Design and Optimization of Test Solutions for Core-based System-On-Chip Benchmark Circuits Using Genetic Algorithm

P. Sakthivel and P. Narayanasamy

Abstract - **The increased usage of embedded pre-designed reusable cores necessitates a core-based test strategy in which cores are tested as separate entities. Test application time is a major issue in System-on-Chip Testing (SOC). Pre-designed cores and reusable modules are popularly used in the design of large and complex systems. As the complexity of system increases, the test application time also significantly increases. Available techniques for testing of core-based SOC do not provide a systematic means of compact test solutions. The test application time must be minimized to transport test data to and from the cores. In this paper, we present a Genetic Algorithm (GA)-based approach to optimize the test vectors for globally asynchronous locally synchronous SOC Benchmark Circuits. This approach provides optimal results comparable to other methods of similar problems. Based on our experiments, the test results for four ITC-02 SOC Test Benchmark circuits are presented. The results of GA-based approach are shown to be superior to the heuristic approaches proposed in the literature.**

Index Terms - **Benchmark Circuit, Core-based design, Genetic Algorithm, System-On-Chip, Test Access Mechanism.**

I. Introduction

Asynchronous design offers a solution to the interconnect problems faced by system-on-chip designers [1], but their adoption has been held back by a lack of methodology and support for post-fabrication testing. Interest in asynchronous circuit design is increasing due to its promise of efficient design [3]. The quiescent nature of asynchronous circuits allows them to remain in a stable state until necessary wire transitions trigger an event to occur [29].

Very Large Scale Integrated (VLSI) circuits designed using modern Computer Aided Design (CAD) tools are becoming faster and larger, incorporating millions of smaller transistors on a chip [2].

Manuscript received March 20, 2007.

P. Sakthivel is with Ramanujan Computing Centre, Anna University, Chennai – 600 025, INDIA (Phone: 91 44 22203133; Fax: 91 44 22201160; E-Mail: psv@annauniv.edu)

P. Narayanasamy is with Department of Computer Science and Engineering, Anna University, Chennai – 600 025, INDIA.

VLSI designs can be divided into two major classes: Synchronous and Asynchronous circuits. Synchronous circuits use global clock signals that are distributed throughout their sub-circuits to ensure correct timing and to synchronize their data processing mechanisms [28, 30,]. Asynchronous circuits contain no global clocks. Their operation is controlled by locally generated signals [26]. Asynchronous circuits [22] have many potential advantages over their synchronous equivalents including lower latency, low power consumption, and lower electromagnetic interference [22]. However, their acceptance into industry has been slow, which may be due to a number of reasons.

Recent advances in Integrated Circuit (IC) design methods and manufacturing technologies have led to the integration of a complete system onto a single IC, called system-on-chip (SOC). These system chips offer advantages such as higher performances, lower power consumption, and decreased size and weight, when compared to their traditional multichip equivalents. SOC technology is the packaging of all the necessary electronic circuits and parts for a System on a single integrated circuit generally known as a Microchip [10]. SOC technology is used in small, increasingly complex consumer electronic devices. Some such devices have more processing power and memory than a typical computer. Many such chips are designed by embedding large reusable building blocks, commonly called cores. Such a design reuse approach speeds up the design process, and allows import of external design expertise. The design of asynchronous circuits has been attracting more interest recently, as clock distribution on a large die becomes increasingly difficult.

The ITRS road map [22] predicts that as a solution to the clock distribution problem, Globally Asynchronous Locally Synchronous (GALS) system will become mainstream in the near future. In a GALS system, a number of synchronous islands of logic communicate asynchronously using a suitable interconnect. Unfortunately, the testability of asynchronous systems is considered to be one of their major drawbacks.

The increased system complexity leads to high-test data volumes, which means long testing times. The testing of SOC is a crucial and time-consuming problem [7] due to the increasing design complexity. Therefore it is important to provide the test designer with support to develop an efficient test solution. Testing SOC becomes an increasing challenge [10] as these devices become more complex. An SOC design is typically built block by block. Efficient testing is also best done by block by block. Recently, pre-designed cores are also used in the SOCs [12]. Traditionally, the chips are tested before they are integrated into a system; the interconnections are tested separately in system test when fault-free chips are already integrated. For system chips, on the other hand, testing of cores and interconnections is performed in a single system test step. Test access becomes also a problem for system chips since the cores are not directly accessible via chip inputs and outputs. Testing individual circuits, individual blocks and individual cores has established technologies. But, available techniques for testing of core-based SOC do not provide a systematic means of synthesizing low overhead test architectures and compact test solutions [14].

Embedded cores such as processors, custom application specific integrated circuits, and memories are increasingly being used to provide SOC solutions to complex integrated circuit design problems [16]. The advances in design methodologies and semiconductor process technologies have led to the development of systems with excessive functionality implemented on a single chip [14].

In a core based design approach, a set of cores (predefined and pre-verified design modules) is integrated into a system using user defined logic and interconnections. In this way, complex systems can be efficiently developed [18]. However, the complexity in the system leads to high-test data volumes and design and optimization of test solution are a must for any test. Hence the following independent problems [14] might be considered:

How to design an infrastructure for the transportation of test data in the system.

How to design a test schedule to minimize test time, considering test conflicts and power constraints.

The testable units in an SOC design are the cores, the User Defined Logic (UDL) and the interconnections. The cores are 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 Test Access Mechanism (TAM) Design. The workflow when developing an SOC test solution can mainly be divided into two consecutive parts: an early design space exploration followed by 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 must be controlled. Otherwise the system may be damaged during test. Furthermore, test resources such as external testers support a limited number of scan-chains and have a limited test memory, which also introduce constraints [11].

Research has been going on in developing techniques for test scheduling, test access mechanism design and testability analysis. In this paper, we propose a new technique using Genetic Algorithm for optimizing the test vector for Globally Asynchronous Locally Synchronous SOC with the objective to minimize the test application time. The aim of our 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 test access mechanism at a relatively low computational cost.

The rest of the paper is organized as follows: The works related to our approach and various issues related to SOC testing and test scheduling techniques are discussed in section 2; Test vector optimization and test scheduling framework based on genetic algorithm is presented in section 3; the characteristics of four ITC-02 SOC benchmark circuits are given in section 4; experimental results for the four benchmark SOC circuits are presented in section 5 and section 6 looks at the conclusion.

II. SOC Test Scheduling Techniques and the Related Works

The basic problem in test scheduling 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 [7]. 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 [13].

Zorian et al [5] proposed a test scheduling technique for fully BISTed systems where test time is minimized while power constraint is considered. 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 [17, 25], each core has its dedicated test source and test sink; and there might not be any conflicts among tests [9]. However, in general, conflicts among tests may occur.

Peng et al [7] proposed a test scheduling technique where test time is minimized for systems with test conflicts. For Core Based Systems a test scheduling technique is proposed by Chakraborthy et al [19]. Choudhary et al [11] proposed an analytic test scheduling technique where test conflicts and power constraints are considered. Iyengar and Chakrabarty [6] proposed a pre-emptive test scheduling technique where the test for a testable unit may be interrupted and resumed later.

The test-application time can be minimized by scheduling the execution of the test sets as concurrently as possible [2]. The basic idea in test scheduling is to determine when each test set should be executed and the main objective is to minimize the test application time.

The scheduling techniques can be classified into the following scheme [4]:

No partitioned testing

Partitioned testing with run to completion, and
Partitioned testing.

A. *SOC Testing*

Integration of a complex system that until recently consisted of multiple Integrated Circuits onto a single Integrated Circuit is known as System-on- Chip [6]. The shrinking of silicon technology leads to an increase in the number of transistors on a chip. This increases the number of faults and test vectors that in turn leads to a serious increase in test time. Test time reduction is one of the research challenges [8, 21] in the SOC design paradigm. The most important issues in the SOC Testing are as follows [10]:

Controlling the whole process of SOC Testing.
Testing the User Defined Logic and Interconnections.
Testing cores with different functionalities coming from different vendors.
Accessing cores from the system's primary inputs and primary outputs.

B. *Test Access Mechanism*

The test access mechanism (TAM) takes care of chip test pattern transport. 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 chip [12, 23].

The following are the four problems structured in order of increasing complexity [4]:

P_W: Design a wrapper for a given core, such that the core testing time is minimized, and the TAM width required for the core is minimized.
P_{AW}: Determine (i) an assignment of cores to TAMs of given widths, and (ii) a wrapper design for each core such that SOC testing time is minimized.
P_{PAW}: Determine (i) a partition of the total TAM width among the given number of TAMs, (ii) an assignment of cores to TAMs of given widths, and (iii) a wrapper design for each core such that SOC testing time is minimized.
P_{NPAW}: Determine (i) the number of TAMs for the SOC, (ii) a partition of the total TAM width among the given number of TAMs, (iii) an assignment of cores to TAMs of given widths, and (iv) a wrapper design for each core such that SOC testing time is minimized.

The above problems are all NP – Hard problems. Therefore, efficient heuristics and other techniques are needed for large problem instances [14]. In this work, we are presenting Genetic Algorithm based approach [16] to effectively solve the problems namely P_{AW} and P_{PAW}.

III. TEST VECTOR OPTIMIZATION BASED ON GENETIC ALGORITHM

Genetic Algorithms can effectively be used to solve the search and optimization problems [27]. In this section the genetic algorithm that is used for generating test sequences for SOC is described [2, 4, 6, 8, 18]. First, the basic idea of the method is given. Then we present the representation of test conditions and the objective function and provide some insight into the parameter settings of the genetic algorithm [15, 20]. 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 by applying genetic operators. Given old generation, new generation is built from it, according to the following operation given in section 3.1, 3.2 and 3.3 [22].

A. *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.

B. *Crossover*

This operator generates two new chromosomes from the couple of selected chromosomes. A random point on the chromosome also known as cross-site is selected. Portions of individuals in a couple are exchanged between themselves to form new chromosomes as follows [24]:

```
for I = 1 to number of entries in the chromosome

    C1(I)   = P1(I) if I <= cross-site
            = P2(I) if I > cross-site

    C2(I)   = P2(I) if I <= cross-site
            = P1(I) if I > cross-site
```

(1) 1-Point Crossover

Construct two new elements C1 and C2 from two parents P1 and P2, where P1 and P2 are split in two parts at a cut position.

(2) 2-Point Crossover

Construct two new elements C1 and C2 from two parents P1 and P2, where P1 and P2 are split in three parts.

(3) Uniform Crossover

Construct two new elements C1 and C2 from two parents P1 and P2, where for each position the value is taken with a certain probability from P1 and P2 respectively.

(4) Free Vertical Crossover

Construct two new elements C1 and C2 from two parents P1 and P2. Determine for each test vector T in two parts at its cut position. The first part of each test vector of C1 is taken from P1 and the second part is taken from P2.

C. Mutation

Construct one new element C from a parent P by copying the whole element and changing a value at a randomly chosen position.

(1) 2-Point Mutation

Perform Mutation two times on the same element.

(2) Mutation with neighbor

Perform Mutation at two adjacent positions on the same element.

D. Overview of the proposed method

The different steps of the proposed method are given as follows [2]:

```
Genetic Algorithm Begin
Generate the initial population of chromosomes, randomly.
Sort the initial population in ascending order of the cost.
While there is no improvement in cost function.
  Do Begin
    Select first 20% chromosome as best class.
    Generate 40% chromosomes using crossover operator.
    Generate 40% chromosomes using mutation o operator
    Sort this generation in ascending order of the cost.
  Do End
Genetic Algorithm End.
```

IV. Characteristics of the SOCs

Table I lists the general characteristics of the SOCs considered in our experiment [6]. This table is organized as follows: Column 1 gives the names of the SOCs. In column 2, the number of modules is listed. Column 3 lists the number of design hierarchy levels. Column 4 lists the total number of I/O terminals in the SOC. Column 5 shows the total number of scan flip flops in the SOC. Column 6 lists the total sum of the test pattern counts of all tests.

Table I: General Characteristics of the ITC-02 SOC Test Benchmark Circuits

Name of SOC	No. of Module s	No. of Levels	No. of I/Os	No. of SFFs	No. of Test Patterns
P34392	20	3	2057	20948	66349
P22810	29	3	4283	24723	24890
P93731	33	3	6943	89973	22987
D695	11	2	1845	6384	881

Table II lists the main characteristics of the module tests of the SOCs [6]. Column 1 again lists the names of the SOCs. In column 2, the summed number of tests over all modules is shown. In columns 3, 4 and 5, the minimum, average and maximum number of test patterns per test is listed.

Table II: Test Characteristics of the ITC-02 SOC Test Benchmark Circuits

Name of SOC	No. of Tests	No. of Pattern Counts		
		Min	Ave	Max
P34392	21	11	3159	12336
P22810	30	1	830	12324
P93731	32	11	718	6127
d695	10	12	88	234

V. Experimental Results

Our experiments were conducted for the ITC-02 SOC benchmark circuits p34392, p22810, p93791 and d695 [6]. The results are obtained for SOC with two partitions. W is the width of Test Access Mechanism. w1 and w2 are the size of the partition 1 and partition 2. The processor time and test time are given under the Heuristics taken from [4]. The processor time and test time given under Genetic Algorithm are the results of our experiment.

A. Results for p34392

The Table III gives the result for SOC p34392 with two partitions. In the Genetic Algorithm approach, the maximum processor time taken is 1.34 seconds and the minimum processor time taken is 0.82 seconds whereas the maximum and minimum processor time taken in Heuristic method is 1 second.

Table III: Results for SOC p34392 with two partitions

W	W1 + W2	(Processor Time(*Seconds*))/ Test Time(*Cycles*)	
		Heuristics	Genetic Algorithm
16	8 + 8	(1)/ 1080940	(1.34)/ 1080900
24	15 + 9	(1)/ 928782	(1.25)/ 928562
32	21 + 11	(1)/ 750490	(1.10)/ 749850
40	24 + 16	(1)/ 721566	(0.97)/ 721450
48	31 + 17	(1)/ 709262	(0.85)/ 708550
56	38 + 18	(1)/ 704659	(0.82)/ 704650
64	18 + 46	(1)/ 700939	(0.88)/ 700800

B. *Results for p22810*

The Table IV gives the result for SOC p22810 with two partitions. In the Genetic Algorithm approach, the maximum processor time taken is 1.32 seconds and the minimum processor time taken is 0.41 seconds whereas the maximum processor time taken is 9 seconds and minimum processor time taken is 1 second in Heuristic method.

Table IV: Results for SOC p22810 with two partitions

W	W1 + W2	(Processor Time(*Seconds*))/ Test Time(*Cycles*)	
		Heuristics	Genetic Algorithm
16	6 + 10	(1)/ 462210	(0.63)/ 461348
24	8 + 16	(2)/ 365947	(1.15)/ 361326
32	10 + 22	(9)/ 312659	(0.41)/ 313891
40	9 + 31	(2)/ 290644	(0.61)/ 287260
48	10 + 38	(1)/ 290644	(1.32)/ 268512
56	24 + 32	(1)/ 290644	(0.51)/ 271429
64	12 + 52	(6)/ 271330	(0.48)/ 260645

C. *Results for p93791*

The Table V gives the result for SOC p93791 with two partitions. In the Genetic Algorithm approach, the maximum processor time taken is 1.46 seconds and the minimum processor time taken is 0.36 seconds whereas the maximum and minimum processor time taken in Heuristic method is 1 second.

Table V: Results for SOC p93791 with two partitions

W	w1 + w2	(Processor Time(*Seconds*))/ Test Time(*Cycles*)	
		Heuristics	Genetic Algorithm
16	8 + 8	(1)/ 1952800	(0.48)/ 1786032
24	12 + 12	(1)/ 1217980	(0.67)/ 1206532
32	9 + 23	(1)/ 894342	(1.36)/ 878971
40	23 + 17	(1)/ 750311	(0.83)/ 735677
48	33 + 15	(1)/ 632474	(0.36)/ 628733
56	46 + 10	(1)/ 524203	(1.46)/ 518686
64	46 + 18	(1)/ 467424	(0.61)/ 462757

D. *Results for d695*

The Table VI gives the result for SOC d695 with two partitions. In the Genetic Algorithm approach, the maximum processor time taken is 0.29 seconds and the minimum processor time taken is 0.12 seconds whereas the maximum and minimum processor time taken in Heuristic method is 1 second.

Table VI: Results for SOC d695 with two partitions

W	W1 + w2	(Processor Time(*Seconds*))/ Test Time(*Cycles*)	
		Heuristics	Genetic Algorithm
16	7 + 9	(1)/ 45055	(0.18)/ 1080900
24	19 + 5	(1)/ 34455	(0.19)/ 928562
32	20 + 12	(1)/ 25828	(0.21)/ 749850
40	8 + 32	(1)/ 22848	(0.12)/ 721450
48	16 + 32	(1)/ 22804	(0.22)/ 708550
56	19 + 37	(1)/ 18940	(0.29)/ 704650
64	41 + 23	(1)/ 18868	(0.16)/ 700800

VI. CONCLUSION

The experimental results are given for four ITC-02 SOC

Benchmark circuits with two partitions. The result gives good approximation compared to Heuristics within a few generations with acceptable processor times. This establishes the suitability of this problem to be solved by Genetic Algorithm. We can apply this technique to all the other SOCs given in [6] having more number of cores with many scan chains and even more number of TAM widths.

References

[1] Aristides Efthymiou, John Bainbridge and Douglas Edwards, Test Pattern Generation and Partial-Scan Methodology for an Asynchronous SoC Interconnect, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* Vol. 13, No. 12, pp. 1384 – 1393.

[2] S. Chattopadhyay and K. Sudharsana Reddy, Genetic Algorithm Based Test Scheduling and Test Access Mechanism Design for System-On-Chips, *Proceedings of the International Conference on VLSI Design*, 2003.

[3] N. Gupta and D.A. Edwards, Synthesis of Asynchronous Circuits Using Early Data Validity, Proceedings *of the International Conference on VLSI Design,* 2005.

[4] Vikram Iyengar, K. Chakrabarthy and Erik. J. Marinissen, Efficient Wrapper/TAM Co-Optimization for Large SOCs, *http://www.ee.duke.edu/~krish/237_iyengar_v.pdf*

[5] Erik Jan Marinissen, Rohit Kapur, Maurice Lousberg, Teresa McLaurin, Mike Ricchetti, and Yervant Zorian, On IEEE P1500's Standard for Embedded Core Test, *Journal of Electronic Testing: Theory and Applications,* Vol. 18, 2002, pp. 365-383.

[6] Erik Jan Marinissen, V. Iyengar and K. Chakrabarthy, A Set of Benchmarks for Modular Testing of SOCs, *http://www.extra.research.philips.com/itc02socbenchm*

[7] Erik Larsson and Zebo Peng, An Integrated Framework for the Design and Optimization of SOC Test Solutions, *Journal of Electronic Testing: Theory and Applications,* Vol. 18, 2002, pp. 385 – 400.

[8] Martin Keim, Nicole Drechsler, Rolf Drechsler and Brend Becker, Combining GAs and Sysmbolic Methods for High Quality Tests of Sequential Circuits, *Journal of Electronic Testing: Theory and Applications*, Vol. 17, 2001, pp. 37-51.

[9] Sandeep Koranne, A Note on System-on-Chip Test Scheduling Formulation, *Journal of Electronic Testing: Theory and Applications*, Vol. 20, 2004, pp. 309 – 313.

[10] Srivaths Ravi, Ganesh Lakshminarayana and Niraj. K. Jha, Testing of Core-Based Systems-on-a-Chip, *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, Vol. 20, No. 3, 2001, pp. 426-439.

[11] Santanu Chattopadhyay and Naveen Choudhary, Genetic Algorithm Based Approach for Low Power Combinational Circuit Testing, *Proceedings of the IEEE International Conference on VLSI Design,* 2003.

[12] Erik Larsson, Klas Arvidsson, Hideo Fujiwara and Zebo Peng, Efficient Test Solutions for Core-Based Designs, *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, Vo. 23, No. 5, 2004, pp. 758-774.

[13] Vivekananda M. Vedula and Jacob A. Abraham, Jayanta Bhadra and Raghuram Tupuri, A Hierarchical Test Generation Approach Using Program Slicing Techniques on Hardware Description Languages, *Journal of Electronic Testing: Theory and Applications,* Vol. 19, 2003, pp. 149-160.

[14] Anuja Sehgal, V. Iyengar and K. Chakrabarthy, SOC Test Planning Using Virtual Test Access Architectures, *IEEE Transactions on Very Large Scale Integration Systems,* Vo. 12, No. 12, 2004, pp. 1263 – 1276.

[15] Indradeep Ghosh and Masahiro Fujita, Automatic Test Pattern Generation for Functional Register-Transfer Level Circuits Using Assignment Decision Diagrams, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* Vol. 20, No. 3, 2001, pp. 402 – 415.

[16] Yuejian Wu and Paul MacDonald, Testing ASIC with Multiple Identical Cores, *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, Vo. 22, No. 3, 2003, pp. 327 – 336.

[17] Han Bin Kim and Dong Sam Ha, A High-Level BIST Synthesis Method Based on a Region-wise Heuristic for an Integer Linear Programming, Proceedings of the *International Test Conference,* pp. 903 – 912, 1999.

[18] V. Iyengar, A. Chandra, S. Schweizer and K. Chakrabarthy, A Unified Approach for SOC Testing Using Test Data Compression and TAM Optimization, *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition*, 2003.

[19] Anshuman Chandra and Krishnendu Chakrabarty, A Unified Approach to Reduce SOC Test Data Volume, Scan Power and Testing Time, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 22, No. 3, 2003, pp. 352 – 362.

[20] Cheng-Wen Wu, SOC Testing Methodology and Practice, *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition*, 2005.

[21] Paulo F. Flores, Horacio C. Neto and Joao P. Marques Silva, An Exact Solution to the Minimum Size Test Pattern Problem, *ACM Transactions on Design Automation of Electronic Systems*, Vol. 6, No. 4, 2001, pp. 629-644.

[22] Aristides Efthymiou, John Bainbridge and Douglas A. Edwards, Adding Testability to an Asynchronous Interconnect for GALS SOC, *Proceedings of the 13th Asian Test Symposium*, 2004.

[23] Irith Pomeranz and Sudhakar M. Reddy, Functional Test Generation for Delay Faults in Combinational Circuits, *ACM Transactions on Design Automation of Electronic Systems,* Vol. 3, No. 2, 1998, pp. 231 – 248.

[24] Hemangee K. Kapoor and Mark B. Josephs, Modelling and Verification of delay-insensitive circuits using CCS and the Concurrency Workbench, *Information Processing Letters,* Vol. 89, 2004, pp.293-296.

[25] Kwang-Ting Cheng, Gate-Level Test Generation for Sequential Circuits (Tutorial and Survey Paper), ACM *Transactions on Design Automation of Electronic Systems,* Vol. 1, No. 4, 1996, pp. 405 – 442.

[26] Eunjung OH, Soo-Hyun KIM, Dong-Ik LEE and Ho-Yong CHOI, High Level Test Generation for Asynchronous Circuits from Signal Transition Graph, *IEICE Transactions on Fundamentals*, Vo.E85-A, No. 12, 2002, pp. 2674 – 2683.

[27] P. Sakthivel, D. Sridharan and P. Narayanasamy, ATPG Algorithm for Synchronous Sequential Circuits Based on HGA and Simulation, *Proceedings of the International Conference on VLSI,* pp. 59 – 65, 2002.

[28] Yin-He SU, Ching-Hwa CHENG and Shih-Chieh CHANG, Novel Techniques for Improving Testability Analysis, *IEICE Transactions on Fundamentals*, Vol. E85-A, No. 12, 2002, pp. 2901-2912.

[29] P. Sakthivel, D. Sridharan and P. Narayanasamy, Testing of Embedded Core Based System on a Chip: Challenges and Directions, *Proceedings of the National Seminar on Global Prosperity Through Information Technology,* pp. 97 – 101, 2002.

[30] Mathew Sacker, Andrew D. Brown, Andrew J. Rushton and Peter R. Wilson, A Behavioral Synthesis System for Asynchronous Circuits, *IEEE Transactions on Very Large Scale Integration Systems*, Vol.12, No.9, 2004, pp. 978 – 994.