Temperature-Gradient-Based Burn-In and Test Scheduling for 3-D Stacked ICs

Nima Aghaee, Zebo Peng and Petru Eles

Linköping University Post Print

N.B.: When citing this work, cite the original article.

©2015 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

Nima Aghaee, Zebo Peng and Petru Eles, Temperature-Gradient-Based Burn-In and Test Scheduling for 3-D Stacked ICs, 2015, IEEE Transactions on Very Large Scale Integration (vlsi) Systems, (23), 12, 2992-3005.

<http://dx.doi.org/10.1109/TVLSI.2014.2380477>

Postprint available at: Linköping University Electronic Press

<http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-123487>

Temperature-Gradient Based Burn-In and Test Scheduling for 3D Stacked ICs

Nima Aghaee, Zebo Peng, and Petru Eles

Abstract—Large temperature gradients exacerbate various types of defects including early-life failures and delay faults. Efficient detection of these defects requires that burn-in and test for delay faults, respectively, are performed when temperature gradients with proper magnitudes are enforced on an Integrated Circuit (IC). This issue is much more important for 3D stacked ICs compared with 2D ICs because of the larger temperature gradients in 3D stacked ICs. In this paper, two methods to efficiently enforce the specified temperature gradients on the IC, for burn-in and delay-fault test, are proposed. The specified temperature gradients are enforced by applying high power stimuli to the cores of the IC under test through the test access mechanism. Therefore, no external heating mechanism is required. The tests, high power stimuli, and cooling intervals are scheduled together based on temperature simulations so that the desired temperature gradients are rapidly enforced. The schedule generation is guided by functions derived from a set of thermal equations. Experimental results demonstrate the efficiency of the proposed methods.

Index Terms—3D Stacked IC test, burn-in, temperature gradients, test scheduling

I. INTRODUCTION

Large temperature gradients (e. g., temperature difference between two adjacent cores) exacerbate various types of defects including early-life failures and delay faults. The capability to detect these temperature-gradient induced defects is crucial for many ICs. In particular, three dimensional ICs exhibit considerably larger temperature gradients compared with normal ICs (for example, three times is reported in [29]) and therefore temperature-gradient based test is necessary for them.

A promising technology for fabricating 3D ICs is based on Through-Silicon Vias (TSV) used for inter-die connections [11], [13], [17], [27]. The ICs fabricated using TSVs are commonly referred to as 3D Stacked IC (3D-SIC) [17]. The important advantages of this technology include high inter-die interconnect densities and low inter-die interconnect wire lengths. This leads to higher operating frequencies at lower power consumptions.

A. Test for Early-Life Failures

Burn-in is a common way of accelerating and detecting early-life failures and it should be done with low cost in a reasonably short time. For this purpose, usually the dies are operated at elevated temperature and voltage. The elevated temperature and voltage speed up the aging and wear mechanisms so that the dies experience their early life before testing. The wear mechanisms that are speeded up include metal stress voiding and electromigration, metal slivers bridging shorts, as well as gate-oxide wear-out and breakdown [23].

Recently, several studies have, however, shown that some wear mechanisms are speeded up more efficiently by large temperature gradients rather than the high temperature itself. A temperature-gradient induced wear mechanism is identified in [25] which shows that a metal layer elevation develops rapidly on the sites that experience large temperature gradients. Moreover, in the atomic flux equation that models the electromigration, temperature gradient is present directly and also indirectly through its effect on the mechanical-stress gradient [21]. Therefore, a burn-in process that has not created the appropriate thermal scenarios does not sufficiently speed up the formation of the defects and, consequently, such early-life defects will go undetected. In order to prevent these test escapes, it is necessary to introduce a burn-in process that enforces appropriate temperature scenarios on the IC. This necessity is more urgent for the ICs that suffer from large temperature gradients, such as 3D-SIC.

3D-SIC technology, similar to other deep submicron technologies, suffers from high power densities. Additionally, power densities are considerably higher in the test mode compared to the functional mode, in particular for core-based designs [6], [33]. Consequently overheating may damage the ICs under test [2], [19], [32]. This means that the application of test stimuli to ICs can raise their temperatures beyond their tolerable limits. This often undesirable effect is, however, utilized in this paper to heat up the IC for burn-in. In our case the stimuli are not necessarily actual test patterns. Instead, they could be specially generated sequences which cause large switching activities. Such stimuli are called heating sequences. The use of the heating sequences to heat up the IC from inside means that special equipment for heating the IC from outside are not necessary. This will lead to large reduction of cost, and also allow for the generation of needed temperature gradients.

Some temperature gradients might be enforced on an IC by applying appropriate inputs to the IC's input ports in the functional mode. This might work, to some extent, for 2D ICs, since from the functional point of view all the required circuitry, including the input ports, are there when the IC enters the test process. For 2D ICs, there are usually two possible stages for burn-in: Wafer-Level Burn-In (WLBI) which is performed before packaging and Die-Level Burn-In (DLBI) performed after packaging [23]. For 3D-SIC, there are more stages, including pre-bond, mid-bond, post-bond, and final stages [26]. At different stages, different defects can be targeted, based on their likelihood and considering the corresponding burn-in costs. A 3D-SIC, in addition to the defects that also exist for 2D, is affected by TSV related defects (e.g., defects related to TSV bonding). Such defects motivate, among other reasons, the use of these extra stages.

Existence of the test stages before the IC is fully assembled is a key difference between the 2D and 3D-SIC burn-in process. In the case of 3D-SIC, using input ports in the functional mode

may benefit burn-in for the post-bond and the final stages similar to 2D ICs. But for the pre-bond or mid-bond stages, the inputs to the die or partially stacked dies are not necessarily the inputs to the IC. The input ports to the unit under test for 3D-SICs, before the final bonding, are likely to include a number of TSVs. The TSVs and test equipment are not designed to support simultaneous application of functional signals, particularly to large number of TSVs (even though they might be designed to allow simple electrical tests for the TSV itself). Therefore, the use of the IC's ports for enforcing the temperature gradients is not possible for the pre-bond and mid-bond stages. Albeit this lack of access in the functional mode, the Test Access Mechanism (TAM) provides access to the cores, in the test mode [1]. Therefore, the heating sequences could be applied using the TAM in order to enforce the desired gradients.

B. Test for Delay Faults

3D-SIC and other deep submicron technologies suffer from a considerably larger number of delay faults as compared with previous technologies. The causes for these delay faults include resistive bridges and vias, power droops, and cross-talk noise effects [5]. Therefore, delay-fault testing is necessary to provide sufficient fault coverage [14]. A large number of pre-bond TSV defects are resistive in nature and, moreover, the mechanical stress caused by TSVs contributes also to delay faults [11], [13]. Therefore, the expected number of delay faults for 3D-SIC is larger than that of 2D ICs.

Since temperature has a significant effect on delay, its impact should be taken into account for delay-fault test. A very important effect of temperature on signal integrity is its effect on the clock network [8]. Delay faults usually occur because of increased clock skew and a major contributor to skew in 3D-SICs is temperature gradient [20]. Since propagation delays depend on temperature, different temperatures on different sites (i.e., temperature gradients) result in clock skew. Temperature gradients may reach up to 50°C in adjacent cores for normal operation and even higher during test [7], [8], [20]. Besides, as mentioned before, the temperature gradients in 3D-SICs are much larger than in 2D ICs [29]. This will exacerbate temperature-gradient related issues including delay faults, in particular, for 3D-SIC. Therefore, the associated tests should be performed when the proper temperature maps are enforced. A temperature map specifies the appropriate temperatures for different sites (e.g., cores) in the IC. These temperatures are to be realized simultaneously in order to enforce the proper temperature gradients. The temperature maps are given along with their corresponding tests. Beside the gradient-based burn-in (discussed in section I.A) the other objective of this paper is to introduce a technique to apply the tests while the corresponding maps are enforced on the IC.

II. RELATED WORKS

Traditionally, burn-in is performed at elevated temperature, which is achieved by special equipment (e.g., temperature chambers) [23]. A more elaborate technique is a hybrid burn-in and test technique that provides tighter temperature control using an active heat-sink, without using temperature chambers [18]. These existing techniques are not able to enforce the specified temperature gradients, especially those with large magnitudes.

Despite the fact that enforcing specific temperature gradients will facilitate the detection of delay-faults, existing methods for delay-fault testing focus only on performing the tests disregarding the temperature gradients [5]. Apart from gradients, creating and maintaining some other kinds of thermal conditions during the test has been addressed previously. These existing techniques are briefly reviewed in the following.

A thermal-aware test scheduling is introduced in [30] for stacked multi-chip modules. The method tries to achieve a vertical uniform temperature distribution throughout the 3D-SIC during the test. The method is based on analytical simplifications of the temperature model [30].

A linear programming approach is used in [19] in order to generate thermally-safe test schedules for 3D-SICs. The proposed method uses a super-position based temperature simulator [19].

Two different approaches for multi-core ICs are introduced in [15] and [31] to guarantee that the cores' temperatures are kept within the specified range when the corresponding tests are being applied. They focus on the temperature of the individual cores that are under test and the temperatures of other cores are neglected. Keeping the temperatures within the specified range is achieved by introducing heating sequences (high-power stimuli are applied) and cooling intervals (no stimuli are applied) into the test schedule.

Speeding up the test while minimizing the damages caused by overheating due to process variation is addressed in [2]. The test temperatures are kept sufficiently low by introducing cooling cycles. A number of test schedules are generated for different variation scenarios. During the test the proper schedule is selected based on the on-chip temperature sensor readouts (adapting to the current thermal variation situation). A fast temperature simulation technique that isolates heavy computations into a single initial phase is also suggested in [2].

The existing methods for controlling the chips' temperatures during test try to respect a global upper temperature limit to prevent overheating [19], [32] or to respect upper and lower bounds for individual cores in order to target temperature dependent defects [15], [31]. In both cases, the temperature bounds are defined for each core independent from other cores and therefore spatial temperature gradients cannot be enforced.

The first proposals to consider temperature-gradients in the burn-in and delay-fault test processes were made by us in [4] and [3], respectively, where the concept of temperature maps were introduced. Preliminary sketches of two simple algorithms to consider such temperature maps for burn-in and delay-fault test, respectively, were reported in [4] and [3]. This paper has built on the preliminary results of [3] and [4] and develops an integrated and systematic framework to address temperature-gradient issues in both burn-in and test processes. It presents also several advanced techniques to compute proper values for the test schedule's period, to order the temperature maps, and to generate high-quality temperature-gradient based burn-in solutions, which were outside the scope of [3] and [4].

The rest of the paper is organized as follows. Section III addresses the temperature-gradient based burn-in. Section IV addresses temperature-gradient based test. Section V suggests an efficient temperature map ordering technique. Section VI presents experimental results. Section VII concludes the paper.

III. TEMPERATURE-GRADIENT BASED BURN-IN

A. Preliminaries and Problem Formulation

As discussed earlier, a temperature map specifies the desired temperature values for different sites (e.g., cores) in an IC. The temperature maps are to be given by the user, who studies the typical temperature-gradient induced failure mechanisms analytically or experimentally [21], [25]. Each map corresponds to a particular temperature condition of an IC, such as large temperature differences between adjacent cores (i.e., large temperature gradients), that can accelerate aging for early-life failures or enlarge the delay fault effect so that they can easily be tested for. There might also be some sites that their temperatures are not important regarding the targeted defects. Such sites are indicated as don't-cares. Even though they are marked as don't-cares, their temperature should, however, be kept below the overheating limit (denoted by $\theta^{\text{overheating}}$) in order to prevent damage.

When the expected locations in the IC simultaneously have the temperature values that are specified by a map, it is said that that temperature map is enforced. The specified temperature maps should be enforced quickly. In case of burn-in, the temperatures should then be enforced for a given period of time to achieve the intended effect. In case of test, a map should be enforced as long as the corresponding tests are being applied. Usually, there are many temperature maps that, therefore, it is important to enforce them rapidly whether the ICs start from the ambient temperature or from another map. The order of the maps has a considerable impact on the overall burn-in/test time, as will be discussed in section V. For the time being, we assume that the maps order is given and focus on other aspects of the problem. In our work, a temperature map will be enforced by applying heating sequences sent through the TAM. Moreover, it is assumed that no test is applied when an IC is kept under a temperature map for burn-in. This will be relaxed in section IV so that the tests can be applied when a map is enforced.

Assume that there are M modules in an IC (on one or multiple dies) and their tests can be started and stopped independently (e.g., the modules are cores with core wrappers in a core-based design). In order to enforce the specified temperature maps, heating sequences are used to heat up some of the modules. The average power of the heating sequence is given by a real number, denoted by p_m^{HS} for module m ($0 \leq m < M$). It is assumed that the TAM only affords W (a positive integer number) modules to be tested simultaneously.

A heating sequence could be a collection of high power tests, being applied one after the other. A heating sequence generates heat the same way that a test does and occupies TAM the same way as a test. In general, heat generation happens with every shift in the scan-chain, since we do not assume scan chain masking during the shift mode. As mentioned above, the simplest way of obtaining heating sequences is to identify and copy the most high-power tests. Moreover, specialized techniques for this purpose can be developed. These methods are, in essence, similar to [9]. The authors in [9] introduced a power-hungry test stimulus generation technique. The produced stimuli cause large switching activity which results in large power density and high temperature. The proposed technique combines a meta-heuristic with power analysis tools to automatically find the power-hungry test programs [9].

Although the authors in [9] focus on programs (software used in circuit's functional mode), generating heating sequences (test patterns for the circuit in test mode) can be done in a similar manner.

A 3D-SIC is usually laid out so that the main blocks (e.g., logic and memory) are placed in a certain distance relative to TSVs to avoid undesirable effects induced by TSVs such as high mechanical stress. Such forbidden areas are called Keep-Out-Zones (KOZ) [11], [13]. A collection of the TSVs placed next to each other (perhaps to overlap the KOZ of different TSVs and save area on the die), is called a *TSV block*. A TSV block may consist of only one TSV if the TSVs are placed far apart.

In this section it is assumed that a module is a single active thermal node. Furthermore, it is assumed that TSV blocks are always thermally don't-care. They do not generate heat (are passive thermal nodes) since TSV drivers are not considered as parts of the TSV blocks.

Assume that the desired temperature map is specified by a *low temperature limit* and a *high temperature limit* for each module and the don't-care modules are declared separately. For example, a temperature map specifies that module m has a low temperature limit equal to θ_m^L and a high temperature limit equal to θ_m^H .

The inputs to the proposed method include temperature maps, IC's temperature model, IC's electrical model (e.g., specification of the TAM and power-related specifications), switching activities of the heating sequences, ambient temperature (θ^{ambient}), and overheating limit ($\theta^{\text{overheating}}$). The output is a schedule that guides the application of the heating sequences to the modules so that their temperatures move into the specified ranges and stay there.

As an example, consider an IC with 3 modules, m_0 , m_1 , and m_2 . Assume that a temperature map is specified as $\theta_0^H = 125^\circ\text{C}$, $\theta_0^L = 115^\circ\text{C}$, $\theta_1^H = 95^\circ\text{C}$, $\theta_1^L = 85^\circ\text{C}$, $\theta_2^H = 65^\circ\text{C}$, and $\theta_2^L = 55^\circ\text{C}$, and no module is specified as don't-care. These temperature limits are shown in Fig. 1a with dashed/dotted lines. A temperature simulation is performed for this IC based on a proper periodic schedule and the simulated temperatures are shown in Fig. 1a. Starting from the ambient temperature ($\theta^{\text{ambient}} = 30^\circ\text{C}$), the modules' temperatures steadily raise until they are inside the specified ranges. As shown in this example, applying heating sequences can drive the IC into a high temperature situation. For example, the temperature of module m_0 has reached 120°C at around 4×10^4 Time Units (TU). A TU consists of 4×10^3 test cycles in this example.

The temperatures around 6×10^4 TU point, are magnified in Fig. 1b. The time interval shown in Fig. 1b corresponds to three periods of the schedule. Since the schedule is periodic, one period captures the entire schedule which is repeated in a cyclic manner. Fig. 1c further magnifies one period of the schedule that starts at t_0 and ends at t_3 . The period length is denoted by τ ($\tau = t_3 - t_0$). One period is divided into three intervals, specified by numbers 0, 1, and 2 in Fig. 1b. They correspond to the time intervals $[t_0 \ t_1]$, $[t_1 \ t_2]$, and $[t_2 \ t_3]$ ¹ in Fig. 1c, respectively. The schedule specifies that the heating sequence for module m_0 is applied only in the $[t_0 \ t_1]$ interval,

¹ The notation $[a \ b]$ is used to represent an interval that ranges from a to b .

Figure 1. Temperature curves for an IC with three modules. (a) Transition followed by steady state response. (b) Temperature curves for three periods of the schedule. (c) Temperature curves for one period in detail.

the $[\tau + t_0 \ \tau + t_1]$ interval, and in general in $[\tau \times k + t_0 \ \tau \times k + t_1]$ intervals ($k = 0, 1, 2 \dots$), assuming that the process starts at time t_0 . The application of the heating sequences for module m_1 and module m_2 are specified in a similar manner by the schedule. For the $[t_0 \ t_3]$ period, the time intervals that the heating sequences are applied are depicted by gray areas in Fig. 1c. In this example, the TAM provides access to one module at a time ($W = 1$). Therefore in interval $[t_0 \ t_1]$ only module m_0 receives heating sequence. Similarly, in $[t_1 \ t_2]$ only m_1 is heated and the same goes for interval $[t_2 \ t_3]$ for m_2 . We need an efficient algorithm to generate such schedules.

B. Steady State Solution

Let us first analyze a simplified situation, where we assume that a steady state power could be provided for the modules. In this case, there is a steady state solution that could generate and maintain the specified temperature map. Providing continuous steady state powers simultaneously for all modules is, however, very likely to be impossible mainly due to TAM limitations. One solution is to use the maximal practical power for each module in combination with a Pulse Width Modulation (PWM) technique. Therefore, the best that can be achieved is a discrete stimulus sequence that has constant long-term average power with small ripples.

This way, the modules have a time-divided multiple access to the TAM. In order to reduce the risk of out of range temperatures due to ripples in the input power, the desired steady state temperatures are defined at the middle of the specified ranges $\theta_m^{SS} = \frac{1}{2} \times (\theta_m^L + \theta_m^H)$. Such ripples could be seen in the temperature curves given in Fig. 1. In order to find the power values that result in the specified temperatures, the IC's temperature model should be analyzed. A widely used temperature model is the lumped element temperature model, as used in HotSpot [16]. Such a model divides an IC into elements represented by nodes. Each node has a heat capacitance modelling its thermal capacity. Adjacent nodes are connected through a heat resistance that models the thermal conductivity between them. They are connected together in a network configuration, similar to an electric circuit. The temperatures correspond to voltages and the heat dissipation corresponds to a current source. A node is called *active* if it directly receives electrical power caused by switching activities. Detailed information of such models can be found in [12], [16].

All the characteristics of a temperature model are captured in two matrices $\mathbf{A}$ and $\mathbf{B}$. The thermal behavior of an IC is captured in the following system of ordinary differential equations [2][16].

$$\mathbf{A} \times \frac{d}{dt} \boldsymbol{\theta} + \mathbf{B} \times \boldsymbol{\theta} = \mathbf{P} \quad (1)$$

In this equation, $\boldsymbol{\theta}$ is the temperature vector and $\mathbf{P}$ is the power vector. Heat transfer among nodes is included in the temperature model and it means that a node can be heated up by its neighboring nodes even if it has no switching activities.

The specified temperature map consists, in fact, of the steady state temperatures that the IC should be kept at. A temperature map could be thought as the targeted steady state temperatures, $\boldsymbol{\theta}^{SS}$, which are composed of the desired steady state temperatures for each module (e.g., θ_m^{SS} for module m). Since $\boldsymbol{\theta}^{SS}$ is, in this case, equivalent to the steady state temperatures, which are considered constant (for a certain amount of time), its derivatives are zero (no variation in time). Therefore, (1) could be written as

$$\mathbf{P}^{SS} = \mathbf{B} \times \boldsymbol{\theta}^{SS}. \quad (2)$$

This means that it is possible to calculate the required powers that lead to the specified temperature map. In order for the specified temperature map to be achievable, the computed steady state power values must satisfy a feasibility and a schedulability condition. The first part of the feasibility condition is that the computed steady state power for module m (p_m^{SS}) should be larger than or equal to the stray power dissipated by the module. The stray power is the sum of the leakage power and the clock networks' power. It is denoted by $\overline{p}_m$ (for module m). The second part of the feasibility condition is that p_m^{SS} should be less than or equal to the average power of the corresponding heating sequence, p_m^{HS} , plus $\overline{p}_m$. The feasibility condition is, therefore, as follows:

$$\forall m, \overline{p}_m \leq p_m^{SS} \leq (p_m^{HS} + \overline{p}_m). \quad (3)$$

Usually the feasibility condition is easily met if the specified temperature map is realistic (i.e., the specified temperature is not lower than the ambient, and not larger than the achievable temperature). Assuming that (3) is satisfied, the schedulability condition which is related to the limited TAM bandwidth should be verified. The challenging problem here is to create the required average power values, $\mathbf{P}^{SS}$, using the available TAM bandwidth. This is done by selectively applying the heating sequences to the modules.

The continuous application of the heating sequence generates an average dynamic power equal to p_m^{HS} . The desired power values, p_m^{SS} , which are smaller than $p_m^{HS} + \overline{p}_m$, are created by applying the heating sequence, p_m^{HS} , for a fraction of a time period. The average power in a period should be made equal to the required steady state power. As mentioned before, this is done using a technique similar to PWM. The ratio of the

duration of heating sequence application to the overall time period is therefore called Duty-cycle (D_m) and its value is calculated using the following equation.

$$D_m = \frac{(P_m^{SS} - \bar{p}_m)}{p_m^{HS}} \quad (4)$$

The duty-cycles might not be achievable if their values are relatively large and if the TAM does not provide sufficient bandwidth. For example, assume a design with two modules, with the duty-cycles $D_0 = 0.6$ and $D_1 = 0.8$. This means that in a period of time equal to 1, we need access to module 0 for 60% of the time and access to module 1 for 80% of the time. Therefore, simultaneous access to more than one module ($0.6 + 0.8 = 1.4$ modules) is needed. This means that the TAM must provide simultaneous access to these two modules otherwise these duty-cycles are not schedulable and the specified temperature map cannot be enforced.

Note that D_m can be divided into pieces; for example $D_1 = 0.8$ could be implemented by first applying the heating sequence for a duration equal to $D_{1,0} = 0.3$ at the middle of the period and later on for a duration of $D_{1,1} = 0.5$ at the end of the same period. The feasibility and schedulability conditions could be written together using the duty cycle concept as follows.

$$\forall m, 0 \leq D_m \leq 1, \text{ and } \sum_{m=0}^{M-1} D_m \leq W \quad (5)$$

In fact, the first line in (5) is identical to the feasibility condition in (3), which is re-written in terms of the duty cycles. The second line in (5) is the schedulability condition, where W is the number of modules that can access the TAM, simultaneously. Given a temperature map that satisfies both feasibility and schedulability conditions, it is relatively simple to develop a schedule to deliver the required duty cycles. One such scheduling algorithm is presented in [4]. It is demonstrated in [4] for the schedules generated by the proposed method, at every moment in time, W modules or less are receiving their heating sequences, which means that the TAM limitation is not exceeded. Furthermore, it is shown that the average of the applied heating sequences for each module is equal to the specified steady state power for it. For example in Fig. 1c, modules m_0 , m_1 , and m_2 receive 50, 35, and 15 percent of p_0^{HS} , p_1^{HS} , and p_2^{HS} plus $\bar{p}_0$, $\bar{p}_1$, and $\bar{p}_2$, respectively. This is indicated by the width of the gray areas as compared with the schedule's period, τ ($\tau = t_3 - t_0$).

As mentioned before, a temperature map may leave the temperatures for some nodes unspecified (don't-care nodes). Besides, the temperatures for inactive thermal nodes (e.g., TSV blocks) are also left unspecified. On the other hand, in order to compute the steady state powers (using (2)), these temperatures should also be known. The proper choice of the temperatures for the don't-care nodes may determine if the temperature map can be achieved or not. The problem of finding proper temperature values for the don't-care nodes can be formulated as a Linear Programming (LP) problem.

In the LP formulation shown in Fig. 2, the duty cycles are decision variables. The main objective is to find a feasible solution. The temperatures, θ_m , should be equal to the temperatures that are specified by the temperature map, θ_m^{SS} . If not specified by the temperature map (don't-care modules) the temperatures should be between the ambient temperature and

the overheating temperature. The relation between the power values, p_m^{SS} , and the duty cycles is defined by (4). The temperatures, θ_m , are computed based on power values, p_m^{SS} , using (2) (by replacing $\mathbf{P}^{SS}$ and $\mathbf{\theta}^{SS}$ with vectors composed of p_m^{SS} and θ_m , respectively). For an inactive module, the power value should be equal to the stray power, $\bar{p}_m$, and therefore the duty cycles should be zero. For an active node, the duty cycles are between zero and one as defined in (5). The duty cycles should satisfy the schedulability condition (i.e., the second line in (5)). Assuming that the LP solver has found a feasible solution and has calculated the duty cycles successfully, a proper period for the PWM-like method has to be computed.

The duty cycles and the scheduling approach, discussed so far, are independent of the schedule's period, τ . They generate the modules' temperatures such that their average equals the specified steady state temperatures. The period, τ , should be short enough so that the fluctuations in the temperatures do not violate the specified limits (θ_m^L and θ_m^H). On the other hand, a longer period is desirable in order to minimize the switching actions in the schedule. An example for the results obtained by the proposed algorithm could be seen in Fig. 1a. After the temperatures have completed their transitions to their new values (after 4×10^4 TU), the proper choice of the period keeps them inside the specified ranges, with a relatively low number of switching actions in the schedule.

In order to find a relatively long period, τ , that albeit being long, keeps the temperature fluctuations inside the specified ranges, two different situations should be considered: (H) heating sequence is applied; and (L) no stimuli are applied. In order to estimate the proper period for situation H, (1) is re-written around the steady state temperature for the heating sequence power, as shown in (6a). For situation L, (6b) is used, instead.

$$\left(\frac{d}{dt}\theta\right)^H = \mathbf{A}^{-1} \times (\mathbf{P}^{HS} + \bar{\mathbf{P}} - \mathbf{B} \times \mathbf{\theta}^{SS}) \quad (6a)$$

$$\left(\frac{d}{dt}\theta\right)^L = \mathbf{A}^{-1} \times (\bar{\mathbf{P}} - \mathbf{B} \times \mathbf{\theta}^{SS}) \quad (6b)$$

An example for (6a) is the tangent line that touches the temperature curve in Fig. 3 at point A (around the steady state temperature). A similar example for (6b) is the tangent line CD in Fig. 3. Equation 6a is then used to estimate the desired value for the period focusing only on the high temperature limit. Assume that the proper τ , only focusing on situation H, is denoted by T_m^H and the proper τ , only focusing on situation L, is denoted by T_m^L . It is safe to assume that $D_m \times T_m^H$ (D_m is the duty cycle) is the amount of time that will result in a near violation situation for module m in situation H. In order to estimate T_m^H , first the derivative on the left side of (6a) is

1. Decision variables: $D_m ; m = 0, 1, \dots, M - 1$
2. Objective: *find a feasible solution*
3. Constraints:
4. $\theta_m = \theta_m^{SS}, \text{ if } m \text{ is specified}$
5. $\theta_{ambient} \leq \theta_m < \theta_{overheating}, \text{ if } m \text{ is don't care}$
6. $D_m = 0, \text{ if } m \text{ is inactive}$
7. $0 \leq D_m \leq 1, \text{ if } m \text{ is active}$
8. $\sum_m D_m \leq W ; m = 0, 1, \dots, M - 1$
9. Equations 2 and 4 relate variables D_m and θ_m .

Figure 2. Linear programming formulation for finding feasible duty cycles.

linearly approximated as follows:

$$\frac{\theta_m^H - \theta_m^{SS}}{D_m \times T_m^H} \cong \left(\frac{d}{dt} \theta \right)_m^H \quad (7)$$

Now, T_m^H is computed for module m as

$$T_m^H \cong (\theta_m^H - \theta_m^{SS}) / \left(D_m \times \left(\frac{d}{dt} \theta \right)_m^H \right). \quad (8)$$

The values for $\left(\frac{d}{dt} \theta \right)_m^H$ are obtained from the right side of (6a) and, consequently, the values for T_m^H are computed using (8). For example, in Fig. 3, when the module is receiving power, the derivative that is represented by a straight line is tangential to the temperature curve at its intersection point with the steady state temperature at point A and later on intersects with the high temperature limit at point B. The period, T_m^H , is then calculated based on the time difference between A and B. The other part of the line that stand between A and the low temperature limit is deliberately left out in order to achieve a shorter period that is safe in most of the situations (e.g., variation in the input power).

In a similar manner values for situation L, T_m^L , are calculated based on (6b) by focusing only on the low temperature limit. Since the temperatures should not violate any of the specified limits, the shortest T_m ($T_m = \min\{T_m^H, T_m^L\}$) is selected as the acceptable period for module m . The actual period, τ , should be the smallest among acceptable periods for all modules ($\tau = \min_m\{T_m\}$) so that none of the temperature limits for the modules is violated.

For the steady state solution, the average powers that are applied when a new map is going to be enforced are the steady state powers that will also maintain that map. This implies that the transition to the new map is very slow since even during the transition these steady state powers are applied. The steady state solution results in an excessively long transition time and therefore a faster solution is necessary.

C. Transient Solution

In this section, a solution to reduce the overall transition time is introduced. We start by looking into the analytic solution for (1) for a duration of time equal to t , as shown below [2]:

$$\theta^t = \alpha(t) \times \theta^0 + \beta(t) \times P^B \quad (9)$$

In the above equation, the initial temperatures are expressed by θ^0 and the temperatures at time t are denoted by θ^t . P^B is the power vector that is assumed to be constant for the time interval t . An intuitive explanation of (9) is that $\alpha(t)$ determines how fast the initial temperatures fade away and $\beta(t)$ determines how fast the input power affects the temperatures. $\alpha(t)$ and $\beta(t)$ are matrices that are computed based on A and B , for a duration of time equal to t , as follows [2]:

$$\alpha(t) = \exp(-A^{-1} \times B \times t) \quad (10a)$$

$$\beta(t) = (I - \alpha(t)) \times B^{-1} \quad (10b)$$

In the rest of this paper $\alpha(t)$ and $\beta(t)$ are represented as α and β , respectively. As mentioned before, achieving a new temperature map in a short time is crucial and, therefore, this transition should happen as fast as possible. Once the IC's temperatures have converged to the specified temperature map, they can be maintained using the steady state powers, P^{SS} , as presented in section III.B.

We would like to extend the steady state solution approach to (9), which includes the transient response, in order to find the

Figure 3. An example for the computation of a safe period so that the temperature limits are not violated.

schedulable power values that result into the shortest transition time. The new problem can be formulated as: Find the shortest transition time, t , and the corresponding power values, P^B , such that the specified map is achievable. The transition time from map μ_i to map μ_j is defined as the time required to construct the temperatures specified by map μ_j starting from temperatures specified by map μ_i .

This problem can be solved using an iterative search that tries different alternatives for t . The algorithm uses the latest information regarding the interval that contains the optimal transition time (denoted by $[\lambda \ \sigma]$). At any step, it is known from the previous steps that the specified map is not achievable for transition times shorter than λ . It is also known that since the temperature map is achievable for a transition time equal to σ , longer transition times are not optimal. Initially λ is set to zero and σ to the transition time for the steady state approach. This steady state transition time is obtained by simulating the temperatures when the steady state schedule is used (similar to Fig. 1). A number of candidate transition times with uniform distances are selected between λ and σ :

$$t_r = \lambda + (r + 1) \times (\sigma - \lambda) / (R + 1) \quad (11)$$

The r -th candidate transition time is denoted by t_r . R is the number of parallel LP solvers and its value is selected based on the degree of parallelism offered by the platform that runs the algorithm. For example, for a machine that supports 8 threads, 8 is a reasonable choice for R . For each candidate t_r , solving the LP formulation determines whether the temperature map is achievable or not. The value of σ is updated to the smallest t_r that leads to schedulable power values. The value of λ is updated to the largest t_r that leads to power values that are not schedulable. Note that if for all the candidate transition times, denoted by t_r ($r = 0, 1, \dots, R - 1$) the map is achievable, then λ remains unchanged. On the other extreme, if none of the t_r 's are schedulable then σ remains unchanged. The algorithm stops when the smallest transition time is found with acceptably low error. The error is bounded to $(\sigma - \lambda)$ and therefore if this difference is smaller than a certain limit then the actual error, too, will be smaller than that limit.

The problem formulation for the LP solver that is used here is similar to the LP formulation in section III.B and Fig. 2, with the following differences: (i) Instead of θ_m s, the temperatures at the end of the transition time, θ_m^t s, are used. (ii) Instead of (2), (9) is used to calculate the temperatures based on the power values. The relation between the power values and the duty cycles is defined by (4) similar to Fig. 2. If the LP solver finds a feasible solution, the temperature map is achievable. This information is then used to update the λ and σ values.

The matrix exponent computation for α , in (10), is performed using techniques proposed in [28]. These techniques are used in order to speed up the repeated recalculations of α and β for alternative transition times. They are based on eigenvalue decomposition utilizing the inherent properties of matrices A and B and replace the excessively time consuming matrix exponent calculations in (10) with simpler operations. Although these techniques speed up the calculations, the required time is still very large, as experimentally shown in section VI.

Even though, the transient solution is an intuitive extension of the steady state solution and greatly outperforms it, it is slow in generating the schedules. Therefore, a new approach that avoids the time-consuming successive calculations of α and β is necessary. Such an approach is proposed in the next section, based on a fast heuristic. Moreover, this new approach is capable of handling a more realistic problem formulation compared with the steady state and transient solutions.

D. Transient-Based Heuristic

So far, temperature maps could only specify temperatures for modules. Therefore, the module's area limits the resolution of the temperature maps. This limitation is relaxed, from this section onward, by allowing the modules to be divided into smaller areas, called *sub-modules*.

1) Support for High Resolution Temperature Maps

Previous techniques require the ability to apply the heating sequence only to a selected thermal node, avoiding application of heating sequences to other nodes. Therefore, the smallest element in the temperature model could not be smaller than the corresponding module (e.g., a core with core wrapper). The method proposed in this section can work even if heating sequence application to a selected node, generates heat in some other nodes. This supports division of a module into a number of sub-modules.

Now we can assume that the overall number of thermal nodes, denoted by N , is larger than or equal to the number of modules ($M \leq N$). In the rest of this paper, the desired temperature maps are specified for the thermal nodes instead of the modules. Consequently, the temperature map specifies that node n has low temperature limit equal to θ_n^L and high temperature limit equal to θ_n^H ($0 \leq n < N$).

The switching activities for heating sequences must be more specific, providing the power breakdown among active thermal nodes. For example, assume that module m is divided into two active thermal nodes n and o . The average power of a heating sequence for active node n is represented by p_n^{HS} . Node o may also receive power, denoted by $p_{n,o}^{HS}$, when heating is targeted for node n . Similarly, when trying to heat up node o with p_o^{HS} , node n is also heated by $p_{o,n}^{HS}$. Furthermore, power dissipation for TSV blocks is now supported, and the TSV drivers/buffers may be placed in TSV blocks and their desired temperatures might also be specified in the temperature maps. Note that a TSV block is not only a TSV. A TSV block includes TSVs, interconnects, bulk silicon, and possibly transistors. The TSV, itself, is a metal rod that is a good conductor of heat and will not generate heat, on its own.

2) Operations During the Transition: Boosting

The transient-based heuristic generates the schedule, offline. The schedule generation process is based on temperature

simulation. The general idea is to apply heating when the simulated temperature is below θ^L and stop the heating before it reaches θ^H . These start/stops events construct the schedule.

The transition time between two maps is wasted time and must be minimized. The fact that the temperatures during transition are not important (except that overheating is avoided) is used to shorten the transition time. The proposed technique may apply heating more than needed. Heating, in this case, is called boosting. Boosting stops when the node reaches the Stop Boosting temperature, θ_n^{SB} . The stop boosting temperature may be higher than the high temperature limit, θ_n^H , but it is always lower than $\theta_{overheating}$.

The following example shows how boosting helps. Assume that node n is initially boosted beyond θ_n^H ($\theta_n^{SB} > \theta_n^H$). Then n does not need to receive heating for a while and this leaves the TAM available for other nodes. Meanwhile, n 's temperature keeps decreasing (naturally). Just before all other nodes are in their specified temperature ranges, n 's temperature drops below θ_n^H . This simplifies and shortens the schedule for the transition period and, therefore, is desirable.

Fig. 4 demonstrates how the transient-based heuristic works, by showing the temperature curve for one of the thermal nodes. The curve starts at the exact moment that the transition to the new map is started. The transition interval ends when all the nodes are in their valid temperature ranges. Since only one of the nodes is shown here, the transition time cannot be observed directly, therefore it is indicated by the gray area at the lower left corner of Fig. 4. Boosting is shown in interval a : the temperature increases beyond θ_n^H and continues to θ_n^{SB} . This helps to achieve a shorter transition time, as discussed before. Apart from boosting, other operations during the transition are similar to the operations after the transition.

3) Operations After the Transition

Just after transition, the map is enforced. But since a node's temperature will naturally decrease the temperature will eventually fall below θ_n^L if no or little power is applied to it. Therefore, a heating sequence should be applied at some point, before the temperature falls out of range. This point is marked with a temperature level named *Heating Trigger* and denoted by θ_n^{HT} for node n ($\theta_n^{HT} > \theta_n^L$). The heating sequence should be applied when the temperature of node n falls below θ_n^{HT} . The difference between θ_n^{HT} and θ_n^L provides sufficient time for the node to wait for gaining access to the TAM without its temperature falling below θ_n^L . In Fig. 4, the heating is required at the beginning of the interval c , but since the TAM is not available, the node waits. At the beginning of the interval d the node has finally gained access to the TAM and the heating begins.

Heating should stop when the temperature reaches θ_n^H . The time it takes to get back to the low temperature limit could be

Figure 4. Transient-based heuristic demonstrated using a temperature curve.

utilized to heat up other nodes that need heating. In a situation that a module consists of multiple active thermal nodes, the heating sequence could only be applied if all of these thermal nodes have temperatures lower than their high temperature limit.

4) TAM Access Management

The nodes that simultaneously require heating should be accommodated within the available bandwidth of the TAM. This bandwidth might not be sufficient for all of them and, therefore, the nodes that need heating more than others should be prioritized. The priorities for using the TAM are determined based on the *regional need for heating* (denoted by d_n around a node n). The value of d_n is recomputed whenever node n needs heating (during offline schedule generation). A node requires heating in the following two situations: (i) When $\theta_n < \theta_n^{HT}$, after the transition, for example the interval c in Fig. 4. (ii) When $\theta_n < \theta_n^{SB}$, during the transition, for example the interval a in Fig. 4. In the following, we explain how to calculate d_n for the situation i. Regional need for heating for situation ii is obtained in a similar manner by replacing θ_n^{HT} with θ_n^{SB} .

Equation 1 is re-written, with the approximate derivatives, as

$$\frac{A \times (\theta_n^{HT} - \theta)}{T} + B \times \theta = \bar{P} + D \times P^{HS}. \quad (12)$$

The input power, P , in (1) is substituted with the stray power, $\bar{P}$, plus the PWM power of heating sequences, $D \times P^{HS}$. Vector D is the vector form of the regional need for heating and consists of d_n s. Equation 12 is written for one test cycle with period T which is a very short time. The equation is then solved for the nodes that need heating as follows.

$$d_n = \frac{\sum_{k=0}^{N-1} a_{n,k} \times (\theta_k^{HT} - \theta_k) + \sum_{k=0}^{N-1} b_{n,k} \times \theta_k - \bar{p}_n}{p_n^{HS}} \quad (13)$$

The regional need for heating, d_n , depends on the required heating for node n (consider the summations when k is equal to n), on the required heating that is related to the adjacent nodes (consider the summations when k denotes an adjacent node to n), and on the average power of the corresponding heating sequence, p_n^{HS} . The regional need for heating for a node has the highest dependency on the node itself, and then a relatively high dependency on the adjacent nodes (this characteristic is captured by the temperature model). The influence of other nodes located far away from the targeted node is small. The heat transfer between nodes is taken into account automatically, since (13) is derived from the temperature equation, (1), and includes the thermal conductances from matrix B . This is reflected by $b_{n,k}$ in (13).

Equation 13 ensures that the priority for using the TAM is given to the regions that need longer heating times, for example because of large $(\theta_n^{HT} - \theta_n)$ and small p_n^{HS} . Furthermore, the locality of this heuristic is helpful because adjacent nodes are likely to be in the same module and therefore these nodes will receive some desirable heating sequence ($p_{n,k}^{HS}$) or heat transferred from module n . An effect of the interplay between priorities could be seen in Fig. 4. The waiting period in the interval f is much shorter than the waiting period in the interval c . The length of a waiting period depends on the other nodes's priorities in addition to the node n 's priority.

As discussed before, the performance of the transient-based heuristic strongly depends on the stop boosting, θ_n^{SB} , and

heating trigger, θ_n^{HT} , temperatures. One example is the priorities calculated using (13), since they depend on θ_n^{HT} after the transition and on θ_n^{SB} during the transition. Efficient values for these temperature levels for each temperature map and each thermal node are found using a Particle Swarm Optimization (PSO) technique. Particle swarm optimization is a well-known iterative population-based optimization metaheuristic. A canonical form of PSO [22] is used in this paper in a straightforward manner.

E. Remarks

The proposed approaches support also heating sequences generated by a Built-In Self-Test (BIST) engine. An example for the use of BIST engines during burn-in in order to achieve high toggle coverage is reported in [10]. Such BIST engines that stimulate high switching activities in a certain area of the IC under burn-in can be used to produce heating sequences online. The only difference, in our context, is that if the BIST engine does not occupy TAM, then it can be scheduled at any time as needed. For instance assuming that module m_k can receive its heating sequence from an adjacent BIST engine that is not occupying TAM, the 8th line for the LP formulation in Fig. 2 should be changed to: $(\sum_{m=0}^{k-1} D_m + \sum_{m=k+1}^{M-1} D_m) \leq W$. The situation for the transient-based heuristic is even simpler, since the algorithm only needs to know that module m_k can receive its heating sequence at any time. Then, m_k does not need to compete with others for TAM access. Consequently, there is no need to evaluate the regional need for heating for m_k .

The proposed approaches in section III make it possible to perform burn-in based on heating sequences without requiring a heat chamber. One of the situations when a heat chamber might be required is for the ICs that are designed to work in an extremely high temperature environment. For example, a microcontroller for a car engine is designed with low power in order not to raise too much its temperature from the very high ambient temperature in the engine area. When such a chip is tested or operates with regular low ambient temperature, it is impossible to have enough power density to boost its temperature to its usual high level in normal working condition.

Another such situation is when some parts of an IC (e.g., package pins, die to pin connections, and the interposer) cannot be heated up sufficiently by input stimuli. In such cases, an extremely hot burn-in condition might be required that is not achievable by exclusive use of heating sequences. Even in such cases the use of the methods proposed in this paper for enforcing the temperature gradients will still be useful. The proposed algorithms do not need any modifications to work under such situations, except to set a large ambient temperature corresponding to the heat chamber temperature.

The proposed transient-based heuristic allows the IC to be divided into a desired number of thermal nodes, N . A large N means high resolution and translates into higher computational effort. Therefore, the resolution can be restricted if the available computational power is limited.

In this paper, as mentioned previously, it is assumed that the temperature maps capable of identifying the gradient-dependent defects are given. Furthermore, it is assumed that the combination of the given tests and heating sequences (mixed with cooling intervals) is capable of enforcing the desired

temperatures on the specified modules.

IV. TEMPERATURE-GRADIENT BASED TEST

For the temperature-gradient based test, the goal is to make sure that the tests are performed when the temperature gradients are enforced. This means that the specified temperature maps should be enforced before the test and then maintained during the test. The straightforward algorithm and the fast heuristic, which are proposed in the following, do this differently.

A. Straightforward Algorithm

This algorithm works by changing between two modes, the *temperature construction* mode and the *test* mode. Initially the temperature construction mode is activated and it enforces the specified temperature map using a method similar to the transient-based heuristic proposed in section III.D. Then the test mode is activated and the tests that are scheduled with a third party algorithm (e.g., scheduling method proposed in [24]) are applied. The test temperatures are simulated at design time and as soon as at least one of the thermal nodes is out of its specified range, the test mode is paused and the temperature construction mode takes over again. When all thermal nodes are brought into the specified temperature ranges, the temperature construction mode is paused and testing resumes.

Similar to the transient-based heuristic, if the temperature of a node is lower than the heating trigger temperature, it should be heated by applying the heating sequence to it. If there are many nodes that need heating (more than what the TAM can support), priority is given to those with higher regional need for heating as defined in section III.D. The construction mode, unlike the transient-based heuristic, should not heat the nodes up to their high temperature limit since the power of the tests that are applied immediately after the construction mode may rapidly heat up the node beyond high temperature limit. Therefore, *Testing Trigger* temperatures which are denoted by θ_n^{TT} for node n ($\theta_n^{HT} < \theta_n^{TT} < \theta_n^H$) are introduced here. During the temperature construction mode, the heating for node n stops as soon as the temperature reaches θ_n^{TT} .

In the test mode, as soon as the temperature of a node reaches the high temperature limit, the test mode is immediately paused, the temperature construction mode is activated and, consequently, a cooling interval is applied. The cooling continues until the node is cooled down to the testing trigger temperature, θ_n^{TT} , and then the node is ready for testing again. The actual activation of the test mode will also depend on the temperatures of the other nodes. Efficient values for testing trigger temperatures, θ_n^{TT} , for each map are found using a particle swarm optimization technique along with θ_n^{SB} and θ_n^{HT} .

The inputs to the methods proposed here include the inputs to the methods proposed in section III as well as the test specifications (e.g., test switching activities). The output is a set of offline schedules, generated based on temperature simulations. There is no need for temperature sensors and the precision of the achieved temperature maps depends on the temperature model. Although, generating such offline schedules is the focus of this paper, the proper values for the heating trigger, θ_n^{HT} , stop boosting temperatures, θ_n^{SB} , and testing trigger temperatures, θ_n^{TT} , which result in a rapid test, could also be considered as the outputs that provide a basis for an online scheduling scenario. In this case, temperature sensors are used and the precision depends on the temperature sensors.

Such online approaches are however beyond the scope of this paper.

The straightforward algorithm is simple, and allows the choice of a desired arbitrary test schedule that is used in the test mode. But the overall test application time offered by this method is very long. Note that the total test application time also includes times spent for temperature construction.

B. Fast Heuristic

The fast heuristic schedules the tests together with the heating sequences such that the specified temperature map is maintained. This way, a shorter test application time can be achieved. An illustrative example for the proposed method is given in Fig. 5 for a single thermal node. The proposed technique has similarities to the temperature construction algorithm in the previous section. For example, stop boosting temperature, θ_n^{SB} , indicates that the boosting should stop, as illustrated at the end of interval *a* in Fig. 5. After being too warm, the node should cool until its temperature gets below the testing trigger temperature, θ_n^{TT} , as shown in interval *b*.

When the temperatures for all of the other thermal nodes covered by module m are between their high temperature limit, θ_n^H , and their heating trigger, θ_n^{HT} ($\theta_n^H > \theta_n^{HT}$), testing may start, as in interval *c* in Fig. 5. All other nodes should be within their temperature limits $[\theta_n^L, \theta_n^H]$. Testing continues until the temperature of at least one of the nodes goes beyond the high temperature limit θ_n^H or falls below the heating trigger θ_n^{HT} . For example at the end of interval *c*, the node is too cold for testing and a heating interval should be introduced. Note that the TAM may no longer be available and therefore the node is waiting for access to TAM in interval *d*.

Finally, when access to the TAM is obtained, the heating sequence is applied in interval *e*. In order to start heating, all nodes covered by a module should be colder than the high temperature limit since the heating sequence for one node is very likely to inject power to other nodes in the same module (as explained in section III.D). Heating continues until the temperature goes beyond the testing trigger temperature and, then, the test resumes as in interval *f* in Fig. 5. When the temperature reaches the high temperature limit, a cooling interval is introduced as in interval *g*. This procedure continues until all tests corresponding to the current temperature map are completed.

As mentioned before, nodes will compete for access to the TAM and, therefore, some of them should be prioritized. First the nodes that require heating (not the tests) are granted access to TAM. This helps to keep the temperatures most of the time within the specified limits and, thus, keep the flow of the tests uninterrupted. Note that if only one node falls out of its specified range, all tests must be interrupted until the map is achieved again. This will waste a lot of time, since the tests for the modules that are in their specified range should also be interrupted. The priorities for the nodes that require heating is

Figure 5. Fast heuristic demonstrated using a temperature curve.

determined based on the regional need for heating as proposed in section III.D.

If the TAM has left with some available bandwidth after the heating sequences are scheduled, the modules that are *thermally qualified* may resume their tests. A module is *thermally qualified* if none of the nodes that correspond to that module are demanded by the previously discussed rules to receive heating, wait for heating, or receive cooling. The priority is given to the modules that are expected to offer long *test endurance*. The test endurance is denoted by e_m for module m , and is defined as

$$e_m = tt_m \times r_m. \quad (14)$$

The test endurance is directly proportional with the *remaining test size* denoted by r_m for module m . The larger the remaining test size, the longer the test endurance. The *thermal tolerance*, denoted by tt_m for module m , is the other contributor to the test endurance. High thermal tolerance, tt_m , indicates that the module is capable of receiving tests for a relatively long time without exceeding the specified thermal limits. Therefore, a module with large thermal tolerance may remain under test for a relatively long time. The thermal tolerance is defined as

$$tt_m = \min_k \{\Delta_k\}. \quad (15)$$

In (15), it is assumed that module m covers K active thermal nodes. Δ_k ($k = 0, 1, \dots, K-1$) denotes the expected thermal distance to a temperature limit for node k and is defined as

$$\Delta_k = \begin{cases} \theta_k^H - \theta_k, & \text{if } \text{upcoming_tests_power} > p_k^{SS} \\ \theta_k - \theta_k^{HT}, & \text{if } \text{upcoming_tests_power} \leq p_k^{SS} \end{cases} \quad (16)$$

As mentioned in section III.B, the desired steady state power p_k^{SS} is the power that results in a temperature equal to $\theta_n^{SS} = \frac{1}{2} \times (\theta_n^L + \theta_n^H)$. Equation 16 indicates that if the upcoming tests have relatively high average power, then it is likely that the thermal node exceeds the high temperature limit and, therefore, the difference between the current temperature, θ_k , and the high temperature limit, θ_k^H , is a good measure for thermal tolerance. Similarly, for a relatively low power test, it is more likely that the temperature falls below the heating trigger in the future. Therefore, the difference between the current temperature, θ_k , and the heating trigger temperature, θ_k^{HT} , is a good measure for thermal tolerance. Thermal tolerance, tt_m , is defined as the smallest Δ_k ($k = 0, 1, \dots, K-1$) since as soon as a single node is out of the specified range $[\theta_n^L, \theta_n^H]$, disregarding of the temperatures of the other nodes, test should be interrupted. Note that if the temperature falls below θ_n^{HT} , only for a node in module m , then the test is interrupted only for module m .

A proper value for the testing trigger temperature, θ_n^{TT} is selected so that the temperature variation during test (caused by the variations in the test power) rarely results in the temperatures below θ_n^{HT} or above θ_n^H . Every time that θ_n^{HT} or θ_n^H are violated, the test must be interrupted and a heating or cooling interval must be introduced, respectively. Since these are time consuming, a proper θ_n^{TT} value helps to obtain a short test application time by reducing the number of interruptions. Besides the testing trigger temperature, stop boosting and heating trigger temperatures (θ_n^{SB} and θ_n^{HT} respectively) have a considerable effect on the test application time and therefore proper values for them should be found. A particle swarm optimization technique is used to find the proper values for θ_n^{SB} ,

θ_n^{HT} , and θ_n^{TT} for each map. Similar to the previous sections, the canonical PSO [22] is utilized in a straightforward manner.

V. TEMPERATURE MAP ORDERING TECHNIQUE

The order in which the maps are enforced has a considerable impact on the overall burn-in and test time. Since there are usually a number of temperature maps to be applied, their ordering is important. In this section we present methods to rapidly obtain a proper order for temperature maps that results in a short burn-in and test time.

To simplify the discussions, let us assume that the temperature map for a thermal node is represented by the middle value of the specified temperature range $\theta_n^{SS} = \frac{1}{2} \times (\theta_n^L + \theta_n^H)$. As an example, assume that an IC has two thermal nodes and the initial temperature is 30°C. The temperatures specified in map μ_0 are denoted by $\{\theta_0^{SS}, \theta_1^{SS}\}^2$. This means that temperatures θ_0^{SS} and θ_1^{SS} are specified by map μ_0 for nodes 0 and 1, respectively. Assume that there are three temperature maps denoted by μ_0 , μ_1 , and μ_2 . These maps specify the following temperatures: $\mu_0 = \{110^\circ\text{C}, 90^\circ\text{C}\}$, $\mu_1 = \{40^\circ\text{C}, 50^\circ\text{C}\}$, and $\mu_2 = \{110^\circ\text{C}, 80^\circ\text{C}\}$, respectively. These temperature maps are represented in Fig. 6 by three points in a Cartesian space. The temperature for node 0 is represented by the horizontal axes, θ_0 , and for node 1 by the vertical axes, θ_1 . The initial order of temperature maps $\{\mu_0, \mu_1, \mu_2\}$ requires a long time to increase the temperature for node 0 from 30 to 110 (a_0 in Fig. 6a), then decrease it to 40 (a_1 in Fig. 6a), and then again increase it from 40 to 110 (a_2 in Fig. 6a). This process will take a long time due to the required large temperature changes. In contrast, it is much faster to work with the maps ordered as $\{\mu_1, \mu_2, \mu_0\}$, since in this case, the required temperature changes consist of smaller temperature variations, as shown in Fig. 6b.

As discussed earlier, in order to minimize the overall transition time for burn in, a particle swarm optimization finds the proper values for stop boosting and heating trigger temperatures (θ_n^{SB} s and θ_n^{HT} s, respectively). The map orders should be optimized along with these temperatures, since all of these factors have a crucial effect on the overall transition time for a given set of temperature maps. The naïve approach to find proper map orders is to introduce them as decision variables into the PSO along with θ_n^{SB} s and θ_n^{HT} s. Experiments showed that this naïve approach takes very long CPU time to complete. Since the optimized values for θ_n^{SB} and θ_n^{HT} depend on the map order, different map orders results in different optimized values for θ_n^{SB} and θ_n^{HT} .

The initial PSO population in the naïve approach consists only of random solutions (random θ_n^{SB} s, θ_n^{HT} s, and random map orders). Introducing a relatively good map order into the initial population of PSO (among other initial solutions that are random) will help to speed up the search. This approach is denoted by A1. The idea for approach A1 is to rapidly find a potentially good map order using some initialization heuristic and introduce it into the initial PSO population. By doing this, the search should speed up while the quality of the final values for θ_n^{SB} s and θ_n^{HT} s are kept reasonably high. In the majority of cases, PSO finds a better map order than the one produced by

² The notation $\{a_0, a_1, \dots, a_K\}$ is used to represent an ordered sequence of elements a_k ($k = 0, 1, \dots, K$).

the initialization heuristic.

It is, in fact, possible to find a potentially good map order without having to go through the time-consuming optimization of θ_n^{SB} s and θ_n^{HT} s. Furthermore, it is possible to do it without the relatively time consuming scheduling procedures for the heating sequences. A temperature map could be considered as a point in an N -dimensional Euclidean space (N is the number of thermal nodes). The thermal distance between two maps is defined as the Euclidean distance between them (e.g., b_2 between maps μ_1 and μ_2 in Fig. 6b). For a sequence of maps, the total thermal distance (TTD) is defined as the sum of the thermal distances between successive maps. For example, TTD for Fig. 6a is approximately 257, while for Fig. 6b it is 108, which is much smaller. In general, a sequence of maps with smaller TTD is expected to have a shorter transition time compared with a sequence with larger TTD.

Note also that the time required to change the temperature differs from node to node depending on the node's location on the IC, the adjacent nodes' temperatures, the heating sequence powers, and so on. Moreover, depending on these factors, the rise time and the fall time for the temperature of a certain node are also different (e.g., in many cases heating up is faster than cooling down, with the same temperature gap). The TTD does not take these differences into account in favor of a simple but meaningful metric that is fast to evaluate. However, when the map order is optimized using PSO, all these once ignored factors are automatically taken into account.

This problem is equivalent to finding the shortest Hamiltonian path in a complete graph whose vertices are temperature maps and the distance between two vertices is their Euclidean distance. Therefore, the initial heuristic based map order that is added to the PSO's initial population in approach A1 is called shortest Hamiltonian path. Due to the reasons discussed previously, this shortest path does not necessarily correspond to the optimal map order.

If A1 is allowed to run for a long time, it will produce very high quality solutions. However, for larger designs, this is unaffordable. We have therefore proposed the A2 approach, which consists of a short run of A1 followed by a post-PSO optimization of map orders. The motivation for this is that PSO optimization in A1 can rapidly identify possible solutions in the near optimal area of the search space but it then becomes very slow. Knowing the near optimal area, other optimization techniques can be deployed to rapidly improve the results. In the followings, the post-PSO optimization for the map orders is discussed.

In the general case, the post-PSO optimization could be excessively time consuming. A greedy heuristic is therefore used to rapidly find a near optimal solution. The greedy approach is characterized by its size, S . This size is the number of alternative partial solutions that are kept at each step (i.e., among the vertices with equal depth in the search tree). A greedy heuristic with size S works as follows. Starting from the root vertex (initial temperature) in the search tree, S vertices (i.e., S temperature maps) that have the shortest partial transition times are selected. This corresponds to the first map in the final map order. Here the full-fledged scheduling is performed to calculate the transition times. Then again S new vertices that have the shortest partial transition times are

Figure 6. The total thermal distance, TTD, for (a) a bad map order, and (b) a good map order.

selected out of the set of vertices that succeed the previous best S vertices. Two maps (in the final map order) are scheduled so far. This procedure repeats until all maps are scheduled. For S equal to one, at each step the map that is the fastest to achieve is selected. A large S slows down the search but it may provide better results. Our experiments showed that 10 is a good choice for S .

Albeit this general case which addresses large and time consuming ICs, for smaller ICs it is possible to find the optimal map order (i.e., exact solution) using an exact algorithm (e.g., branch and bound). Since a relatively good solution is already found by PSO in approach A1, we can skip many paths in the search tree that result in a larger transition time, without wasting time to fully schedule them. For example assuming that the map order in Fig. 6b is already found by A1, there is no need to schedule a_2 (in Fig. 6a) at all. Scheduling a_1 may also be aborted before completion since the overall transition time of this path in the search tree exceeds the overall transition time of the path corresponding to Fig. 6b before it even gets to vertex μ_1 . Note that in this algorithm, the edges are actual transition times and not the Euclidean distances. Albeit significant acceleration achieved by utilizing the near optimal result from A1 approach, larger examples are excessively time consuming and therefore finding their optimal solution is not practical.

Although this section has focused on map ordering for the temperature-gradient based burn-in, the map ordering for the delay test is very similar and the same technique can be used.

VI. EXPERIMENTAL RESULTS

We have first performed experiments to demonstrate that the proposed technique can rapidly and accurately achieve the specified temperature maps. For this purpose temperature simulations with HotSpot [16] based on the generated schedules are performed. Consider an IC with three modules and two temperature maps. The first map requires average temperatures of 120°C, 90°C, and 60°C for modules m_0 , m_1 , and m_2 , respectively ($\mu_0 = \{120, 90, 60\}$). The second map requires 90°C, 60°C, and 120°C, respectively ($\mu_1 = \{90, 60, 120\}$). Both maps assume a temperature range of 10°C (e.g., 120°C means [115° 125°]).

The temperature curves for the steady state solution and the transient-based heuristic are plotted in Fig. 7a and b, respectively. Boosting of temperature at module m_2 , which happens around 5×10^3 TU in Fig. 7b, is magnified in Fig. 7c. As discussed earlier, the temperature is allowed to rise beyond the high temperature limit during the boost. The burn-in intervals for the first and second maps are marked with bi_0 and

bi_1 , respectively. The transition time from the ambient temperatures to the first map (denoted by tr_0) and from the first map to the second one (denoted by tr_1) are much shorter for the transient-based heuristic. Consequently, the total time for burn-in process performed by the transient-based heuristic is about 4×10^4 TU shorter, compared to the steady state solution.

The proposed temperature-gradient based methods for burn-in and delay-fault test are evaluated for twelve experimental ICs with one to three layers as detailed in Table I, columns 2, 3, and 4. The one-layer experimental ICs (row 1 to 4 in Table I) are bare dies and could represent the pre-bond test stage. The ICs that have two layers (row 5 to 8) could represent mid-bond test stage. The ICs with three layers (row 9 to 12) could represent post-bond test stage. There are two, four, eight, and 16 physical modules per layer for different dies, resulting in the total number of modules ranging from two to 48, as given in column 3. There are one, two, and three TSV blocks per layer on the dies, resulting in the total number of TSV blocks given in column 4, ranging from one to nine. Each TSV block hosts a relatively large number of TSVs. The dies are assumed to be stacked in a face to back configuration.

The temperature models are extracted using an approach similar to the method proposed in [12] for 3D Stacked IC. This is an extended form of the technique used by HotSpot [16] for normal 2D ICs. The heating patterns' switching activities are generated using Markov chains, similarly as in [32]. The temperature maps specify the valid temperature ranges for nodes in the temperature model. The valid ranges are randomly selected between 35°C and 95°C, and some modules/nodes are randomly selected to be don't-care. Only temperature maps that can be achieved in practice are considered. An example for a temperature map that cannot be achieved is one that requires a central node with very low temperature and its adjacent nodes with very high temperature. In this case the temperature gradient is huge and it probably will require negative power (active cooling) for the central node.

A. Evaluation of Transient-Based and Fast Heuristics

The transient solution and the transient-based heuristic for burn-in (sections III.C and III.D, respectively) are evaluated and compared with the steady state solution (section III.B). The transient-based method is capable of handling temperature models having multiple nodes per module, while the steady state and transient solutions only support one thermal node per module. In order to have comparable experiments, the temperature model that is supported by the steady state method is used in the experiments for burn-in. The CPU time to generate the schedules for the transient-based method for all of the twelve experimental ICs together is about 12 minutes while the transient solution takes 17 minutes and steady state method completes in 2 seconds. As discussed earlier, the time required to bring the IC into a thermal situation that complies with the first temperature map and then to the next map until all maps are applied is defined as the overall transition time in this work. The percentage change in overall transition time offered by the transient solution and the transient-based heuristic, compared with the steady state solution, are given in columns 5 and 6 of Table I, respectively. Considerable speed up (78% in average) is achieved by the transient-based heuristic and moreover, it also outperforms the transient solution.

Figure 7. Temperature curves for (a) steady state solution and (b) transient-based heuristic. (c) Magnified module m_2 's curve shows initial boosting.

The fast heuristic for test (section IV.B) is evaluated and compared with the straightforward method (section IV.A). The temperature model used for these experiments has multiple nodes per module. First, the total CPU times for generating the schedules are compared. The percentage change in CPU time required by the fast heuristic compared with the straightforward method is -36%. The overall CPU time depends on the interaction between the computational complexity of a single decision point in the schedule and the schedule length. The experimental results indicate that since the fast heuristic method makes better decisions, compared with the straightforward method, the overall length of the schedule is reduced considerably and therefore the overall CPU time is also reduced (-36% in average). This happens despite of the fast heuristic's higher computational complexity for individual decision points. In fact, the schedule length is an important contributor to the CPU time, since longer schedules require longer temperature simulations and temperature simulation is, per se, very time

TABLE I
EVALUATION OF TRANSIENT-BASED AND FAST HEURISTICS

IC Number	IC Specifications			Percentage change in overall transition time ^a		Percentage change in test time ^b
	Number of layers	Number of modules	Number of TSV blocks	Transient solution	Transient-based heuristic	Fast heuristic
1	1	2	1	-83.88	-97.82	-16.97
2	1	4	1	-68.35	-73.05	-39.69
3	1	8	2	-64.97	-69.95	-63.35
4	1	16	3	-56.93	-62.63	-94.77
5	2	4	2	-64.37	-68.37	-8.70
6	2	8	2	-58.32	-65.94	-60.80
7	2	16	4	-57.19	-63.82	-78.17
8	2	32	6	-43.99	-55.14	-95.04
9	3	6	3	-70.44	-97.18	-75.90
10	3	12	3	-57.15	-93.17	-84.81
11	3	24	6	-84.56	-95.87	-87.08
12	3	48	9	-92.06	-94.52	-94.72
Average				-66.85	-78.12	-66.67

^a Percentage change is reported in comparison with the steady state solution (section III.B)

^b Percentage change is reported in comparison with the straightforward algorithm (section IV.A)

consuming.

The total time required to enforce a temperature map and maintain it while the tests are being applied, in addition to the time spent applying the corresponding tests is defined as the test time in this work. The percentage change in test time offered by the fast heuristic compared with the straightforward method is given in column 7 of Table I, which shows that considerable speed up (67% in average) is achieved.

The CPU times for the transient-based heuristic for different number of modules are given in Fig. 8. Even though they grow rapidly with the increase in the number of modules, for an IC with 48 modules it is still relatively short (480 sec). The CPU times for the fast heuristic will be relatively larger since the tests are also scheduled along with the heating sequences. The increase rate in the CPU times, as shown in Fig. 8, is tolerable similar to the transient-based heuristic. This was expected since these algorithms are very similar.

B. Evaluation of Temperature Map Ordering Technique

Percentage change in CPU time for A1 approach compared with the naïve approach is -266% in average. Furthermore, the overall transition time achieved by A1 is 18% smaller than the overall transition time achieved by the naïve approach.

Optimal map orders are found for some of the experimental ICs to be used for comparison purposes. It was not practical to find optimal map orders for all the experimental ICs because of the excessive search time that relatively large ICs require. The overall transition times achieved by A1 are around 23% larger than the overall transition times offered by the optimal map orders. As mentioned before, this shows that the map orders found by A1 are close to optimal, but A2 can do better. In the following A2 that includes post-PSO optimization is compared with A1 that terminates after the PSO optimization.

The greedy approach with a population size of one ($S = 1$) is used to find map orders for all of the experimental ICs. The results show 16% improvement over the A1 results, but it is 13% worse than the optimal. Increasing the population size to ten ($S = 10$), improves the results furthermore, so that there is 21% improvement over the A1 and it is only 7% worse than the optimal, but it almost doubles the search time. In short, A1 finds map orders that result in overall transition time around 23% worse than optimal. The post-PSO optimization in A2 improves the map orders by 21%, which means that it is very close to the optimum.

VII. CONCLUSIONS

Early-life failures and delay faults that are dependent on temperature-gradients introduce additional challenges to achieve efficient burn-in and delay-fault test. The negative effects of temperature gradients are more pronounced for 3D-SIC technology, since their magnitude is much larger. The challenge for burn-in is that some defects develop and cause early-life failures very rapidly when the IC is working with certain temperature maps that include large temperature gradients, which are difficult to enforce by traditional burn-in methods. The challenge for delay-fault test is that some defects can be detected only when a certain temperature map is enforced on the IC.

In order to effectively detect these defects, it is necessary to construct and maintain the specified temperature maps during burn-in and delay-fault test. The methods proposed in this paper

Figure 8. CPU time for the transient-based heuristic and the fast heuristic. utilize the available test access mechanisms in order to do so. The specified temperature maps are constructed and maintained by selectively applying high-power stimuli to the IC. Therefore, there is no need for expensive equipment to heat up the chip externally. To our knowledge, this is the first technique to achieve temperature maps for burn-in and test without any external heating mechanism.

For burn-in, a steady state solution is introduced that is fast to generate the schedules, but the schedules are slow to achieve the specified temperatures. A schedule in this case consists of a single periodic schedule for each map. The steady state solution has been extended to the transient solution which is slow in generating the schedules, but constructs the maps faster. Finally, the transient-based heuristic is proposed to support a more precise temperature model, and offer a shorter overall transition time by generating schedules that rapidly bring the IC to the specified temperature conditions. The experiments indicate that this method outperforms the transient solution. Moreover, this method is 78% faster than the steady state solution in realizing the specified temperature maps.

For delay-fault test, a straightforward method is proposed that is based on two working modes, the temperature construction mode and the test mode. The temperature construction mode works similar to the transient-based method for burn-in and brings the IC to the specified temperature conditions. Then the test mode applies the tests according to a given test schedule until the IC's temperatures exits the specified range, when the temperature construction mode is activated again. This continues until all tests are performed. Furthermore, another method (fast heuristic) has been developed to schedule the heating and cooling intervals mixed with the tests. Therefore, the test time offered by this method is reduced. The experiments indicate that fast heuristic is 67% faster in performing the tests compared with the straightforward method.

The order of the temperature maps has a considerable effect on the overall burn-in and test time. Therefore map orders need to be optimized, since they affect the optimal values for other decision variables. Experiments for map ordering show that the introduction of an initialization heuristic that adds an initial map order to the PSO's initial population speeds up the search time by 266% in average. Furthermore, the overall transition time

improves by 18% in average for burn-in. The overall transition times are further improved by 21% through introduction of a post-PSO optimization stage that consists of a greedy approach.

REFERENCES

- [1] S. Adham and E. J. Marinissen, IEEE P1838 web site. <http://grouper.ieee.org/groups/3Dtest/>.
- [2] N. Aghaee, Z. Peng, and P. Eles, "Process-variation and temperature aware SoC test scheduling technique," *Journal of Electronic Testing (JETTA)*, 2013, vol. 29, no. 4, pp. 499–520.
- [3] N. Aghaee, Z. Peng, and P. Eles, "Temperature-gradient based test scheduling for 3D stacked ICs," *International Conference on Electronics, Circuits, and Systems (ICECS)*, 2013, pp. 405–408.
- [4] N. Aghaee, Z. Peng, and P. Eles, "An efficient temperature-gradient based burn-in technique for 3D stacked ICs," *Design, Automation, and Test in Europe (DATE)*, 2014, pp. 575.1–575.4.
- [5] N. Ahmed, M. Tehranipoor, and C. P. Ravikumar, "Enhanced launch-off-capture transition fault testing," *International Test Conference (ITC)*, 2005, pp. 255–264.
- [6] Y. Bonhomme, P. Girard, C. Landrault, and S. Pravossoudovitch, "Test power: a big issue in large SOC designs," *Electronic Design, Test, and Applications (DELTA)*, 2002, pp. 447–449.
- [7] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, V. De, "Parameter variations and impact on circuits and microarchitecture," *Design Automation Conference (DAC)*, 2003, pp. 338–342.
- [8] S. A. Bota, M. Rosales, J. L. Rosello, A. Keshavarzi, and J. Segura, "Within die temperature gradient impact on clock-skew: a new type of delay-fault mechanism," *International Test Conference (ITC)*, 2004, pp. 1276–1283.
- [9] A. Calimera, E. Macii, D. Ravotto, E. Sanchez, and M. Sonza Reorda, "Generating power-hungry test programs for power-aware validation of pipelined processors," *23rd symposium on Integrated circuits and system design (SBCCI)*, 2010, pp. 61–66.
- [10] A. Carbine and D. Feltham, "Pentium® Pro processor design for test and debug," *International Test Conference (ITC)*, 1997, pp. 294–303.
- [11] K. Chakrabarty, S. Deutsch, H. Thapliyal, and F. Ye, "TSV defects and TSV-induced circuit failures: The third dimension in test and design-for-test," *International Reliability Physics Symposium (IRPS)*, 2012, pp. 5F.1.1–5F.1.12.
- [12] A. K. Coskun, J. L. Ayala, D. Atienza, T. S. Rosing, and Y. Leblebici, "Dynamic thermal management in 3D multicore architectures," *Design, Automation, and Test in Europe (DATE)*, 2009, pp. 1410–1415.
- [13] S. Deutsch, K. Chakrabarty, S. Panth, and S. K. Lim, "TSV stress-aware ATPG for 3D stacked ICs," *Asian Test Symposium (ATS)*, 2012, pp. 31–36.
- [14] D. Goswami and N. Mukherjee, "At-speed scan tests: reality or fantasy?" *International Test Conference (ITC)*, 2007, Panel 1.
- [15] Z. He, Z. Peng, and P. Eles, "Multi-temperature testing for core-based system-on-chip," *Design, Automation, and Test in Europe (DATE)*, 2010, pp. 208–213.
- [16] W. Huang, M. R. Stan, K. Skadron, K. Sankaranarayanan, S. Ghosh, and S. Velusamy, "Compact temperature modeling for temperature-aware design," *Design Automation Conference (DAC)*, 2004, pp. 878–883.
- [17] E. J. Marinissen, "Challenges and emerging solutions in testing TSV-based 2.5D- and 3D-stacked ICs," *Design, Automation, and Test in Europe (DATE)*, 2012, pp. 1277–1282.
- [18] M. Miller, "Next Generation Burn-in and Test System for Athlon Microprocessors: Hybrid Burn-in," session 5, Bits workshop, 2001.
- [19] S. K. Millican and K. K. Saluja, "Linear programming formulations for thermal-aware test scheduling of 3D-stacked integrated circuits," *Asian Test Symposium (ATS)*, 2012, pp. 37–42.
- [20] M. Mondal, et al., "Thermally robust clocking schemes for 3D integrated circuits," *Design, Automation, and Test in Europe (DATE)*, 2007, pp. 1–6.
- [21] J. Pak, M. Pathak, S. K. Lim, and D. Z. Pan, "Modeling of electromigration in through-silicon-via based 3D IC," *Electronic Components and Technology Conference (ECTC)*, 2011, pp. 1420–1427.
- [22] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization, an overview," *Swarm Intell*, 2007, vol. 1, no. 1, pp. 33–57.
- [23] O. Semenov, A. Vassighi, M. Sachdev, A. Keshavarzi, and C. F. Hawkins, "Effect of CMOS technology scaling on thermal management during burn-in," *IEEE Trans. Semicond. Manuf. (SM)*, 2003, vol. 16, no. 4, pp. 686–695.
- [24] B. SenGupta, U. Ingelsson, and E. Larsson, "Scheduling tests for 3D stacked chips under power constraints," *Journal of Electronic Testing (JETTA)*, 2012, vol. 28, no. 1, pp. 121–135.
- [25] T. Smorodin, J. Wilde, P. Alpern, and M. Stecher, "A temperature-gradient-induced failure mechanism in metallization under fast thermal cycling," *IEEE Trans. Device Mater. Rel. (DMR)*, 2008, vol. 8, no. 3, pp. 590–599.
- [26] M. Taouil, S. Hamdioui, K. Beenakker, and E. J. Marinissen, "Test impact on the overall die-to-wafer 3D stacked IC cost," *Journal of Electronic Testing (JETTA)*, 2012, vol. 28, no. 1, pp. 15–25.
- [27] A. Todri et al., "A Study of tapered 3-D TSVs for power and thermal integrity," *IEEE Trans. VLSI Syst. (VLSI)*, 2013, vol. 21, no. 2.
- [28] I. Ukhov, M. Bao, P. Eles, and Z. Peng, "Steady-state dynamic temperature analysis and reliability optimization for embedded multiprocessor systems," *Design Automation Conference (DAC)*, 2012, pp. 197–204.
- [29] G. Van der Plas, et al., "Verifying electrical/ thermal/ thermo-mechanical behavior of a 3D stack - Challenges and solutions," *Custom Integrated Circuits Conference (CICC)*, 2010, pp. 1–4.
- [30] N. S. Vinay, I. Rawat, E. Larsson, M. S. Gaur, and V. Singh, "Thermal aware test scheduling for stacked multi-chip-modules," *East-West Design and Test Symposium (EWDTS)*, 2010, pp. 343–349.
- [31] C. Yao, K. K. Saluja, and P. Ramanathan, "Temperature dependent test scheduling for multi-core system-on-chip," *Asian Test Symposium (ATS)*, 2011, pp. 27–32.
- [32] C. Yao, K. K. Saluja, and P. Ramanathan, "Thermal aware test scheduling using on-chip temperature sensors," *VLSI Design (VLSID)*, 2011, pp. 376–381.
- [33] Y. A. Zorian, "Distributed BIST control scheme for complex VLSI devices," *VLSI Test Symposium (VTS)*, 1993, pp. 4–9.