# An enhanced Strategy for Functional Stress Pattern Generation for System-on-Chip Reliability Characterization

M. de Carvalho, P. Bernardi, E. Sanchez, M. Sonza Reorda
Dipartimento di Automatica e Informatica – Politecnico di Torino, Torino (TO), Italy

## Abstract

*Reliability characterization is the industrial process intended to measure the useful life period and failure rate of a component population by exploiting stress mechanisms.*

*The paper describes a methodology for the automatic generation of stress programs to be used during the reliability characterization process of Systems-on-Chip (SoC). The proposed methodology is composed of a two-phase strategy; first an evolutionary algorithm (EA) works on the SoC's description at Register-Transfer-Level (RTL) by evaluating high-level metrics to quickly progress to a sufficient stress quality level; then evolution is continued on the gate-level description towards a better stress quality.*

*The proposed methodology was experimented on a SoC manufactured in a 90nm technology including an 8051 processor. The proposed strategy reduces significantly the generation times and quickly improves the stress quality values with respect to the previous methodology.*

## 1. Introduction

Reliability characterization performed on a population of semiconductor devices has the goal to provide electrical and reliability measurements useful to individuate and classify physical failure data models along the entire component lifecycle. This process, known as *operating life testing* (OLT) [1], turns out to be fundamental for understanding the characteristics of advanced materials and device manufacturing processes. The importance of OLT is even greater for devices working in critical environments, e.g., the automotive and the space ones, and it is expected to continue growing due to the predictable technology scaling for the next generations of VLSI circuits.

With respect to burn-in screening (BI) [12], OLT is mainly performed on test chips and monitors useful life and wear-out failures as well. Beyond their different purposes, BI and OLT are differently implemented. BI requires the devices to be stimulated at high temperature for few hours, either monitoring their outputs (test during burn-in, or TDBI) [11] or performing a successive test with automatic test equipment (ATE). On the contrary, OLT consists of several interleaved stress and test/diagnosis phases [7]; application engineers managing OLT normally have to plan for very long runs that can reach up to thousands of hours of duration and have to deal with the selection of proper stimuli for effectively stress the considered device and have to carefully plan for test/diagnosis measurements.

Another significant difference among BI and OLT is in the application of voltage stress that is obtained by toggling circuit node values [6]. BI requires the application of stress patterns owning high stressing capability all over the inspected device in order to rapidly exacerbate the insurgence of latent failures. Conversely, patterns used to induce voltage stress during OLT have to mimic the mission-like behavior of the component for inducing realistic failure modes related to the entire life of the inspected device.

In this paper, we concentrate on generating stress patterns for an effective SoC stress phase during OLT phases; similarly to the previous work, such patterns are functional test programs executed by the programmable resources available on chip, which maximizes the switching activity while complaining with the IC's normal functionalities. In the approach, the quality of the stress program under generation is calculated using some suitable metrics which are strictly related to the toggle activity of the circuit; these metrics, already discussed in [8], suit to measure how strong and uniform is the applied functional stress.

We propose a generation framework based on an EA [8], accounting on two-phases is considered to optimize the overall process. First, the EA operates at the RTL hardware description to quickly produce programs with a sufficient stress quality; in this phase, the generated programs are evaluated according to high-level metrics based on information gathered at RTL. These values are known to be correlated with gate-level ones, and are much easier to compute than the latter; therefore, this preliminary process is fast. After achieving this initial result, the generated stress patterns are optimized by operating on the gate-level description, thus requiring a high computational effort.

The benefits in terms of time gained with respect to the previous approach in [9] are evaluated on a 90nm technology System-on-Chip including an 8051

microcontroller core. Experimental results show that the generation time is reduced significantly and high stress quality achieved in a short time.

The rest of the paper is structured as follows: Section 2 introduces background concepts on reliability modeling, measurements and flows; Section 3 outlines the proposed methodology; Section 4 shows the comparative results collected at the gate-level and RTL. Finally, conclusions are drawn in Section 5.

## 2. Background

Borrowing some concepts from physics it is possible to say that stress is the internal resistance, or counterforce, of a material to the distorting effects of an external force. This counterforce tends to return the atoms to their normal positions. The total resistance developed equals the external force applied. This resistance is known as *stress*.

A force or load applied to a material distorts it in some way, independently of its strength. If it is a light load, then the distortion will most probably disappear when load is removed. This distortion is known as *strain*. So, stress attempts to overcome the material's interatomic forces holding it together. As stress continues to increase, the material starts deteriorating proportionally to the applied stress. This limit of applied stress is defined as the material's elastic region. If even more stress is applied to the material, this external force produces irrecoverable damages and this corresponds to the material's plastic region of deformation. Such application occurs when sufficient energy has been incremented to overcome internal forces [14] . Although it is impossible to measure the intensity of this stress, the external load and area to which it is applied can be measured. For a semiconductor device, it can be observed that physical stress modifies the molecules of the poly-silicon crystalline structure as well as the metal interconnections. Thus, it may be susceptible to failures as it creates specific faster paths for electrons, whose velocity augments the electric field and causes atoms to be pushed together or apart from each other. This phenomenon is denominated electronmigration. These atoms' movements create open or short circuits on the integrated circuit interconnections and flaws appear as the malfunctioning of the device [15].

On the other hand, temperature stress uniformly accelerates the aging of Integrated Circuits, and an effective stress phase requires a functional stress pattern, which means toggling circuit node values, even without monitoring the device responses.

### 2.1. Functional stress quality evaluation metrics

Generally speaking, a stress procedure capable of effectively stressing a device under test has to activate transitions on the circuit nodes [6] [12] in such a way that:

- the number of transitions per node is maximized in a given time slot
- all nodes show the same (or almost the same) number of transitions in a given time slot.

These considerations are valid for any considered circuit and for any kind of selected stress pattern (e.g., scan, BIST, functional, etc.) meaning that the applied stress should ideally be strong and uniform over the whole set of circuit nodes [6].

Let us define $T_i$ as the number of transitions taking place on the circuit node *i* during test application and *TD* as the Toggle Distribution resulting from monitoring $T_i$ over the set of all nets. Suitable metrics to measure the strength of a stress pattern are the TD average, *AVG(TD)*, and the TD variance, *VAR(TD)*. In order to comply with the underlined stress quality requirements, *TD* has to show:

- High *AVG(TD)*, implying high stress effectiveness on the circuit components
- Low *VAR(TD)*, enabling stressing the circuit components as uniformly as possible.

In the specific case of the SoC reliability characterization process, functional stress methods provides some significant advantages: in particular, with respect to scan based methods [6], the adoption of *functional stress programs* allows to reproduce a mission-like behavior of the analyzed SoC, thus

- Inducing more realistic failure modes
- Intrinsically avoiding power consumption concerns deriving from excessive nodes activity.

In a functional stress approach, the executed stress program should ideally guarantee that:

- The number of activated gates is as high as possible
- Switching activity stress applied to each core is as intense as possible
- The cores composing the SoC are equally stressed.

In the SoC scenario, some constraints are imposed by the processor core running a functional program. In fact, some components inevitably show more activity than others. For example, control and decode units are continuously required to process instructions, while functional units are active only when the SoC processor executes specific instructions. For this reason, we refine the aforementioned metrics by separately considering the contribution of each module

composing the SoC. Therefore, it is proposed to generate functional stress patterns for these SoC units that continuously process instructions.

Indeed, stress effectiveness over SoC and its units is obtained by running functional stress programs and it is maximized when obeying the following order:

- The percentage of activated gates inside each module *j* is **maximized**
- *AVG(TD$_j$)* of each module *j* is **maximized**
- *VAR(TD$_j$)* of each module *j* is **minimized**.

### 2.2. Functional program generation approach by means of evolutionary algorithm

A feasible way to automatically generate functional programs employs an EA, the basis of the exploited EA are presented in [8]. Such approach optimizes a population by imitating the natural process of biological evolution, as shown in Figure 1. Following this perspective, an assembly program is an *individual* of a population (i.e., a collection of assembly programs) which is handled by the tool. The initial population, a set of random programs that are not yet stress effective, is iteratively refined mimicking the Darwinian Theory: new individuals are generated by *mutation* (an individual is slightly modified) or by *recombination* (two or more individuals are mixed in some way); the best performing individuals are selected for survival. The process is halted after a certain number of steps, called *generations*, or when a steady state is reached. The best individual is provided as the output. Each individual is characterized by a quality value called fitness, and it is computed at each generation step by an evaluator.

An EA based methodology for the automatic generation of functional stress patterns is introduced in [9]. Even though the method achieved good results, it is highly time consuming, since the evaluation step is based on slow gate-level simulations. In this case, fitness values proposed by the evaluator and used to screen individuals are:

- The percentage of activated gates of each module *j* is **maximized**
- *AVG(TD$_j$)* of each module *j* is **maximized**
- *VAR(TD$_j$)* of each module *j* is **minimized**.

## 3. Proposed approach

In this paper, we concentrate on the stress phase of OLT procedures suitable for SoC reliability characterization. The main goal is to generate a set of functional programs executed by the SoC processor core able to stimulate the majority of its internal gates as intense and constant as possible. Such programs are referred as *functional stress patterns*. With respect to [9], where only gate-level simulations are employed, we propose an EA based methodology that achieves better results in a shorter time.

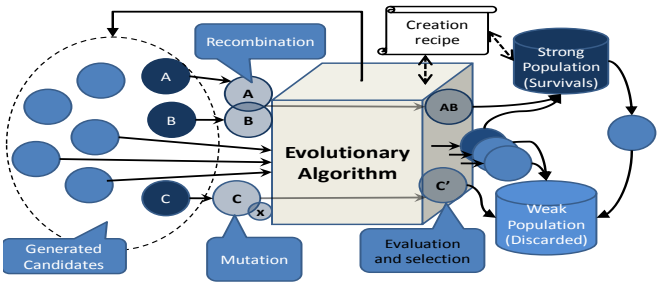


**Figure 1.** EA: Optimization of a population

The new strategy is composed of the following phases both based on stress programs evolution through EA:

**1)** The first phase aims at quickly generating stress patterns by working at RTL until sufficient stress quality values are reached.

**2)** The second phase starts from the stress quality level acquired in phase 1, and aims at improving the stress pattern population by working at gate-level:

**a)** initially using the fitness order as in phase 1.

**b)** then inverting fitness priorities for further refinement of the stress quality.

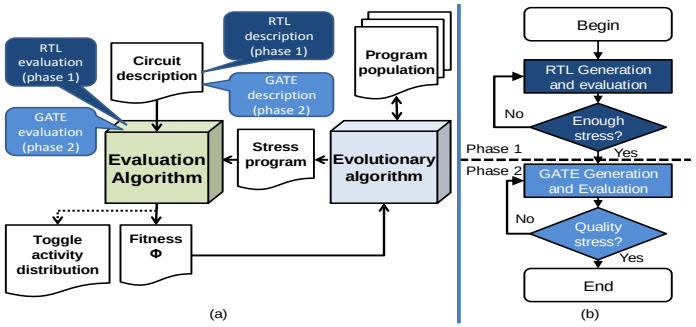


**Figure 2.** Proposed approach; (a) Generation flow, (b) Decision making process

The approach is illustrated in figure 2. Initially the framework in figure 2a has the phase 1 setup configured (dark balloons), and it iterates until sufficient stress quality is achieved, flow of phase 1 shown in figure 2b. Then phase 2 setup is configured (lighter balloons) and iterations continue until the functional stress patterns have achieved a desired quality level of stress within the program population. Figure 2a represents the proposed framework for the generation flow and Figure 2b the flow employed during each phase.

The proposed methodology is feasible thanks to the strong correlation among RTL signals and gate-level nets. Thus,

high switching activity of FFs' inputs and outputs of the sequential logic increases activity on the combinational one.

### 3.1. Phase 1: Fast stress pattern generation at RTL

This initial phase is intended to quickly generate stress patterns until a sufficient stress quality is met. The optimized methodology evaluates results based on the transitions of RTL signals gathered performing a logic simulation at RTL.

The timing optimization lies on the speed-up of the evaluation algorithm which simulates the SoC at RTL instead of performing the original task using the gate-level simulation, which is very slow.

The meaningfulness of this phase is granted because there is a correlation between the activities seen at RTL signals with the ones shown by the gate-level nets. In the detail, it is straightforward that signals in the RTL description are corresponding to Flip-Flops (FF) in the synthesized circuit; therefore, if signals are toggled, FFs will toggle accordingly. Moreover, the amount of activity of logic levels in the gate-level description is a direct consequence of the FFs transitions (i.e., if FFs move a lot, gates will show high activity). This correlation factor guarantees that the same stress program executed by the SoC at RTL produces proportional switching activity results to the SoC's gate-level activities.

However, information regarding the switching activity gathered at RTL tends to saturate after a while, partially losing switching activity correlation between RTL and gate-level. This is particularly true when dealing with complex circuits that are described at RTL with a few lines of code, as in the case of a logic multiplier.

Taking in consideration the above ideas, the first phase of the proposed approach is stopped when the first stress metric has progressed less than a given threshold K over a series of N consecutive generations.

### 3.2. Phase 2: Initial stress qualities refinement at gate-level

The second phase starts as soon as the stop condition is reached in phase 1. In this situation, to evaluate the stress metrics of RTL signals do not provide useful switching activity information to generate quality stress patterns, and so a thorough observation is needed to improve the current population. Therefore, the SoC RTL description is replaced by the gate-level one, allowing the evaluation of every single toggled gate in the design. In this new phase, the feedback parameters are also changed to gate-level values. The metrics priorities, based on a TD of gate-level nets, are described in the numbered list of section 2.1.

During the second phase, simulations are slower since the SoC gate-level description is used. Consequently, the number of generations performed by the EA is smaller. However, this phase is very important to discover new instruction sequences able to stimulate gates that could not be observed in phase 1.

Once again, the stop condition depends on the saturation of all stress metrics values or the observation of constant stress quality. Since simulation is slow, a long period of time is usually required before reaching one of the stop conditions.

***Final stress qualities refinement at gate-level***

At a certain level of evolution, the number of stimulated gates within a module is slowly increased and the tool starts generating programs corresponding to strong but non-uniform stress over the SoC, according to the list of fitness values described in section 2.1. It means that few gates switch intensively while others are at all activated. Such characteristics are critical for the reliability characterization of a SoC, as specified in section 2.1. So, the feedback stress quality metrics 2 and 3, corresponding respectively to $AVG(TD_j)$ and $VAR(TD_j)$, are swapped to guarantee uniformity of toggles among gates. Thus the metrics priorities become:

1) The percentage (%) of stimulated gates (**to be maximized**)
2) $VAR(TD_j)$ (**to be minimized**)
3) $AVG(TD_j)$ (**to be maximized**)

Where $TD_j$ is the toggle distribution of gate-level nets of module *j*.

## 4. Case study and results

In order to demonstrate the effectiveness of the approach, we present the results obtained on a SoC test-chip manufactured by STMicroelectronics in a 90nm technology; the reliability characterization flow principles described in the previous sections were implemented according to the requirements of a Massively Parallel Burn-In tester by ELES, the ART200 equipment.

The test-chip design was developed with the aim of maximizing the technologic diversity and thus includes different types of functional cores

- An 8-bit microcontroller
- A 64Kx8 bit SRAM storing instructions and data
- A combinational 16x16 parallel multiplier.

The test-chip also includes an IEEE 1500 SECT-based structure and a IEEE 1149.1-compliant TAP controller that

enable transferring the stress program at low frequency and executing it at 40 MHz exploiting the BI equipment free-running clock commodity [10]. The internal structure of the chip is shown in Figure 3.

The stress program generation was performed by a tool called µGP3 that implements the EA described in section 2.3, and exploiting a combination of Mentor Modeltech and an ad-hoc C++ tool for fitness computation. The elaboration provided a single functional program to be executed by the 8-bit microcontroller included in the chip.

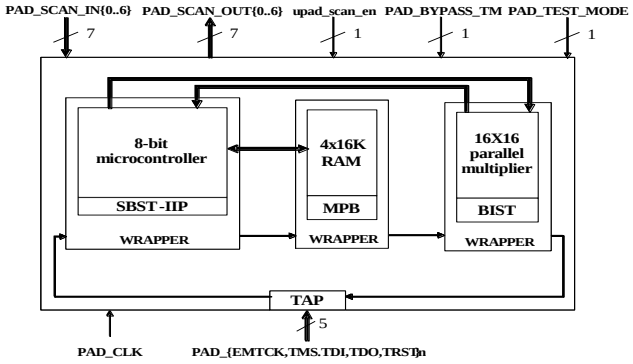


**Figure 3.** Block diagram of the test-chip

For the sake of generating patterns according to the considerations in section 2.1, we purposely applied the approach to the test-chip structure on the following components:

- The entire System-on-Chip (**SOC**)
- Single modules:
  - Microcontroller control unit (**FSM**)
  - Arithmetic Logic Unit (**ALU**)
  - RAM controller (**RAM**)
  - Memory controller (**MEM**)

The approach applied to the memory controller (MEM) produced results useful to exemplify the behavior of the tool during the two phases, even from a graphical point of view. Figure 4 shows the evolution on the number of stimulated signals/gates, in percentage, caused by the stress patterns executions at the core. During phase one, these programs were generated using the SoC RTL description framework (bottom curve) and evaluated on the SoC gate-level (top curve). In this figure, is noticeable the correlation among RTL signals and gate-level nets, since both curves have approximately the same behavior during phase 1. Then, when the stop condition in phase 1 is reached (corresponding in our case to K=2% and N=30), the evolution is continued in phase 2.

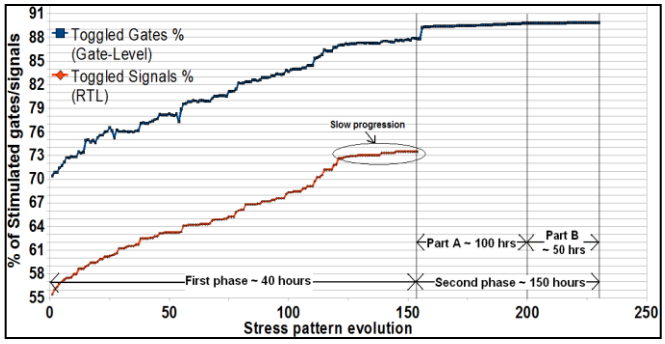


**Figure 4.** Stimulated gates (in %) evolution of the MEM

In phase 2, the automatic generation was performed using SoC gate-level description, allowing refining the obtained stress quality up to this point. In part B of phase 2, the feedback stress quality parameter order is inverted (as discussed in section 3.2) to preserve and improve stress qualities.

Figure 5 shows the *AVG(TD$_{MEM}$)* evolution of the MEM module. Figure 6 shows the *VAR(TD$_{MEM}$)* evolution of both phases and the effect produced by the metrics inversion.

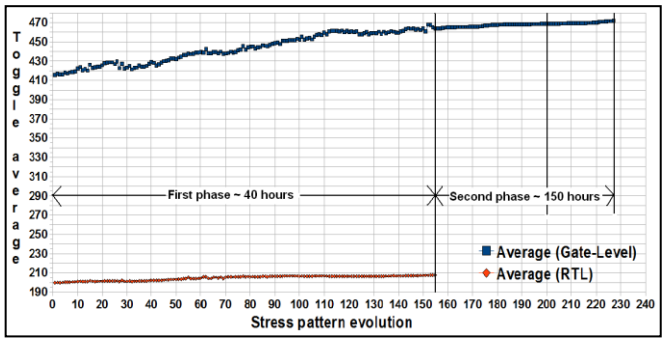


**Figure 5.** *AVG(TD$_{MEM}$)* evolution

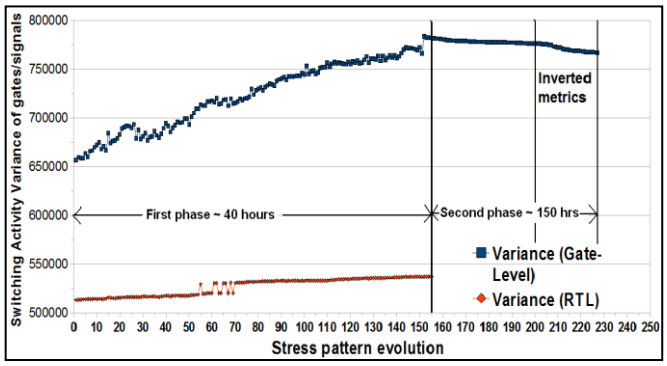


**Figure 6.** *VAR(TD$_{MEM}$)* evolution

Final results drawn in the second column of Table I, shows the best stress patterns results of each SoC module, which were generated and evaluated by the two-phase strategy. In order to provide comparative results among the suggested approach and the previous methodology in [9],

both procedures were launched at the same time. A preliminary stress quality level was reached by the proposed approach at around 190 hours of continuously pattern generation and evaluation, and then results of both methodologies were compared. This allowed measuring the stress quality disparity among both procedures within that period; After 190 hours the proposed approach reached high quality values, while the ones obtained by the previous methodology were still far from the best; The latter had to be executed up to 850 hours to reach comparable results.

If we consider results produced on the FSM module by both approaches, it is noticeable that after 190 hours the new method was able to activate 98.94% of this module's gates. This value is better than the one obtained by the previous method [9] at the end of the same period (49.37%) and also after 850 hours (94.06%).

The proposed method was faster because a single generation cycle lasts 4 seconds when simulating the SoC RTL description, and about 45 seconds for the gate-level one. Since the previous method uses only the SoC gate-level description, simulations slowed the pattern generations.

**Table I.** Final results obtained for each SoC module**.**

| Module | | **Proposed approach** Final results (~190 hours) | Previous approach [9] | |
|---|---|---|---|---|
| | | | ~190 hours | ~850 hours |
| **SOC** | % | 76.16 | 71.70 | 75.91 |
| | Avg | 313 | 259 | 308 |
| | Var | 991,571 | 689,854 | 978,282 |
| **FSM** | % | 98.94 | 49.37 | 94.06 |
| | Avg | 978 | 898 | 966 |
| | Var | 1,838,778 | 1,497,723 | 1,786,832 |
| **ALU** | % | 97.69 | 91.98 | 98.14 |
| | Avg | 169 | 138 | 167 |
| | Var | 132,697 | 119,564 | 131,186 |
| **RAM** | % | 77.62 | 29.22 | 38.78 |
| | Avg | 395 | 178 | 192 |
| | Var | 1,349,509 | 549,755 | 575,106 |
| **MEM** | % | 89.91 | 69,41 | 93.33 |
| | Avg | 473 | 411 | 1,508 |
| | Var | 766,894 | 645,636 | 5,817,757 |

## 5. Conclusions

In this paper a new method for automatic generation of functional stress patterns suitable for reliability characterization of SoC was introduced. We proposed enhancements to a previous methodology by adopting a new strategy; initially we use an RTL generation-based technique to speed-up the generation and quickly achieve improved results; secondly, we resort to a gate-level pattern generation approach to refine the results. It was demonstrated the feasibility of the improved methodology by proving the equivalence of gate-level and RTL approaches.

Results obtained by the proposed and original approaches were compared. The adopted strategy is much faster and achieves high level quality stress in a shorter time than the previous methodology.

## 6. References


[1] A. Birolini, "Reliability Engineering, Theory and Practice", SpringerVerlag, 3rd edition, 1999

[2] A. Vassighi, et al. , "CMOS IC Technology Scaling and Its Impact on Burn-In", IEEE Transactions on Device and Materials Reliability, vol. 4, no. 2, June 2004, pp. 208-221

[3] M. Elbert, et al. ,"Stress testing and reliability", IEEE Southcon/94, 1994, pp. 357-362

[4] K. Roy, et al., "Stress testing of combinational VLSI circuits using existing test sets", IEEE International Symposium on VLSI Technology, Systems and Applications, 1995, pp. 93-98

[5] V. Dabholkar, S. Chakravarty, J. Najm, J. Patel, "Cyclic stress tests for full scan circuits", IEEE VLSI Test Symposium, 1995, pp. 89-94

[6] A. Benso, A. Bosio, S. Di Carlo, G. Di Natale, P. Prinetto, "ATPG for Dynamic Burn-In Test in Full-Scan Circuits", IEEE Asian Test Symposium, 2006, pp. 75-82

[7] D. Appello, et al. , "An Innovative and Low-Cost Industrial Flow for Reliability Characterization of SoCs", IEEE European Test Symposium, 2008, pp. 321-327

[8] F. Corno, et al. , "Automatic Test Program Generation - a Case Study", IEEE Design & Test of Computers, 2004, Vol. 21, n. 2, pp. 102-109

[9] D. Appello, et al. , "Automatic Functional Stress Pattern Generation for SoC Reliability Characterization", IEEE European Test Symposium, 2009, pp 93-98.

[10] P. Bernardi, M. Rebaudengo, M. Sonza Reorda, "Using Infrastructure IPs to support SW-based Self-Test of Processor Cores", IEEE International Workshop on Microprocessor Test and Verification, 2004, pp. 22-27

[11] S. Bahukudumbi, K. Chakrabarty, "Test-Pattern Ordering for Wafer-Level Test-During-Burn-In", IEEE VLSI Test Symposium, 2008, pp. 193-198

[12] Huang *et al.*, "Maximization of Power Dissipation Under Random Excitation For Burn-In Testing", IEEE International Test Conference, 1998, pp. 567-576

[13] D. Gizopoulos, Y. Zorian, A. Paschalis, "Embedded Processor-Based Self-Test", Springer-Verlag, New York (USA), 2004

[14] Patrick dt, O'Connor, Wiley and Chichester,it "Practical Reliability Engineering", Wiley,2002

[15] John Wiley and Sons, "Wiley Encyclopedia of Electrical and Electronics Engineering - Life Testing", J.Webster,1999