Journal of Computer Science 5 (4): 290-296, 2009
ISSN 1549-3636
© 2009 Science Publications

Optimization of Test Scheduling and Test Access for ITC-02 SOC Benchmark Circuits

[1]P. Sakthivel, [2]R. Delhi Babu and [3]P. Narayanasamy
[1]Department of Electronics and Communication Engineering,
Anna University Chennai, Chennai 600025, India
[2]Department of Computer Science and Engineering,
Sri Sivasubramaniya Nadar College of Engineering, Chennai-603 110, India
[3]Department of Computer Science and Engineering, Anna University Chennai,
Chennai 600025, India

Abstract: Problem statement: This study presented the optimized test scheduling and test access for ITC-02 SOC benchmark circuits using genetic algorithm. In the scheduling procedure of SOC, scheduling problem was formulated as a sequence of two problems and solved. **Approach:** Test access mechanism width was partitioned into two and three partitions and the applications of test vectors and test vector assignments for different partitions were scheduled using different operators of genetic algorithm. **Results:** The test application time was calculated in terms of CPU time cycles for two and three partitions of twelve ITC-02 SOC benchmark circuits and the results were compared with the integer linear programming approach. **Conclusion:** The results showed that the genetic algorithm based approach gives better results.

Key words: System-on-chip, test scheduling, test access mechanism, integer linear programming, genetic algorithm, test wrapper

INTRODUCTION

In many of the earlier research studies[2,3,5,9] of the test scheduling for the SOC benchmark circuits[15], scheduling was done using functional bus as the medium for test vector transportation and buffers are inserted between each core to store the test vectors and applying it to each core as per the given constraints and obtained schedule. The buffer size is the hardware overhead and considered as a constraint in the test scheduling. The CLP method[12] was used to schedule the test application.

In this research study, the hardware overhead is not considered as a constraint. Since cores in an SOC are not directly accessible via chip inputs and outputs, special access mechanisms are required to test them at the system level. For each core in the SOC, a Test Access Mechanism (TAM) is built around each core and test vectors are applied through these TAMs. There is conceptual test access architecture for embedded cores[11] with the source, sink and test access mechanism. The TAM is used to deliver test vector from the source to the cores and also to deliver responses from cores to the sink. Test scheduling for various widths of TAM and various number of partitions are carried out.

The general problem of SOC test integration[7,10] includes the design and optimization of wrapper and TAM architectures and test scheduling. Test wrappers form the interface between cores and TAM. TAM transport test data between SOC pins and test wrappers. Test scheduling determines the order in which tests are applied. The focus is on wrapper and TAM co-design to minimize testing time under TAM width constraints[16,17].

In a core based design approach[9], a set of cores is integrated into a system using UDL and interconnections. In this way, complex systems can be efficiently developed. However, the complexity in the system leads to high-test data volumes. So, the design and optimization of test solution are very much important for any test. Hence the following two independent problems are considered:

- Design of an infrastructure for the transportation of test data in the system
- Design of a test schedule to minimize test time

The testable units in an SOC design are the cores, the UDL and the interconnections[8]. The cores are

Corresponding Author: P. Sakthivel, Department of Electronics and Communication Engineering, Anna University Chennai, Chennai 600025, India

usually delivered with predefined test methods and test sets, while the test sets for UDL and interconnections are to be generated prior to test scheduling and TAM Design. The workflow when developing an SOC test solution can mainly be divided into two consecutive parts[10,11] namely (i) An early design space exploration and (ii) An extensive optimization of the final solution. During the process, conflicts and limitations must be carefully considered. For instance, tests may be in conflict with each other due to the sharing of test resources and power consumption. Otherwise the system may be damaged during test. Further, test resources such as external testers support a limited number of scan-chains and limited memory.

Research has been going on in developing techniques for test scheduling, TAM design and testability analysis[5,6]. In this study, a new technique is proposed using Genetic Algorithm for optimizing the test vector for Globally Asynchronous Locally Synchronous (GALS) SOC with the objective to minimize the test application time. The aim of the proposed approach is to reduce the gap between the design space exploration and the extensive optimization that is to produce a high quality solution in respect of test time and TAM at a relatively low computational cost. Earlier research[14] has studied wrapper design or TAM optimization as independent problems. They have not addressed the issue of sizing the TAM to minimize SOC testing time. Alternative approaches that combine TAM design with test scheduling do not address the problem of wrapper design and its relationship to TAM optimization[18,19].

The GA based approach to solve the problems of test scheduling optimization for wrapper design and TAM is presented here. This approach provides improved results, comparable to the existing ILP approach.

The study related to our approach and various issues related to SOC testing and test scheduling techniques, Test vector optimization and test scheduling framework based on genetic algorithm, the experimental results for the 12 SOC benchmark circuits of ITC-02 are presented.

MATERIALS AND METHODS

Soc test scheduling: The basic problem in test scheduling[4] is to assign a start time for all tests. In order to minimize the test application time, tests are scheduled as concurrent as possible. However, various types of constraints must be considered. A test to be scheduled consists of a set of test vectors produced or stored at a test source. The test response from the test is evaluated at a test sink. When applying a test, a test conflict may occur, which must be considered during the scheduling process. For instance, often a testable unit is tested by several test sets. If several tests are used for a testable unit, only one test can be applied to the testable unit at a time.

The tests are scheduled in sessions where tests at cores placed physically close to each other are grouped in the same test session. In a fully BISTed system[2], each core has its dedicated test source and test sink and there might not be any conflicts among tests. However, in general, conflicts among tests may occur during testing.

The test-application time can be minimized by scheduling the execution of the test sets as concurrently as possible. The basic idea in test scheduling is to determine when each test set should be executed. The main objective is to minimize the test application time.

Proposed test access mechanism: The test access mechanism takes care of chip test pattern transport[13,14]. It can be used to transport test stimuli from the test pattern source to the core under test and to transport test responses from the core under test to the test pattern sink. The TAM is, by definition, implemented on the chip.

The wrapper and TAM are structured into the following two problems in the order of increasing complexity[1].

P_A: To determine the test bus assignment to each cores. The TAM is partitioned into different test buses and the problem here is to identify the bus assignment to each core in the SOC.

P_{PA}: To determine a Partition of the total TAM width among given number of TAM and to determine the test bus assignment to each core (P_A). The size of the TAM is given and the TAM should be divided into many partitions according to the requirement. The number of partition required should be obtained first and it will be given as an input to the problem (P_A). Then the problem (P_A) will determine the test bus assignment to each core in the SOC.

Genetic Algorithm Based Problem Formulation for P_A: The problem (P_A) is formulated in such a way that the Genetic Algorithm is used to optimize the solution. In the formulation of P_A, number of cores (N) in SOC and number of test buses (B) of TAM of widths w_1, w_2, w_3, …, w_B are considered. The main objective is to determine the assignment of cores to test buses of TAM such that the assignment is used for test application for

SOC and the total testing time is minimized. Distributing the cores of SOC equally among test buses of TAM and taking the permutations of cores of SOC assigned to test buses of TAM can obtain initial populations for Genetic Algorithm. Then the GA (Selection, Crossover and Mutation) is applied on the initial population to generate new chromosomes (children). The solution to the above problem obtained as a set of chromosome (child) consists of integers in the range 1 to B. Each value in the chromosome set represents the core assignment to the test bus. The 'i'th element of the chromosome set represents the bus number of TAM to which core 'i' of SOC is assigned.

Genetic algorithm based problem formulation for P_{PA}: The problem (P_{PA}) is formulated as a sequence of two problems both of which is solved using Genetic Algorithm. In the formulation of P_{PA}, number of cores (N) of SOC and number of test Buses (B) of TAM of widths w_1, w_2, w_3, …, w_B are considered. The objectives are (i) To determine the distribution of the total TAM width among the given number of TAM and (ii) To determine the assignment of cores of SOC to the test buses of TAM. A chromosome in our approach consists of two parts. (i) The assignment of cores of SOC to test buses of TAM which is a set of integer numbers with 'i'th element representing the test bus number for which the core 'i' of SOC is assigned. (2) The chromosome is the bus width distribution of each test bus of TAM, which is also set of integer numbers where the total of all the integers is equal to the size of TAM. The 'j'th entry of the set represents width of the test bus 'j', such that sum of these widths is equal to TAM width.

Function for total time: Total time is the time required to test all the cores in the system, which is given below. If the core 'i' of SOC is assigned to test bus 'j' of the TAM, then the testing time for core 'i' of SOC is given by:

$$T_i(W_j) = (1 + \max\{L_{wi}, L_{wo}\}) * V_{ni} + \min\{L_{wi}, L_{wo}\}$$

Where:
T_i = Test application time of core "i" in SOC
W_j = Width of test bus 'j'
L_{wi} = Length of the longest wrapper scan-in chain
L_{wo} = Length of the longest wrapper scan-out chain
V_{ni} = Number of test vector for core 'i'

Total test cycles needed to test all the cores in the SOC is:

$$T = \{\sum T_i(W_j) * b_{ij}\},\ 1 <= i <= N \text{ and } 1 <= j <= B$$

where, b_{ij} a binary variable defined as follows:

$$b_{ij} = \begin{cases} 1 & \text{if core 'i' is assigned to bus 'j'} \\ 0 & \text{otherwise} \end{cases}$$

The above problem is NP-Hard problem[1]. Therefore, efficient heuristics and other techniques are needed for large problem instances. In this study, genetic algorithm based approach to effectively solve these problems namely P_A and P_{PA} are presented.

Test vector optimization based on genetic algorithm: Genetic Algorithms can effectively be used to solve the search and optimization problems. The genetic algorithm that is used for generating test sequences for SOC is described. First, the basic idea of the method is given. Then the representation of test conditions, the objective function and some insights into the parameter settings of the genetic algorithm are presented. GAs consist of population of solutions called chromosomes. Here the chromosomes are an encoding of the solution to a given problem. The algorithm proceeds in steps called generations. During each generation, a new population of individuals is created from the old population by applying genetic operators. Given old generation, new generation is built, according to the genetic operations such as selection, 1-point crossover, 2-point crossover, uniform crossover, weight based crossover, 1-point mutation, 2-point mutation and mutation with neighbor.

Selection: This operator selects the individuals from the old generation. The fitness of an individual determines its chances to reproduce. The individual with a better performance possesses higher chances of getting selected. For each parent, two elements are chosen randomly. Only these elements are evaluated by the objective function. The element with higher ranking is selected. Thus, for the selection of two parents only four elements are evaluated instead of the whole population. Various selection schemes such as roulette wheel selection, stochastic universal selection and binary tournament selection with and without replacement are used depend upon the requirement. The objective of the GA is to converge to an optimal individual and selection pressure is the driving force which determines the rate of converges. A high selection pressure will cause the population to converge quickly, possibly at the expense of a suboptimal result. The GA selects individual with probability proportional to their fitness.

Crossover: Once two chromosomes are selected, the crossover operator is used to generate two offspring. The details about 1-point crossover, 2-point crossover, uniform crossover and weight-based crossover operators are illustrated in the Chapter 3. Crossover combines the schemata or building blocks from two different solutions in various combinations. Smaller good building blocks are converted into progressively larger good building blocks over time until a completely good solution is found.

Point mutation: The 1-point Mutation produces incremental random changes in the offspring generated through crossover. Mutation may be done by flipping a bit. One new element C from a parent P is constructed by copying the whole element and changing a bit at a randomly chosen position.

Point mutation: The 2-point mutation is performing 1-point Mutation two times on the same chromosome one after the other. The values of two bits are changed by the 2-point mutation.

Mutation with neighbor: 1-pont Mutation is performed at two adjacent positions on the same element instead of randomly selected positions as in 2-point mutation. The values of two adjacent bits are changed by the mutation with neighbor operation. In the Genetic Algorithm mutation serves the crucial role of replacing the gene values lost from the population during the selection process so that they can be tried in a new context or of providing the gene values that were not present in the initial population.

Pseudo code for the proposed genetic algorithm based method: The pseudo code of the proposed GA based algorithm is shown in the Fig. 1.

```
Genetic Algorithm
Begin
randomly generate the initial population of chromosomes;
arrange the initial population in increasing order of the test
cycle;
while (no improvement in the function for total time)
do
select 10% population of chromosomes as best class;
generate 15% chromosomes using1-point  mutation;
generate 15% chromosomes using mutation with neighbor;
generate 10% chromosomes using 2-point  mutation;
generate 15% chromosomes using 1-point crossover;
generate 15% chromosomes using 2-point crossover;
generate 10% chromosomes using uniform crossover;
generate 10% chromosomes using weight based crossover;
sort the new population in increasing order of the cost.
Enddo
End.
```

Fig. 1: The GA based test vector optimization algorithm

RESULTS

The experiments were conducted for the ITC-02 SOC benchmark circuits. The results were obtained for each of the benchmark circuits by partitioning TAM width into two and three partition. W is the width of Test Access Mechanism. w1, w2 and w3 are the size of the partition 1, partition 2 and partition 3. The vector assignment in the Table 1 is the information about the bus assignment ("1" in the "ith" position indicates the "bus 1" or "partition 1" of size "w1" is assigned to the "ith" core for the transportation of test vector) for test vector transportation of each core in the SOC. ILP cycles are the result of the existing algorithm, which utilized the integer linear programming techniques to solve the SOC test scheduling problem. GA cycles are the result of the proposed experiment, which utilizes Genetic algorithm to solve the problem, In the Table 1, the results of ILP and GA for SOC u226 is presented for the partition size of 16, 24, 32, 40, 48, 56 and 64 bits. The TAM is partitioned into 2 parts. The optimized scheduling of test vectors are obtained for the proposed GA-based method and the required amount of test time that is the number of CPU cycles are obtained and tabulated. These values are also plotted for each partition against the number of CPU cycles in the Fig. 2. From the results and comparison graph, it is observed that the amount of CPU cycle required for the GA-based method is relatively less than the ILP-based method. Further, if the size of the TAM gets increased, the amount of time required for test application also gets reduced.

Table 1: Results of ILP versus proposed GA approach for SOC u226 with two partitions

W	W1+w2	Vector assignment	ILP cycles	GA cycles
16	8+8	1,1,2,1,2,2,1,1,2	38400	36340
24	11+13	2,2,1,1,2,2,1,2,1	38324	35942
32	12+20	1,1,1,1,2,2,2,2,2	37430	35690
40	18+22	2,2,2,1,1,1,1,1,2	37112	34987
48	24+24	2,1,1,2,1,1,1,1,2	36985	34439
56	30+26	1,1,1,2,2,2,1,1,1	35876	33856
64	48+16	2,1,1,2,1,1,2,1,1	34678	32560
Average		36972	34830	

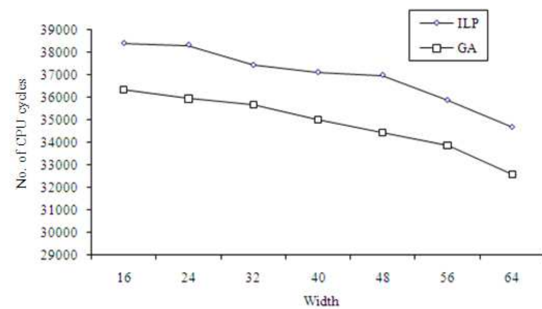

Fig. 2: Comparison of ILP with GA for SOC with two partitions

Table 2: Results of ILP versus proposed GA Approach for SOC u226 with three partitions

W	w1+w2+w3	Vector assignment	ILP cycles	GA cycles
16	5+5+6	1,2,3,1,1,1,1,3,2	34439	31234
24	8+8+8	1,1,1,2,2,2,3,3,3	32567	29345
32	10+12+10	2,3,1,1,2,3,3,2,1	30456	27430
40	15+15+10	3,3,2,2,1,3,2,1,1	29876	26345
48	15+25+8	2,3,2,1,1,3,3,1,1	28976	25987
56	20+12+24	2,3,1,1,3,2,1,3,3	28123	25234
64	32+16+16	1,3,2,2,1,3,3,2,1	27154	24126
Average			30227	27100

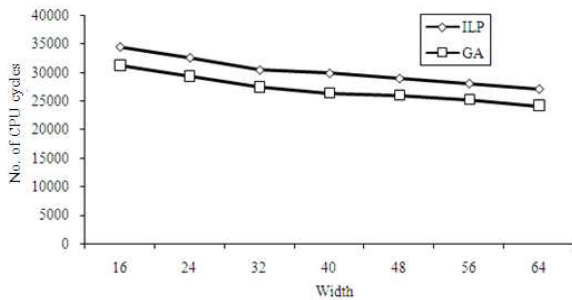

Fig. 3: Comparison of ILP with GA for SOC with three partitions

In the Table 2, the results of ILP and GA for SOC u226 is presented for the partition size of 16, 24, 32, 40, 48, 56 and 64 bits. The TAM is partitioned into 3 parts. The optimized scheduling of test vectors are obtained for the proposed GA-based method and the required amount of test time that is the number of CPU cycles are obtained and tabulated.

These values are also plotted for each partition against the number of CPU cycles in the Fig. 3. From the results and comparison graph, it is observed that the amount of CPU cycle required for the GA-based method is relatively less than the ILP-based method. Further, if the size of the TAM gets increased, the amount of time required for test application gets reduced. Another important result obtained from the Table 1 and 2 is, if the number of partition gets increased, the amount of test application time gets reduced.

In both the cases of GA based approach for SOC u226 with two partitions and GA based approach for SOC u226 with three partitions; the amount of test application time gets reduced.

DISCUSSION

Genetic Algorithms work by evolving a population of individuals over a number of generations. A fitness value is assigned to each individual in the population, where the fitness computation depends on the application.

Table 3: Average CPU Cycles for benchmark circuit with 2 partitions

	Average CPU cycles	
Circuit	ILP	GA
f2126	22786	18708
d695	25419	21699
q12710	27179	25084
h953	34394	32269
a586710	36613	33507
u226	36972	34830
d281	43770	39716
g1023	53274	49188
p34392	56482	52037
p22810	58643	54330
t512505	62452	56666
p93791	66433	61301

In the GA based test scheduling and TAM optimization, the initial population is randomly generated over a number of generations. The fitness function "improvement in the total test application time" is checked for each generation. The fitness function is not satisfied, the individuals are selected from the population for reproduction, crossed to generate new individuals and the new individuals are mutated to the population repeatedly until the fitness function is satisfied. During each generation of the Genetic Algorithm, the new individual may completely replace the old individuals in the population or new individual may be combined with the old individuals in the population. Since selection is biased toward more highly fit individuals, the average fitness of the population next. The fitness of the best individual is also chosen as a solution after several generations. The genetic algorithm uses two basic processes "inheritance" or the "passing features from one generation to the next" and "competition" or "survival of the fittest" which results in weeding out the bad features from individuals in the population. Due to these reasons, the GA based method produces improved results for the problems (P_A) and (P_{PA}).

The number of CPU cycles is obtained for the ITC-02 SOC Benchmark circuits given in[15] with the TAM width as 16, 24, 32, 40, 48, 56 and 56 bits and by dividing the TAM into 2 and 3 partitions. The average values of CPU cycles are obtained for GA based method and tabulated in the Table 3 and 4 for 12 ITC-SOC benchmark circuits for the TAM partition of 2 and 3 respectively along with the CPU cycles of ILP method. The comparison graph for the GA based method and ILP based method are shown in the Fig. 4 and 5 respectively. For all the circuits, the GA based method outperforms the ILP based method. The number of CPU cycle is relatively reduced for 3-partitions than 2-partitions of TAM. This is due to the faster and parallel transportation of test vector when the partition of TAM gets increased.

Table 4: Average CPU cycles for benchmark circuit with 3 partitions

	Average CPU cycles	
Circuit	ILP	GA
f2126	19581	15450
q12710	24508	21933
d695	25070	20069
u226	30227	27100
h953	30382	27426
a586710	34135	30273
d281	37134	33384
g1023	49457	45373
p34392	56050	50277
p22810	56600	51883
t512505	57616	52114
p93791	61357	53893

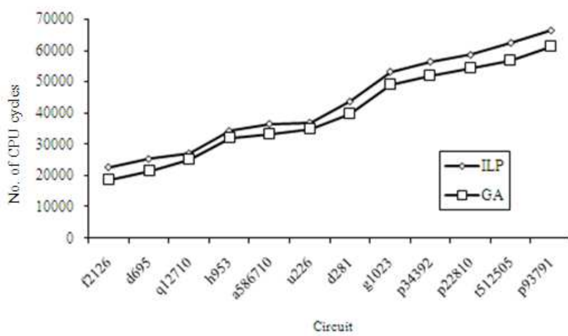

Fig. 4: Average CPU cycles for benchmark circuit with 2 partitions

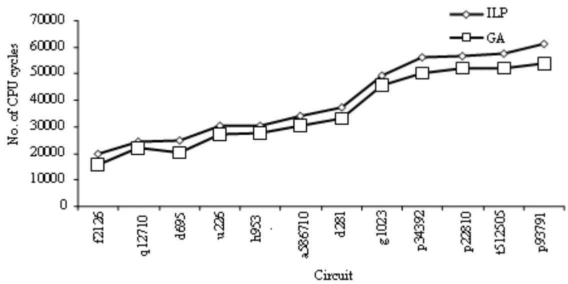

Fig. 5: Average CPU cycles for benchmark circuit with 3 partitions

When the numbers of partitions of TAM are increased, the possibility for parallel transportation of test vectors also increased and it naturally reduces the total test application time.

CONCLUSION

The investigation of the results show that the GA based approach produces the required partition of TAM width and vector assignment for the cores in SOC, such that the testing time is less than the ILP based approach. The experimental results are given for twelve ITC-02 SOC Benchmark circuits with two partitions and three partitions. The result gives good approximation compared to ILP within a few generations with acceptable processor times.

Further, the comparison of results of 12 ITC-02 SOC benchmarks circuits in Table 4 shows that the test application time for circuit increases with the complexity of the circuit in both the ILP and GA-based methods. The GA based-method takes less amount of test application time. This establishes the suitability of this problem to be solved by genetic algorithm. This technique can be applied to all the SOC benchmarks with more number of TAM widths and partitions. The results of proposed GA-based approach are found to be better than the results of the ILP methods available in the literature.

REFERENCES

1. Aho, A.V., J.E. Hopcroft and J.D. Ullman, 2004. The Design and Analysis of Computer Algorithms: Pearson Education. New Delhi, ISBN: 10: 0201000296.
2. Chandra, A. and K. Chakrabarty, 2003. A unified approach to reduce SOC test data volume, scan power and testing time. IEEE. Trans. Comput. Aided Des. Integrat. Circ. Syst., 22: 352-361. DOI: 10.1109/TCAD.2002.807895
3. Chandramouli, R. and S. Pateras, 1996. Testing systems on a chip. Proceedings of the IEEE Spectrum, Nov. 1996, IEEE Press Piscataway, New Jersey, USA., pp: 42-47. http://portal.acm.org/citation.cfm?id=277234
4. Crouch, A.L., 1999. Design-for-Test for Digital IC's and Embedded Core Systems. Prentice Hall PTR, New Jersey, ISBN: 0-13-084827-1.
5. Fabrizio, F., F. Franco, S. Donatella, M. Enrico and P. Massimo, 1997. Testing core-based systems: A symbolic methodology. Proceedings of the IEEE Design and Test of Computers, Oct.-Dec. 1997, IEEE Computer Society, USA., pp: 69-77. http://doi.ieeecomputersociety.org/0.110910.1109/54.632883
6. Gerez, S.H., 2004. Algorithms for VLSI Design Automation. John Wiley and Sons, New Delhi, ISBN: 81-265-0837-X, pp: 348.
7. Goldberg, D.E., 2003. Genetic Algorithms in Search, Optimization and Machine Learning. Pearson Education, New Delhi, ISBN: 10: 0201157675.

8. Gupta, R.K. and Y. Zorian, 1997. Introducing core-based system design. Proceeding of the IEEE Design and Test of Computers, Oct.-Dec. 1997, IEEE Computer Society, USA., pp: 15-25. http://doi.ieeecomputersociety.org/10.110910.1109/54.632877
9. Iyengar, V., K. Chakrabarty and E.J. Marinissen, 2003. Efficient test access mechanism optimization for system-on-chip. IEEE. Trans. Comput. Aided Des. Integrat. Circ. Syst., 22: 635-642. http://ieeexplore.ieee.org/iel5/43/26910/01196206.pdf?arnumber=1196206
10. Julien, P., E. Larsson, Z. Peng, M.L. Flottes and R. Bruno, 2003. An efficient approach to SOC wrapper design, TAM configuration and test scheduling. Proceedings of the IEEE European Test Workshop, May 25-28, Maastricht, The Netherlands, pp: 117-122. http://portal.acm.org/citation.cfm?id=943229
11. Koranne, S., 2004. A note on system-on-chip test scheduling formulation. J. Elect. Test Theor. Applied, 20: 309-313. http://portal.acm.org/citation.cfm?id=993054
12. Larsson, A., E. Larsson, E. Petru and Z. Peng, 2005. SOC Test scheduling with test set sharing and broadcasting. Proceedings of the IEEE Asian Test Symposium, Dec. 18-21, IEEE Xplore Press, Kolkata, pp: 162-167. http://ieeexplore.ieee.org/iel5/10525/33306/01575424.pdf?isnumber=33306
13. Larsson, E. and Z. Peng, 2002. An integrated framework for the design and optimization of SOC test solutions. J. Elect. Test Theor. Applied, 18: 385-400. http://portal.acm.org/citation.cfm?id=608939
14. Larsson, E., K. Arvidsson, H. Fujiwara and Z. Peng, 2004. Efficient test solutions for core-based designs. IEEE. Trans. Comput. Aided Des. Integrat. Circ. Syst., 23: 758-774. DOI: 10.1109/TCAD.2004.826560
15. Marinissen, E.J., V. Iyengar and K. Chakrabarthy, 2002. A set of benchmarks for modular testing of SOCs. Proceedings of the IEEE International Test Conference, (ITC'02), IEEE Xplore Press, Baltimore, pp: 519-528. http://ieeexplore.ieee.org/iel5/8073/22329/01041802.pdf
16. Mazumder, P. and E.M. Rudnick, 2003. Genetic Algorithms for VLSI Design, Layout and Test Automation. Pearson Education, New Delhi, ISBN: 10: 0-13-011566-5.
17. Merrill, H. and J.A. Rowson, 1996. Blocking in a system on a chip. IEEE. Spectrum, 33: 35-41. DOI: 10.1109/6.542273
18. Ravi, S., L. Ganesh and N.K. Jha, 2001. Testing of core-based systems-on-a-chip. IEEE. Trans. Comput. Aided Des. Integrat. Circ. Syst., 20: 426-439. http://ieeexplore.ieee.org/iel5/43/19726/00913760.pdf
19. Sehgal, A., V. Iyengar and K. Chakrabarty, 2004. SOC test planning using virtual test access architectures. IEEE. Trans. Very Large Scale Integrat. Syst., 12: 1263-1275. DOI: 10.1109/TVLSI.2004.834228