Thermal-Aware Testing of VLSI Systems

Registration seminar report submitted in partial fulfillment of the requirements for the degree of

Master of Science (by Research)

in

Dept. of Electronics & Electrical Communication Engineering

by

Rajit Karmakar

[Roll No: 12EC72P02]

Under the supervision of

Prof. Santanu Chattopadhyay

Dept. of Electronics & Electrical Communication Engineering

Indian Institute of Technology, Kharagpur

June, 2014

ABSTRACT

This work addresses the issue of power-aware and thermal-aware test scheduling of cores in a System-on-chip (SoC). While the existing approaches either use a fixed power value for the entire test session of a core, or the cycle-accurate power values, the proposed work divides the power profiles of cores into fixed-sized windows. This reduces the number of power values to be handled by the test scheduling approaches, while reducing the amount of pessimistic over-estimations of instantaneous power. As a result, the power model can be integrated with more exhaustive meta-search techniques for generating power constrained test schedules. A multi-frequency test environment has been proposed to facilitate testing of cores at different frequency levels. Test Application Time (TAT) can be further improved by using multi-frequency test approach over single-frequency test approach. A superposition principle-based thermal model has been incorporated with the test scheduling algorithm to ensure thermal safety during testing. Although this thermal model avoids run-time thermal simulations, it has the capability of estimating temperature of a core very fast with high degree of accuracy. These two proposed power and thermal models, integrated with a Particle Swarm Optimization (PSO) based test scheduling technique, generates a power and thermal safe test schedule for SoC.

I. INTRODUCTION

System-on-Chip (SoC) paradigm has evolved to address the problem of high design turnaround time of complex VLSI systems. Design time is reduced via core-based system design. Pre-designed logic blocks (processor, memory etc.), commonly known as Intellectual Property (IP) cores, are integrated on the same silicon floor by the system integrator. Compared to a board-based system, all communication between the system components have become on-chip. This aids in improving the system performance. Large numbers of IP cores are being integrated onto a single chip. The approach, though reduces design time, may increase the testing time significantly. With increasing complexity of present day's integrated circuits (IC), the amount of test data needed to test the ICs have increased dramatically. More test patterns are required to improve fault coverage targeting delay faults, stuck-at faults and some other subtle faults. Automatic Test Equipment (ATE) has to accommodate the large amount of test data, increasing the memory size as well as memory cost. To test the chip, system integrator has no option but to apply all the test patterns specified by the vendors of individual cores, making the test data volume huge and increasing the Test Application Time (TAT) significantly.

In SoC testing the test engineers are limited by the following.

- Limited number of channels of Automated Test Equipment (ATE) to transport test patterns to the chip.
- Limited amount of on-chip test resources. It may be a few signal lines running through the chip to carry the test signals.
- Different number of input-output pins (including scan input/output) for different cores coming from different IP vendors. Scan chain lengths are also varying.
- Huge amount of test data (pattern and response) to be stored at the ATE and transported to/from the chip.

Test cost increases mainly because of the total Test Application Time (TAT) needed to complete testing of all the cores of the SoC and large amount of memory requirement to store huge amount of test data. Test engineers mainly focus on reducing TAT by efficient test scheduling of the cores and reducing the total memory requirement by storing test data in a compressed manner in the ATE.

So, the SoC testing problem can be broadly divided into two categories.

- Test Scheduling
- Test Data Compression

To keep TAT within a reasonable limit, one solution is to perform parallel testing of several cores. The test engineer may need to test some modules simultaneously, which are running in exclusive fashion during normal system operation. Overlapped testing of cores may be prohibited by the system power limit. Moreover, for a single module, generally test mode power is much higher than functional mode power. Successive input signals in functional mode are often highly correlated to each other. In test mode, successive test patterns are made as uncorrelated as possible with the expectation that further patterns will excite different regions of the module, possibly identifying newer faults. A set of cores can be tested in parallel only if it does not violate the system level power limitation at every point of the schedule. Determining such a schedule is difficult as it is necessary to ensure power limit validity at every time instant. It is worth mentioning that excessive power consumption causes overheating, hence may cause a permanent damage to the chip. Thus, ensuring power validation cannot be ignored.

It is worth mentioning that ensuring power validation may not be sufficient enough to guarantee thermal safety of the chip. Although, power has a major contribution in the temperature increase of a chip, there are some other parameters like floorplan, power density of cores, core size and their relative positions etc, which play important role to determine the temperature of the chip. It may be noted that, power minimization may reduce overall temperature of the circuit, but, does not necessarily minimize peak temperature, which causes local hotspot due to non-uniform spatial power distribution in the circuit and hence permanent damage of it. Thus, thermal safety has to be ensured during testing. Although advanced cooling techniques can effectively solve the high temperature problems, they substantially increase the overall system cost and/or require larger area. Other solution is to incorporate some thermal model that takes care of thermal safety during test scheduling.

So the SoC test scheduling problem can again be classified into three sub-categories.

- Basic Test Scheduling (with only resource constraint).
- Power-Aware Test Scheduling (with resource and power constraints).
- Thermal-Aware Test Scheduling (with resource, power and temperature constraints).

Test data compression stores huge amount of test data (T_D) in the ATE in a compressed form (T_E), helps to reduce the total memory requirement. The compressed data (T_E) has to pass through a decompressor to get back original uncompressed (T_D) test patterns before being applied to the circuit under test (CUT).

II. LITERATURE SURVEY

In the literature, solution to the test scheduling problem has been approached by first designing wrappers around the cores [1-3]. If a core has a total of k number of input/output/scan pins and allocated only b-bits ($b \leq k$) of test signal lines, the wrapper design algorithm [2] joins some of these pins over chains, so that test data can be transported b-bits at a time. Naturally, decreasing b will increase test time. Some of the approaches [2] [4] [17] [23-24] perform a complete partitioning of the on-chip test resources and allocate one partition to each core. On the other hand approaches suggested in [6-16] [18-22] do not perform a complete partitioning; test resources are allocated to cores depending on availability. The second approach provides more flexibility, since deciding an optimal partitioning itself is an NP-complete problem [7]. In both the approaches, scheduling is performed to determine the duration at which individual cores are to be tested. A core can be tested only if resources are available for the duration of its test. Since with different amount of allocated test resources, test time varies, it gives rise to a bin packing problem [6]. Bin width is equal to the total number of test signal lines. Testing of each core is represented as a rectangle having width equal to the number of signal lines allocated and height corresponding to the associated test time. An architecture-independent theoretical lower bound on test time has been presented in [5]. An *evolutionary algorithm* (EA) based approach, using sequence pair representation and a distributed bin-packing approach has been presented in [8]. Simulated annealing (SA) based two dimensional bin-packing algorithm to solve SoC test scheduling problem has been proposed in [9]. The work also proposes a wrapper design procedure for cores with no scan cells. B*-tree based floorplanning technique has been used in [10] transforming the scheduling problem to that of floorplanning. Height and width of the floorplan represent the TAM width and test time of the core respectively. The idea is to find a fixed height, minimum width floorplan, while there should be no overlap between the tiles in the floorplan. Simulated annealing has been used to solve this problem. Two stage genetic algorithm (GA) based approach to solve SoC test scheduling problem has been presented in [11]. Solutions are represented by sequence pairs. The first stage takes care of the cores with longer test time and wider TAM width, while the remaining cores are adjusted in the second stage of the solution. Final solution is obtained through evolution over generations. Ant Colony Optimization (ACO) has been used in [12] to solve the SoC scheduling problem, while the work [13] uses another Genetic Algorithm (GA) based approach to select the rectangles to pack in a bin for the rectangular 2D-bin packing approach.

The problem with most of the approaches reported above is that they do not consider power as a constraint during the scheduling process. For the safety of the chip during testing, the power consumption should not go beyond a certain limit. Several works in the literature consider power restricted test schedule generation. Different power models of a core have also been proposed. Conventionally, a global peak power model has been assumed [14-21] for the entire test session of a core. On the other extreme, a cycle-accurate power model has been followed in [23-24]. In [15], rectangular 3-D bin packing approach has been adopted to solve SoC test scheduling problem under power constraint. The authors, in [16], have considered multiple test sets for testing of a core. An Integer Linear Programming (ILP) based solution has been presented in [17]. Here the authors have presented a *reward model* that allows the system integrator to incorporate *preferences* arising from place-and route constraints, in allocating cores to test buses. A greedy algorithm based technique has been proposed in [18]. A genetic algorithm based approach, to minimize TAT has been reported in [19]. Here the authors have used chromosome structure, mutation and crossover operators to select representative rectangles for rectangular 3-D bin packing approach to solve the scheduling problem optimally. A shuffle frog-leaping algorithm (SFLA) that follows evolution of frog population has been used in [20] to solve scheduling problem under power constraint. However, all these power aware test scheduling techniques have used global peak power model. Authors in [22] have merged test power minimization problem with the test scheduling problem. They have reordered the test vectors to generate test sequences in such a way that, there will be an initial long low power part followed by a short high power part. This *two local peak power approximation model* (2LP-PAM) minimizes the power overestimation compared to a single global peak power model. The power profile of a core is represented by two local peak power values in this approach. However, the approach is limited by the fact that this pattern reordering may not be possible as the ATE is preloaded with test patterns.

Several recent works have attempted to solve the test scheduling problem by using different voltage levels and also by varying the operating clock frequency. Multiple voltage testing of cores, where cores are placed in multiple voltage islands, has been presented in [25]. The formulated test scheduling problem has been solved using ILP and also via a greedy approach. Another ILP based method, considering different operating frequencies for different test sessions to minimize test time has been presented in [26]. In [27] both supply voltage as well as operating frequency has been varied to get better schedule. This dynamic voltage frequency scaling (DVFS) method can be used properly to slow down or speed up individual tests to get better test compatibility under power constraint, during scheduling. A pre-emptive and a non-pre-emptive heuristic approach using DVFS method for session-less test scheduling has been presented in [28].

Several other approaches have done multi-frequency testing to bridge the gap between ATE frequency and the cores with different operating frequencies. Wrapper-design for cores with multiple clock domains has been proposed in [29]. It can handle cores with different operating frequencies for different scan chains embedded within it. Several other approaches, using virtual TAM for bandwidth matching and test data rate synchronization between ATE and the cores operating at different frequencies have been presented in the literature [30-32]. A test data multiplexer (TDM) / test data demultiplexer (TDdeM) based approach has been used in [30] to fulfil the gap between the frequency of ATE and the cores. A dynamic reconfigurable multi-port ATE based multi-frequency test scheduling strategy has been proposed in [33]. However, all these works consider fixed average or peak power consumption of the core during scheduling.

Thermal-aware SoC test scheduling problem has drawn the attention of many researchers in recent time. The authors in [34] have used a test session based thermal model for test scheduling. Their RC model based approach tries to maximize heat dissipation through lateral neighborhood of active cores, in a test session. The work assumes negligible heat transfer between two active neighboring cores. The assumption may not be valid, as we have shown later in this paper that, the temperature of a core largely depends on the parallelly active neighbours. The concept of thermal ground for idle cores does not hold, as there is a fair amount of leakage power consumption of the idle cores, which actually increase their ambient temperature. In [35] the authors have tried to minimize the temperature of the hottest core to achieve a balanced thermal distribution across the chip during testing. The floorplan information has been used during scheduling to avoid concurrent testing of two hot neighboring cores. They have tried to maintain the temperature of each core below a threshold value, during testing. Hand crafted floorplan of the chip has been used for experimentation. In [36], the authors have presented two algorithms for thermal aware testing. First, an Integer Linear Programming (ILP) based solution has been proposed. It produces optimal solution, but is computationally inefficient. It requires large number of thermal simulations to get a thermal safe test schedule. A faster thermo-resistive model based heuristic has also been proposed, however the model also has the drawbacks similar to [34].

Test set partitioning and interleaving based SoC test scheduling strategies have been presented in [37, 38]. In [37] the authors have divided the total test set into several smaller identical test sub-sequences. The idea behind this approach is that if a core is tested for a long period of time, the temperature of the core will increase. To restrict the temperature of the core under a certain limit, equal length cooling periods have been introduced in between the test sub-sequences. A constraint logic programming (CLP) model and a heuristic approach have been used to solve this problem. In [38] the authors have removed the restrictions on the equality of the length of the test sub-sequences as well as the cooling periods. This makes the procedure more efficient. However, as it may be observed from the simulation result of Fig.6 (ii) (Section C.3), temperature of a core increases very fast and reaches its peak value quickly. Thereafter not much variation in temperature is observed with time. So the concept of increase of temperature with time is not always true. Thus, we need to incorporate more numbers of cooling periods between the test sub-sequences, adding extra overhead in scheduling. This may result in longer test application time.

A simplified thermal model based approach has been presented in [39]. The model reduces the number of thermal simulations required to determine thermal violations. The idea behind this work is to test the hotter cores as fast as possible (small test duration) to minimize its effect on the surrounding cores. They have considered Test Access Mechanism (TAM) dependent average power model of the cores. Scheduling problem has been converted into a 3-D bin packing one. A fast thermal simulator ISAC has been incorporated in [40]. Here, the authors have considered test set partitioning and interleaving based approach for scheduling. They have considered lateral temperature dependency between the cores. A Mixed Integer Linear Programming (MILP) based thermal solution has been proposed in [41]. The authors have also presented a seed-based clustering approach in this paper. A test-schedule reshaping, test-set partitioning and interleaving based approach has been enumerated in [42]. The authors have tried to minimize the temperature of the hottest core by an iterative process of test reshaping. Test set partitioning, interleaving and bandwidth matching based techniques have been used to cool down the temperature of the cores, which fail to reduce their temperature during the test reshaping process. Fixed width TAM architecture and cycle-accurate power model have been used in this paper. A computationally tractable thermal model based on average power consumption, thermal resistances and test overlapping time has been incorporated for the purpose of calculation simplicity. A thermal-aware test scheduling in an abort-on-first-fail (AOFF) environment, suitable for volume production, has been presented in [43]. In this kind of environment, if a fault is detected in any of the circuit under test (CUT), the test process gets terminated. Test set partitioning and interleaving based approach to minimize the expected test application time (ETAT) has been presented in this paper.

Multi-temperature SoC testing has first been proposed in [44]. As different faults can be detected in different temperature levels, it is important to test a SoC at different temperature levels, to detect all the faults. The authors have tested all the cores in a certain temperature interval. The temperature during testing cannot go beyond the temperature interval. Some passive cooling sequences and heating sequences have been introduced here, to keep the temperature of a core within a certain interval during testing. The concept of global temperature range for the SoC, presented in this paper, has been modified in [45]. Here, the authors have considered different temperature ranges for each individual test. They have also considered multiple tests of a core in different temperature ranges. A list scheduling based algorithm has been used to minimize TAT. On-chip temperature sensor for thermal-aware test scheduling has been proposed in [46]. To overcome the inaccuracies of fast thermal models, the authors have suggested reading the temperature data from on chip sensors and use these data for dynamic test scheduling. However, non-availability of such sensor data may be a problem in this type of testing.

Superposition principle based thermal models have been presented in [47, 48]. In these works, the linearity of RC thermal model has been used to calculate thermal profile of a core during scheduling, using superposition principle. As the CPU execution time is the main bottleneck of thermal simulation during scheduling, they have tried to avoid invoking the thermal simulator in the scheduling process. Instead, they have used the HotSpot [49] thermal simulator to create offline thermal profiles of the cores. These thermal profiles are used for scheduling purpose. This type of thermal model is fast. They have also partitioned the tests with different power and different test lengths, to get a better thermal and power constrained test schedule.

Test vector compression schemes fall broadly into three categories [50]:

- ***Code-based schemes:*** It uses data compression codes to encode test cubes. This involves partitioning the original data into symbols, and then replacing each symbol with a code word to form the compressed data. To perform decompression, a decoder simply converts each code word in the compressed data back into the corresponding symbol. Code-based schemes can again be classified into four categories. a) Run-length-based codes (fixed length [51], and variable length code [52-54]), b) Dictionary codes [55-60], c) Statistical codes [61] and d) Constructive codes [62-63].
- ***Linear-decompression-based schemes****:* It decompresses the data using only linear operations (that is LFSRs and XOR networks). A linear decompressor can generate test vector Y if and only if there exists a solution to the system of linear equations $AX = Y$, where A is the characteristic matrix for the linear decompressor and X is a set of free variables shifted in from the tester. Encoding a test cube using a linear decompressor requires solving a system of linear equations consisting of one equation for each specified bit, to find the free variable assignments needed to generate the test cube. If no solution exists, then the test cube is unencodable (that is, it does not exist in the output space of the linear decompressor). Linear-decompressor-based codes can be classified into two categories. a) Combinational linear decompressor-based codes [64-66] and b) Sequential linear decompressor-based codes [67].
- ***Broadcast-scan-based schemes:*** It relies on broadcasting the same values to multiple scan chains. This is actually a special degenerate case of linear decompression in which the decompressor consists of only fan-out wires. Given a particular test cube, the probability of encoding it with a linear decompressor that uses XORs is higher because it has a more diverse output space with fewer linear dependencies than a fan-out network. Reconfigurable broadcast scan-based codes can be classified into two categories. a) Static Reconfiguration [68-70] and b) Dynamic reconfiguration [71-72].

It is worth mentioning that, more than 95% bits of the test data are don’t-cares [73]. To get better compression, most of the test compression techniques take the advantage of the flexibility of the don’t-care bits, which can be flipped either to ‘0’ or ‘1’ to get more numbers of matching sub-vectors from the test vectors without compromising fault coverage.

However, temperature is another major concern during testing. Temperature can be reduced by efficient don’t-care filling. Several works in the literature have tried to minimize temperature of the IC during testing by efficiently filling the don’t-care bits present in the test patterns [74-75]. 0-fill, 1-fill, minimum-transition-fill and random-fill are some of the popular don’t-care bit filling techniques. Other thermal-aware don’t-care bit filling works [74] attempt to reduce temperature using their own cost function to fill up the don’t-care bits.

It is interesting to note that, both test data compression and thermal-aware don’t-care filling concentrate on filling the don’t-care bits efficiently to get best compression and low temperature testing respectively. In the literature these two problems have been solved separately. The test data compression works try to fill up the don’t-care bits to get better compression ratio ignoring the thermal effect, while the thermal-aware don’t-care filling works try to minimize the temperature by efficient don’t-care bit filling without taking care of test data compression.

In recent times, some works have addressed the problem of high power consumption in test data compression schemes [76-78]. However, it may be noted that, power minimization may reduce overall temperature of the circuit, but, does not necessarily minimize peak temperature, which causes local hotspot due to non-uniform spatial power distribution in the circuit and hence permanent damage of it. Peak temperature of a block depends not only on power consumption, but also on heat exchange between adjacent blocks. So, temperature of a circuit requires special attention.

III. OBJECTIVE AND SCOPE OF THE WORK

Objective of the proposed research work is to find out an efficient test scheduling strategy that can schedule all the cores of a SoC with minimal Test Application Time (TAT). The scheduling strategy must satisfy all the resources, power and thermal constraints during scheduling to get a power and thermal safe test schedule with minimal TAT. Another objective of this work is to find out an efficient test data compression strategy that can produce good compression ratio as well as reduce the temperature at the time of testing of the chip. Specifically the following works will be targeted.

- ***Basic test scheduling*:** As mentioned earlier, test scheduling problem is NP-complete. So a Particle-Swarm-Optimization (PSO) guided bin-packing heuristic can be used to find a test schedule with minimal TAT. Faster convergence of PSO than other meta-search techniques like GA, ACO, SA etc. makes PSO a useful search technique to solve NP-complete problem efficiently.
- ***Single-frequency power-aware test scheduling*:** Among all the power models presented in the literature, conventional global peak power model may generate a too restrictive schedule inhibiting potential parallel testing of some cores, even if the sum of exact instantaneous power values of them is well below the power limit. Cycle-accurate power model does exact power sums, however, the scheduling algorithm needs to handle large number of power values. Handling large number of power values and keeping track of power profile for each clock cycle during scheduling makes this approach extremely time consuming. An intermediary approach that considers peak power values over a time-window to approximate the core power over that interval of test time can be a viable solution to reduce computational complexity of cycle-accurate power model. This window-based peak power model, though introduces some inaccuracy in the power model, the reduction in the number of power values can be helpful in designing a Particle Swarm Optimization (PSO) based search procedure to evolve better test schedules than many of the contemporary SoC testing approaches.

- ***Multi-frequency power-aware test scheduling*:** Multi-frequency test scheduling solves the problem of frequency mismatch between ATE and the cores operating at different frequencies. This approach can use the flexibility of selecting different operating frequencies for the cores to reduce TAT efficiently. Moreover, the works presented in the literature follow global-peak power model, which may generate too restrictive test schedule considering lots of false power. Incorporation of window-based peak power model with multi-frequency approach can further improve the TAT.

- ***Thermal-aware test scheduling*:** As popular thermal simulator Hotspot cannot be used inside the scheduling procedure because of its high time consumption, thermal model with the capability of temperature estimation can be the alternative solution for faster schedule generation. While the inaccuracy of most of the thermal models presented in the literature fails to estimate temperature accurately, a superposition principle based thermal model that uses the linearity of Hotspot, can be useful to estimate the temperature accurately. However, care must be taken to consider the thermal interdependencies between the neighbour cores. Moreover, in the absence of the accurate information regarding area, power and test data of the ITC'02 SoC benchmarks, most of the works in the literature approximate all these information. This introduces lots of inaccuracy in the thermal profile of the SoC. This problem can be overcome by designing new SoCs with all the required details. ISCAS'85, ISCAS'89 and ITC'99 benchmarks can be used as cores of the new SoC. A PSO guided test schedule with superposition principle based thermal model can generate test schedule with minimal TAT respecting resource, power and thermal constraints.
- ***Thermal-aware test data compression*:** Test data compression techniques use the flexibility of having large number of don't-care bits (about 95% [73]) of the test vectors to get high compression ratio. These don't-care bits are filled suitably (either '0' or '1') to extract more number of matching sub-vectors from the test vectors. Instead of storing all the similar sub-vectors, a single representative sub-vector is stored to save memory requirement. However, the don't-care bits can also be filled intelligently to reduce the temperature during testing. It may happen that for a particular don't-care bit, compression-aware don't-care filling and thermal-aware don't-care filling suggest different values. A trade-off between compression and temperature reduction can be achieved by efficiently filling the don't-care bits.

Out of these, basic test scheduling, power-aware test scheduling and thermal-aware test scheduling have been completed. In the following these works have been presented.

IV. WORK DONE SO FAR

A. Basic Test Scheduling

A.1 Problem Formulation:

Suppose a SoC with N cores C_1, C_2 ...C_N is to be tested with a maximum of W_{max} TAM resources. The test scheduling problem is to allocate TAM resources and test times to the cores so that, the total test application time (*TAT*) is minimized.

Due to its flexibility of TAM width attachment to individual cores, rectangular 2-D bin packing has evolved as a popular method to solve the test scheduling problem for embedded cores [7]. Each core C_i ($1 \leq i \leq N$) is represented by a set of wrapper configurations R_i. The test resource requirement of core C_i with j^{th} wrapper configuration can be represented by a rectangle whose height and width represent allocated TAM width (w_{ij}) and the corresponding test time ($T(w_{ij})$) respectively. To get a schedule for the full SoC, the rectangles are to be packed into a bin of fixed height (W_{max}) so that *TAT* (width of the bin) is minimized. Bin packing is NP-complete [7]. In the following we have presented a PSO based approach to solve the scheduling problem.

A.2 Test Schedule Generation:

The process consists of the following components.

- Generation of test rectangles for individual cores.
- Selecting one test rectangle for each core.
- Scheduling the selected test rectangles.

The first step is a stand-alone one, while the next two stages need to be solved in an integrated fashion. We have used the *Design_Wrapper* algorithm [2] to generate different wrapper configurations for each core. For a core, the wrapper configurations corresponding to only the pareto-optimal TAM width [2] are noted. Corresponding window-based power profiles are determined. Selection of one test rectangle per core has been performed using PSO. Each particle gives a set of rectangles, one for each core. Fitness of the particle has been evaluated by performing a scheduling of these rectangles.

A.2.1 Particle Swarm Optimization Formulation:

PSO is a population based evolutionary technique designed by Eberhart and Kennedy [79]. It starts with an initial population of particles. Each particle corresponds to a solution to the optimization problem being solved. Each particle has its fitness value. Particles evolve over generations guided by self and group-intelligence, and also via their inertia. Any PSO formulation involves choosing a proper representation of the particles, their fitness calculation, and defining an evolution policy. Next, we elaborate each of these in the context of power-aware test scheduling of cores in a SoC.

A.2.1.1 Particle Structure

Each core has a set of test rectangles. It may be noted that considering the power values; we can say that each core has a set of test cuboids associated with it – the dimensions being wrapper width, test time, and power. However, since we are not considering pattern reordering, the two cuboids of a core cannot differ only in their power values – we will conveniently call them rectangles only. Let the number of cores in the SoC be N and the maximum number of rectangles for any core is M. Let B $=\lceil \log_2 M \rceil$. A particle consists of $N \times B$ number of bits. First B bits identify the test rectangle selected for the first core, second B bits for the second core, and so on. Fig.1. shows a simple particle with N = 4 and B = 4. In this case test rectangles 9, 2, 8 and 13 are selected for cores 1, 2, 3 and 4 respectively.

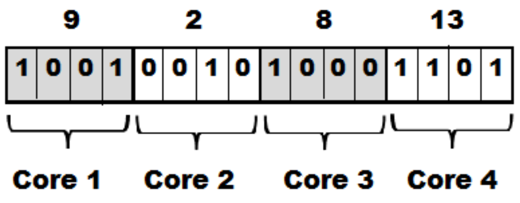

Fig.1.Sample particle structure of 4cores with W_{max} =16 (B = 4)

For the initial generation, particles are generated randomly; however care has been taken to ensure that the indices generated for a core are always within the total number of rectangles of it. Fitness of a particle is equal to the total test time (TAT) of the SoC after scheduling the test rectangles using the procedure noted in Section A.2.1.3.

A.2.1.2 Evolution of a particle

In a PSO formulation, evolution of a particle is guided by three factors – its own intelligence, global (swarm) intelligence, and the inertia factor. A particle always remembers its history about its best structure over generations. This is called the local best (*pbest*) of the particle. In a particular generation, the particle with the best fitness values is the global best (*gbest*) of the generation. For the initial generation, *pbest* of each particle is initialized to itself, while the *gbest* of the generation is the best one of the first generation. In the successive generations, new particles are created using the *replace* operator noted next.

The *replace* operator attempts to align a particle with its *pbest* and the *gbest* particles, with some probability. For the sake of this alignment, the *replace* operator is applied at each individual bit position of a particle. For bit position i of a particle, the bit is replaced by the corresponding bit of *pbest* particle with probability α. After the operator has been applied for *pbest*, the same is done with respect to *gbest* with probability of replacement, β. After both the replacement operators have been applied to all bit positions for a core, a consistency check is performed. If the new rectangle number for the core becomes larger than the total number of rectangles available for the core, the rectangle number is reverted back to its value in the original particle. In our experimentation, we have kept both α and β values at 0.1. Fig.2 shows an example alignment of a particle towards its local best.

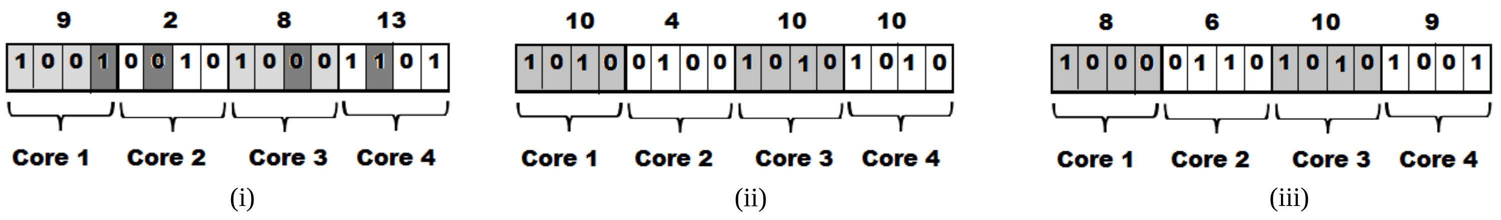

Fig.2. (i) current particle, (ii) local best and (iii) evolved particle

A.2.1.3 Scheduling of test rectangles

The algorithm takes as input the rectangle set corresponding to the particle and the maximum TAM width W_{max}. It performs a scheduling of the rectangles, honouring the constraints that at no instant of time, the total TAM width requirement exceeds W_{max}. The resulting total test time (TAT) is the fitness of the particle. At any point of time, the algorithm maintains the following data structures to arrive at a decision about scheduling the next core.

- ***Break_Point_List*** **(*BP*):** A set of time instants at which the power requirement of the schedule has changed from its value in the previous instant. The next core can be scheduled at any time of the breakpoints, $bp_k \in BP$.

- ***Available_TAM_Width_Info*** **(*ATW*):** A set with cardinality same as BP. The value atw_k is equal to the total free TAM width available at break point instant bp_k.

As the till unscheduled cores get scheduled, the list *BP* and *ATW* also get updated. The bin packing procedure also needs to prioritize the next unscheduled rectangle to be selected for packing (scheduling). For this purpose the rectangles are sorted on their area values (TAM width (w) × test time (T)) in a descending order. The break-point list *BP* is scanned from the minimum to the maximum value.

To make the schedule compact, we try to utilize any available TAM resource at every break-point. Hence, for the break-point bp_k, the algorithm scans the unscheduled rectangle list to check for the largest rectangle that can be scheduled at bp_k. If none are feasible, the algorithm advances to the next break-point. When rectangles corresponding to all cores have been scheduled, the maximum end time of testing of all cores gives the total test application time. The algorithm to produce the schedule is presented next.

Algorithm Schedule_Rectangles

```
Inputs:
    List of rectangles to be scheduled
    $W_{max}$, the maximum TAM width
Var:
    BP: A list of break points.
    ATW: List of available TAM widths at each break point $bp \in BP$.
Begin
    $BP \leftarrow \{0\}$
    $ATW \leftarrow \{0\}$
    Sort list of rectangles on decreasing area
    Mark all rectangles as unscheduled
    While there exists unscheduled rectangles do
        While all entries of BP not checked do
            Let $bp_k$ be the next entry of BP
            $atw_k \leftarrow ATW[bp_k]$
            Check if any rectangle picked up in sorted order can be scheduled at $bp_k$ with available TAM
            resource.
            If yes then
                Update BP, ATW and Rectangle List
                Mark corresponding rectangle scheduled
            Else then
                Continue with next $bp_k \in BP$
        End while
    End while
    Return the maximum test end time among all rectangles.
End
```

A.3 Experimental results:

In this section we present results of experimentation with ITC'02 [80-81] benchmark SoCs. As noted in Section 2, several techniques have been developed to solve the basic test scheduling problem of SoC. Among them evolutionary approaches [8-13] do better than other iterative heuristics. Since PSO is an evolutionary algorithm, we have compared our results with several other approaches [8-13] noted in Table I. All these approaches use a flexible TAM architecture. It may be noted that our PSO performs better than all other approaches for benchmark *p93791*.

TABLE I: Comparisons of PSO with other scheduling procedure for benchmark *p93791* for different W_{max}

p93791	LBT [5]	GA [13]	ACO [12]	B*-SA [10]	EA(C) [8]	EA(nC) [8]	SA [9]	PSO
16	1746657	1750830	1747504	1782067	1754980	1754980	1757452	**1720725**
24	1164442	1170620	1175988	1190565	1171190	1184630	1169945	**1150467**
32	873334	877073	891103	890092	886038	900388	878493	**864460**
40	698670	704272	716112	707664	706820	724758	718005	**695200**
48	582227	587117	598286	609580	600986	611029	594575	**579761**
56	499053	505586	517692	517017	501057	520868	509041	**496559**
64	436673	442455	452951	452245	445748	458389	447974	**435838**

B. POWER-AWARE TEST SCHEDULING

B.A. SINGLE-FREQUENCY POWER-AWARE TEST SCHEDULING:

B.A.1 Problem Formulation:

Suppose a SoC with N cores C_1, C_2 ...C_N is to be tested with a maximum of W_{max} TAM resources and a maximum power limit P_{max}. The test scheduling problem is to allocate TAM resources and test times to the cores so that, the total test application time (*TAT*) is minimized and the power consumed by the SoC at each instant of time during test is less than P_{max}.

Power constrained test scheduling takes the basic scheduling problem to a 3-D bin packing problem, where power represents the third dimension. We have used the window-based peak power profile for each wrapper configuration of each core.

B.A.2 Window-Based Peak Power Model:

During testing of a core, test pattern bits shift into the wrapper scan-in chains, launch the test (single capture cycle) and corresponding test responses shift out through the wrapper scan-out chains. Transitions occur in the scan cells at the time of shift-in, launch-and-capture and shift-out operations. While a test response shifts out, the next test stimulus shifts in. We consider the total number of transitions in the wrapper chains in a particular clock cycle as the power consumption of that cycle.

B.A.2.1 Transition Count in Multiple Wrapper Chains

Let a certain wrapper configuration of a core C_i have l wrapper chains with wrapper scan-in lengths SI_1, SI_2, SI_3.... SI_l and wrapper scan-out lengths SO_1, SO_2, SO_3.... SO_l. If the core is tested with p test patterns, total test time (T) can be calculated as [1],

$$T = (1 + \max(SI_i, SO_i)) * p + \min(SI_i, SO_i), \quad (1 \le i \le l) \qquad \text{(Eqn. 1)}$$

The length of test stimuli is equal to the sum of primary inputs (*PI*), Bidirectional lines (*Bidir*) and number of scan cells (*SC*) while the length of test responses is given by the sum of primary outputs (*PO*), *Bidir* and *SC*s. The test stimuli and test responses split themselves according to the length of the wrapper-scan-in and wrapper-scan-out chains, respectively, in case of multiple wrapper-chain configurations. Sometimes it is necessary to pad some extra bits to the test patterns to match its length with the quantity in Eqn. (1). Cycle accurate power values can be computed for all sets of wrapper configurations of each core.

For bigger circuits with longer test time values, it is very difficult to handle a large number of associated power values in the test schedule generation process. From the cycle accurate power profiles it can also be observed that for most of the time instants, there is not significant variation in the power profile in the neighbourhood of it. This has motivated us to make the power-model coarse-grained via windowing.

In window-based peak power model, we partition the total test time of a core into some smaller sized time windows and consider a single peak power in that interval to represent the power value for that interval. The window-based power model gives us a faster test scheduling strategy than cycle-accurate power model since now we need to work with less number of power values. A single global peak-power model introduces high amount of overestimation compared to the actual instantaneous power values. The window-based power model has the capability to reduce the amount of overestimation. The accuracy depends on the window-size to a large extent. In our formulation the length of the window interval has been kept flexible and can be tuned according to the requirements and computational resources available. Obviously schedule generation time depends on the length of the window interval. Very small window interval leads towards the cycle accurate power model, while large window-size incorporates more false power values in the schedule. Fig.3 shows the concept of window-based power model. It shows the cycle-accurate power values, global peak power of 15 units and the window-based model with window-size 10. For example, for the time interval 0 to 10, the peak value is 8 and thus, it has been used as the power value for the entire window. From the figure it can be observed that the amount of overestimation of power is much less in the window-based model compared to the global peak power model. For the example in Fig.2, the number of power values to be remembered is only 5 in the window-based model, compared to 50 for the cycle-accurate case. For the full 50 cycle operation, overestimation in global peak power model is 353 (computed by summing up the differences between the global peak of 15 and the instantaneous power values), while in window-based model, it reduces to 143.

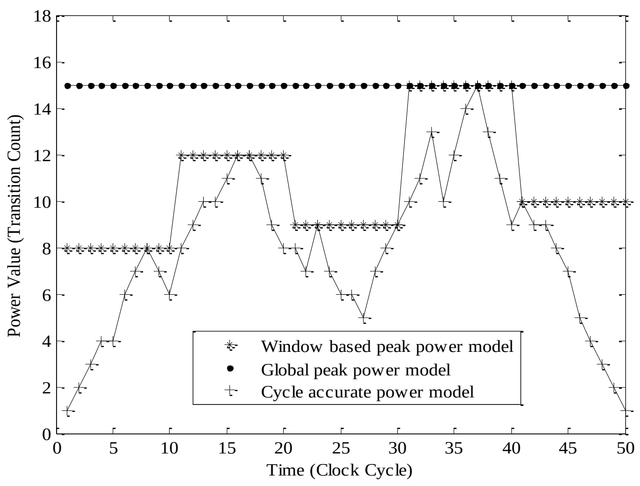

Fig.3.Comparison of window-based power model with other power models

B.A.3 Test Schedule Generation:

The PSO guided *Schedule_Rectangle* algorithm used for basic test scheduling (Section A.2) is used with added power constraint to solve the problem. A *Power_Tracker* (*PT*) keeps track of the power profile at the time of scheduling. It is a data-structure with cardinality same as *BP* and holds the total power consumed by the already scheduled cores at the corresponding break point instant. Power requirements are checked to ensure satisfaction of power limit at every break-point till the end of schedule for the current core. *PT* gets updated with *BP* and *ATW* when an unscheduled core gets scheduled.

B.A.4 Experimental results:

Since many of the power-aware SoC tests scheduling approaches consider a single power value for the entire duration of testing a core, first we show results of our approach using this global peak power model. Next the test scheduling result of window-based peak power model is presented.

B.A.4.1 Test scheduling with global peak power model:

The peak power values for cores are taken as those mentioned in [16]. Table II note the power-aware test scheduling results for the benchmark *p93791*, corresponding to different power limits. While the work reported in [16, 19-21] consider only this global peak power values, [23-24] have reported scheduling results for both global and cycle-accurate power models. Our results, noted under the column marked “PSO GP” are better than other techniques including the cycle-accurate model of [23- 24], for most of the cases.

B.A.4.2 Test scheduling with window-based peak power model:

Working with the window-based power model requires detailed knowledge about the test patterns and the corresponding core power profiles. In the absence of this information for the benchmarks, we have randomly generated the test pattern sets for cores, guided by the number of inputs, bidirectional lines, outputs, scan cells. Appropriate numbers of scan chains have been generated and transition in them for the test sets have been calculated for each cycle. Total number of scan transitions in a cycle has been taken as the measure of power consumption by the core at that cycle. Since power profiles are also dependent on the wrapper configurations, we have also computed the global peak values for individual wrapper configurations of cores. This we call TAM width dependent global peak power (marked as TP in Table III). Window-based peak power is marked as WP in the table. For different power limits, test time results have been noted in Table V, for benchmark *p93791*. Here also, in most of the cases, results corresponding to the window-based peak power model are better than those for simple global peak power model (GP) and TAM-width dependent peak-power.

TABLE II. Comparisons of PSO with other scheduling procedure for *p93791* with different P_{max} and W_{max}

p93791	P_{max} = 10000						
W_{max}	MC [16]	Cycle Accurate [23,24]		GA [19]	ACO [21]	SFLA [20]	PSO GP
		GP	CA				
16	1827819	-	-	1770954	1815761	1803224	**1744001**
24	1220469	-	-	1192015	1227691	1210668	**1175050**
32	1117385	-	-	1033988	1031103	1222874	**980153**
40	1091210	-	-	841594	-	-	871920
48	691866	-	-	609751	639255	626725	**606544**
56	629051	-	-	573720	573720	573720	**556735**
64	568734	-	-	552410	-	-	**441808**
p93791	P_{max} = 20000						
16	1827819	1835416	1829232	1759656	1787856	**1660342**	1730385
24	1220469	1233680	1233716	1174517	187161	1203156	**1159733**
32	957921	932323	934069	886869	912503	993822	**871780**
40	821575	766353	769378	712053	-	-	**702252**
48	658132	640602	640615	600632	623013	611527	**583762**
56	549669	550636	539815	508947	573720	598228	**507406**
64	493599	485297	492463	450977	-	-	**444511**
p93791	P_{max} = 25000						
16	1827819	1829176	1829232	1756326	-	-	**1730030**
24	1220469	1233680	1233716	1173939	-	-	**1155331**
32	965383	929974	934069	883885	-	-	**868726**
40	821475	748140	748154	708150	-	-	**697041**
48	639217	612046	625476	599339	-	-	**580743**
56	549669	545322	539815	508437	-	-	**500678**
64	493599	485297	481893	449657	-	-	**438386**

TABLE III. Comparisons of GP and TP with WP using PSO for different ITC'02 SoCs for different P_{max} and W_{max}

p93791	P_{max} = 15000			P_{max} = 20000			P_{max} = 25000		
W_{max}	GP	TP	WP	GP	TP	WP	GP	TP	WP
16	1780678	1775971	**1771797**	1769798	1775971	**1769673**	1766795	1772665	**1763891**
24	1199309	1188150	**1178986**	1197543	1185322	**1178986**	1195045	1183200	**1178741**
32	894150	897678	**888064**	890300	891100	**888064**	890300	891100	**888038**
40	722541	722286	**718005**	722541	721215	**710508**	718005	718005	**710508**
48	603951	602658	**592200**	600710	599453	**592200**	600710	592798	**592200**
56	520719	520868	**513658**	516802	517238	**512188**	512614	514462	**509565**
64	456848	452943	**450523**	451659	452943	**449633**	451659	**447244**	**447244**

B.B MULTI-FREQUENCY POWER-AWARE TEST SCHEDULING:

B.B.1 Problem Formulation:

Suppose a SoC with N cores C_1, C_2, C_3...C_N is to be tested with a maximum of W_{max} TAM resources and a maximum power limit P_{max}. Each core can be tested at any of the five frequency levels (*f/4*, *f/2*, *f*, *2f* and *4f*). The test scheduling problem is to choose appropriate operating frequencies of individual cores and allocate TAM resources and test times to them so that, the total test application time (TAT) is minimized. Also the power consumed by the SoC at each instant of time during test has to be less than P_{max}.

B.B.2 Multi-frequency Test Infrastructure:

In modern days SoCs, most of the cores have the facilities of being tested at multiple frequency levels. This allows us to shift the test data from ATE to the scan chains at different possible frequencies. Different shift frequencies of a core produce different power profile of the core. Power consumption of a core can be described by the following equation.

$$p = {}^{1}\!/_{2}\, C V_{DD}^{2} \alpha f \qquad \text{(Eqn. 2)}$$

Where, C is the output capacitance, V_{DD} is the supply voltage, α is the number of switching activities in the scan chains and f is the shifting frequency of the scan chains. As power is directly proportional to the shifting frequency, an increase in the shift frequency increases the power by the same proportion. However, the test time of the core reduces by same ratio. Similarly for low frequency operation although the power consumption reduces, test time of the core increases. These variations of the test time and power profile with shift frequency of cores introduce lots of flexibilities in the schedule. A test controller needs to select a suitable operating frequency of a core among the available frequency ranges to get a better schedule.

However, it should be noted that ATE can be operated only at a single frequency, while the cores can be tested at different frequency levels. This frequency mismatch between cores and the ATE can be resolved by bandwidth matching between ATE and cores. The cores, operating at higher frequencies should be provided with more resources than normal frequency operation condition, while the low frequency cores require less resource to fulfil the bandwidth requirement criteria.

B.B.2.1 Multi-frequency architecture

To control the test data rate between ATE and the cores operating at different frequency level, TAM architecture needs to be modified. We propose a multiplexer based architecture that coordinates between ATE and cores, to select proper resources. Figure 4 shows the MUX based multi-frequency architecture. For the sake of simplicity we have assumed that a core can operate at five frequency levels ($f/4$, $f/2$, f, $2f$ and $4f$), while the ATE can send data only at a single frequency f. Let, n be the number of TAM wires required to test the core if both the ATE and the core operate at frequency f. Suppose a core operates at frequency $4f$, total bandwidth (BW) required to test the core is BW = $n \times 4f$. So, the ATE, which sends data at a rate of f, has to allocate $4n$ numbers of TAM lines to test the core to fulfil the bandwidth criteria. A test controller decides the operating frequency of individual cores and generates a three bit signal, which chooses the appropriate number of TAM resources required to test the core. A 4:1 and a 2:1 MUX are used for this purpose.

Test controller generated signals and corresponding variation in required TAM, test time and power values have been noted in Table IV.

TABLE IV. Variation of different test parameters with operating frequency of circuit under test

Controller signal			Core frequency	ATE frequency	TAM required	Test time	Power
0	×	×	f	f	n	t	p
1	0	0	$f/4$	f	$n/4$	$4t$	$p/4$
1	0	1	$f/2$	f	$n/2$	$2t$	$p/2$
1	1	0	$2f$	f	$2n$	$t/2$	$2p$
1	1	1	$4f$	f	$4n$	$t/4$	$4p$

Unlike [30] we do not use test data multiplexer (TDM) or test data demultiplexer (TDdeM) to overcome the frequency gap between ATE and core. Instead our simple MUX based approach can dynamically select the operating frequency of the core during scheduling and allocate the resources. However, it may be noted that the test controller is restricted by the maximum bandwidth of $BW_{max} = W_{max} \times f_{\text{ATE}}$ (f_{ATE} is the frequency of ATE, i.e. f). Total allocated resource to all the cores tested parallelly is also limited to W_{max}.

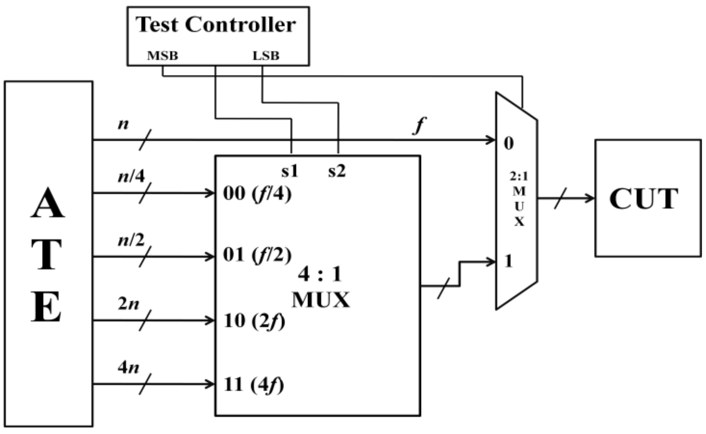

Fig.4 Multi-frequency architecture

B.B.3 Power Model:

The window-based peak power model mentioned in section B.A.2 is used for this purpose.

B.B.4 Test Schedule Generation:

The process consists of the following components.

- Generation of test rectangle (TR_i, $1\leq i \leq N$) for each core.
- Selection of one test rectangle for each core.
- Selection of operating frequency for each core and calculation of frequency factor $FF_{C_i} = \frac{f_{C_i}}{f_{\text{ATE}}}$ ($1\leq i \leq N$).
- Modification of each test rectangle according to the corresponding frequency factor.
- Scheduling the modified test rectangles (MTR_i, $1\leq i \leq N$).

B.B.4.1 Particle Swarm Optimization Formulation:

The particle structure of single-frequency test schedule is modified by incorporating frequency part into it, for the purpose of multi-frequency test scheduling. The modified particle structure is presented next.

B.B.4.1.1 Particle Structure

Let the number of cores in the SoC be N, the maximum number of rectangles for any core be M and the number of frequency values be F. Let $B = \lceil log_2 M \rceil$ and $K = \lceil log_2 F \rceil$. A particle consists of $B \times N + K \times N$ number of bits. First $B \times N$ bits select the rectangle indices for the cores, while next $K \times N$ bits select their corresponding frequencies. The decimal equivalent of first B bits identifies the test rectangle selected for the first core, second B bits for the second core, and so on. The decimal equivalent of each K-bit value of remaining $K \times N$ bits selects a frequency. For example, for K = 3 and $F = 5$, values are in the range 0 to 4 ($0 \leftarrow f/4$, $1 \leftarrow f/2$, $2 \leftarrow f$, $3 \leftarrow 2f$ *and* $4\leftarrow 4f$). Figure 5(i) shows a simple particle with $N = 4$ and $B = 4$, $F = 5$. In this case test rectangles 9, 2, 8 and 13 are selected for cores 1, 2, 3 and 4 and their corresponding frequencies are $f/2$, $2f$, $4f$ and f respectively. For the initial generation, particles are generated randomly; however care has been taken to ensure that the indices generated for a core are always within the total number of rectangles of it and also the bandwidth ($w_{ij} \times f_{C_i}$) allocated to any core C_i is within the limit of maximum allowable bandwidth value of $W_{max} \times f_{ATE}$. Figure 5 shows an example alignment of a particle towards its local best. The dark shaded bits of the initial particle (Fig. 5(i)) are replaced by the bits in the same position as the local best particle (Fig. 5(ii)) to get the evolved particle (Fig. 5(iii)).Fitness of a particle is equal to the total test time (TAT) of the SoC after scheduling the test rectangles using the procedure noted in Section B.B.4.1.2.

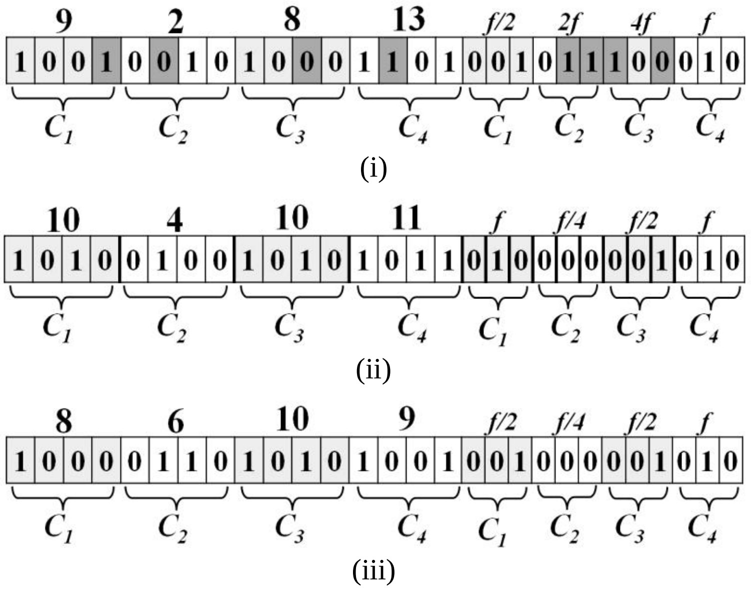

Fig. 5 Particle structure; (i) initial particle, (ii) local best particle and (iii) evolved particle

B.B.4.1.2 Scheduling of test rectangles:

The test rectangles can be scheduled using the method mentioned in Section A.2.1.3.

improvement %

B.B.5 Experimental Results:

Table V shows comparative study between window-based peak power model and global peak power model as well as between single frequency testing and multifrequency testing. To have a better understanding of the effect of different power models on TAT, we have compared the window-based power model results with global peak power model, keeping multi-frequency test environment as constant for both the cases. Similarly the comparison between multifrequency approach and single frequency approach has been carried out under same window-based peak power profile. In Table V the global peak multi-frequency approach is marked as GPMF, while WPSF and WPMF indicate window-based peak power single frequency approach and window-based peak power multi-frequency approach respectively.

Table V. Comparison of different test scheduling approaches for different P_{max} and W_{max} for benchmark *p93791*

p93791	P_{max} = 12000			P_{max} = 15000			P_{max} = 20000			P_{max} = 25000		
W_{max}	**GPMF**	**WPSF**	**WPMF**	**GPMF**	**WPSF**	**WPMF**	**GPMF**	**WPSF**	**WPMF**	**GPMF**	**WPSF**	**WPMF**
16	1555974	NS	**1500754**	1541789	1771797	**1500754**	1520095	1769673	**1479144**	1483264	1763891	**1479713**
24	1097268	NS	**1086643**	1086643	1178986	**1078119**	1078119	1178986	**1072684**	1074274	1178741	**1052881**
32	862866	NS	**845676**	862749	888064	**835715**	836302	888064	**835442**	836302	888038	**835442**
40	704919	NS	**704271**	700943	718005	**691604**	674746	710508	**664247**	666076	710508	**664247**
48	590197	NS	**589752**	588713	592200	**580268**	586362	592200	**581577**	582601	592200	**572976**
56	515930	NS	**513953**	513953	513658	**513658**	505525	512188	**498674**	503408	509565	**460304**
64	454672	NS	**448202**	442768	450523	**441430**	440796	449633	**440241**	440796	447244	**440241**

C. Thermal-Aware Test Scheduling

C.1 Problem Formulation:

Suppose a SoC with N cores C_1, C_2 ...C_N is to be tested with a maximum of W_{max} TAM resources, a maximum power limit P_{max}, and a maximum temperature limit T_{max}. The test scheduling problem is to allocate TAM resources and test times to the cores so that, the total test application time (*TAT*) is minimized, while the power consumption during testing remains under P_{max} and the maximum temperature of any core does not cross the temperature upper bound T_{max}.

C.2 Power Model:

The window-based peak power model mentioned in Section B.A.2 is used here.

C.3 Superposition Principle Based Thermal Model:

Temperature of a core can be estimated using well known thermal simulators, such as, HotSpot [49]. HotSpot is a tool which invokes the floorplan and the power profile of a chip as input and provides the thermal profile of the chip as the output. Thermal profile includes transient analysis as well as steady state analysis of the temperature distribution of the chip. Transient analysis records the intermediate change in the thermal profile due to the change of the power profile over time, while steady state analysis records the final steady state temperature.

As test scheduling problem is NP-Hard, finding an optimal schedule for SoCs with large number of embedded cores using any meta-search technique is time consuming. Most of the meta-search techniques are based on iterative methods, which try to find better solution over several iterations. Temperature of cores in the scheduling interval is required to get a thermal safe test schedule. It requires online thermal simulation of the cores at the time of scheduling. However, one major drawback of the thermal simulator HotSpot is its execution time, which restricts us to invoke it inside the scheduling procedure. An alternate solution is to incorporate some kind of thermal model in the scheduling procedure, which can predict the temperature of the cores during scheduling without online Hotspot thermal simulations. We have used a superposition principle based thermal model, which takes the help of linearity of thermal model of HotSpot. Our thermal model uses offline HotSpot thermal simulations for each core in different possible conditions and creates thermal databases for all those conditions. These database information are used for the scheduling purpose. This superposition principle based thermal model, although do not use online Hotspot simulation, produces same temperature as Hotspot simulator produces. It helps to generate much faster thermal safe test schedule than those uses online Hotspot simulations. This thermal superposition model has been suggested in [47, 48], however, our model is more general as it includes both heating and cooling of cores, subject to the conditions of its neighbors.

To get the exact thermal behavior of a core, it is very much important to observe different conditions, which can change the temperature of a core. The temperature of a core C_i can increase due to the following activities.

- Before C_i starts its testing, its ambient temperature increases due to its leakage power consumption (Fig. 6(i)).
- During the period of testing of the core C_i, its temperature increases (Fig. 6(ii)).
- If any numbers of other cores C_j ($1\leq j \leq N$) ($j \neq i$) are tested in parallel with C_i, due to the lateral spreading of heat from C_j, the temperature of C_i increases (Fig. 6(ii)).
- Ambient temperature of C_i increases due to the effect of all the cores which have already been scheduled before C_i starts their testing (Fig. 6(iii)).

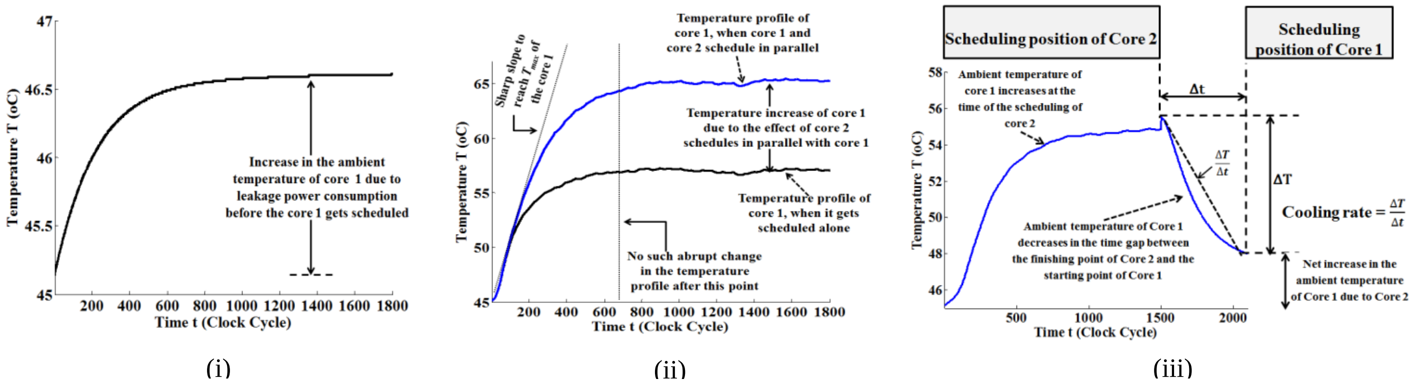

Fig.6. (i) Pre scheduling Increase of the ambient temperature of core 1 of SoC *k10* due to leakage power
(ii) Temperature profile of core 1 of SoC *k10* during its scheduling
(iii) Temperature profile of core 1 as an effect of core 2 schedules before core1 of SoC *k10*

Fig. 6(i) shows the increase in the ambient temperature due to leakage power consumptions of core 1 before it starts its testing in SoC *k10.* It may be noted from the simulation result that, after certain clock cycles, there is no further increase in the ambient temperature due to leakage power consumption.

To check the exact heating effect on a core during its test, different conditions need to be considered. First we have considered the situation in which only core 1 is active and all other cores are idle and measured the temperature increase of core 1, because of its own heating. After that we have shown the heating effects because of lateral heat spread from neighboring cores, which are tested in parallel. Fig.4 shows the extra heating effect on core 1, if it is active in parallel with core 2. It may be noted that, lateral heat spreading of core 2 causes an extra temperature increase of up to 8ºC of core 1, when it is tested in parallel with core 2. An important observation from Fig. 6(ii) is that the temperature rises to its peak value quickly, and after that the temperature profile is almost flat and no such abrupt change can be noticed. It is quite justified from the fact that there are not that many fluctuations in the middle of the power profile as the total number of switching activities does not vary much from one cycle to the next cycle, hence little variations in the dynamic power consumptions in the middle.

Fig. 6(iii) shows the effect of already scheduled cores on the ambient temperature of a core, considered to be scheduled next. When core 1 is idle and core 2 is being tested, the ambient temperature of core 1 increases by 10ºC, because of lateral heat spreading from core 2. Now, if core 1 wants to start its test just after core 2 has finished, the ambient temperature of core 1 will be 10ºC more than its normal ambient temperature. This will increase the maximum temperature of core 1 and if it is more than the temperature budget of the SoC, it may cause a permanent damage in the chip due to burning. However, if core 1 is starts its test after a certain time interval after the completion of the test of core 2, in that time gap, its ambient temperature reduces towards its normal ambient temperature, which leads to more thermal safe schedule than the previous one. Thermal model should be able to handle all these conditions and generate a test schedule that does not violate the thermal constraint.

The objective of the superposition principle based thermal model is to sum up the individual effects of other cores on the temperature of a particular core to get their cumulative effect on the temperature of the core. The operation of our superposition principle based thermal model can be described by the following steps.

- Calculate the increase in the ambient temperature due to leakage power consumption for each core. Create a thermal database of it.
- For each core C_i ($1 \leq i \leq N$) calculate its temperature rise when it is tested alone. Then calculate extra increase in temperature of C_i due to core C_j ($1 \leq j \leq N$) ($i \neq j$), when C_j is tested in parallel with C_i. This extra increase in temperature of C_i is calculated for each C_j separately. Create a thermal database of it.

The database is basically a matrix of size $N \times N$. For each row i in the matrix, the (i, i) element stores the value of temperature increase of the i^{th} core, while it is tested alone. All the other elements (i , j) of i^{th} row store the values of extra increase of temperature of core C_i because of parallel testing of core C_j with core C_i. Superposition principle is applied to calculate the actual temperature increase of core C_i considering the other cores, which are active in parallel with C_i during its scheduling.

- For each core C_i $(1 \leq i \leq N)$ calculate its ambient temperature increase due to the core C_j $(1 \leq j \leq N)$ $(i \neq j)$, which is scheduled prior to C_i. Let us assume, the ambient temperature increase is *T1*. Also calculate the cooling rate for the combination of C_i and C_j. The ambient temperature of C_i reduces in the time gap between the end point of C_j and start point of C_i. Cooling rate can be calculated from the slope of the temperature decrease curve as mentioned in Fig. 6(iii). Although the temperature decrease portion of the curve is not exactly linear, still a linear prediction of this portion of the curve can be a good approximation to calculate the simplified cooling rate as mentioned in Fig. 6(iii). If the time gap is Δt and the temperature decrease in Δt time is ΔT then,

$$\textbf{Cooling rate} = \frac{\Delta T}{\Delta t} \qquad \text{(Eqn. 3)}$$

So the actual increase in the ambient temperature of C_i is $(T1 - \Delta T)$. This is done for each C_j corresponding to each C_i. Two thermal database matrices of size $N \times N$ store all these information. Superposition principle is applied to calculate actual increase in temperature of C_i, considering all C_j that are scheduled prior to C_i.

C.4 Test Schedule Generation:

The overall scheduling strategy can be described by the following steps.

C.4.1 Step I:

The first step is a stand-alone one, while the next stages need to be solved in an integrated fashion.

- ***Wrapper Design***: We have used the *Design_Wrapper* algorithm [2] to generate different wrapper configurations for each core. For a core, the wrapper configurations corresponding to only the pareto-optimal TAM width [2] are noted.
- ***Power Profile Generation***: Window-based power profile corresponding to each pareto-optimal point of each core is determined using the method mentioned in Section B.A.2.
- ***Thermal database creation***: As mentioned in the thermal model in Section C.3, our scheduling strategy does not invoke any thermal simulator during scheduling. Instead of that, we create four thermal databases to store different thermal values using offline HotSpot [49] thermal simulations for each core. *Thermal_Database_1 (TD1)* stores the values of increase in the ambient temperature due to leakage power consumption of each core. *Thermal_Database_2 (TD2)* is used to store the temperature values of self testing as well as parallel testing. The increase in the ambient temperature of a core because of previously scheduled cores is stored using two thermal databases *Thermal_Database_3 (TD3)* and *Thermal_Database_4 (TD4).* Cooling rates of each core can be calculated using *TD3* and *TD4.* The procedures of generation of thermal databases are mentioned in *Procedure_TD1, Procedure_TD2, Procedure_TD3* and *Procedure_TD4.*

Procedure_TD1

```
1. for each core $C_i$ $(1 \leq i \leq N)$
2.     Create power profile $P_{leakage_i}$ like this
3.         Leakage power of $C_i$ for 2000 clock cycles
4.         No power for all other cores $C_j$ $(j \neq i)$
5.     Feed $P_{leakage_i}$ and floorplan of the SoC to HotSpot
6.     Calculate the maximum temperature value $T_{leakage_i}$ from the transient and steady state responses of $C_i$
7.     Enter $T_{leakage_i}$ in TD1.
8. end for
```

Procedure_TD2

```
1.  for each core $C_i$ $(1 \le i \le N)$
2.      Create power profile $P_{self_i}$ like this
3.          Dynamic power of $C_i$ for the clock cycles required to test $C_i$
4.          Leakage power for all other cores $C_j$ $(j \ne i)$ for the same number of clock cycles
5.      Feed $P_{self_i}$ and floorplan of the SoC to HotSpot
6.      Calculate the maximum temperature value $T_{self_i}$ from the transient and steady state responses of $C_i$
7.      Enter $T_{self_i}$ in the $(i , i)$ position of the matrix TD2.
8.      for each core $C_j$ $(1 \le j \le N)$ $(j \ne i)$
9.          Create power profile $P_{parallel_{ij}}$ like this
10.             Dynamic power of $C_i$ and $C_j$ for the clock cycles required to test $C_i$
11.             Leakage power for all other cores $C_k$ $(k \ne j \ne i)$ for the same number of clock cycles
12.         Feed $P_{parallel_{ij}}$ and floorplan of the SoC to HotSpot
13.         Calculate the maximum temperature value $T_{parallel_{ij}}$ from the transient and steady state responses of
            $C_i$
14.         $T_{extra_{ij}} = T_{parallel_{ij}} - T_{self_i}$
15.         Enter $T_{extra_{ij}}$ in the $(i , j)$ position of the matrix TD2.
16.     end for
17. end for
```

Procedure_TD3

```
1.  for each core $C_i$ $(1 \le i \le N)$
2.     for each core $C_j$ $(1 \le j \le N)$ $(j \ne i)$
3.         Create power profile $P_{preschedule1_{ij}}$ like this
4.             Dynamic power of $C_j$ for the clock cycles required to test $C_j$
5.             Leakage power for all other cores $C_k$ $(k \ne j)$ including $C_i$ for the same number of clock cycles
6.         Feed $P_{preschedule1_{ij}}$ and floorplan of the SoC to HotSpot
7.         Calculate the maximum temperature value $T_{preschedule1_{ij}}$ from the transient and steady state responses
           of $C_i$
8.         Enter $T_{preschedule1_{ij}}$ in the $(i , j)$ position of the matrix TD3.
9.     end for
10. end for
```

Procedure_TD4

```
1.  for each core $C_i$ $(1 \le i \le N)$
2.      for each core $C_j$ $(1 \le j \le N)$ $(j \ne i)$
3.          Create power profile $P_{preschedule2_{ij}}$ like this
4.              Dynamic power of $C_j$ for the clock cycles required to test $C_j$
5               Leakage power of $C_j$ for next 2000 number of clock cycles
6.              Leakage power for all other cores $C_k$ $(k \ne j)$ including $C_i$ for the total number of clock cycles
7.          Feed $P_{preschedule2_{ij}}$ and floorplan of the SoC to HotSpot
8.          Calculate the final temperature value $T_{preschedule2_{ij}}$ of $C_i$
9.          Enter $T_{preschedule2_{ij}}$ in the $(i , j)$ position of the matrix TD4.
10.     end for
11. end for
```

In *Procedure_TD1,* $P_{leakage_i}$ considers the leakage power of core C_i for 2000 clock cycles and no power for other cores. It may be noted that for all the cores we have worked with, there is no variation in the thermal profile after 2000 clock cycles, due to only leakage power consumption. So, considering only 2000 clock cycles is sufficient to get the effect of leakage power on temperature. $T_{leakage_i}$ is the maximum increase in ambient temperature of C_i due to leakage power consumption. This value is stored in the i^{th} position of an array of size *N*, named *TD1*.

In *Procedure_TD2,* T_{self_i} is the temperature increase of core C_i, when no other cores are tested in parallel with C_i. $T_{parallel_{ij}}$ is the maximum increase in temperature of C_i, when it is tested in parallel with C_j. $T_{extra_{ij}}$ is the extra increase in temperature of C_i as an effect of parallel testing of C_j with C_i. Suppose for example, cores C_1, C_3 and C_7 are active in parallel. So, the temperature of core C_1 can be calculated by adding the (1, 1), (1, 3) and (1, 7) elements of the matrix *TD2*. .

In *Procedure_TD3,* $P_{preschedule1_{ij}}$ considers core C_j as active and C_i as the idle core. $T_{preschedule1_{ij}}$ calculates the maximum increase in the ambient temperature of C_i as an effect of earlier scheduling of C_j. In *Procedure_TD4,* we allow C_i to cool down towards its normal ambient temperature for 2000 clock cycles. $T_{preschedule2_{ij}}$ is the ambient temperature of C_i after 2000 clock cycles. The cooling rate of C_i for the combination of C_i and C_j can be calculated as

$$\text{Cooling Rate } (CR_{ij}) = \frac{T_{preschedule1_{ij}} - T_{preschedule2_{ij}}}{2000} \qquad \text{(Eqn. 4)}$$

This is basically the difference between $(i, j)^{\text{th}}$ elements of *TD3* and *TD4* divided by 2000. We can calculate the actual reduction in the ambient temperature of C_i, by multiplying CR_{ij} with the time gap (*TG*) between end point of C_j and start point of C_i. Net increase in ambient temperature of C_i because of earlier scheduling of C_j can be calculated as

$$\text{Net Ambient Increase } (NAI_{ij}) = T_{preschedule1_{ij}} - (CR_{ij} \times TG) \qquad \text{(Eqn. 5)}$$

C.4.2 Step II:

In this step selection of one test rectangle per core has been performed using PSO. The PSO mentioned in Section A.2.1 is used for this purpose.

C.4.3 Step III:

C.4.3.1 Scheduling of test rectangles:

The algorithm takes as input the rectangle set corresponding to the particle, the maximum TAM width W_{max}, the maximum power limit P_{max} and maximum allowable temperature T_{max}. It performs a scheduling of the rectangles, honouring the constraints that at no instant of time, the total TAM width requirement exceeds W_{max}, and the instantaneous power value does not exceed P_{max}. Also the maximum temperature of individual cores does not exceed T_{max}. The resulting total test time (TAT) is the fitness of the particle. At any point of time, the algorithm maintains the following data structures to arrive at a decision about scheduling the next core.

- *Break_Point_List* (*BP*): A set of time instants at which the power requirement of the schedule has changed from its value in the previous instant. The next core can be scheduled at any time of the breakpoints, $bp_k \in BP$.
- *Available_TAM_Width_Info* (*ATW*): A set with cardinality same as BP. The value atw_k is equal to the total free TAM width available at break point instant bp_k.
- *Power_Tracker* (*PT*): This is also a set with cardinality same as *BP* and holds the total power consumed by the already scheduled cores at the corresponding break point instant.
- *Temperature_Tracker* (*TT*): It keeps track of the temperature of each core during scheduling.

A *Power_Violation_Checker (PVC)* checks whether there is any power violation during scheduling. If at any particular point in the schedule, a core does not respect *PVC,* it does not get the permission to get scheduled at that point. Similarly a *Thermal_Violation_Checker (TVC)* checks whether any core is violating thermal budget or not. *TVC* checks the scheduling position of a core and uses superposition principle to add different thermal values from *TD1, TD2* and *TD3,* corresponding to that core to calculate the exact temperature of the core. Suppose a core starts its scheduling from the 0th clock cycle, there will be no leakage power consumption of the core before it starts scheduling. Naturally, the increase in the ambient temperature should not be considered for that particular core, while the same effect has to be considered for the cores which remain idle initially and consume leakage power. Similarly, when a new core schedules, the temperature profiles of the already scheduled cores also get modified because of the newly scheduled core. *TVC* checks all these possible conditions and decides whether the new core can be scheduled at that break-point bp_k or not. The temperature calculation procedure of *TVC* is mentioned next in *Procedure_TVC.*

Procedure_TVC

```
1.  if (a new core $C_i$ wants to schedule at any $bp_k$)
2.      Calculate temperature $T_i$ of $C_i$ like this
3.      if (idle time of $C_i$ > 1000 clock cycles)
4.          Add $T_{leakage_i}$ from $TD1$ to $T_i$
5.          Check thermal violation $C_i$
6.          if (thermal violation)
7.              Do not schedule $C_i$ at $bp_k$
8.          end if
9.      end if
10.     Add $T_{self_i}$ from $TD2$ to $T_i$
11.     Check thermal violation of $C_i$
12.         if (thermal violation)
13.             Do not schedule $C_i$ at $bp_k$
14.         end if
15.     for each core $C_j$ $(1 \leq j \leq N)$ $(j \neq i)$
16.         Check parallelism between $C_i$ and $C_j$
17.         if (parallel)
18.             Add $T_{extra_{ij}}$ to $T_i$
19.             Add $T_{extra_{ji}}$ to $T_j$
20.             Check thermal violation $C_i$ and $C_j$
21.                 if (thermal violation)
22.                     Do not schedule $C_i$ at $bp_k$
23.                 end if
24.         else
25.             Calculate $CR_{ij}$
26.             Find scheduling time gap ($TG$) between $C_j$ and $C_i$
27.             Net increase in ambient temperature of $C_i$ is ($T_{preschedule1_{ij}}$- ($CR_{ij} \times TG$))
28.         Add it to $T_i$
29.             Check thermal violation of $C_i$
30.             if (thermal violation)
31.                 Do not schedule $C_i$ at $bp_k$
32.             end if
33.         end if
34.     end for
35. end if
```

As the till unscheduled cores get scheduled, the list *BP*, *ATW*, *PT* and *TT* also get updated. The bin packing procedure also needs to prioritize the next unscheduled rectangle to be selected for packing (scheduling). For this purpose the rectangles are sorted on their area values (TAM width (w) × test time (T)) in a descending order. The break-point list *BP* is scanned from the minimum to the maximum value.

To make the schedule compact, we try to utilize any available TAM resource, power budget and temperature budget at every break-point. Hence, for the break-point bp_k, the algorithm scans the unscheduled rectangle list to check for the largest rectangle that can be scheduled at bp_k. If none are feasible, the algorithm advances to the next break-point. Power requirements and temperature validations are also checked to ensure satisfaction of power limit and temperature limit at every break-point till the end of schedule for the current core. When rectangles corresponding to all cores have been scheduled, the maximum end time of testing of all cores gives the total test application time. The algorithm to produce the schedule is presented next.

Test Scheduling Algorithm

Input: $LIST_C$, W_{max}, P_{max}, T_{max}.
Output: A schedule of all the cores respecting all the constraints with minimum *TAT*.

```
1. begin
2.     while (all cores are not scheduled) do
3.         $bp_k \leftarrow$ Break point with minimum time value.
4.         $atw_k \leftarrow$ Available tam resource at $bp_k$.
```

```
5.      while ($atw_k$ > 0) do
6.        Set $Area_{max}$ = 0
7.        for all unscheduled cores $C_i$ do
8.          if ($w_{ij} \leq atw_k$) then
9.            if ($P_{ij}$ respects $PVC$) then
10.             if ($C_i$ and all earlier scheduled cores respect $TVC$) then
11.               Calculate $Area_{ij} = w_{ij} * T(w_{ij})$
12.             end if
13.           end if
14.           if ($Area_{ij} > Area_{max}$) then
15.             Set $Area_{max} = Area_{ij}$
16.             Set $C_{imax} = C_i$
17.           end if
18.         end if
19.       end for
20.       Select $C_{imax}$ for scheduling at $bp_k$
21.       Update $BP$
22.       Update $ATW$
23.       Update $PT$ with $P_{ij}$
24.       Update $TT$ with current temperature values of all scheduled cores
25.       $Start_{Ci} = bp_k$
26.       $End_{Ci} = bp_k + T(w_{ij})$
27.       Mark $C_i$ as scheduled
28.     end while
29.     Remove $bp_k$ and $atw_k$ from $BP$ and $ATW$ respectively
30.   end while
31. end
```

C.5 Experimental Result:

We have divided this section into two parts. Fewer details of the traditional ITC'02 SoC benchmarks have motivated us to form new SoCs with sufficient details to observe exact thermal behavior of them. The 1st sub-section describes the details of our newly formed SoCs, while the next sub-section presents the results of our experimentations on those SoCs.

C.5.1 Experimental setup:

To get exact thermal behaviour of cores in a chip, the input power profiles of the cores and the floorplan of the chip must be exact and specific. So we have developed two SoCs named *k10* and *k25* having 10 and 25 cores respectively using *ITC'99* [82], *ISCAS 89* [83] and *ISCAS 85* circuits.

The infrastructure of formation of SoCs can be described by the following steps.

1) Each circuit in *Verilog* format (.v) is taken as input and mapped to Faraday 90nm standard cell library [84]. The circuits are synthesized using *Synopsys Design Vision Compiler* [37] to generate gate-level netlist from the *Verilog* design description.
2) All the flip-flops are replaced with scan flip-flops using *Synopsys DFT Compiler* for the testability purpose.
3) Multiple scan chains are inserted to reduce the test application time.
4) Dynamic and leakage powers are extracted from *Synopsys Design Vision Report Power* tool [85].
5) Area of each core is calculated using *Cadence Encounter Compiler* [86].
6) Exact floorplan of the SoCs are calculated using Floorplanning tool *Blobb* and *Plottool*. As different cores have different sizes, there are some empty spaces in the floorplan of the SoCs. These empty spaces are packed with zero power boxes to support temperature simulation using HotSpot tool. The packing of empty spaces with zero power boxes are done using a C program.
7) For most of the circuits, test vectors are generated using *Synopsys TetraMax ATPG* [87]. For larger circuits like *b22, b17* from *ITC'99*, *Atalanta-M-2.0* [88] has been used to generate test vectors. Test pattern compression is not performed.

C.5.2 *Results:*

In this part, simulation results of our experimentation are shown with newly formed SoCs. Window-based peak power model and superposition principle based thermal model are used along with PSO guided test scheduling algorithm to obtain thermal-aware test schedule. Table VI shows the results for newly formed SoCs *k10*.

We have tried to show a comparison of our thermal model with the thermal model presented in [47, 48]. For this purpose we have considered same SoC information (mentioned in experimental setup) and compared the efficiency between the two thermal models. Different test schedules have been generated considering both the thermal models. Post-schedule Hotspot thermal simulations of the schedules have been carried out to finally check whether there are any thermal violations in the schedules or not. Table VII presents minimum allowable temperature values to get a valid test schedule for both the thermal models and their corresponding maximum temperature values obtained from post-schedule Hotspot simulations. It may be noted that, although the thermal model presented in [47, 48] produces valid test schedules for lesser temperature than our thermal model, however it does not ensure thermal safe test schedule, as it violates thermal limits in most of the cases. On the other hand, our thermal model violates the thermal limit for none of the cases. This is because of our thermal model takes care of the heating effect of pre-scheduled cores on a particular core, which the thermal model of [47, 48] might not have considered. This shows the efficiency of our thermal model.

TABLE VI: Variation in Test Application Time (TAT) for different SoCs with the variation of P_{max} and T_{max}

	P_{max} = 1.3 Watt		P_{max} = 1.5 Watt		P_{max} = 2 Watt		P_{max} = 5 Watt	
	T_{max} (°C)	*TAT*	T_{max} (°C)	*TAT*	T_{max} (°C)	TAT	T_{max} (°C)	*TAT*
k10 W_{max} = 64	72	43154	72	42494	72	42365	72	42365
	80	41424	80	41388	80	41347	80	40924
	85	39349	85	37875	85	37875	85	37875
	90	39349	90	37875	90	37875	90	37875
	95	39349	95	36984	95	36110	95	35256
	100	39349	100	35880	100	34810	100	34459
	105	39349	105	35865	105	34810	105	34072
	110	39349	110	35732	110	34268	110	34021

TABLE VII: Comparison of our thermal model with the thermal model presented in [47, 48]

k10	Thermal Model [47, 48]					Our Thermal Model				
W_{max}	P_{max} (Watt)	T_{max} (°C)	TAT (Clock cycle)	Max temp (°C)	Thermal Violation	P_{max} (Watt)	T_{max} (°C)	TAT (Clock cycle)	Max temp (°C)	Thermal Violation
24	1.3	70	93935	**71.11**	**Yes**	1.3	72	99428	70.92	No
	1.5	70	93935	**71.12**	**Yes**	1.5	72	99428	70.92	No
	2	70	92835	**71.05**	**Yes**	2	72	97752	71.54	No
	5	70	90468	**71.17**	**Yes**	5	72	96925	70.92	No
32	1.3	70	70161	**70.74**	**Yes**	1.3	72	73571	70.74	No
	1.5	70	69727	**71.56**	**Yes**	1.5	72	72238	70.74	No
	2	70	68870	**70.74**	**Yes**	2	72	72109	70.74	No
	5	70	68126	**71.3**	**Yes**	5	72	71423	71.81	No
40	1.3	70	58435	**70.66**	**Yes**	1.3	72	61804	70.66	No
	1.5	70	57858	**70.66**	**Yes**	1.5	72	58270	70.66	No
	2	70	57539	**70.66**	**Yes**	2	72	57355	70.66	No
	5	70	56053	**71.42**	**Yes**	5	72	57355	71.77	No
48	1.3	70	49802	**70.79**	**Yes**	1.3	72	54666	70.66	No
	1.5	70	49802	**70.8**	**Yes**	1.5	72	54534	70.66	No
	2	70	49802	**70.86**	**Yes**	2	72	54516	70.66	No
	5	70	49802	**70.86**	**Yes**	5	72	52288	70.66	No
56	1.3	70	49802	**70.74**	**Yes**	1.3	72	50285	70.91	No
	1.5	70	46475	**70.91**	**Yes**	1.5	72	50285	70.91	No
	2	70	44698	**70.93**	**Yes**	2	72	50285	70.91	No
	5	70	44591	**70.91**	**Yes**	5	72	49010	70.66	No
64	1.3	70	44043	**71.2**	**Yes**	1.3	72	43154	70.66	No
	1.5	70	43530	**70.91**	**Yes**	1.5	72	42494	70.91	No
	2	70	44043	**71.32**	**Yes**	2	72	42365	71.1	No
	5	70	43018	**70.91**	**Yes**	5	72	42365	70.66	No

V. WORK TO BE DONE

Another major challenge in SoC testing is the storage cost of huge test data volume. Test data can be stored in a compressed manner in the ATE by efficiently filling the don't-care bits. At the same time intelligent don't-care filling is able to generate low temperature test set. Although, both of the problems rely on don't-care bit filling, most of the existing works have considered them as separate problems. The main focus of our future work is-

- ***Thermal-aware test data compression***:

The test data compression works try to fill up the don't-care bits to get better compression ratio ignoring the thermal effect, while the thermal-aware don't-care filling works try to minimize the temperature by efficient don't-care bit filling without taking care of test data compression. It may so happen that, for a particular don't-care bit, better compression can be achieved if it is filled with '0' value, while a '1' value for that bit produces lower temperature during testing. A trade-off between these two don't-care bit filling techniques is required to obtain good compression ratio with reasonably low temperature.

Dictionary-based coding technique is a popular method for test data compression. In this method a clique-partition heuristic is used to generate different matching test slices from the original test slices. However, different clique selections and don't-care filling techniques generate different clique sets with a variation in compression ratio as well as temperate profile of the chip.

The main objective of this work is to develop a thermal-aware test data compression technique, which can fill up the don't-care bits smartly taking care of compression and temperature simultaneously to bridges the gap between test data compression and thermal-aware don't-care filling techniques.

PUBLICATIONS SO FAR

1. Rajit Karmakar, Aditya Agarwal and Santanu Chattopadhyay, "Particle Swarm Optimization Guided Multi-frequency Power-Aware System-on-Chip Test Scheduling Using Window-Based Peak Power Model", VLSI Design and Test (VDAT), 2014

2. Rajit Karmakar and Santanu Chattopadhyay, "Window Based Peak Power Model and Particle Swarm Optimization Guided 3-D Bin Packing for SoC Test Scheduling", communicated in Elsevier Integration, the VLSI Journal.

3. Rajit Karmakar and Santanu Chattopadhyay, "Particle Swarm Optimization Guided Bin-Packing Approach for Power- and Thermal-Aware Test Scheduling of System-on-Chip", communicated in IEEE Transaction on Computers.

References

[1] E. Marinissen, S. Goel, and M. Lousberg, “Wrapper design for embedded core test,” in Test Conference, 2000. Proceedings. International, 2000, pp. 911–920.

[2] V. Iyengar, K. Chakrabarty, and E. Marinissen, “Test wrapper and test access mechanism co-optimization for system-on chip,” in Test Conference, 2001. Proceedings. International, 2001, pp. 1023–1032.

[3] A. Sehgal, S. Goel, E. Marinissen, and K. Chakrabarty, “Ieee p1500- compliant test wrapper design for hierarchical cores,” in Test Conference, 2004. Proceedings. ITC 2004. International, Oct 2004, pp. 1203–1212.

[4] V. Iyengar, K. Chakrabarty, and E. Marinissen, “Efficient wrapper/tam co-optimization for large socs,” in Design, Automation and Test in Europe Conference and Exhibition, 2002. Proceedings, 2002, pp. 491–498.

[5] S. Goel and E. Marinissen, “Effective and efficient test architecture design for socs,” in Test Conference, 2002. Proceedings. International, 2002, pp. 529–538.

[6] Y. Huang, W.-T. Cheng, C.-C. Tsai, N. Mukherjee, O. Samman, Y. Zaidan, and S. Reddy, “Resource allocation and test scheduling for concurrent test of core-based soc design,” in Test Symposium, 2001. Proceedings. 10th Asian, 2001, pp. 265–270.

[7] V. Iyengar, K. Chakrabarty, and E. Marinissen, “On using rectangle packing for soc wrapper/tam co-optimization,” in VLSI Test Symposium, 2002. (VTS 2002). Proceedings 20th IEEE, 2002, pp. 253–258.

[8] Y. Xia, M. Chrzanowska-Jeske, B. Wang, and M. Jeske, “Using a distributed rectangle bin-packing approach for core-based soc test scheduling with power constraints,” in Computer Aided Design, 2003. ICCAD-2003. International Conference on, Nov 2003, pp. 100–105.

[9] W. Zou, S. Reddy, I. Pomeranz, and Y. Huang, “Soc test scheduling using simulated annealing,” in VLSI Test Symposium, 2003. Proceedings. 21st, April 2003, pp. 325–330.

[10] J. Wuu, T.-C. Chen, and Y.-W. Chang, “Soc test scheduling using the b*-tree based floorplanning technique,” in Design Automation Conference, 2005. Proceedings of the ASP-DAC 2005. Asia and South Pacific, vol. 2, Jan 2005, pp. 1188–1191 Vol. 2.

[11] Y. Yu, X. Y. Peng, Y. Peng. “A test scheduling algorithm based on two-stage ga.” International Symposium on Instrumentation Science and Technology, pp. 658–662, 2006.

[12] J.-H. Ahn and S. Kang, “Soc test scheduling algorithm using aco-based rectangle packing,” in Computational Intelligence, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2006, vol. 4114, pp. 655–660

[13] C. Giri, S. Sarkar, and S. Chattopadhyay, “Test scheduling for core-based socs using genetic algorithm based heuristic approach,” in Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2007, vol. 4682, pp. 1032–1041.

[14] R. Chou, K. Saluja, and V. Agrawal, “Scheduling tests for vlsi systems under power constraints,” Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, vol. 5, no. 2, pp. 175–185, June 1997.

[15] Y. Huang, S. Reddy, W.-T. Cheng, P. Reuter, N. Mukherjee, C.-C. Tsai, O. Samman, and Y. Zaidan, “Optimal core wrapper width selection and soc test scheduling based on 3-d bin packing algorithm,” in Test Conference, 2002. Proceedings. International, 2002, pp. 74–82.

[16] J. Pouget, E. Larsson, and Z. Peng, “Soc test time minimization under multiple constraints,” in Test Symposium, 2003. ATS 2003. 12th Asian, Nov 2003, pp. 312–317.

[17]] K. Chakrabarty, “Design of system-on-a-chip test access architectures under place-and-route and power constraints,” in Design Automation Conference, 2000. Proceedings 2000, 2000, pp. 432–437.

[18] E. Larsson and Z. Peng, “Test scheduling and scan-chain division under power constraint,” in Test Symposium, 2001. Proceedings. 10th Asian, 2001, pp. 259–264.

[19] C. Giri, S. Sarkar, and S. Chattopadhyay, “A genetic algorithm based heuristic technique for power constrained test scheduling in core-based socs,” in Very Large Scale Integration, 2007. VLSI - SoC 2007. IFIP International Conference on, Oct 2007, pp. 320–323.

[20] X. L. Cui, X. M. Shi, H. Li, and C. L. Lee, “A shuffle frog-leaping algorithm for test scheduling of 2d/3d soc,” in Solid-State and Integrated Circuit Technology (ICSICT), 2012 IEEE 11th International Conference on, Oct 2012, pp. 1–3.

[21] X. L. Cui, W. Cheng, C LI, “Effective soc test scheduling under peak power constraints based on the ant colony algorithm,” In Proceedings IEEE International Conference on Test and Measurement (ICTM), pp. 189,192, 2010.

[22] P. Rosinger, B. Al-Hashimi, and N. Nicolici, “Power profile manipulation: a new approach for reducing test application time under power constraints,” Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, vol. 21, no. 10, pp. 1217–1225, Oct 2002.

[23] S. Samii, E. Larsson, K. Chakrabarty, and Z. Peng, “Cycle-accurate test power modeling and its application to soc test scheduling,” in Test Conference, 2006. ITC ’06. IEEE International, Oct 2006, pp. 1–10.

[24] S. Samii, M. Selkala, E. Larsson, K. Chakrabarty, and Z. Peng, “Cycleaccurate test power modeling and its application to soc test architecture design and scheduling,” Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, vol. 27, no. 5, pp. 973–977, May 2008.

[25] X. Kavousianos, K. Chakrabarty, A. Jain, and R. Parekhji, “Test schedule optimization for multicore socs: Handling dynamic voltage scaling and multiple voltage islands,” Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, vol. 31, no. 11, pp. 1754–1766, Nov 2012.

[26] V. Sheshadri, V. Agrawal, and P. Agrawal, “Optimal power-constrained soc test schedules with customizable clock rates,” in SOC Conference (SOCC), 2012 IEEE International, Sept 2012, pp. 271–276.

[27] S. Millican and K. Saluja, “Formulating optimal test scheduling problem with dynamic voltage and frequency scaling,” in Test Symposium (ATS), 2013 22nd Asian, Nov 2013, pp. 165–170.

[28] V. Sheshadri, V. Agrawal, and P. Agrawal, “Power-aware soc test optimization through dynamic voltage and frequency scaling,” in Very Large Scale Integration (VLSI-SoC), 2013 IFIP/IEEE 21st International Conference on, Oct 2013, pp. 102–107.

[29] X. Qiang and N. Nicolici, “Wrapper design for multifrequency ip cores,” in Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, vol. 13, no. 6, 2005, pp. 678,685.

[30] T. Yoneda, K. Masuda, and H. Fujiwara, “Power-constrained test scheduling for multi-clock domain socs,” in Design, Automation and Test in Europe, 2006. DATE ’06., vol. 1, March 2006, pp. 1,6,6–10.

[31] Z. Dan, H. Ronghua, T. Yoneda, and H. Fujiwara, “Power-aware multi-frequency heterogeneous soc test framework design with floorceiling packing,” in Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium, 27-30 May 2007, pp. 2942,2945.

[32] Z. Dan, U. Chandran, T. Yoneda, and H. Fujiwara, “Shelf packing to the design and optimization of a power-aware multi-frequency wrapper architecture for modular ip cores,” in Design Automation Conference, 2007. ASP-DAC ’07. Asia and South Pacific, 23-26 Jan 2007, pp. 714,719.

[33] Z. Dan, H. Ronghua, and H. Fujiwara, “Multi-frequency modular testing of socs by dynamically reconfiguring multi-port ate,” in Asian Test Symposium, 2007. ATS ’07. 16th, 8-11 Oct 2007, pp. 107,110.

[34] P. Rosinger, B. Al-Hashimi, and K. Chakrabarty, “Rapid generation of thermal-safe test schedules,” in Design, Automation and Test in Europe, 2005. Proceedings, March 2005, pp. 840–845 Vol. 2.

[35] C. Liu, K. Veeraraghavan, and V. Iyengar, “Thermal-aware test scheduling and hot spot temperature minimization for core-based systems,” in Defect and Fault Tolerance in VLSI Systems, 2005. DFT 2005. 20th IEEE International Symposium on, Oct 2005, pp. 552–560.

[36] P. Rosinger, B. Al-Hashimi, and K. Chakrabarty, “Thermal-safe test scheduling for core-based system-on-chip integrated circuits,” Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, vol. 25, no. 11, pp. 2502–2512, Nov 2006.

[37] Z. He, Z. Peng, P. Eles, P. Rosinger, and B. Al-Hashimi, “Thermal-aware soc test scheduling with test set partitioning and interleaving,” Journal of Electronic Testing, vol. 24, no. 1-3, pp. 247–257, 2008.

[38] Z. He, Z. Peng, and P. Eles, “A heuristic for thermal-safe soc test scheduling,” in Test Conference, 2007. ITC 2007. IEEE International, Oct 2007, pp. 1–10.

[39] T. Yu, T. Yoneda, K. Chakrabarty, and H. Fujiwara, “Thermal-safe test access mechanism and wrapper co-optimization for system-on-chip,” in Asian Test Symposium, 2007. ATS ’07. 16th, Oct 2007, pp. 187–192.

[40] Z. He, Z. Peng, and P. Eles, “Simulation-driven thermal-safe test time minimization for system-on-chip,” in Asian Test Symposium, 2008. ATS ’08. 17th, Nov 2008, pp. 283–288.

[41] D. Bild, S. Misra, T. Chantemy, P. Kumar, R. Dick, X. Huy, L. Shangz, and A. Choudhary, “Temperature-aware test scheduling for multiprocessor systems-on-chip,” in Computer-Aided Design, 2008. ICCAD 2008. IEEE/ACM International Conference on, Nov 2008, pp. 59–66.

[42] T. Yu, T. Yoneda, K. Chakrabarty, and H. Fujiwara, “Test infrastructure design for core-based system-on-chip under cycle-accurate thermal constraints,” in Design Automation Conference, 2009. ASP-DAC 2009. Asia and South Pacific, Jan 2009, pp. 793–798.

[43] Z. He, Z. Peng, and P. Eles, “Thermal-aware test scheduling for core-based soc in an abort-on-first-fail test environment,” in Digital System Design, Architectures, Methods and Tools, 2009. DSD ’09. 12th Euromicro Conference on, Aug 2009, pp. 239–246.

[44] Z. He; Z. Peng; P. Eles, “Multi-temperature testing for core-based system-on-chip,” in Design, Automation Test in Europe Conference Exhibition (DATE), 2010, March 2010, pp. 208–213.

[45] C. Yao, K. Saluja, and P. Ramanathan, “Temperature dependent test scheduling for multi-core system-on-chip,” in Test Symposium (ATS), 2011 20th Asian, Nov 2011, pp. 27–32.

[46] C. Yao, K. Saluja, and P. Ramanathan, “Thermal-aware test scheduling using on-chip temperature sensors,” in VLSI Design (VLSI Design), 2011 24th International Conference on, Jan 2011, pp. 376–381.

[47] C. Yao, K. Saluja, and P. Ramanathan, “Power and thermal constrained test scheduling under deep submicron technologies,” Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, vol. 30, no. 2, pp. 317 322, Feb 2011.

[48] C. Yao, K. Saluja, and P. Ramanathan, “Partition based soc test scheduling with thermal and power constraints under deep submicron technologies,” in Asian Test Symposium, 2009. ATS ’09., Nov 2009, pp. 281–286.

[49] M. R. Stan, K. Skadron, M. Barcella, W. Huang, K. Sankaranarayanan, and S. Velusamy, “Hotspot: a dynamic compact thermal model at the processor-architecture level.” Microelectronics Journal, vol. 34, no. 12, pp. 1153–1165, 2003.

[50] N. Touba, “Survey of test vector compression techniques,” Design Test of Computers, IEEE, vol. 23, no. 4, pp. 294–303, April 2006.

[51] A. Jas and N. Touba, “Test vector decompression via cyclical scan chains and its application to testing core-based designs,” in Test Conference,1998. Proceedings., International, Oct 1998, pp. 458–464.

[52] A. Chandra and K. Chakrabarty, “System-on-a-chip test-data compression and decompression architectures based on golomb codes,” Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, vol. 20, no. 3, pp. 355–368, Mar 2001.

[53] A. Chandra and K. Chakrabarty, “Test data compression and test resource partitioning for system-on-a-chip using frequency-directed runlength (fdr) codes,” Computers, IEEE Transactions on, vol. 52, no. 8, pp. 1076–1088, Aug 2003.

[54] P. Gonciari, B. Al-Hashimi, and N. Nicolici, “Variable-length input huffman coding for system-on-a-chip test,” Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, vol. 22, no. 6, pp. 783–796, June 2003.

[55] S. Reddy, K. Miyase, S. Kajihara, and I. Pomeranz, “On test data volume reduction for multiple scan chain designs,” in VLSI Test Symposium, 2002. (VTS 2002). Proceedings 20th IEEE, 2002, pp. 103–108.

[56] L. Li, K. Chakrabarty, and N. A. Touba, “Test data compression using dictionaries with selective entries and fixed-length indices,” ACM Trans. Des. Autom. Electron. Syst., vol. 8, no. 4, pp. 470–490, Oct. 2003.

[57] A. Wurtenberger, C. Tautermann, and S. Hellebrand, “Data compression for multiple scan chains using dictionaries with corrections,” in Test Conference, 2004. Proceedings. ITC 2004. International, Oct 2004, pp.926–935.
[58] L. Li and K. Chakrabarty, “Test data compression using dictionaries with fixed-length indices [soc testing],” in VLSI Test Symposium, 2003. Proceedings. 21st, April 2003, pp. 219–224.
[59] P. Sismanoglou and D. Nikolos, “Input test data compression based on the reuse of parts of dictionary entries: Static and dynamic approaches,” Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, vol. 32, no. 11, pp. 1762–1775, Nov 2013.
[60] K. Basu and P. Mishra, “Test data compression using efficient bitmask and dictionary selection methods,” Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, vol. 18, no. 9, pp. 1277–1286, Sept 2010.
[61] A. Jas, J. Ghosh-Dastidar, M.-E. Ng, and N. Touba, “An efficient test vector compression scheme using selective huffman coding,” Computer- Aided Design of Integrated Circuits and Systems, IEEE Transactions on, vol. 22, no. 6, pp. 797–806, June 2003.
[62] S. Reda and A. Orailoglu, “Reducing test application time through test data mutation encoding,” in Design, Automation and Test in Europe Conference and Exhibition, 2002. Proceedings, 2002, pp. 387–393.
[63] Z. Wang and K. Chakrabarty, “Test data compression for ip embedded cores using selective encoding of scan slices,” in Test Conference, 2005. Proceedings. ITC 2005. IEEE International, Nov 2005, pp. 10 pp.–590.
[64] I. Bayraktaroglu and A. Orailoglu, “Concurrent application of compaction and compression for test time and data volume reduction in scan designs,” Computers, IEEE Transactions on, vol. 52, no. 11, pp.1480–1489, Nov 2003.
[65] S. Mitra and K. S. Kim, “Xpand: an efficient test stimulus compression technique,” Computers, IEEE Transactions on, vol. 55, no. 2, pp. 163–173, Feb 2006.
[66] C. V. Krishna and N. Touba, “Adjustable width linear combinational scan vector decompression,” in Computer Aided Design, 2003. ICCAD-2003. International Conference on, Nov 2003, pp. 863–866.
[67] G. Mrugalski, J. Rajski, and J. Tyszer, “Ring generators - new devices for embedded test applications,” Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, vol. 23, no. 9, pp. 1306–1320, Sept 2004.
[68] A. Pandey and J. Patel, “Reconfiguration technique for reducing test time and test data volume in illinois scan architecture based designs,” in VLSI Test Symposium, 2002. (VTS 2002). Proceedings 20th IEEE, 2002, pp. 9–15.
[69] S. Samaranayake, E. Gizdarski, N. Sitchinava, F. Neuveux, R. Kapur, and T. Williams, “A reconfigurable shared scan-in architecture,” in VLSI Test Symposium, 2003. Proceedings. 21st, April 2003, pp. 9–14.
[70] H. Tang, S. Reddy, and I. Pomeranz, “On reducing test data volume and test application time for multiple scan chain designs,” in Test Conference, 2003. Proceedings. ITC 2003. International, vol. 1, Sept 2003, pp. 1079–1088.
[71] N. Sitchinava, E. Gizdarski, S. Samaranayake, F. Neuveux, R. Kapur, and T. Williams, “Changing the scan enable during shift,” in VLSI Test Symposium, 2004. Proceedings. 22nd IEEE, April 2004, pp. 73–78.
[72] L.-T. Wang, X. Wen, H. Furukawa, F.-S. Hsu, S.-H. Lin, S.-W. Tsai, K. Abdel-Hafez, and S. Wu, “Virtualscan: a new compressed scan technology for test cost reduction,” in Test Conference, 2004. Proceedings. ITC 2004. International, Oct 2004, pp. 916–925.
[73] K. Butler, J. Saxena, A. Jain, T. Fryars, J. Lewis, and G. Hetherington, “Minimizing power consumption in scan testing: pattern generation and dft techniques,” in Test Conference, 2004. Proceedings. ITC 2004. International, Oct 2004, pp. 355–364.
[74] A. Dutta, S. Kundu, and S. Chattopadhyay, “Thermal aware don’t care filling to reduce peak temperature and thermal variance during testing,” in Test Symposium (ATS), 2013 22nd Asian, Nov 2013, pp. 25–30.
[75] T. Yoneda, M. Nakao, I. Inoue, Y. Sato, and H. Fujiwara, “Temperature variation- aware test pattern optimization,” in European Test Symposium (ETS), 2011 16th IEEE, May 2011, pp. 214–214.
[76] M. Chen and A. Orailoglu, “Scan power reduction for linear test compression schemes through seed selection,” Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, vol. 20, no. 12, pp. 2170–2183, Dec 2012.
[77] X. Liu and Q. Xu, “On x-variable filling and flipping for capture-power reduction in linear decompressor-based test compression environment,” Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, vol. 31, no. 11, pp. 1743–1753, Nov 2012.
[78] A. Chandra and K. Chakrabarty, “Reduction of soc test data volume, scan power and testing time using alternating run-length codes,” in Design Automation Conference, 2002. Proceedings. 39th, 2002, pp. 673– 678.
[79] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in Neural Networks, 1995. Proceedings., IEEE International Conference on, vol. 4, Nov 1995, pp. 1942–1948 vol.4.
[80] E. Marinissen, V. Iyengar, and K. Chakrabarty, “A set of benchmarks for modular testing of socs,” in Test Conference, 2002. Proceedings. International, 2002, pp. 519–528.
[81] ITC’02 SOC Test Benchmarks, http://itc02socbenchm.pratt.duke.edu/.
[82] http://www.cad.polito.it/downloads/tools/itc99.html
[83] W.-T. Cheng and S. Davidson, “Sequential circuit test generator (stg) benchmark results,” in Circuits and Systems, 1989., IEEE International Symposium on, May 1989, pp. 1939–1941 vol.3.
[84] Faraday, http://www.faraday.com.tw/AIP/ips/90library.html as on 17 January 2011.
[85] Design Vision. , “User Guide,” Version 2002.05, Synopsys Inc.,2002.
[86] Encounter, “Encounter user guide,” Product Version 7.1.2, Cadence Inc., 2008.
[87] TetraMax ATPG Guide, Synopsys Inc., 2006.
[88] http://ddd.fit.cvut.cz/prj/Atalanta-M/man.html