sensitive defect is a resistive open which is a major cause of test escapes [1]. It occurs when a connection between two circuit nodes has a conductance high enough to be considered as connected at normal temperatures. But at high temperatures the conductance decreases so much that the connection is considered as disconnected. This may occur since usually most of interconnects on the chip are made from metals and usually the conductance of those metals has negative temperature coefficient. Therefore, it is expected that a large number of such defects appear at high temperatures. On the other hand, we have other defects that manifest themselves differently with respect to temperature. For example, in [1] a defect ("Dark Via") is reported that "had previously passed all production tests, but then failed a monitor test at cold temperature". Several other defects are also identified in [1] that similarly appear only at low temperatures.

Beside the temperature coefficient for conductivity of the material, thermal expansion may also contribute to temperature-dependent defects [1, 2]. The "Dark Via" defect, which appears at low temperature, could be seen as voids between interconnect and via [1]. This observation could be explained with thermal expansion in metals (it fills up the voids and increases the conductivity) [2, 4]. This effect is illustrated in Fig. 1, where large voids at low temperature shrink at high temperature because of thermal expansion. Therefore, the conductance of the via may increase albeit the reduced conductivity of the via's

1063-9667/14 $31.00 © 2014 IEEE
DOI 10.1109/VLSID.2014.13

constructing material. Similar defects also exist for other technologies and materials. For example, some defects for a different technology are studied in [5] and "interface voids" are mentioned along with "sidewall voids" and "bulk voids" (shown also in Fig. 1) as temperature-dependent defects. Moreover, similar to possible temperature-dependent mechanisms for open defects, one may think of temperature dependent mechanisms for short or bridging defects.

Another type of temperature-dependent defect that is hard-to-diagnose is silicide open [6]. Silicide is used to make local interconnects. In its perfect condition, such a local interconnect has a positive temperature coefficient for resistance, but a defective one will have it as negative [6]. Detecting such defects at normal temperature is difficult since their difference is not recognizable. In order to provide good defect coverage, a simple diagnosis solution is necessary. Performing the test at low temperatures is suggested in [6] as a good solution since there will be a recognizable difference between the perfect and the defective chips at low temperature.

Resistive-open and stuck-open defects are experimentally studied in [7]. The resistive-opens occur more frequently (39 samples) compared to stuck-open defects (11 samples) [7]. The effects of the environmental parameters are studied and some diagnosis schemes are proposed in [7]. It is concluded that by knowing the location of the defects and the materials involved in those defects, the proper test temperatures can be found and the appropriate test patterns can be generated [7]. Such test temperatures and test patterns are the inputs to the adaptive test scheduling method proposed in this paper.

The behavior of temperature-sensitive defects is also analyzed experimentally in [8]. Some effective screening methods are proposed in [8] based on the comparison of test responses at multiple temperatures and low cost screening alternatives are proposed. Temperature-sensitive defects are expected to become more frequent in future technologies and therefore it is important to develop effective test methodologies for them [8]. The adaptive test scheduling approach that is presented in this paper will facilitate mass application of tests that are designed for temperature-sensitive defects.

The detection approaches for resistive bridging (short) defects are studied in [9, 10]. It is suggested in [9] that low-temperature and low-voltage tests improve the test quality. It is suggested that there exist appropriate combinations of these tests that provide satisfactory test coverage for different types of defects.

The effect of test temperature on the quality of the tests is studied in [11]. A low cost test methodology which utilizes low-voltage and low-temperature testing is also proposed in [11]. Moreover, the method proposed in [11] determines the appropriate test conditions for the best test quality and lowest cost.

Interconnect malfunctions (e.g., opens and shorts) are not the only sources of temperature-dependent defects; transistor malfunctions are also a source of concern. This issue is studied in [12] and the impact of temperature is demonstrated. The thermal behavior of a transistor depends on its quiescent point and therefore higher or lower temperatures, per se, do not imply better or worst results. Usually, in order to minimize the effect of the temperature, transistors are biased at the Zero-Temperature-Coefficient (ZTC) point. ZTC is a point such that the temperature will not affect the transistors. The problem is that there will be variations in the actual quiescent points of the manufactured transistors and therefore temperature will affect them. It is concluded in [12] that multi-temperature test will provide a significant improvement in test resolution.

There is yet another important aspect of temperature-dependent defects, namely the reliability issue. Some imperfections in the chip (e.g., some resistive opens or shorts) will not hinder the normal operation of the chip just after the fabrication, at the time that the manufacturing test is performed. But these imperfections are reliability threats because they are weak points in the circuit that wear out quickly and will lead to failures during the expected lifetime of the chip [1, 12]. These imperfections can be identified by multi-temperature testing.

Figure 1. Voids in a via create a resistive open. (a) Large voids at low temperature. (b) At high temperature, materials expand and voids shrink.

We have discussed temperature-dependent defects and the necessity of multi-temperature testing. The test application time for multi-temperature testing is much longer than the normal testing and therefore the test cost is higher. This becomes a serious cost issue, in particular in situations that the normal test application time is already very long, as it is for Systems-on-Chip (SoC).

B. Thermal Issues Concerning SoC Test Scheduling

SoCs are made of a number of cores integrated on a single chip. The number of cores is usually large and the mechanism that provides access to the cores for testing is a limiting factor for the test parallelism and speed. Apart from the restrictive test access mechanisms, the high test temperature is usually another issue which limits the test speed (in order to assure thermal-safe test). As a result, SoC test takes a very long time and consequently it is very costly. Therefore, efficient SoC test schedules should be designed to minimize the test application time, in particular in the case of multi-temperature test.

A multi-temperature test scheduling for SoCs is introduced in [2] which assumes that tests should be applied in predefined temperature ranges. The proposed scheduling approach in [2] minimizes the test application time and ensures that tests are only applied within the valid temperature ranges. For this purpose the temperatures of the cores are simulated. Based on the simulated temperatures, heating or cooling intervals are introduced into the schedule [2]. The method proposed in [2] is based on partitioning and interleaving and therefore when a core is having its cooling interval, other cores may utilize the test access mechanism's capacity that has been just made available.

Another temperature-dependent test scheduling scheme for SoCs is introduced in [3]. It assumes also that tests should be applied in different specified temperature ranges. The proposed scheduling approach in [3] is based on list scheduling and assumes that tests run always to completion without any interrupts. The initial list order is determined based on the lowest valid temperatures for the tests. The list schedule determines the optimal earliest start times for tests. The test application time is minimized and it is ensured that tests are applied within correct temperature ranges [3].

The proposed methods in [2, 3] provide satisfactory results when the temperature at a certain test cycle could be assumed to be identical for all chips of the same design. However, chips manufactured with deep submicron technologies are likely to have different temperatures at the same test cycle because of process variation. Process variation includes variations in the geometry of the chips' components and variation in the properties of the chips' materials. For example, the effective channel length may vary and result in variation of the threshold voltage and sub-threshold leakage. These variations will result in differences in the leakage current which is an important contributor to the overall chips' power consumption. Consequently, the chips will experience power and temperature variations [13].

The negative effect of temperature variations on the thermal safety of the SoCs during test is addressed by the scheduling methods proposed in [14–16]. These methods try to limit the cores' maximum temperatures so that the test damages caused by overheating during the test process are minimized. The method proposed in [14] addresses the temperature variation issue by assuming that the variation is only observed from chip to chip (inter-die process variation). Moreover, in [14] it is assumed that the chips' temperature differences compared to a reference chip do not vary with time. This assumption is relaxed in [15, 16] where the temperature is also considered to be varying from core to core (intra-die process variation). An overview of test methods that address the negative consequences of process variation on the test is given in [17]. However none of these methods addresses multi-temperature testing.

In this paper, the negative effect of intra-die and time-variant temperature variations on multi-temperature testing is addressed with an adaptive test scheduling scheme. In order to acquire the actual cores' temperatures recurrently during the test, on-chip temperature sensors (which are usually present on the chip for its normal operation) are utilized. Modern chips are usually equipped with multiple temperature sensors (e.g., IBM's Power5 processor is reported to have 24 temperature sensors already in 2004 [18]).

On-chip temperature sensors are also used for the thermal-safe scheduling approach proposed in [19], which reduces the test application time compared to a static schedule. A heuristic is suggested to generate the static schedule that the method is based on. However, the proposed method in [19], during the test, requires numerous thermal simulations and sensor readouts and assumes that tests run to completion without interrupts, which is a very restrictive assumption.

There exist already efficient methods for multi-temperature SoC test scheduling, for example methods proposed in [2, 3]. Also, there exist methods to cope with the negative effects of process variation for thermal-safe SoC test scheduling, for example the approaches proposed in [14–16]. But, to our best knowledge, there exists no method for multi-temperature SoC test scheduling that effectively handles the thermal consequences of process variation. This problem is clarified in the next section and is the focus of this paper.

III. MOTIVATIONAL EXAMPLE

Let us consider a test that should be performed in a temperature range specified by an *Upper limit* and a *Lower limit*, as shown in Fig. 2. Assume that during the application of this test, no heating interval is required, and therefore, as is shown in Fig. 2, only testing and cooling intervals are sufficient to keep the temperature within the specified range. Assume that two chips, C_n and C_w, of a given design are subject to this test.

When the process variation is negligible, there is no noticeable temperature difference between C_n and C_w as shown in Fig. 2a. Therefore, a single schedule will work fine for both of them. The schedule that is generated based on thermal simulations predicts when the proper time to apply the test is and when it is necessary to introduce a cooling interval. Scheduling methods for this situation (when the process variation is negligible) are proposed in [2].

The effect of process variation on the chips' temperatures could be seen as temperature deviations. For example, as shown in Fig. 2b, the two chips initially have identical temperatures, but after time i_3, while C_n continues to work as normal, C_w becomes warmer than normal. Consequently, the test for C_w is applied out of the valid temperature range. In order to prevent such violation, a straightforward solution is to use a more conservative schedule which assumes a smaller upper limit and a larger lower limit (i.e., a narrower temperature range compared to the specified range). This solution does not require a new scheduling algorithm; the existing algorithms can be supplied with the modified upper and lower limits. The schedules generated this way are called *static* in this paper.

For example, assume that the original valid range is from 70℃ to 100℃. In order to make sure that even the deviated chips will be tested within the specified temperature range, the static scheduler assumes a valid range from 80℃ to 90℃. Now, as an example, even a chip that is deviated by −10℃ from what it should be, is tested actually between 70℃ to 80℃, which is within the original specified ranges. The problem with static schedules is that they require too many longer heating and cooling intervals and consequently their test application times will be very long. For example, a longer heating interval will be needed to heat the chips up to 80℃ instead of 70℃ before starting the actual test.

The balance between the test application time and the thermal range violations is more delicate for multi-temperature testing (this paper) compared to thermal-safe testing (e.g., [15]). In the case of thermal-safe test, there exists only an *Upper limit*. Therefore, it is possible to have a conservative schedule similar to S_2 (Fig. 2g) that provides a safe test for a large number of chips (C_w and all other chips with lower temperatures, including C_n). This safety comes with a longer test application time. For example, there are two longer cooling intervals after time i_4 in Fig. 2c compared to a single shorter cooling interval in Fig. 2a. Therefore, the test application time for Fig. 2c (j_4) is longer than Fig. 2a (i_7).

However, unlike the thermal-safe testing, in the case of multi-temperature testing, S_2 will not provide an in-range test for a large number of chips. For example, the temperature of C_n will fall below the *Lower limit* if tested with S_2 (e.g., around j_1 and j_3). Therefore, a test schedule that works for a large number

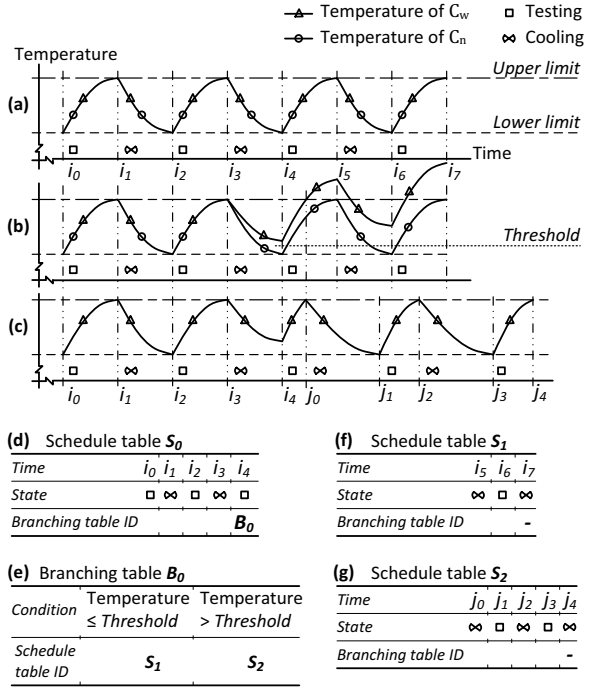

Figure 2. Test temperatures and schedules (plots are only illustrative). (a) There is no temperature variation. (b) There is time-variant temperature variation and C_w is not completely tested within the specified range. (c) Initially, test is performed using the schedule table S_0 (d), then after i_4 by referring to the branching table B_0(e), test of C_n continues with the schedule table S_1 (f); but unlike (a,b), test of C_w continues with the dedicated schedule table S_2 (g), which is longer but assures that tests are applied within the valid temperature range.

of chips, similar to the thermal-safe case, should be also conservative with respect to the low temperatures. This means that the test application time will be longer than j_4 in order to support a similar amount of chips tested within the valid thermal range.

The solution proposed in this paper is to test C_n and C_w up to time i_4 using a single schedule table S_0 (Fig. 2d). Just before C_w exceeds the *Upper limit*, temperature of the chip under test is acquired through a temperature sensor. Then, by referring to the branching table B_0 (Fig. 2e) and the comparison performed with a *Threshold* temperature, it will be known which chip (C_n or C_w) is actually under the test. The test for C_n continues as normal with schedule S_1 while C_w continues the test with a longer schedule, S_2 . This way, the test is performed within the specified temperature range for all chips and the overall test application time is kept as short as possible.

This type of schedule that includes branching and schedule tables is called a schedule tree. The schedule tables could be considered as the tree's edges that are connected through nodes that are the branching tables. During the test, such a schedule tree is used to guide the test process by indicating the appropriate times to test, to heat up, and to cool down. The schedule might be conservative (e.g., S_2 in Fig. 2g) or fast (e.g., S_1 in Fig. 2f). The schedule that is actually used (a path in the tree) is selected gradually during the test in accordance with the actual thermal situation of the cores. Therefore, such a scheduling scheme is an adaptive approach based on the temperature sensor readings which happen at nodes. The gaps between the sensors readouts (i.e., edges) are filled using thermal simulations to predict the temperatures. The efficiency of such an adaptive approach depends therefore on the quality of the schedule tree. We propose a technique to generate high-quality schedule trees in the next section.

IV. Adaptive Multi-temperature Test Scheduling

In order to generate a high-quality schedule tree, a cost function is required (section IV.A). Then, an approach to quantify the temperature variation is needed in order to be able to evaluate its effect on the cost (section IV.B). A heuristic is also needed to generate the schedule tables (section IV.C). Finally, a constructive method is used to generate the schedule trees (section IV.D).

A. Cost Function

The cost function that captures the costs related to (1) test application time, (2) tests that are not applied within valid temperature ranges, and (3) overheating, is defined as

$$CF = \left[\frac{TAT}{ATS} + BC_{oor} \times ORP + BC_{oht} \times TOP\right]/(1 - TOP)\ . \quad (1)$$

In the cost function, TAT stands for Test Application Time and ATS represents Applied Test Size. The first term of the cost function, TAT/ATS, captures the test efficiency which is related to the test application time divided by the applied test size. It indicates how much time is needed to apply a unit amount of test patterns.

There is a valid temperature range for every test and the tests that are applied outside that range may suffer from substantially lower effectiveness. This means that defective chips may pass the test (test escape). The negative impact caused by such undetected defects is included in the cost function considering the probabilities of those defects. The 2nd term of the cost function, $BC_{oor} \times ORP$, represents this cost based on the Out of Range Probability (ORP) that is the probability of tests being applied outside the specified thermal ranges. BC_{oor} is the Balancing Coefficient that balances the cost of the chips that are tested Out-Of-Range (their test is applied when their temperature is not within the specified range) against the rest of the cost components. Its value is larger for costlier test escapes.

The cost related to overheated chips is captured in the 3rd term of the cost function based on the Test Overheating Probability (TOP). BC_{oht} is the Balancing Coefficient for OverHeaTing that is used to balance the costs originated from overheating against the other costs. Its value will be larger for costlier chips.

All the three terms of the cost function should be computed with respect to the chips that are not overheated (overheated chips will be discarded). Therefore, they should be defined per non-overheated chip and therefore the sum of these terms is divided by $(1 - TOP)$.

A test schedule that is not conservative (e.g., S_1 in Fig. 2b) will typically have a smaller TAT and a larger ORP. If it happens that the temperature goes even beyond the thermal-safety limit (larger than the *Upper limit*), then the violation is counted as overheating and it will contribute to a larger TOP. It means that a chip could be overheated or out-of-range but not both (these are two mutually exclusive events). On the other hand, a more conservative test schedule (e.g., S_2 in Fig. 2c) will typically offer a smaller ORP and TOP, but a longer TAT. The overall cost that is computed by the cost function will determine which situation is better globally. In practice the coefficients in the cost function (BC_{oht} and BC_{oor}) are obtained based on the knowledge of the specific chips and the test facility.

B. Temperature Variation Model

The temperature variations are quantified based on the concept of temperature errors. A temperature error is defined as the difference between the actual temperature (measured using a temperature sensor) and the expected temperature. The expected temperatures are temperatures without any variations or errors (could be estimated by thermal simulation). The temperature errors have different causes, including process variation, voltage variations, ambient temperature fluctuations, and simulator imprecision. Since either the complete information about these factors is unavailable or they are random in nature, the temperature errors can only be modeled stochastically. The model used in this paper is meant to be general and not exclusive to certain types of variations. This error model is similar to the temperature error model used in [16].

The temperature error model estimates the error for each core at each test cycle (the probabilities of different error values are provided by a model). For example in Fig 2b, the error for C_w at time i_7 is larger than its error at time i_5 and the error for C_n is zero for all times. This information could be put in a stochastic framework. In this case it could be said that for a population of chips (C_n and C_w) the expected error value at time i_7 is larger than at time i_5 and it is zero before i_3. An example of a time-dependent phenomenon supported by this model is the observation that after sensor readouts, as time goes by, the information about temperature becomes less precise.

The temperature variation model is provided as an input to the adaptive scheduling approach. Based on the data from the temperature variation model, the stochastic values that are used in the cost function are calculated.

C. Schedule Table Generation

The role of a schedule table is to determine for each core when to run the test, when to pause and when to apply the heating sequence. The heuristic that is used to generate the schedule tables is described briefly in the following. The cores with lower temperature and longer tests have the priority to start the test before the others. As many cores as the test access mechanism allows are tested in parallel. The test of a core stops as soon as its temperature goes beyond the *Upper limit* or falls below the *Lower limit*. This frees the resources that previously were occupied by these cores from the test access mechanism. In case the test access mechanism allows, some other cores that satisfy the thermal constrains will start/resume their tests. Again, cores

with lower temperatures and longer remaining tests are given higher priority to start/resume before the others.

If the temperature of a core that is selected to start/resume its tests is lower than the *Lower limit*, a heating interval starts. The heating interval continues until the core's temperature surpasses the *Lower limit* or the Middle temperature, Θ_M, depending on the average power of the following test cycles. The middle temperature, Θ_M, is located halfway between the *Lower limit* and the *Upper limit*. If the average power of the following tests is larger than the THresholD Power, P_{THD}, the heating continues to (and stops at) the *Lower limit*, since the temperature will further increase when the tests are applied. Otherwise (if the average power of the following tests is smaller than or equal to P_{THD}) heating should be stopped when the temperature reaches Θ_M, since the temperature will not significantly increase when the following tests are applied.

The threshold power, P_{THD}, is defined as the power that results in a steady state temperature equal to the middle temperature, Θ_M. An iterative method could be used to estimate the P_{THD} based on thermal simulations as in [2]. Instead, an analytical solution which is much faster is used in this paper. Assume that $\boldsymbol{P}_{THD}$ is the power vector that should be found and $\boldsymbol{\Theta}_M$ is the middle temperatures vector. The mathematical representation of the thermal model used in this paper is a system of ordinary differential equations as shown below.

$$\boldsymbol{A} \times \frac{d}{dt}\boldsymbol{\Theta} + \boldsymbol{B} \times \boldsymbol{\Theta} = \boldsymbol{P} \quad (2)$$

The properties of the thermal model (e.g., thermal capacitance and conductance) are encapsulated into matrices $\boldsymbol{A}$ and $\boldsymbol{B}$. The temperatures are represented by $\boldsymbol{\Theta}$ and the powers by $\boldsymbol{P}$. Since variations in the temperatures are negligible for the steady state solution ($\frac{d}{dt}\boldsymbol{\Theta} \cong 0$), assuming that $\boldsymbol{P}_{THD}$ results in $\boldsymbol{\Theta}_M$, equation 2 could be rewritten as

$$\boldsymbol{P}_{THD} = \boldsymbol{B} \times \boldsymbol{\Theta}_M \,. \quad (3)$$

This way, the threshold powers are computed without time consuming iterative thermal simulations which are used in [2].

In case the core's temperature is higher than the *Upper limit*, a cooling interval is introduced. The core cannot resume its test until its temperature sinks below a Special temperature, Θ_s. The value of Θ_s has a considerable impact on the test application time and should be selected carefully. In this paper, Particle Swarm Optimization (PSO) is used to find the proper values for Θ_s for each test so that the total test application time (taking all cores into account) is minimized.

The schedule table generation algorithm takes the tests for the cores, the corresponding thermal limits (e.g., *Upper limit* and *Lower limit*), the threshold powers, $\boldsymbol{P}_{THD}$ (computed using equation 3), and the special temperatures, $\boldsymbol{\Theta}_s$ (found using PSO), as inputs. An example for such inputs is given in Fig. 3. This algorithm generates the schedule tables according to the principles discussed earlier. A schedule table constitutes an edge in a schedule tree and therefore the process of schedule tree generation extensively uses the schedule table generation algorithm. This will be further explained in the next section.

D. Schedule Tree Generation

A constructive approach is used to generate the schedule trees. In each step, small partial trees are added to the leaves of the current incomplete schedule trees (at the very beginning, the current incomplete tree is just a node). The incomplete trees keep growing until all of the tests are scheduled.

Test	Core	*High limit*	*Low limit*	$\boldsymbol{\Theta}_M$	$\boldsymbol{P}_{THD}$	$\boldsymbol{\Theta}_s$
1	1	95	85	90	62.17	90.25
2	1	75	65	70	42.48	74.17
3	2	85	75	80	52.34	81.80

Figure 3. An example for test thermal specifications. The temperatures are in Celsius and the powers are in Watt.

The small partial trees are dedicatedly designed for the growing tree's leaves that they are going to be fused to (i.e., position that they take in the final schedule tree). The initial data that a partial tree inherits from the leaf at fusion point includes the cores' temperatures and states (testing, cooling, or heating), the remaining tests, and the temperature deviation probabilities at the fusion point. This information enables the partial tree to be scheduled independently. When a partial tree is being scheduled, the final cost value for the complete tree could only be estimated (based on equation 1). For example, the test application time is only known for the partial tree, since the final schedule tree is not completed yet.

As mentioned before, the scheduling of the partial trees is performed at two separate levels, at one level the topologies are investigated and at the other level the scheduled partial trees are evaluated. In the beginning, a range of topologies for partial trees is considered. For every topology, the best scheduled partial tree is found using Particle Swarm Optimization (PSO). A number of alternative partial trees are evaluated at every step by PSO based on a partial cost function.

As mentioned earlier, an edge in a schedule tree is generated using the schedule table generation algorithm. In order to take the effect of the temperature variation into account, an exclusive representative temperature error (could be thought as the expected error) is used to estimate the core's current temperature that is used by adaptive thermal-safe method, similar to [16]. The representative temperature errors work in the same manner as safety margins that are introduced in [14] in order to enforce the temperature limit. Unlike the thermal-safe testing, the multi-temperature testing should satisfy a *Lower limit* in addition to the *Upper limit* and thus a single representative temperature error is not sufficient. Therefore, in this paper two representative temperature errors are used to generate a single schedule table, one to represent the warmer cores and the other to represent the colder cores. The warmer representatives help to enforce the *Upper limit* and the colder representatives help to enforce the *Lower limit*.

The probability that tests are applied within the specified temperature ranges is computed using the statistics provided by the temperature variation model. All the steps in the procedure of generating the schedule tree are guided by the cost function (equation 1) and therefore the expected test application time and the probability of the out-of-range tests are minimized.

V. EXPERIMENTAL RESULTS

The proposed adaptive method is evaluated and is compared with a modified state-of-the-art static method [2]. The experiments are performed on ten chips which have five to fifty cores. Multiple experiments are performed for each chip. The averages of the results obtained from these experiments are the basis for comparisons reported in this section. The experiments are based on models of the SoCs and their tests. The lateral heat transfer among cores and the temperature-dependent static powers are taken into account. The switching activities are generated using Markov chains similar to [19] with random averages and random lengths. The temperature ranges for tests are generated randomly. The balancing coefficients for overheating (BC_{oht}) and for out-of-range test (BC_{oor}) are set to 10. This does not mean that the cost of an overheated chip is equal to the cost of a test escape, since BC_{oor} also includes the probability of the defects.

The percentage changes achieved by the adaptive method compared to the static method are reported in Table I. In average, 71% reduction in cost is achieved. This cost reduction comes from the reduced test application time (20% in average)

TABLE I. PERCENTAGE CHANGES ACHEIVED BY ADAPTIVE METHOD COMPARED TO A STATIC METHOD

Chip	Percentage Change					
Number of Cores	*Cost*	*Required ATE Memory*	*Test Application Time*	*Overheating Probability*	*Out-of-Range Probability*	*CPU Time*
5	-84.4	65.1	-29.4	-84.6	-75.4	3240.0
10	-46.5	54.5	-9.8	-6.2	-65.9	4913.3
15	-34.2	103.8	-7.0	-1.1	-43.6	3364.9
20	-67.4	67.9	-22.4	-8.1	-5.7	2081.3
25	-82.7	79.1	-27.3	-2.0	2.3	1078.8
30	-84.6	40.2	-23.2	-0.4	-47.3	1393.8
35	-89.1	119.6	-15.7	-0.0	1.8	848.8
40	-53.5	57.0	-20.5	-1.6	5.2	2122.2
45	-83.2	79.4	-19.9	-0.1	-0.3	1283.7
50	-89.8	12.0	-23.4	-0.6	-0.8	1010.4
Average	-71.5	77.9	-19.9	-10.5	-23.0	2133.7

combined with the reduced overheating probability (10%) and reduced probability of tests being applied out of the specified temperature range (23%). These improvements are achieved at the cost of increased ATE memory (78% in average), since the adaptive schedule trees are larger than the static schedules. Moreover, the CPU time that is required to generate the adaptive schedules is larger than the time that is required to generate the static schedules (21.34 times in average), which are based on a fast on-the-fly scheduling heuristic.

The adaptive method always offers a lower cost, even if some of the parameters that contribute to the cost become actually worse compared to the static schedule. For example, for the experimental chip with 40 cores (8th row in Table I) the out-of-range probability has actually increased by 5.2%. Even though this increase, on its own, increases a particular cost component, but the reductions achieved by the reduced test application time (20.5%) and the reduced overheating probability (1.6%) are dominating and the overall cost decreases by 53.5%. Since the increases in the CPU times are large, it is necessary to make sure that the CPU time remains affordable for chips with large number of cores. Therefore, a fast simulation approach similar to [16] is used in this paper. The average CPU time is plotted in Fig. 4 for the experimental chips, which shows that the growth rate is in an acceptable range.

We have also studied the effectiveness of the proposed method in reducing the test cost for different process-variation situations, and the results are illustrated in Fig. 5. The vertical axis is the test cost (equation 1) for a chip with 15 cores. The horizontal axis is the variance of the Gaussian distribution that is assumed to characterize the process variation in this experiment. For large variations, the static method is incapable of applying the tests within the specified thermal limits and consequently the test cost is very large. Our adaptive method offers a substantially lower cost for all non-zero variations and the saving keeps on growing for larger variations, which are expected for the future technologies.

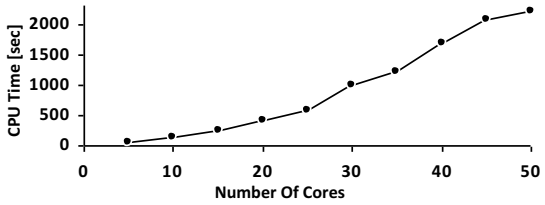

Figure 4. CPU (Design) times for the adaptive method.

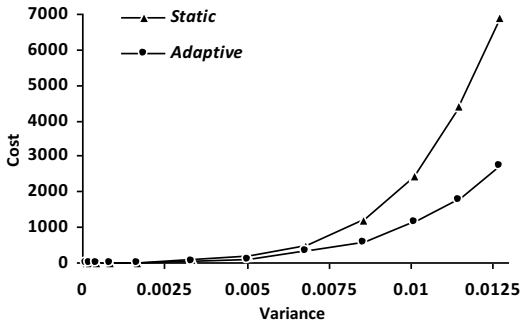

Figure 5. Cost versus variation.

VI. CONCLUSIONS

Temperature-dependent defects and thermal consequences of process variation are two important challenges for the test of core-based chips that have to be addressed as the technology scales down to the deep submicron domain. This paper presents an adaptive test scheduling approach to deal with these coinciding issues, simultaneously. The proposed method, based on thermal simulations in the design phase, generates a number of efficient test schedules, each corresponding to a different intra-die time-dependent temperature error situation. These schedules are put together in a schedule tree and are stored to be used during the test. During the test, the actual cores' temperatures are monitored using on-chip temperature sensors and the chip under test is tested using the most compatible test schedule (a selected path from the root to one of the leaves in the schedule tree).

The proposed approach reduces the test costs by decreasing the average test application time and by increasing the likelihood that the tests are applied at the correct temperatures. The cost reduction is demonstrated by experiments on a number of chips having up to fifty cores. The experiments indicate that the test application time could be reduced about 20% in average, the probability of tests being applied out of valid temperature range could be reduced about 23% in average, and the overheating probability could be reduced about 10% in average.

REFERENCES

[1] W. Needham, C. Prunty, and E. H. Yeoh, "High volume microprocessor test escapes, an analysis of defects our tests are missing," ITC, 1998.

[2] Z. He, Z. Peng, and P. Eles, "Multi-temperature testing for core-based system-on-chip," DATE, 2010.

[3] C. Yao, K. K. Saluja, and P. Ramanathan, "Temperature dependent test scheduling for multi-core system-on-chip," ATS, 2011.

[4] J. Segura and C. F. Hawkins, "CMOS electronics: how it works, how it fails," Wiley-IEEE Press, 2004, pp. 305.

[5] E. Zschech, E. Langer, H-J. Engelmann, and K. Dittmar, "Physical failure analysis in semiconductor industry—challenges of the copper interconnect process," Materials Science in Semiconductor Processing, 2002, vol. 5, no. 4–5.

[6] C. W. Tseng, E. J. McCluskey, X. Shao, J. Wu, and D. M. Wu, "Cold delay defect screening," VTS, 2000.

[7] J. C.-M. Li, C. W. Tseng, and E. J. McCluskey, "Testing for resistive open and stuck open," ITC, 2001.

[8] C. Schuermyer, J. Ruffler, R. Daasch, and R. Madge, "Minimum testing requirements to screen temperature dependent defects," ITC, 2004.

[9] P. Engelke et al., "On detection of resistive bridging defects by low-temperature and low-voltage testing," IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, 2008, vol. 27, no. 2.

[10] S. Kundu, P. Engelke, I. Polian, and B. Becker, "On detection of resistive bridging defects by low-temperature and low-voltage testing," ATS, 2005.

[11] L. Jagan et al., "Impact of temperature on test quality," VLSID, 2010.

[12] E. Long, W. R. Daasch, R. Madge, and B. Benware, "Detection of temperature sensitive defects using ZTC," VTS, 2004.

[13] J.H. Choi, J. Murthy, and K. Roy, "The effect of process variation on device temperature in finFET circuits," ICCAD, 2007.

[14] N. Aghaee, Z. He, Z. Peng, and P. Eles, "Temperature-aware SoC test scheduling considering inter-chip process variation," ATS, 2010.

[15] N. Aghaee, Z. Peng, and P. Eles, "Adaptive temperature-aware SoC test scheduling considering process variation," DSD, 2011.

[16] N. Aghaee, Z. Peng, and P. Eles, "Process-variation and temperature aware SoC test scheduling technique," Journal of Electronic Testing, 2013, vol. 29, no. 4.

[17] E. Yilmaz, S. Ozev, O. Sinanoglu, and P. Maxwell, "Adaptive testing: conquering process variations," ETS, 2012.

[18] J.G. Clabes et al., "Design and implementation of the POWER5 microprocessor," DAC, 2004.

[19] C. Yao, K. K. Saluja, and P. Ramanathan, "Thermal-aware test scheduling using on-chip temperature sensors," VLSID, 2011.