A Test-Ordering Based Temperature-Cycling Acceleration Technique for 3D Stacked ICs

Nima Aghaee, Zebo Peng and Petru Eles

Linköping University Post Print

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

The original publication is available at www.springerlink.com:

Nima Aghaee, Zebo Peng and Petru Eles, A Test-Ordering Based Temperature-Cycling Acceleration Technique for 3D Stacked ICs, 2015, Journal of electronic testing, (31), 5, 503-523.

<http://dx.doi.org/10.1007/s10836-015-5541-5>

Copyright: Springer Verlag (Germany)

<http://www.springerlink.com/?MUD=MP>

Postprint available at: Linköping University Electronic Press

<http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-123489>

A Test-Ordering Based Temperature-Cycling Acceleration Technique for 3D Stacked ICs

Nima Aghaee*, Zebo Peng, and Petru Eles

*Embedded Systems Laboratory (ESLAB), Department of Computer and Information Science,
Linköping University, 58183 Linköping, Sweden*

{nima.aghaee, zebo.peng, petru.eles}@liu.se

+46 13 28 2231 (phone*)

Abstract In a modern three-dimensional integrated circuit (3D IC), vertically stacked dies are interconnected using through silicon vias. 3D ICs are subject to undesirable temperature-cycling phenomena such as through silicon via protrusion as well as void formation and growth. These cycling effects that occur during early life result in opens, resistive opens, and stress induced carrier mobility reduction. Consequently these early-life failures lead to products that fail shortly after the start of their use. Artificially-accelerated temperature cycling, before the manufacturing test, helps to detect such early-life failures that are otherwise undetectable. A test-ordering based temperature-cycling acceleration technique is introduced in this paper that integrates a temperature-cycling acceleration procedure with pre-, mid-, and post-bond tests for 3D ICs. Moreover, it reduces the need for costly temperature chamber based temperature-cycling acceleration methods. All these result in a reduction in the overall test costs. The proposed method is a test-ordering and schedule based solution that enforces the required temperature cycling effect and simultaneously performs the tests whenever appropriate. Experimental results demonstrate the efficiency of the proposed technique.

Keywords Temperature Cycling Test, Test Scheduling, Test Ordering, 3D Stacked IC

1 Introduction

Large and frequent temperature changes (i.e., temperature cycling) create fatigue and wearout in Integrated Circuits (ICs). Temperature-cycling affects ICs by causing various damages including solder joint fatigue, fracture in bond wires, and die deformation [1]. In addition to these undesirable effects, 3D stacked ICs (3D-SIC) suffer from defects related to Through Silicon Vias (TSV). TSV protrusion and void formation in TSV are two of such defects. These effects are worsened by temperature cycling [2], [3], [4]. Furthermore, some other defects including resistive opens and stress induced carrier mobility reduction can also be worsened by temperature cycling.

Temperature-cycling exacerbates a number of defect mechanisms, as pointed out above. Therefore, operating the dies under intensive temperature cycling can effectively accelerate such failures so that they can be detected by the subsequent test, before the 3D-SIC is delivered to the customers. This procedure is called temperature-cycling acceleration [5], [6]. An example for the impact of temperature cycling on 3D-SIC is the protrusion of TSVs out of the die surface. Right after TSV fabrication, there is normally no protrusion and the TSVs have about the same length as the die's thickness. However, after a few temperature-cycles an increase in the TSV length may be observed. The TSV length will continue to increase with the number of cycles [2], [3]. After a certain amount of temperature cycling, the TSV length approaches a maximum level. Further temperature cycling will have almost no effect on the TSV length, afterwards. The TSV protrusion can be further exacerbated by the electrical current it carries [2], [3]. Therefore, operating the IC during this procedure (letting the current to flow) speeds up the cycling acceleration.

The existing procedure for temperature-cycling acceleration is based on one or multiple temperature chambers [6]. Although this procedure is usually affordable for 2D ICs, it is likely to be too expensive for 3D-SICs. Due to TSV-related defects, a larger number of dies manufactured to be a part of a 3D-SIC may require cycling acceleration compared with 2D ICs. Moreover, 3D-SIC manufacturing process includes multiple bonding stages. Corresponding to these bonding stages, pre-, mid-, or post-bond tests are introduced in order to avoid: (1) wasting a good die bonded to a bad die or stack, (2) wasting bonding effort for bonding bad dies or stacks, and (3) wasting packaging effort spent on a bad stack. Based on the cost breakdown, temperature-cycling acceleration could be beneficial at one or multiple test stages. Integrating the temperature-cycling acceleration with the tests that are performed at different stages and eliminating the need for temperature chambers will reduce the overall manufacturing costs.

Modern core-based system-on-chips, including 3D-SICs, experience excessively large test power densities [7], [8], especially since the tests are mostly scan-based. High power densities lead to excessively high temperatures, in particular for the middle dies in a 3D-SIC. Therefore, temperatures should be taken into account when planning the test process [9], [10]. This otherwise undesirable thermal effect is, however, utilized in this paper to generate large amounts of temperature-cycling. Temperature-cycling acceleration is achieved by frequent switching between high power tests that heat up the IC and pauses that allow for cooling.

A deliberate pause for cooling is called a cooling interval. It is the time interval that no stimuli are applied to a core and, therefore, the core's temperature decreases. Some cooling intervals are usually present in the original test schedule for thermal-safety reasons. More intensive temperature-cycling acceleration can be achieved by introducing additional cooling intervals and stronger heating sequences into the process. A stronger heating sequence consists of stimuli that generate larger switching activities in a core and, therefore, increases the core's temperature faster than usual. The mixture of cooling intervals and heating sequences can generate the required temperature-cycling acceleration effect.

A test sequence's bit streams define the circuit-under-test's power dissipation in combination with the previously applied test sequence (circuit's state) as well as the core's power-related properties. Consequently, the power dissipation generated by a series of tests depends on the order in which they are applied [11]. This phenomenon is employed in this paper in order to produce extreme power values for tests as well as heating sequences and consequently achieve a high speed temperature-cycling process.

This paper presents a schedule-based technique that integrates temperature cycling acceleration with testing procedure. The cycling acceleration is achieved by mixing heating sequences and cooling intervals with test sequences in an efficient order. Furthermore, tests and heating sequences are reordered so that a rapid testing and acceleration process is achieved. The proposed technique is in contrast with the existing approaches that are based on temperature chambers and can be impractical for 3D-SICs due to their unaffordable costs and limitations.

The rest of the paper is organized as follows. The related works are reviewed in Section 2. The preliminaries are introduced in Section 3. Section 4 presents motivational examples. Section 5 describes the problem formulation. Section 6 introduces a baseline method, the three-phase approach. Section 7 is the proposed integrated approach. Section 8 presents the experimental results and Section 9 presents the conclusion. A quick reference guide including abbreviations and notations is given in Section 10.

2 Related Works

2.1 Power Issues during Test

A large portion of modern ICs are core-based designs. Besides, the growing portion of 3D stacked ICs are also core-based. The main test techniques for all these ICs are scan based. A circuit's switching activity depends on the changes of its bits caused by the difference between the current and the previous test as well as the state of the scan chain's flip flops. The scan chain's state is mainly determined by the previous tests. This phenomenon is utilized in a number of test scheduling techniques to modify the tests' power profiles, as reviewed below.

A test power reduction technique based on test vector ordering is proposed in [11], [12]. The objective is to minimize the tests' average switching activity. It is demonstrated that the test ordering problem is NP-hard. Consequently, a greedy approach for finding a low-power test order is proposed. An elaborate power model based on the transition count in the scan chain is also used [11], [12].

A test ordering technique for power reduction is also proposed in [13]. A close connection between the actual number of transitions and the Hamming distance between tests is confirmed. Consequently, a fast algorithm to calculate Hamming distances is used instead of the actual transition count which is hard to calculate. A greedy heuristic is then used to find a low power test order [13].

Another test ordering technique for power reduction is proposed in [14]. The circuit-under-tests' switching activities are approximated by Hamming distances between the subsequent tests. The problem is equivalent to a travelling salesman problem. An ILP estimation and Christofides algorithm are employed to find a low-power test order [14].

A method for reducing the Test Application Time (TAT) while respecting a power budget is proposed in [15]. The method focuses on the test power peaks. These peak values depend on the order of the tests. The tests are reordered so that the power peaks for different cores are not overlapping. This leads to a minimized TAT under power constraints [15].

Reducing power variations in order to reduce the temperature variations during burn-in is discussed in [16], [17]. The variation is reduced through test reordering. An ILP approach as well as a greedy algorithm are used to properly reorder the tests. An efficient transition counting method is proposed to rapidly estimate the test power values [16], [17].

Peak power reduction by reordering the tests is studied in [18]. The peak power values are represented by a complete directed graph. Consequently a number of graph based techniques are employed to reduce the peak power. Removing those edges that their peak power is larger than a certain threshold is one of the pre-processing techniques. After that, the remaining graph is searched for a Hamiltonian path. Other techniques such as repeating a test, adding an all-zero test, and adding an all-one test are also studied [18].

2.2 Temperature Cycling and Scheduling Techniques

Existing temperature-cycling acceleration techniques are based on using one or multiple temperature chambers followed by the final test [5], [6]. This approach, in many cases, is too expensive to be performed at pre-, mid-, and post-bond stages for 3D-SICs. The shortcomings of the traditional approach include costs for running the temperature chambers as well as the time and equipment required for handling the dies/stacks between test equipment and chambers. In order to avoid these costs, in current practice, some or even all of the temperature-cycling acceleration operations are avoided. Therefore, the temperature-cycling related early-life failure rates in the final products will not be as low as it can be.

Our proposed approach for temperature cycling acceleration is mainly based on test scheduling. A number of test thermal issues that are handled using schedule-based techniques are reviewed as follows.

A burn-in technique is proposed in [19] to enforce specific temperature gradients on an IC. This results in an effective burn-in process for gradient-dependent early-life defects. A test technique is proposed in [20] to perform tests while specific temperature gradients are enforced on the IC. This helps to detect gradient-dependent defects that are usually related to signal delay and clock jitter. Temperature-gradients locations and magnitudes are represented by temperature-maps. An efficient temperature map ordering technique is proposed in [21]. The proper map order leads to faster burn-in and shorter test application time.

The usage of heating sequences is already discussed in the introduction which can be used in the temperature map enforcement approach. Heating sequences can be simply obtained by cloning of the high power tests. But, in order to have more effective heating sequences, input stimuli that generate even larger switching activities must be found. Authors in [22] have introduced an automated framework for finding high power test programs. The proposed approach is based on a meta-heuristic that generates alternative test programs and evaluates their power consumption. The alternatives with promising power consumptions are then used to generate the next generation of the test program alternatives [22]. A similar approach may be used in order to generate high power heating sequences.

A linear programming approach is used in [9] to generate thermally-safe test schedules for 3D-SICs. A temperature-based test partitioning technique is introduced in [23] in order to generate thermally-safe test schedules with a minimal test application time. A thermal-aware test scheduling approach is introduced in [10] for stacked multi-chip ICs. It minimizes the vertical temperature differences among different dies throughout the 3D IC during the test.

Two different methods for detecting temperature-dependent defects are introduced in [24] and [25]. These methods perform the tests only when the cores' temperatures are kept within the specified range for the particular test. The focus of these papers is on the temperature of the individual cores that are under test and the temperatures of other cores are not considered.

Speeding up the test by carefully planning safety margins that counteract negative effects of process variation is addressed in [26], [27]. The test temperatures are kept sufficiently low by introducing cooling intervals into the test schedule. The cooling intervals are carefully planned using temperature simulations. In addition to a fast temperature simulation technique, an adaptive scheduling approach is proposed in [27].

These existing methods for managing the chips' temperatures focus on keeping the temperatures under a global upper temperature limit (to prevent overheating) or to respect upper and lower bounds for cores (in order to target temperature-dependent or gradient-dependent defects). In all above cases, cores' temperatures are considered independent of their cycling effects.

Since burn-in was mentioned above, it must be pointed out that the temperature-cycling is different from the conventional burn-in. These two aim at accelerating different aging mechanisms. Cycling acceleration will not accelerate aging mechanisms identical to those that burn-in does and vice versa. To briefly explain this difference, let us focus only on two aging mechanisms. During burn-in the device is operated in a very hot environment with increased voltage to accelerate electromigration. This must continue for a relatively long time to allow for sufficient migration (detectable atomic built-up or depletion). On the contrary, simply operating the device at a single temperature does not create cycling-related material fatigue. It is the variation of the mechanical stress (as a result of varying temperature) that does it. The required amounts of burn-in and cycling are decided based on analytical, experimental, and empirical studies that are outside the scope of this paper. In this paper we solely focus on temperature-cycling and assume that the required amount of cycling is given by user.

The first proposal to integrate temperature cycling acceleration with test procedure was made by us in [28]. The current paper has built on the preliminary results introduced in [28] and develops an efficient technique to order the tests and heating sequences to achieve a high speed temperature cycling process. Furthermore, an accurate cycling related acceleration model that, also, includes Arrhenius acceleration is used in this paper. Additionally, this paper offers a technique to efficiently mix the remaining normal tests¹ with cycling tests¹. The proposed technique provides controlled temperature-cycling acceleration without utilizing temperature chambers.

3 Preliminaries

3.1 Circuit under Test and Test Access Mechanism

It is assumed that there are M modules (cores) in the 3D-SIC under test. These modules are located on different levels of stacked dies. The modules that are on different layers are connected using TSVs. Tests for each module can be started and stopped independent of other modules. The modules could be cores with core wrappers in a core-based design. The extension of this scenario to 3D-SIC is proposed as the IEEE P1838 standard [29]. Test stimuli are, therefore, transferred through a Test Access Mechanism (TAM) to the relevant module. It is assumed that the TAM only affords W (a positive integer number) modules to be tested at the same time. Other modules, therefore, have to queue up and wait for TAM access.

3.2 Thermal Model

In order to obtain the temperature values from power values, a thermal model that describes the thermal behavior of the IC must be used. The model used in this paper is HotSpot [30] and its extension for 3D ICs [31]:

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

All the characteristics of the thermal model are captured in two matrices $\mathbf{A}$ and $\mathbf{B}$. $\boldsymbol{\theta}$ is the temperature vector and $\mathbf{P}$ is the power. $\boldsymbol{\theta}$ and $\mathbf{P}$ consist of θ_m s and P_m s, respectively, put together in a vector format. Index m indicates the relevant module. There are a total of M modules ($m = 0, 1, \dots, M-1$). Equation 1 can be solved for time-domain assuming that the power values are constant during a period of time equal to τ , as follows [27]:

$$\boldsymbol{\theta}^\tau = \boldsymbol{\alpha} \times \boldsymbol{\theta}^0 + \boldsymbol{\beta} \times \mathbf{P} \quad (2)$$

The initial temperature is expressed by $\boldsymbol{\theta}^0$ and the temperature after a period of τ seconds (note that a fraction of a second is used in practice) is represented by $\boldsymbol{\theta}^\tau$. Matrices $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are obtained as follows [27]:

$$\boldsymbol{\alpha} = \exp(-\mathbf{A}^{-1} \times \mathbf{B} \times \tau) \quad (3a)$$

and

$$\boldsymbol{\beta} = (\mathbf{I} - \boldsymbol{\alpha}) \times \mathbf{B}^{-1} \quad (3b)$$

¹Tests targeting the cycling-dependent defects are called cycling tests and all other tests are called normal tests.

The identity matrix is denoted by $\mathbf{I}$. The above equations are explained in the following case study, assuming that there is only one module ($M = 1$) with its heat capacitance denoted by C (analogous to $\mathbf{A}$). The heat resistance between the module and the ambient is equal to R (analogous to $\mathbf{B}^{-1}$). In this case, Equation 2 can be re-written as:

$$\theta^\tau = \theta^0 \cdot \exp\left(-\frac{\tau}{R \cdot C}\right) + P \cdot R \cdot \left(1 - \exp\left(-\frac{\tau}{R \cdot C}\right)\right) \quad (4)$$

Since there is only one module, the vectors and matrices are reduced to scalar values. A larger initial temperature (θ^0), power (P), or resistance (R) results in higher final temperature (θ^τ), if other factors are kept unchanged. A larger period (τ) means that the contribution of the initial temperature is smaller while the effect of power on the final temperature is larger. In the vector form, increasing the period translates into a decreased α and an increased β . A large time-constant ($R \cdot C$) means that the initial temperature takes longer to lose its effect while power takes longer to noticeably affect the final temperature. In the vector form, increasing the time-constant translates into an increased α and a decreased β .

3.3 Temperature Cycling Model

The effect of temperature cycling can be described based on the Amount of Temperature Cycling induced fatigue (denoted by ATC_m for module m). Based on the Arrhenius-Coffin-Manson model [1], [32], ATC is estimated as:

$$ATC_m \cong \frac{\mathcal{N}_m}{k_0} \times \left(\frac{\Delta\theta_m - \theta^\epsilon}{k_1}\right)^\gamma \times \exp\left(\frac{\bar{\theta}_m}{k_2}\right) \quad (5)$$

Considering module m , in this equation $\mathcal{N}_m$ is the number of temperature cycles and $\Delta\theta_m$ is the amplitude of temperature changes during cycling. In the above equation, a regular cycling pattern is assumed. It means that the temperature monotonically increases from an arbitrary temperature θ_m^a to $\theta_m^a + \Delta\theta_m$ and then monotonically decreases back to θ_m^a . Usually, when the actual temperature curve is only a bit different from a regular pattern, the average amplitude is used for $\Delta\theta_m$. $\Delta\theta_m$ must be larger than θ^ϵ (a very small threshold value) in order to be considered in the temperature cycling calculations. However, it is not unusual to completely ignore θ^ϵ since the typical temperature changes are much larger than θ^ϵ . The effect of the average temperature is captured in the exponential term. The average temperature is expressed by $\bar{\theta}_m$. θ^ϵ , k_0 , k_1 , k_2 , and γ are constants that are obtained analytically or empirically by reliability analysts. A comprehensive explanation and details of Equation 5 can be found in [1] and [32]. As Equation 5 suggests, a large number of cycles, $\mathcal{N}_m$, a large temperature swing, $\Delta\theta_m$, or a large average temperature, $\bar{\theta}_m$, will result in a large cycling effect.

4 Motivational Examples

4.1 ATC Rate for a Simple Scenario

As an example, consider an IC with two modules ($M = 2$). Assume that the TAM can only support one module to be tested at a time ($W=1$). Assume that $\theta^{overheating} = 150^\circ\text{C}$ and $\theta^{ambient} = 30^\circ\text{C}$. The required amounts of temperature cycling are ATC_0^R and ATC_1^R for modules m_0 and m_1 , respectively. In this paper, tests that target cycling-dependent defects are called cycling tests and the other tests are called normal tests. Cycling tests can only be applied after the required amount of temperature cycling, ATC_m^R , is achieved.

A three-phase approach is introduced here: In phase 1, normal tests are scheduled. A thermal aware scheduling of tests based on the proposed approach in [33] is used. The corresponding temperature curves are shown in Fig. 1 (green for m_0 and blue for m_1). The normal tests for module m end at t_m^0 . Phase 1 starts at time 0 and end at t^0 that is defined as $\max_m\{t_m^0\}$.

Phase 2 starts by evaluating the ATC generated in phase 1. This value is less than the required ATC_m^R in this example. Therefore, phase 2 will generate additional temperature cycling. This is done by applying the heating sequences and cooling intervals. Corresponding temperature cycles can be seen in Fig. 1 from t^0 to $\hat{t}$. Time-point $\hat{t}_m$ marks the point when the required ATC_m^R is achieved for module m . Phase 2 ends when all required ATCs for all modules are met. This point is marked with $\hat{t}$ that is defined as $\max_m\{\hat{t}_m\}$. After this, phase 3 starts by applying the cycling tests. Phase 3 ends when all the cycling tests are complete. This point is marked with TAT .

Fig. 1 Temperature curves for the three-phase approach. Curves are illustrative.

Always, a small TAT is desirable. Test application time from 0 to t^0 and from $\hat{t}$ to TAT is already minimized by the given third-party test scheduling algorithm. The only TAT reduction opportunity in this three-phase approach is to speed up phase 2. This means that a large ATC should be achieved in a short time. Therefore, $ATC_m(t)/t$ should be maximized. Here we assume a uniform periodic temperature profile that means all cycles have the same amplitude. Moreover, for this motivational example we assume that in Equation 5: $k_0 = 1$, $k_1 = 1$, $k_2 \gg \bar{\theta}_m$, and $\theta^\epsilon \ll \Delta\theta_m$.

Since it is assumed that $k_2 \gg \bar{\theta}_m$, the exponential term can be ignored for the moment. Furthermore, since it is assumed that $\theta^\epsilon \ll \Delta\theta_m$, θ^ϵ could also be ignored. The ATC rate (denoted by ρ_m for module m) can, therefore, be defined as:

$$\rho_m = \frac{ATC_m(t)}{t} = \frac{N_m(t)}{t} \times (\Delta\theta_m)^\gamma \quad (6)$$

Frequency of temperature changes (i. e., the number of cycles per time unit) depends on the physical properties of the system and the amplitude of temperature changes, $\Delta\theta_m$. It is possible to achieve a high frequency (i.e., a large $\frac{N_m(t)}{t}$) if $\Delta\theta_m$ is small. A large amplitude on the other hand, may increase the ATC, only if it dominates the resulted reduction in the frequency.

4.2 Optimal Cycling in a Simplified Scenario

In order to clarify the tradeoff between the frequency and the amplitude of the temperature cycling, the physical properties of the system should be captured in the ATC rate equation (Equation 6). In the following this is done for a simple IC with only one module. The thermal model for such a case was discussed in Section 3.2, Equation 4. Remember that C is the heat capacitance and R is the thermal resistance between the module and the ambient. Assume that the heating sequence generates a power equal to P and the power during a cooling interval is zero. Assume that the temperature varies between $\mu - \sigma$ and $\mu + \sigma$. Both μ and σ are positive real numbers.

The period of a temperature cycle is denoted by T . This period consists of a rise time denoted by T_r plus a fall time denoted by T_f . T_r is the time the temperature takes to increase from $\mu - \sigma$ to $\mu + \sigma$. T_f is the time taken to decrease from $\mu + \sigma$ to $\mu - \sigma$. These values are calculated as follows. First, the system's differential equation is solved in the time domain similar to Equation 4 for a period of t (i.e., $\tau = t$):

$$\theta^t = \theta^0 \cdot \exp\left(-\frac{t}{R \cdot C}\right) + P \cdot R \cdot \left(1 - \exp\left(-\frac{t}{R \cdot C}\right)\right) \quad (7)$$

Let us denote $R \cdot C$ by RC and $P \cdot R$ by PR . For heating:

$$(\mu + \sigma) = (\mu - \sigma) \exp\left(-\frac{T_r}{RC}\right) + PR \left(1 - \exp\left(-\frac{T_r}{RC}\right)\right). \quad (8)$$

Then

$$T_r = RC \times \ln\left(\frac{\mu - \sigma - PR}{\mu + \sigma - PR}\right). \quad (9a)$$

Similarly for cooling, T_f can be calculated:

$$T_f = RC \times \ln\left(\frac{\mu + \sigma}{\mu - \sigma}\right). \quad (9b)$$

The period, T , is calculated as follows:

$$T = T_r + T_f = RC \times \ln\left(\frac{(\mu - \sigma - PR)(\mu + \sigma)}{(\mu + \sigma - PR)(\mu - \sigma)}\right). \quad (10)$$

Now, the ATC rate (Equation 6) could be re-written incorporating the physical properties of the system:

$$\rho_m = \frac{(2\sigma)^\gamma}{RC \times \ln\left(\frac{(\mu-\sigma-PR)(\mu+\sigma)}{(\mu+\sigma-PR)(\mu-\sigma)}\right)} = \frac{(2\sigma)^\gamma}{RC \times \ln(\xi)}. \quad (11)$$

Let us first focus on the optimal value for μ , assuming that σ is constant. In this case optimality happens when the denominator in Equation 7 is minimized. Considering a realistic situation, this is equivalent to finding the minimum for

$$\xi = \frac{(\mu-\sigma-PR)(\mu+\sigma)}{(\mu+\sigma-PR)(\mu-\sigma)}. \quad (12)$$

Following a closed-form approach:

$$\frac{d}{d\mu} \xi = 0 \quad \rightarrow \quad \frac{(2\mu-PR)((\mu+\sigma-PR)(\mu-\sigma) - (\mu-\sigma-PR)(\mu+\sigma))}{((\mu+\sigma-PR)(\mu-\sigma))^2} = 0 \quad (13)$$

The valid solution is $\mu = PR/2$. Here for the sake of simplicity, the ambient temperature was not included in the equations. Since the temperature model is a Linear Time-Invariant (LTI) system [27], the ambient temperature can be added later on. Assume that power and resistance values are so that $PR = 120^\circ\text{C}$. This means that considering the ambient temperature (30°C), the IC's temperature will increase to 150°C if no control is applied. Thus, the optimal value for μ is $\mu_{\text{optimal}} = \frac{120^\circ\text{C}}{2} + 30^\circ\text{C} = 90^\circ\text{C}$.

The resulted equations for finding the optimal value for σ do not have a simple closed form. Therefore, a numerical method is employed. The ATC rate ρ versus σ for $\mu = 90^\circ\text{C}$ is plotted in Fig. 2. If $\gamma = 4$ and $RC = 50 \mu\text{s}$, then the ATC rate is maximal at $\sigma_{\text{optimal}} = 55.6^\circ\text{C}$. For values of σ less than σ_{optimal} the ATC rate increases by increase in σ . This is due to the increase in amplitude, $(\Delta\theta_m)^\gamma$, dominating the decrease in frequency, $\mathcal{N}_m(t)/t$, in Equation 6. For larger σ values the ATC rate decreases by increase in σ . This is due to the increase in amplitude, $(\Delta\theta_m)^\gamma$, being dominated by the decrease in frequency, $\mathcal{N}_m(t)/t$. In other words, a very large temperature cycle takes too much time to complete.

If the assumption that $k_2 \gg \overline{\theta_m}$ does not hold, the temperature cycling rate equation, Equation 11, will be as follows:

$$\rho_m = \frac{(2\sigma)^\gamma}{RC \times \ln\left(\frac{(\mu-\sigma-PR)(\mu+\sigma)}{(\mu+\sigma-PR)(\mu-\sigma)}\right)} \times e^{\mu/K_2} = \frac{(2\sigma)^\gamma}{RC \times \ln(\xi)} \times e^{\mu/K_2}. \quad (14)$$

The inclusion of the exponential (Arrhenius) term results in a larger (or equal) optimal μ_{optimal} value. Since both the exponential term and Equation 11 are increasing when μ is smaller than $PR/2$, the optimal value cannot happen for a μ smaller than $PR/2$. After this point, the value of Equation 11 decreases while the exponential term is increasing. The optimal μ can be in this region ($\mu \geq PR/2$). Besides, the introduction of the exponential term leads to dependency of the optimal μ on the value of σ .

In the general case (without assumptions made for the motivational examples), the optimal value for μ could be very different compared with the μ_{optimal} obtained here. Moreover, the assumptions made for obtaining Equation 6 will not be valid and therefore the situation will be more complicated than discussed in the above paragraph. In such situations a numerical approach is best suited to find the optimal values for μ and σ . Moreover, in the general case, there are multiple modules competing for access to TAM and their interference makes the problem even more complicated, so complex that a heuristic is the only practical solution to deal with the problem.

4.3 Effect of the Test Application Order

In general, the circuit under test's consumed power depends on the order in which the tests are performed. Let us consider the scan chain itself. Different orders of the tests will result in different transition counts and thus different power values. Consider a 4-bit scan chain as shown in Fig. 3.

Fig. 2 ATC rate, ρ , versus σ for three-phase approach.

Fig. 3 Test orders: (a) A low power order. (b) A high power order.

Assume that 0101, 1111, and 1010 are the test stimuli. The order 1010-1111-0101, as shown in Fig. 3a, results in 12 transitions in the scan chain during shift-in. Another test order, 1111-1010-0101, as shown in Fig. 3b, results in 22 transitions and thus higher power dissipation. Assuming that the temperature of the core should be reduced, arranging the tests in their low power order may avoid an additional cooling interval. Alternatively, if the core is in its heating interval of the cycling process, the high power arrangement may replace an unnecessary heating sequence application. This will ensure that TAM is not unnecessarily occupied by dummy heating sequences. Both situations help to shorten the test application time.

5 Problem Formulation

As discussed before, along with pre-, mid-, or, post-bond tests, temperature-cycling acceleration might be beneficial. In this case, there will be tests that target cycling-dependent defects (i.e. cycling tests) in addition to other tests (i.e., normal tests). Normal tests are scheduled along with heating and cooling intervals in order to generate the required amount of temperature cycling. The cycling tests can be performed afterward.

The amount of temperature cycling can be easily calculated using Equation 5 if the temperature swings in a uniform periodic manner similar to Fig. 4a. In Fig. 4a five cycles with amplitudes equal to $\Delta\theta$ can be identified. In the general case, for example when the IC is under test, the temperature fluctuations are irregular, as shown in Fig. 4b. In this case, identifying cycles and their amplitudes is not straightforward. For such irregular patterns, the number and amplitudes of the cycles are calculated using the widely used Rainflow-counting algorithm [34].

As mentioned previously, the required amount of temperature cycling is denoted by ATC_m^R . The current amount of temperature cycling generated by normal tests or heating sequences (e.g., phase 1 and phase 2 in Fig. 1), up to a given time, t , is denoted by $ATC_m(t)$. For a certain test schedule, the temperature curves are obtained using temperature simulations. Then a fast version of Rainflow-counting algorithm, introduced in [35], calculates $ATC_m(t)$. Assuming that for $t < \hat{t}_m$, $ATC_m(t) < ATC_m^R$, only normal tests can be performed before time $\hat{t}_m$. The cycling tests can only be performed after the required amount of cycling (ATC_m^R) has been applied. Therefore, after time $\hat{t}_m$, cycling tests can be performed too. The test application time, TAT_m , marks the point that testing module m is complete. TAT_m consists of the time spent before and after time $\hat{t}_m$. The goal is to generate a schedule with a minimal overall TAT. The overall test application time is defined as $\max_m \{TAT_m\}$.

As previously discussed, the power dissipation during a test depends on the previous test, among other factors. Assuming that test $s_{m,s}$ for module m immediately follows test $s_{m,u}$, the dynamic power is expressed by $p_{m,u-s}^d$. The overall power dissipation (in the circuit under test), denoted by $p_{m,u-s}$, consists of the dynamic power, $p_{m,u-s}^d$, plus the stray power, denoted by $\widehat{p}_m$ ($p_{m,u-s} = p_{m,u-s}^d + \widehat{p}_m$). The dynamic power is caused by the circuit under tests' switching activities. The stray power is defined, in this paper, as the sum of all power values that their dissipations cannot be independently controlled with existing test controls. This includes the leakage power as well as the clock networks' power. Stray power's exact value depends on the module's current temperature since the leakage power depends on the temperature. In this paper, the stray power (including temperature dependent leakage) is taken into account.

Fig. 4 Temperature patterns: (a) Uniform periodic. (b) Irregular

It is assumed that module m has N_m tests including both normal and cycling tests. Relevant test properties can be captured in a test graph. Consider an IC that consists of two modules ($M = 2$). Assume that module m_0 has two tests ($N_0 = 2$) as shown in Fig. 5a. Module m_1 has three tests ($N_1 = 3$) as shown in Fig. 5b. Assume that one of the tests for module m_0 is a normal test (the node is marked with N) and the other is a cycling test (marked with C). A node that corresponds to a heating sequence (marked with H) is also included in the test graph. Tests and the heating sequence for module m_1 are marked in a similar manner. Total test powers are shown on the edges in Fig. 5. Usually, in the general case, there are a number of normal and cycling tests in addition to a number of heating sequences.

At each time point, during the test, there could be some tests that cannot be performed. This is due to a number of reasons including the limited capacity of the TAM as well as the cycling tests that cannot be performed before the required ATC is applied. A validity checker is used to make sure that the scheduling algorithm takes these limitations into account. The validity checker updates the set of Valid Tests (VaT) if a new test can be performed in parallel with the tests that are already selected for the current time point. It also makes sure that any test that cannot be applied in parallel with the currently selected tests does not remain in VaT. This is based on the knowledge of previously applied tests as well as the partial set of tests selected to be applied next. Moreover, the current amount of the ATC is also taken into account. For example, assume that in Fig. 5 normal tests ($s_{0,0}$ and $s_{1,0}$) have been performed previously. Assume that $s_{0,1}$ is already selected to be applied next and the required ATC for m_1 is already achieved. In this case VaT is $\{s_{1,1}, s_{1,2}, s_{1,3}\}$. Meaning that $s_{1,1}$, $s_{1,2}$, or $s_{1,3}$ can be applied in parallel with $s_{0,1}$ without violating TAM limit or ATC requirement. Although using $s_{1,3}$ (i.e., the heating sequence) does not make sense since the required ATC is already achieved, it would be a valid choice from the VaT's point of view. Note that the heating sequences can be applied repeatedly, as needed, while repeating the tests is usually unnecessary.

The goal is to schedule the tests so that all the cycling tests are performed after the required amount of ATC is achieved and the overall test application time (including the cycling process) is minimized. This is achieved by scheduling and reordering the tests and the heating sequences. High power test stimuli and heating sequences can increase the modules' temperatures. A module may become so hot that unrealistic failures show up and even the device gets damaged. In order to avoid these undesirable overheating situations, the modules' temperatures must be kept below the overheating temperature ($\theta^{overheating}$) at any time. The overheating temperature is equal to the temperature limit minus a safety margin to ensure thermal safety. The power dissipation during a pause is equal to the stray power, $\widehat{p}_m$ (including leakage).

The problem can be described as follows. The inputs to the suggested technique include the IC's thermal model, the IC's electrical model (e.g., specification of the TAM and power-related specifications), the test graph (i.e., the cycling tests, normal tests, and the switching activities of the tests and heating sequences), the ambient temperature ($\theta^{ambient}$), and the required amount of temperature cycling, ATC_m^R . The objective is to minimize the test application time. The output is the corresponding schedule that guides the application of the tests and heating sequences in proper order so that all the tests are performed rapidly and correctly.

The generated schedule will imply, for each of the modules, a certain ordering of the test graph's node. The ordering can be represented by a directed path in each of the original test graphs (e.g., graphs in Fig. 5). This directed path must visit each test node at least once and may visit heating nodes as many times as needed. Applying a test or a heating sequence is equivalent to visiting the corresponding test or the heating node. The test ordering and scheduling can also be viewed as converting the original test graph into a final path-graph. A path-graph is defined as a graph with only one directed path that connects all the nodes. There is no other edge in a path-graph except those on this unique path. The final path-graph must include all of the test nodes, while the heating

Fig. 5 Test graphs for (a) module m_0 (b) module m_1 . Test graphs consist of normal (N), cycling (C), and heating (H) nodes.

nodes are included as needed. The ordering algorithm decides at which point to insert a node taken from the original test graph into the final path-graph.

6 Three-Phase Approach

The basics of the three-phase approach are briefly explained in Section 4.1. Section 4.2 presents a technique to find the best temperature interval ($\mu - \sigma$ to $\mu + \sigma$) for a simplified scenario. As discussed before, if the coefficient k_2 (in Equation 5) is much larger than the average temperature ($k_2 \gg \overline{\theta_m}$) and the high temperature level ($\mu + \sigma$) is smaller than the overheating temperature, $\theta_{overheating}$, everything in Section 4 would be fine. However, often these assumptions are not valid, for example the overheating temperature may be relatively low compared with $\mu + \sigma$. For the example in Section 4.2, $\mu + \sigma$ is equal to 145.6°C while the overheating temperature might be 120°C. There are some other complications, as well. In practice there are a number of modules, instead of one, and their temperatures depend on each other due to heat transfer. Furthermore, the power values fluctuate with time. Besides, power values include the stray powers that depend on the temperature due to the temperature dependent leakage currents. Additionally, the modules may not be able to receive their heating sequences at desired times due to the TAM limitation. New approaches capable of taking all these situations into account are, therefore, proposed in the following.

As discussed in Section 4.1, in phase 1 and 3 the tests are scheduled using a thermally safe third-party algorithm. It is assumed that these algorithms perform optimization to reduce the test application time. Our focus will therefore be on phase 2 where new algorithms can be designed to minimize the test application time. This was demonstrated using a small example in Section 4.2. Assume that in phase 2 the temperature of module m is intended to swing between a low temperature level θ_m^L and a high temperature level θ_m^H ($\theta_m^L < \theta_m^H$). In comparison with the example in Section 4.2, θ_m^L and θ_m^H have roles similar to that of $\mu - \sigma$ and $\mu + \sigma$, respectively.

The heating sequences are assumed to be powerful enough to raise the module's temperature to θ_m^H . The high temperature level should always be lower than the overheating temperature ($\theta_m^H < \theta_{overheating}$) to avoid any kind of damage. Since all the normal tests and all the cycling tests are to be separately scheduled using third party algorithms and then performed in two isolated phases (phase 1 and 3), there is no need to represent them in the test graph. Consequently the test graph reduces to only include the heating nodes (nodes marked with H in Fig. 5). This simplifies the problem of finding a proper path in this reduced graph. A greedy approach is used here and the heating node that offers the highest heating power is selected to follow the current node.

Immediately after the temperature reaching its peak at θ_m^H , a cooling interval is introduced to reduce the temperature back to θ_m^L . Then, for the sake of a fast cycling, the heating sequence must be immediately applied again. However, the TAM might not be available at this moment. Consequently, the temperature may fall below θ_m^L from time to time. An on-the-fly approach is used to schedule the heating sequences for phase 2 based on the simulated temperatures. The temperatures that are obtained by simulation are then compared with θ_m^L and θ_m^H in order to generate the schedule. Heating sequences for different modules will compete for access to TAM. The priority is decided based on the following equation.

$$\pi_m = (\theta_m^L - \theta_m) \times \frac{ATC_m^R}{\epsilon + ATC_m} \quad (15)$$

The priority is higher if the module's current temperature is much below θ_m^L . Note that the priorities are calculated only for modules that need heating, therefore $\theta_m < \theta_m^L$. The reason for the inclusion of this difference term (i.e., $\theta_m^L - \theta_m$) in the priority assessment is that if a module gets really cold, it takes too much time to warm it up again. Therefore, it is a good idea to give a higher priority to the colder modules. A module that has a large amount of temperature cycling left to fill has also a higher priority. This is indicated by $\frac{ATC_m^R}{\epsilon + ATC_m}$. Such a module is likely to need a relatively long time to achieve its required ATC. Consequently, it is likely that at the later stages of phase 2 this module remains alone. This implies that the interleaving opportunities for TAM access will be reduced. Consequently TAM utilization may decrease and test application time may increase. A small value, ϵ , is added to the denominator in order to prevent numerical problems when ATC is zero (e.g., at the beginning of phase 2, if there have not been any normal test). Both θ_m and ATC_m depend on time and are shortened forms of $\theta_m(t)$ and $ATC_m(t)$, respectively, at time t .

The test application time for the schedules generated by this on-the-fly approach depends on θ_m^L and θ_m^H . These temperature levels could assume a range of values provided that $\theta_m^{stray} \leq \theta_m^L < \theta_m^H < \theta_{overheating}$. The temperature that corresponds to the stray power is called stray temperature and is denoted by θ_m^{stray} (always $\theta_{ambient} \leq \theta_m^{stray} < \theta_{overheating}$). Temperature of a module cannot be lower than this because of the stray power dissipation (including leakage). The combination of these temperature levels (θ_m^L and θ_m^H) among different modules affects the test application time. The proper values for these decision variables will be found in an external optimization loop, as shown in Fig. 6. In the inner scheduling loop, the temperature levels (i.e., decision variables) defined by the outer optimization loop are used to generate the schedule. In Fig. 6, the scheduler boxes inside the dashed box represent multiple copies of the inner scheduling algorithm. However only one of such schedulers is sufficient to perform the optimization, multiple of them are used in parallel to speed up the procedure.

The outer optimization loop makes use of a Particle Swarm Optimization (PSO) algorithm. PSO is a well-known iterative population-based optimization metaheuristic. For each alternative solution in the PSO's population, on-the-fly scheduling is performed (inside the dashed box in Fig. 6) to compute the cost function (i.e., TAT). A canonical form of PSO [36] is used in this paper in a straightforward manner. The algorithm starts from a random initial population, similar to other population based metaheuristics (e.g., evolutionary methods). The population is referred to as a *swarm* in PSO terms. An individual in the population is referred to as a *particle*. Each particle goes through a number of alternative solutions, one at a time, as the algorithm iterates. Each particle has a *location* in the search space (i.e., the current alternative solution). A particle records the best solution it has ever encountered, the *local best*. The swarm records the best solution its particles have ever encountered, the *global best*. Based on these best solutions and the previous alternative solution a *velocity* is determined which also incorporates some randomization [36]. Velocity is the vector that determines the next location for a particle. The particles move throughout the search space in a guided random manner until they converge to a near optimal solution.

7 Integrated Approach

Let us assume, now, that the orders in which normal test nodes (e.g., nodes marked with N in Fig. 5) must be visited are given. Furthermore, assume that the order for heating sequence nodes (e.g., nodes marked with H in Fig. 5) are also given. This means that the original test graph is broken down into a number of sub-graphs. This includes two separate directed path-graphs, one for normal tests and the other for the heating sequences among other sub-graphs. This simplified scenario which involves two separate path-graphs will be discussed first and a path-graph scheduling algorithm will be introduced in Sections 7.1–3. Afterwards, Section 7.4 explains how to employ this path-graph scheduling algorithm to solve the original problem that involves the original test graph (i.e., the problem formulation in Section 5). Fig. 7 shows how these components are put together. An example in the following paragraphs (using Fig. 8) explains some of the blocks of Fig. 7. The remaining blocks are explained later on.

Fig. 6 Particle swarm optimization algorithm used to minimize the test application time. Inside the dashed box, copies of the scheduling heuristics are performed in parallel for a number of particles.

Fig. 7 Integrated scheduling approach. The decision variables are highlighted with gray.

A small example, in Fig. 8, explains how all these blocks work together to generate a schedule¹. Let us assume that path-graph scheduling (i.e., Path-graph scheduling block in Fig. 7) determines that the module m_0 must receive heating at i_0 test cycle. Test cycles are shown in Fig. 8f. It asks test graph node ordering (i.e., Node ordering block in Fig. 7) for options. Test graph ordering replies by two options (as shown in Fig. 8d): The first option is $[s_{0,0}, s_{0,2}]$ that is a path-graph consisting of high power normal test nodes. The second option is $[s_{0,4}, s_{0,6}]$ that consists of heating nodes. This interaction is depicted in Fig. 7 as the loop between path-graph scheduling block and node ordering block. The output of the node ordering block is monitored to determine if all tests are completed.

The path-graph scheduling decides to go on with $[s_{0,0}, s_{0,2}]$. Now, the power values are known and temperatures simulation is performed to obtain the temperatures. This interaction is depicted in Fig. 7 as the loop between path-graph scheduling block and temperature simulator block. The simulated temperatures are plotted in Fig. 8ab. As module m_0 heats up, module m_1 is slightly warmed up by the transferred heat from m_0 . It is assumed that the die in this example consists of only two modules. Moreover, it is assumed that the test access mechanism provides access to only one of the modules at a time. The module that occupies the TAM is depicted in Fig. 8c.

Every decision (i.e., change in the schedule) is recorded in the schedule as a new entry. Each entry consists of the corresponding cycle in addition to the node and state for each and every module. For example a decision was made at cycle i_0 to start $s_{0,0}$. This is registered in the schedule as shown in Fig. 8f-j. Applying $s_{0,0}$ continues smoothly to the end and then $s_{0,2}$ starts (at i_1) as previously suggested by the node ordering block.

At cycle i_2 the temperature of m_0 reaches the high level and cooling is required. Node ordering block is consulted and it returns $[s_{0,8}, s_{0,7}]$ that consists of low power normal tests. The other alternative is a pause (cooling interval). Since the application of $s_{0,2}$ is not complete, application of low power normal tests is not possible. Therefore, a cooling interval is introduced. This frees the TAM that the other module can utilize. Node ordering block suggests either $[s_{1,4}, s_{1,2}]$ or $[s_{1,5}, s_{1,8}]$. The scheduler decides to go with $s_{1,4}$, a new entry for i_2 cycle is added to the schedule and then the simulations and scheduling continue. Note that if the temperature reaches the overheating limit (that is higher than the high level discussed here and therefore is not shown in Fig. 8) only a pause can be selected (definitely not a low power test).

At cycle i_4 the temperature of m_1 reaches the high level and cooling is required. Node ordering block is consulted and it returns $[s_{1,9}, s_{1,6}]$ that consists of low power normal tests. The other alternative, as always for cooling, is a pause. Since the application of $s_{1,2}$ is not complete, application of low power normal tests is not possible. Therefore, a cooling interval is introduced. This frees the

¹ This example is not exact. The exact explanation is presented later on.

Fig. 8 An example for schedule generation. Curves are illustrative.

TAM that the other module can utilize. Since $s_{0,2}$ was pending, it is resumed and there is no need to consult the node ordering block at the moment. However, it is consulted later on at i_5 .

At cycle i_6 the temperature of m_0 reaches the high level and cooling is required. The node ordering block is consulted and it returns $[s_{0,7}, s_{0,8}]$ that consists of low power normal tests. Obviously, the other alternative is a pause. This time the application of $s_{0,3}$ is complete and, therefore, $s_{0,7}$ can actually be selected. However, the path-graph scheduler decides that, in any case, a pause is better. Note that before a node is started or resumed, its validity (VaT as discussed in section 5) is checked. If not in the VaT list, either another alternative must be selected or the module must wait until incompatible tests are complete. The above process, as explained in Fig.8, continues until all tests are performed.

7.1 Path-Graph Scheduling Algorithm

The test application time could be reduced if normal tests (phase 1) are integrated into the temperature-cycling acceleration process (phase 2). For example, a test can be employed to heat a module and avoid an unnecessary inclusion of a heating node. It may happen that a test is not powerful enough to increase the modules' temperature to θ_m^H and yet it is beneficial to include it to partially heat the module. A heating node is introduced afterwards to rapidly increase the temperature up to θ_m^H . Similar to this heating scenario, a mixed cooling scenario is also possible. The benefit of these mixing scenarios is that although the temperature will change slowly (increasing the test application time), a part of the tests is being applied (decreasing the TAT). In a mixed cooling scenario, a low power test is introduced when the temperature must decrease to create a cycle. Albeit the decrease in the module's temperature, the temperature may not decrease to θ_m^L . A cooling interval is then introduced to complete the cycle.

Assume that a high power test is being applied in a heating scenario as shown in Fig. 9a. Assume that the high-power test's power for the current time interval is denoted by P_m^{HT} . This power rapidly increases the temperature at the beginning. Assume that this level of power is applied for a long time. In this case a steady state temperature equal to θ_m^{SS} will eventually be reached. As the current temperature approaches θ_m^{SS} , the heating rate decreases. The derivative of the temperature (i.e., heating rate) is shown in the lower part of Fig. 9a. When the difference between the heating-sequence's heating rate and the test's heating rate increases beyond a certain threshold ($threshold^H$

Fig. 9 Thresholds in the integrated approach: (a) Heating and (b) Cooling.

in Fig. 9a), it is time to switch to the heating sequence. This will rapidly increase the temperature to θ_m^H . Temperature caused by heating sequence (shown as the red curve in Fig. 9a) introduces a heating rate much larger than that of the test. Therefore, it is better to save the rest of the tests for a time that the initial temperature is lower and the tests can offer a large heating rate. The rate of temperature change (heating rate in this case) is $\frac{d}{dt}\theta$. Therefore the condition on heating rate is:

$$\frac{d}{dt}\theta^{HS} - \frac{d}{dt}\theta^{HT} > threshold^H. \quad (16)$$

The temperature when the heating sequence is applied is denoted by θ^{HS} . When the high-power test is applied, the temperature is denoted by θ^{HT} . The heating rate can be calculated based on the current temperature and upcoming power values using Equation 1:

$$\frac{d}{dt}\theta^{HT} = A^{-1} \times (P^{HT} - B \times \theta^{HT}). \quad (17)$$

Combining Equation 16, Equation 17, and the equivalent of Equation 17 for the heating sequences results in:

$$A^{-1} \times [(P^{HS} - B \times \theta^{HS}) - (P^{HT} - B \times \theta^{HT})] > threshold^H \quad (18)$$

Considering the fact that at the moment of decision making, there is only one actual temperature, θ ($\theta = \theta^{HS} = \theta^{HT}$), the condition can be further simplified to:

$$A^{-1} \times (P^{HS} - P^{HT}) > threshold^H. \quad (19)$$

This could be re-written to have the condition expressed for the power values:

$$(P^{HS} - P^{HT}) > A \times threshold^H. \quad (20)$$

Renaming ($A \times threshold^H$) to ψ results in:

$$P^{HS} - P^{HT} > \psi. \quad (21a)$$

Similarly, for the situation that the temperature must decrease (as shown in Fig. 9b), the proper condition for switching from a test to a cooling interval is:

$$P^{LT} - \hat{P} > \varphi. \quad (21b)$$

The power of the low-power test is denoted by P^{LT} and the power of the cooling interval (i.e., the stray power) is denoted by $\hat{P}$. Switching to the cooling interval when indicated by the above equation speeds up the cooling. This way, the normal tests are employed in an efficient way during temperature-cycling process so that the overall test application time is further reduced.

According to Equations 21ab, the scheduling heuristic does not need to compute the derivatives of the upcoming tests' temperatures. Instead, it is sufficient to compare the upcoming power values. Whenever the inequality in Equation 21a is satisfied, test nodes are followed by heating nodes and whenever the inequality in Equation 21b is satisfied, the testing is paused for cooling purpose. The variables ψ_m and φ_m (elements that construct ψ and φ vectors), are to be optimized along with θ_m^H and θ_m^L , in the outer optimization loop, to achieve a short test application time. These variables are optimized using a canonical form of particle swarm optimization similar to the one explained in Section 6. The path-graph scheduling is shown in Fig. 7 as a part of the scheduling algorithm. Since the optimization process is similar to the PSO discussed in Section 6, Fig. 7, as a whole, can be viewed as one of the scheduler boxes shown inside the dashed box in Fig. 6. The alternative decision variables shown above Fig. 7 come from Fig. 6.

7.2 Length of the Power Averaging Window

The average upcoming powers (i.e., P^{LT} , P^{HS} , and P^{HT}) can be calculated for a short segment of the tests or heating sequences that immediately follows. The shortest length of this segment is denoted by λ_m for module m . Having a much shorter segment than λ_m leads to higher computational effort without a significant improvement in the accuracy. Taking multiple λ_m s into account helps to obtain a long-term estimate of the power values. A much longer minimal segment length than λ_m is not desirable since an accurate estimate becomes unlikely to achieve.

The proper value of λ_m depends on the dynamics of the system. Consider a λ_m that corresponds to $100x$ percent ($0 < x < 1$) of the final response to a step input. Here, the final response is the steady state temperature and the step input is when zero input power is followed by a constant power. Assuming a constant power, the temperature equation in the time-domain can be written according to Equations 2 and 3. We assume that the step response starts from the initial temperature equal to zero ($\theta^0 = \mathbf{0}$). Replacing θ^τ with the $100x$ percent of the final temperature results in

$$x \times \theta^{SS} = \beta(t) \times P. \quad (22)$$

Since the steady state situation means negligible variations in the temperature, the temperature derivative can be assumed zero ($\frac{d}{dt}\theta^{SS} = \mathbf{0}$). By combining this observation with Equation 1, the steady state temperature can be described as:

$$\theta^{SS} = B^{-1} \times P. \quad (23)$$

Replacing θ^{SS} from the above equation and β from Equation 3b in Equation 22 results in

$$x \times I \times B^{-1} \times P = (I - \alpha(\tau)) \times B^{-1} \times P. \quad (24)$$

Here we are going to replace a scalar time, τ , with a matrix of time, Λ . Besides, we assume that the equivalence of the sides in the above equation is achieved by satisfying the following equation (Equation 25). These assumptions work for estimating the values of λ_m 's [37].

$$x \times I = I - \alpha(\Lambda). \quad (25)$$

Replacing α from Equation 3a results in

$$\exp(-A^{-1} \times B \times \Lambda) = (1 - x) \times I. \quad (26)$$

And finally

$$\Lambda = B^{-1} \times A \times \ln(1/(1 - x)). \quad (27)$$

$(B^{-1} \times A)$ is the time constants matrix [37] (analogous to RC in Equations 4, 7–11, and 14 for a single-element case) and Λ is the matrix that contains the values of λ_m s. A diagonal element in Λ (i.e., $\lambda_{m,m}$ that is denoted by λ_m) represents the proper minimal length for averaging the upcoming test powers for module m . A λ_m 's value obtained this way is not too short and will contain the required information. On the other hand, the use of such λ_m values prevents the temperature changes that are larger than $x \times \theta_m^{SS}$ from going unnoticed. This percentage, x , is only used for estimating the upcoming tests' average powers. The temperature simulations are always performed based on the original power sequence. Therefore, the value of x will not affect them.

A set of experiments reported in [28] evaluate the accuracy of λ_m values estimated using Equation 27. The accurate value for λ_m is obtained based on high quality temperature simulations. The average error is found to be around five percent. Besides, for 95 percent of the samples, the error is smaller than 14 percent. This confirms that the above estimates have sufficient accuracy, in practice.

7.3 Priorities for TAM Access

Normal tests, heating sequences, and cycling tests may compete for access to TAM. The priority for letting module m to access TAM is assigned based on the following criterion.

$$\pi_m = (\theta_m^L - \theta_m) \times \frac{ATC_m^R}{\epsilon + ATC_m} \times r_m \quad (28)$$

Similar to Equation 15, the priority is higher for the colder modules and for the modules with larger remaining ATC. Moreover, a module's priority is higher if it's current amount of remaining tests (denoted by r_m) is larger. Both normal and cycling tests are taken into account for r_m calculation. The motivation for inclusion of r_m , similar to that of ATC_m , is to avoid a small number of modules running long after all other modules have completed their tests. Such a scenario implies inefficient use of TAM due to lack of interleaving opportunities. In the above equation, θ_m^L is used to calculate

the priority for a module running a heating sequence. For normal or cycling tests, instead of θ_m^L , a “stop cooling temperature”, θ_m^S , similar to the one introduced in [33] is used in Equation 28. In case of the cycling tests, $\frac{ATC_m^R}{\epsilon + ATC_m}$ is replaced with one (removed from Equation 28) since the value of ATC is not relevant anymore (after the required ATC is achieved). The priorities are calculated based on dynamically changing amount of temperature cycling, temperatures, and the size of the remaining tests. These values are sent from the path-graph scheduling box in Fig. 7 and the resulted priorities are sent back to it.

7.4 Node Ordering in the Test Graph

The path-graph scheduling algorithm cannot be directly employed to solve the problem that involves the original test graph (e.g., Fig. 5). The path-graph scheduling needs to know, at certain time points, the order of the test nodes that will follow and sometimes also the order of the heating nodes. A path-graph format is usually used to represent the node order for different sub-graphs. These orders may change during the scheduling, as different nodes are being included in the schedule’s final path-graph. A node ordering technique is introduced in this section to determine the proper node orders, over and over again, during the scheduling process. This node ordering technique, put together with the path-graph scheduling algorithm (Section 7.1), solves the problem that involves the original test graph, as shown in Fig. 7.

For example, consider a test sub-graph with three normal and three heating nodes, as shown in Fig. 10a. The graph is simplified for the situation in which the required ATC is not achieved yet. Therefore, all cycling tests can be safely removed, for the moment, from the original test graph. Assume that the node $s_{0,0}$ (a normal test) is already included in the schedule. Assume that after $s_{0,0}$ completion, the temperature must increase in order to create a cycle. The path-graph scheduling only needs to know what the sequence of the normal test nodes (e.g., $[s_{0,2}, s_{0,1}]$ in Fig. 10b) would be if it decides to continue the schedule with the high power tests. Furthermore, it needs to know what the sequence of the heating nodes (e.g., $[s_{0,4}, s_{0,3}, s_{0,5}]$ in Fig. 10b) would be if it decides to continue with the heating sequences. Based on the average power of these upcoming tests and heating sequences, the path-graph scheduling algorithm decides which node to include in the schedule, next.

For the above heating case, the high power orders for tests and heating nodes are desirable. Similar to the above heating scenario, a node ordering is performed also for the cooling scenario. In this case, there are no heating nodes, and a low power order for the test nodes is the only thing to be determined.

Let us continue with the test ordering for a heating scenario. Assume that just one node can be considered at a time to determine the high power order. Continuing with the previous example where $s_{0,0}$ is already selected, if $\overline{p_{0,0-1}}$ is large than $\overline{p_{0,0-2}}$, then $s_{0,1}$ is selected to immediately follow $s_{0,0}$. Since only one node is left, the node ordering for the test nodes must be $[s_{0,1}, s_{0,2}]$.

Instead of only one node at a time, two nodes at a time, also, can be considered to determine the high power order. In this case, the decision is made based on the average power value for two nodes. For example if

$$\frac{(\overline{p_{0,0-2}} + \overline{p_{0,2-1}})}{2} > \frac{(\overline{p_{0,0-1}} + \overline{p_{0,1-2}})}{2}$$

Then, the node ordering for the test nodes must be $[s_{0,2}, s_{0,1}]$. The average power value, if node $s_{0,2}$ follows node $s_{0,1}$ is denoted by $P_{1,2}^0$. Therefore, in the above example, $P_{2,1}^0 > P_{1,2}^0$. The number of nodes taken into account at a time could be larger. Moreover, it might be helpful to consider only a part of the test sequence at the beginning of a node. Therefore, the ordering criterion can be generalized to consider the power values inside a power assessment window. The length of the

Fig. 10 Node ordering: (a) Original graph (b) Ordered during the scheduling process for the time point that comes just after $s_{0,0}$ has been scheduled.

power assessment window is $\omega^H \times \lambda_m$ cycles (of high-power test or heating sequence). Assume that a node consists of η samples and $\omega^H \times \lambda_m$ cycles is equal to v nodes plus κ cycles ($\omega^H \times \lambda_m = \eta \times v + \kappa$). This means that χ nodes ($\chi = v + 1$) will be involved. Assuming $[s_{0,1}, s_{0,2}, \dots, s_{0,v}, s_{0,\chi}]$ as the supposed node order, its average power is:

$$P_{1,2,\dots,\chi}^m = \frac{\eta \times \sum_{i=1}^v \overline{p_{m,j-i}} + \sum_{k=0}^{\kappa-1} p_{m,v-\chi}(k)}{\omega^H}. \quad (29)$$

It is assumed that node i is visited immediately after node j ($j = i - 1$ for $P_{1,2,\dots,\chi}^m$). Note that unlike the test nodes that are visited only once, the heating nodes may be repeated as needed. For example the heating nodes could have $[s_{0,3}, s_{0,3}, s_{0,3}]$ as the order, although this has not happened in Fig. 10b. Similar to ω^H , which is for heating situations, ω^C is used for the cooling situations. We use ω in this paper to refer to both of the ω^H and ω^C , indifferently.

A small ω results in fast schedule generation, but the generated schedules might not be as short as they would have been with a large ω . A large ω , on the other hand, results in a slow schedule generation. Moreover, a too large ω may delay the use of some of the best heating sequences so much so that they are left unused at the end. The proper ω^H and ω^C values are obtained in the external optimization loop. In the inner ordering/scheduling loop, the ω values defined by the outer optimization loop are used to generate the orders and the schedule. The outer optimization loop consists of a particle swarm optimization algorithm as described before.

After the required amount of cycling is achieved, the remaining normal tests and cycling tests must be performed. In this case, heating nodes as well as the already applied normal tests can be safely removed from the original test graph. The newly created test sub-graph must be converted to a path-graph whenever the path-graph scheduling algorithm (detailed in Section 7.1) demands a new node.

The module temperature may be high due to its previous activities or because of a high temperature in adjacent modules (heat transfer among modules). When the module's temperature is too high and close to the overheating limit ($\theta^{\text{overheating}}$), it might be helpful to find a node ordering that swiftly reduces the power. This is important in a short-time window and moreover usually a rather low-power test sequence may be found if the power-assessment window is rather short. It might not matter if this node order results in a higher test power some time later, since then the module might be cold. In such an emergency situation a short power-assessment window, denoted by ω^E , is used. There is an emergency situation if the current temperature is larger than the emergency temperature limit, denoted by θ^E . If the temperature is less than θ^E then the situation is ordinary. In any case, the nodes must be ordered in a low power configuration as detailed above. The length of power-assessment window in this ordinary situation is denoted by ω^O . It might be helpful to have a long ordinary power-assessment window ($\omega^O > \omega^E$) to avoid large switching activities in a long-term sense (as opposed to short-term low-power in emergency situations). The value of θ_m^E is optimized in the outer optimization loop along with ω^H , ω^C , ω^E , ω^O , ψ_m , φ_m , θ_m^S , θ_m^L , and θ_m^H . This outer optimization loop is similar to Fig. 6 and sends the alternative decision variables to the integrated scheduling algorithm, as shown in Fig. 7. The alternative decision variables shown coming to Fig. 7 are from Fig. 6 and the generated schedule and its corresponding TAT shown going out of Fig. 7 are used in Fig. 6.

The search to find the best order (e.g., a path in a graph similar to Fig. 10a) is performed using a branch and bound approach which searches the graph down to a depth equal to $\omega \times \lambda$ cycles. The cost function (average power similar to Equation 29) can be replaced with the accumulated power since all the alternatives have the same ω . When a low power order is required (corresponding to the situation in which ω^C , ω^E , or ω^O are used), the search can be very fast, since after a relatively good path is found, the bad candidates' accumulated power values rapidly exceed the already found relatively small power value. Consequently, the inferior candidate paths are rapidly discarded. The search for heating situations may take longer but, nevertheless, the overall schedule generation procedure is adequately fast.

7.5 Remarks

The proposed technique uses temperature simulations in order to generate a test schedule that has certain temperature characteristics. We are using a good simulator and therefore there is no large temperature error. Temperature error is defined as the difference between the actual temperature and the simulated one. Since the error is minor, a safety margin is sufficient to prevent overheating, as suggested in section 5.

Moreover, a separate safety mechanism can be added on top of the proposed technique. For example, a temperature-sensor-based system that halts the testing procedure whenever the temperature is too high. Since the temperature error is small, this safety mechanism will seldom interfere with the testing procedure. Therefore, its impact on the average test application time and the applied amount of temperature cycling is negligible.

To ensure that a sufficient amount of cycling has been applied before the related tests, a slightly larger amount of require cycling can be assumed. The situation with large temperature errors is outside the scope of this paper. However, one can think of sensor-based approaches similar to the adaptive techniques introduced in [27].

It is assumed that a node in the test graph can be paused and resumed. This is required for on-demand cooling as well as partitioning and interleaving. In other words a session-less testing scheme is used. A certain module can pause and resume its test but it cannot change to a different node before it completes the node that it has already started.

A node in a test graph may consist of a single test vector or a number of vectors that are applied one after the other. In general, a node in the test graph consists of a single test vector and therefore the test graph is large. Albeit the test-graphs' large sizes, the scheduling heuristic is capable of handling them since it is very fast. However, if the number of test vectors is excessively large, then the schedule generation may become slow. In such a situation, multiple test vectors can be grouped into a single node in the test graph. Ideally, nodes that their different orders do not cause large power dissipation differences should be grouped together. This reduces both the computational effort and the loss of ordering effectiveness. The test-vector clustering problem (i.e., how to group the test vectors into the nodes of a test graph) is, however, outside the scope of this paper.

It should also be mentioned that there can be scenarios that using a chamber-based technique is required. For example, after the packaging, to perform cycling tests targeting the IC features that are external to dies, a chamber-based technique is required.

A chamber-based approach enforces, however, the maximal cycling acceleration on all modules. This may lead to longer overall test time and unnecessary aging of modules that require less cycling acceleration. The integrated approach, on the other hand, can be faster and cheaper than the chamber-based approach. Moreover, it supports different amounts of temperature cycling for different modules. For example one module can receive very little cycling acceleration, while another module receives a very large cycling acceleration, as needed.

8 Experimental Results

Experiments have been performed to demonstrate that the proposed technique can efficiently achieve desired temperature-cycling accelerations. Moreover, it is demonstrated that the proposed integrated approach offers a smaller test application time and, therefore, outperforms the three-phase approach. However, if the normal or cycling test schedules provided by a third-party have to be used, three-phase approach must be chosen. In the following, first the cycling acceleration effect is demonstrated in section 8.1 and then, in section 8.2, the performance of the proposed approach is discussed.

8.1 Cycling Acceleration

The proposed integrated approach is used to perform tests and cycling acceleration for an IC with two modules, as a demonstrator example. It is assumed that the TAM in this example can only support one module to be tested at a time ($W=1$). The corresponding temperature curves are plotted in Fig. 11a. At the beginning (before 20ms) there are many normal tests that are properly mixed with heating sequences and cooling intervals in order to create a high cycling rate. As time goes on, the number of normal tests that can be effectively used reduces and therefore the majority of cycling is generated by mix of heating sequences and cooling intervals (which is, in general, faster and more effective). Around 100ms the required amounts of temperature cycling for the two modules are met and the cycling tests as well as the remaining normal tests can be applied until all tests are performed (around 110ms).

As more and more temperature cycles are performed (as in Fig. 11a), the amount of temperature cycling accumulates as suggested by the increasing accelerated time in Fig. 11b. The vertical axis in Fig. 11b is the accelerated cycling time and the horizontal axis is the actual time. Moreover, the temperature curves in Fig. 11a are used to compare the proposed integrated approach (Alternative 1, below) with a chamber-based technique (Alternative 2):

Fig. 11 Cycling Acceleration.

Alternative 1

Let us evaluate our proposed integrated approach here. A middle section of the temperature curve from Fig. 11a is magnified in Fig. 11c for module m_0 . Temperature swings between 65°C and 83°C resulting in a temperature-cycle amplitude equal to 18°C ($\Delta\theta_m = 18^\circ\text{C}$). The average temperature is approximately 74°C ($\bar{\theta}_m \cong 74^\circ\text{C}$). A temperature cycle happens in $(1318 - 1266) \times 4000$ test cycles. The test is performed at 100MHz. Therefore:

$$\text{cycle rate} \cong \frac{1}{(1318-1266) \times 4000 / (100 \times 10^6)} = \frac{1}{2.08 \times 10^{-3}}.$$

Equation 5 should be used to calculate the ATC value. Here we assume that $\frac{1}{k_0} \cong 10^{-6}$, $k_1 \cong 1$, $\theta^\epsilon \cong 0$, $\gamma \cong 4$, and $\frac{1}{k_2} \cong 1.07 \times 10^{-4}$. Therefore, the ATC value achieved in a second is:

$$ATC^{A1} \cong \frac{1}{2.08 \times 10^{-3}} \times 10^{-6} \times (18)^4 \times \exp(74 \times 1.07 \times 10^{-4}) \cong 50.87$$

Alternative 2

Assume a chamber-based approach that uses Thermotron® test chamber number SE-400-15-15. According to its specifications, this chamber can create a temperature-cycle, similar to that of Alternative 1, in approximately 380 seconds. Therefore:

$$\Delta\theta_m = 18^\circ\text{C} \text{ and } \bar{\theta}_m \cong 74^\circ\text{C}, \text{ cycle rate} \cong \frac{1}{380}.$$

The ATC value achieved in a second is:

$$ATC^{A2} \cong \frac{1}{380} \times 10^{-6} \times (18)^4 \times \exp(74 \times 1.07 \times 10^{-4}) \cong 278 \times 10^{-6}.$$

The amount of temperature cycling per second achieved by chamber-based technique is around 278×10^{-6} while the integrated approach achieves around 50.87. This is a large margin (almost 180000 times²), however as mentioned in section 7.5 there are some cases that a chamber-based technique must be used (e.g., for cycling test of the die-external package components).

² Although other chamber setups may perform better, their corresponding margins will be still very large.

8.2 Performance of the Integrated Approach

The proposed techniques are evaluated on a set of 24 experimental ICs as detailed in Table 1. Column 1 indicates the IC's serial numbers. These ICs have one to four stacked dies (column 2). The ICs with one layer (number 1 to 6) correspond to dies at the pre-bond test stage. The ICs with more than one layer represent a mid-bond or a post-bond test stage. Each die accommodates 2, 12, 20, 30, 42, and 49 modules resulting in 2 to 196 modules per IC as shown in column 3. The number of the corresponding nodes in the test graphs is between 60 and 5880 and test sizes are between 234 kB and 22 MB. The thermal models are extracted using an approach similar to [38]. A fast temperature simulation scheme similar to [27] is used. The switching activities for tests and heating sequences are generated using Markov chains, similar to [39]. All experiments are performed on a desktop computer with Intel® Xeon® W3520 processor and 8 GB of memory.

The required amount of cycling is separately defined for each module in the experimental ICs. It does not depend on the scheduling method and therefore both the three-phase and integrated approaches must enforce identical amounts of temperature cycling. Since before enforcing ATC_m^R , cycling tests cannot be performed, the cycling operations continue until ATC_m^R is enforced. This implies that it is sufficient to compare the test application times.

The integrated approach achieves shorter TAT compared with the three-phase approach for all of the experimental ICs. Two types of the three-phase approach are considered. Type 1 does not use any ordering technique. Type 2 uses a test ordering technique. The percentage changes compared with the three-phase approach without test ordering (type 1) and with test ordering (type 2) are

Table 1 Experimental Results

IC specifications			Percentage change in TATs		Percentage change in CPU times	
			Integrated approach compared with:		Integrated approach compared with:	
Number of layers	Number of modules		Three-phase w/o ordering ¹	Three-phase with ordering ²	Three-phase w/o ordering ³	Three-phase with ordering ⁴
1	1	2	-22.99	-21.75	0.00	0.00
2	1	12	-17.02	-14.81	0.00	0.00
3	1	20	-29.73	-2.12	0.00	0.00
4	1	30	-22.24	-5.51	300.00	300.00
5	1	42	-12.65	-10.08	200.00	200.00
6	1	49	-33.33	-32.19	500.00	500.00
7	2	4	-19.08	-19.08	0.00	0.00
8	2	24	-2.13	-2.13	200.00	200.00
9	2	40	-4.58	-3.74	500.00	500.00
10	2	60	-8.09	-7.34	350.00	350.00
11	2	84	-9.81	-9.52	325.00	325.00
12	2	98	-7.11	-2.94	300.00	300.00
13	3	6	-38.16	-35.60	0.00	0.00
14	3	36	-32.47	-20.71	400.00	400.00
15	3	60	-15.05	-13.85	333.33	333.33
16	3	90	-9.034	-8.92	283.33	283.33
17	3	126	-10.88	-7.45	169.23	191.67
18	3	147	-22.32	-12.98	83.33	83.33
19	4	8	-52.83	-44.92	0.00	0.00
20	4	48	-22.12	-21.95	25.00	25.00
21	4	80	-25.04	-24.86	13.04	13.04
22	4	120	-16.22	-16.03	18.48	15.96
23	4	168	-16.97	-16.96	14.43	18.08
24	4	196	-22.99	-13.99	20.50	26.87
Average			-19.70	-15.39	168.15	169.40

¹ Percentage change in the integrated approach's test application time (TAT^I) compared with the 3-phase approach without test ordering (TAT^{3wo0}) $100 \times \frac{TAT^I - TAT^{3wo0}}{TAT^{3wo0}}$

² Percentage change in the integrated approach's test application time (TAT^I) compared with the 3-phase approach with test ordering (TAT^{30}) $100 \times \frac{TAT^I - TAT^{30}}{TAT^{30}}$

³ Percentage change in the integrated approach's CPU time (CPU^I) compared with the 3-phase approach without test ordering (CPU^{3wo0}) $100 \times \frac{CPU^I - CPU^{3wo0}}{CPU^{3wo0}}$

⁴ Percentage change in the integrated approach's CPU time (CPU^I) compared with the 3-phase approach with test ordering (CPU^{30}) $100 \times \frac{CPU^I - CPU^{30}}{CPU^{30}}$

reported in columns 4 and 5, respectively. In average, the proposed technique outperforms the three-phase technique with ordering and the three-phase technique without ordering by about 15 percent and 20 percent, respectively. Since the test ordering offers a reduced TAT, the improvements achieved by the integrated approach are larger when compared with the three-phase without test-ordering approach.

Compared with the three-phase approach, the integrated approach is more complicated and therefore it takes more time to run. The CPU times are rounded to seconds and then percentage changes are calculated. The percentage changes in the CPU times are reported in columns 6 and 7 for the integrated approach compared with the three-phase approach without and with test ordering, respectively. In average, the proposed technique is slower from the three-phase technique with ordering and the three-phase technique without ordering by about 169 and 168 in percentage change, respectively.

The CPU times for the three-phase approaches with and without test ordering are comparable. Compared with the three-phase without ordering approach, the three-phase with ordering approach is more complicated for the decision-points² in the schedule. This is due to the time taken to search for a good node order. On the other hand, the three-phase with node ordering approach offers slightly shorter schedules which mean less decision-points. Consequently, sometimes shorter schedules compensate for the time-consuming node ordering operations, but not always. Therefore, sometimes the number in columns 6 is smaller than the number in column 7, for example for IC number 17 and sometimes larger, for example for IC number 22.

Note that in these experiments for CPU times we included the scheduling times for the normal and cycling tests. If the schedules would have been provided by a third-party, then the actual CPU times for the three-phase approaches will be smaller. Consequently, the numbers reported in columns 6 and 7 could be larger than the current values. Shorter CPU time can be considered as an advantage for the three-phase approach.

CPU times, in general, grow with the tests size as shown in Fig. 12a. Moreover, CPU times grow also with the number of modules and layers as shown in Fig. 12b. The data points in Fig. 12a represent multiples of test sizes used in Fig 12b. The growth rates are however acceptably low and the scheduling process for the largest IC (number 24) takes less than 5 minutes to complete.

9 Conclusions

Temperature-cycling acceleration is a useful technique to help the detection of cycling-dependent early-life failures. These failures are usually not considered as a major issue for conventional 2D ICs. Therefore, cycling acceleration is only recommended when a high degree of reliability is crucial. Recent studies have shown that the cycling-dependent early-life failures can be a major issue for 3D stacked ICs. The existing cycling acceleration procedures are very costly since they are usually performed using temperature chambers. In this paper we propose an inexpensive technique

Fig. 12 CPU time growth (a) with test size, (b) with IC complexity.

²A decision-point is a point that a module's state (testing/heating/cooling) or test/heating node may change.

to order the tests and heating sequences so that required temperature cycling effects can be achieved without the use of temperature chambers in a short time.

For this purpose tests are ordered differently based on the required power for the related situation. When a module's temperature must increase to generate a temperature cycle, a high-power ordering of the tests and heating sequences is considered. For the situation that the temperature must decrease, a low-power ordering of the tests is used, instead. During the tests, after required cycling is achieved, depending on the current module's temperature a long term or a short term low-power ordering of the tests is selected. All these help to achieve a short test application time, as demonstrated by the experiments. Consequently, this integrated approach is well-suited to be integrated into pre-, mid-, and post-bond test stages for 3D stacked ICs.

10 Quick Reference

Notation	Description	Equations
$\mathbf{A}$	Represents heat capacitances in the thermal model	1, 3, 17–20, 27
ATC	Amount of temperature cycling	5–6, 15, 28
ATC_m^R	Required amount of temperature cycling for module m	15, 28
$\mathbf{B}$	Represents thermal conductance (related to heat transfer) in the thermal model	1, 3, 17–18, 23–27
C	Heat capacitance in a simple single module case (analogous to $\mathbf{A}$)	4, 7–11, 14
$\mathbf{I}$	Identity matrix	3, 24–26
k_0 k_1 k_2	Constants in ATC equation	5
$\mathcal{N}_m$	Number of temperature cycles for module m	5–6
$\mathbf{P}$ P	Power value(s) in a general case	1–2, 4, 7–14, 22–24
$\mathbf{P}^{HS}$	Heating sequences' powers	18–21
$\mathbf{P}^{HT}$	High-power tests' powers	17–21
$\mathbf{P}^{LT}$	Low-power tests' powers	21
$\hat{\mathbf{P}}$	Stray power that includes leakage (static) power as well as clock network power.	21
PSO	Particle Swarm Optimization [36]	
R	Thermal resistance in a simple single module case (analogous to $\mathbf{B}^{-1}$)	4, 7–14
r_m	Remaining tests' size	28
$s_{m,n}$	Node n in module m 's test graph (same as n –th test for module m)	
TAM	Test Access Mechanism	
TAT	Test Application Time	
α	Transfer matrix for initial temperatures	2–3, 24–25
β	Transfer matrix for power values	2–3, 22
γ	Constant in ATC equation	5–6, 14
θ	Temperatures vector in a general case	1, 17
$\theta^{ambient}$	Ambient temperature	
$\theta^{overheating}$	Overheating temperature limit	
θ_m^{stray}	Stray temperature (caused by stray power, $\hat{\mathbf{P}}$)	
θ^0 θ^0	Initial temperature(s)	2, 4, 7
θ^τ θ^τ	Final temperature(s) after τ seconds	2, 4, 7
θ^{ss}	Steady state temperatures	22–23
θ_m^H	High cycling temperature limit for module m	
θ_m^L	Low cycling temperature limit for module m	15, 28

θ_m^E	Emergency temperature limit for module m	
θ_m^S	Stop cooling temperature limit for module m	
θ^ϵ	Threshold for temperature cycle amplitudes	5
$\overline{\theta}_m$	Average cycling temperature for module m	5
$\Delta\theta_m$	The amplitude of temperature cycles for module m	5–6
$\frac{d}{dt}\theta^{HS}$	Heating rate of the heating sequences	16–17
$\frac{d}{dt}\theta^{HT}$	Heating rate of the high-power tests	16–17
κ	Remaining number of cycles when full nodes are subtracted from the length of power assessment window ($\omega^H \times \lambda$). $\omega^H \times \lambda$ covers ν nodes plus κ cycles.	29
Λ	Minimal segment lengths for power averaging/assessment (Vector and element formats)	25–27
λ_m		
μ	Average cycling temperature for single module example	8–14
ν	Number of full nodes in the power assessment window ($\omega^H \times \lambda$). $\omega^H \times \lambda$ covers ν nodes plus κ cycles.	29
π_m	TAM access priority for module m	15, 28
ρ_m	ATC rate	6, 11, 14
σ	Half of temperature cycle amplitude for single module example	8–14
τ	Time period between the initial temperatures and the final temperature	3–4
φ	Threshold on power difference to decide between cooling interval or low-power test application	21
χ	Number of nodes involved in the power assessment window ($\omega^H \times \lambda$). $\chi = \begin{cases} \nu + 1, & \kappa > 0 \\ \nu, & \text{otherwise} \end{cases}$	29
ψ	Threshold on power difference to decide between heating sequence or high-power test application	21
ω	Length of the power assessment window. $\omega \times \lambda$ cycles are considered. Refers indifferently to ω^C , ω^E , ω^H , or ω^O .	
ω^C	ω for node ordering in Cooling situation.	
ω^E	ω for node ordering in thermal Emergency situation.	
ω^H	ω for node ordering in Heating situation.	
ω^O	ω for node ordering in Ordinary situation.	

11 References

1. Jeduc (2010) Failure mechanisms and models for semiconductor devices. <http://www.jedec.org/standards-documents/docs/jep-122e>. .
2. Kumar P, Dutta I, Bakir MS (2012) Interfacial effects during thermal cycling of Cu-filled Through-Silicon Vias (TSV). *J Electron Mater* 41:322–335.
3. Zhang D, Hummler K, Smith L, Lu JJ-Q (2013) Backside TSV protrusion induced by thermal shock and thermal cycling. *Electron. Proc. 63rd IEEE Compon. Technol. Conf.* pp 1407–1413.
4. Okoro C, Lau JW, Golshany F, Hummler K, Obeng YS (2014) A detailed failure analysis examination of the effect of thermal cycling on Cu TSV reliability. *IEEE Trans Electron Devices* 61:15–22.
5. Mil (2004) Temperature cycling (MIL-STD-883; Method 1010). In: DLA Land Marit. MilSpecs Draw. <http://www.landandmaritime.dla.mil/programs/milspec/ListDocs.aspx?BasicDoc=MIL-STD-883>. .
6. Jeduc (2009) Temperature cycling. Jeduc solid state technology association, <http://www.jedec.org/standards-documents/docs/jesd-22-a104d>. .
7. Bonhomme Y, Girard P, Landrault C, Pravossoudovitch S (2002) Test power: a big issue in large SOC designs. *Proc. 1st IEEE Int. Workshop Electron. Des. Test Appl.* pp 447–449.
8. Zorian Y (1993) A distributed BIST control scheme for complex VLSI devices. *Proc. 11th IEEE Annu. VLSI Test Symp.* pp 4–9.
9. Millican SK, Saluja KK (2012) Linear Programming Formulations for Thermal-Aware Test Scheduling of 3D-Stacked Integrated Circuits. *Proc. 21st IEEE Asian Test Symp.* pp 37–42.
10. Vinay NS, Rawat I, Larsson E, Gaur MS, Singh V (2010) Thermal aware test scheduling for stacked multi-chip-modules. *Proc. IEEE East-West Des. Test Symp.* pp 343–349.
11. Chakravarty S, Dabholkar VP (1994) Two techniques for minimizing power dissipation in scan circuits during test application. *Proc. 3rd IEEE Asian Test Symp.* pp 324–329.
12. Dabholkar V, Chakravarty S, Pomeranz I, Reddy S (1998) Techniques for minimizing power dissipation in scan and combinational circuits during test application. *IEEE Trans Comput-Aided Des Integr Circuits Syst* 17:1325–1333.
13. Girard P, Landrault C, Pravossoudovitch S, Severac D (1997) Reduction of power consumption during test application by test vector ordering [VLSI circuits]. *Electron Lett* 33:1752–1754.
14. Flores P, Costa J, Neto H, Monteiro J, Marques-Silva J (1999) Assignment and reordering of incompletely specified pattern sequences targeting minimum power dissipation. *Proc. 12th Int. Conf. VLSI Des.* pp 37–41.
15. Rosinger PM, Al-Hashimi BM, Nicolici N (2002) Power profile manipulation: a new approach for reducing test application time under power constraints. *IEEE Trans Comput-Aided Des Integr Circuits Syst* 21:1217–1225.
16. Bahukudumbi S, Chakrabarty K (2008) Test-pattern ordering for wafer-level test during burn-in. *Proc. 26th IEEE VLSI Test Symp.* pp 193–198.
17. Bahukudumbi S, Chakrabarty K (2009) Power management using test-pattern ordering for wafer-level test during burn-in. *IEEE Trans Very Large Scale Integr VLSI Syst* 17:1730–1741.
18. Tudu JT, Larsson E, Singh V, Agrawal VD (2009) On minimization of peak power for scan circuit during test. *Proc. 14th IEEE Eur. Test Symp.* pp 25–30.
19. Aghaee N, Peng Z, Eles P (2014) An efficient temperature-gradient based burn-in technique for 3D stacked ICs. *Proc. Des. Autom. Test Eur. Conf. Exhib.* pp 1–4.
20. Aghaee N, Peng Z, Eles P (2013) Temperature-gradient based test scheduling for 3D stacked ICs. *Proc. 20th IEEE Int. Conf. Electron. Circuits Syst.* pp 405–408.
21. Aghaee N, Peng Z, Eles P (2015) Temperature-Gradient-Based Burn-In and Test Scheduling for 3-D Stacked ICs. *IEEE Trans Very Large Scale Integr VLSI Syst* PP:1–1.

22. Calimera A, Macii E, Ravotto D, Sanchez E, Sonza Reorda M (2010) Generating power-hungry test programs for power-aware validation of pipelined processors. Proc. 23rd ACM Symp. Integr. Circuits Syst. Des. pp 61–66.
23. Millican SK, Saluja KK (2014) A test partitioning technique for scheduling tests for thermally constrained 3D integrated circuits. Proc. 27th Int. Conf. VLSI Des. pp 20–25.
24. He Z, Peng Z, Eles P (2010) Multi-temperature testing for core-based system-on-chip. Proc. Des. Autom. Test Eur. Conf. Exhib. pp 208–213.
25. Yao C, Saluja KK, Ramanathan P (2011) Temperature Dependent Test Scheduling for Multi-core System-on-Chip. Proc. 20th IEEE Asian Test Symp. 2011. pp 27–32.
26. Aghaee N, Zhiyuan He, Zebo Peng, Eles P (2010) Temperature-Aware SoC Test Scheduling Considering Inter-Chip Process Variation. Proc. 19th IEEE Asian Test Symp. pp 395–398.
27. Aghaee N, Peng Z, Eles P (2013) Process-variation and temperature aware SoC test scheduling technique. J Electron Test 29:499–520.
28. Aghaee N, Peng Z, Eles P (2015) An integrated temperature-cycling acceleration and test technique for 3D stacked ICs. Proc. 20th Asia S. Pac. Des. Autom. Conf. pp 526–531.
29. IEEE SA - P1838 - Standard for test access architecture for three-dimensional stacked integrated circuits. <http://standards.ieee.org/develop/project/1838.html> .
30. Huang W, Stan MR, Skadron K, Sankaranarayanan K, Ghosh S, Velusam S (2004) Compact thermal modeling for temperature-aware design. Proc. 41st ACM Annu. Des. Autom. Conf. pp 878–883.
31. Meng J, Kawakami K, Coskun AK (2012) Optimizing energy efficiency of 3-D multicore systems with stacked DRAM under power and thermal constraints. Proc. 49th ACM Annu. Des. Autom. Conf. pp 648–655.
32. Held M, Jacob P, Nicoletti G, Scacco P, Poech M-H (1997) Fast power cycling test of IGBT modules in traction application. Proc. Int. Conf. Power Electron. Drive Syst. pp 425–430 vol.1.
33. He Z, Peng Z, Eles P (2008) Simulation-driven thermal-safe test time minimization for System-on-Chip. Proc. 17th IEEE Asian Test Symp. pp 283–288.
34. Matsuishi M, Endo T (1968) Fatigue of metals subjected to varying stress. Jpn Soc Mech Eng Fukuoka Jpn 37–40.
35. Musallam M, Johnson CM (2012) An efficient implementation of the Rainflow counting algorithm for life consumption estimation. IEEE Trans Reliab 61:978–986.
36. Poli R, Kennedy J, Blackwell T (2007) Particle swarm optimization. Swarm Intell 1:33–57.
37. Lin T-M, Mead CA (1984) Signal delay in general RC networks. IEEE Trans Comput-Aided Des Integr Circuits Syst 3:331–349.
38. Coskun AK, Ayala JL, Atienza D, Rosing TS, Leblebici Y (2009) Dynamic thermal management in 3D multicore architectures. Des. Autom. Test Eur. Conf. Exhib. pp 1410–1415.
39. Yao C, Saluja KK, Ramanathan P (2011) Thermal-aware test scheduling using on-chip temperature sensors. Proc. 24th Int. Conf. On. VLSI Des. pp 376–381.