# A New Design-for-Test Technique For Reducing SOC Test Time

C.V.Guru Rao

Department of Computer Science and Engg
Indian Institute of Technology
Kharagpur,WB 721302
Tel: +91-08712-579276
Fax: +91-08712-540263
e-mail: guru_cv_rao@hotmail.com

D.Roy Chowdhury

Department of Computer Science and Engg
Indian Institute of Technology
Kharagpur,WB 721302
Tel: +91-03222-3490
Fax: +91-03222-55303
e-mail: drc@cse.iitkgp.ernet.in

***Abstract*— This paper introduces a new design-for-test(DFT) technique for system-on-chip(SOC) designs. It aims to provide the test designer with details of test scheduling, test access mechanism (TAM) design and an integrated test strategy in order to implement an efficient test solution. Post-synthesis simulations are carried out on the net lists of ISCAS'89 benchmark SOCs to prove the allegiance of the proposed algorithm and to realize the DFT. Experiments resulted in a significant reduction of the test time.**

## I. Introduction

THE popularity of core-based system-on-chip(SOC) design is increasing among design teams all over the world. This new design strategy involves pre-designed and pre-verified modules or cores. Designing a multi million gate system using these cores is the most effective way to reduce the design time and cost. Most importantly, the use of embedded cores shortens the time-to-market for these new systems due to design reuse. One major difference with the conventional design is that a designer needs to embed cores that are not manufactured and hence are untested.

Though use of this approach seems inevitable, there are a number of significant challenges affecting implementation. The major one at this time is testing of these complex chips. Solutions need new types of test architectures that are able to manage the testing of up to a 100 million transistors at a higher speeds with lower hardware.

The web site [1] of IEEE Working group on P1500 Standard Embedded Core Based Testing (SECT) provides the necessary background of the system-chip test challenges. Zorian et al.[2], [3], have provided a survey of the various TAM implementations and a conceptual architecture with a source and sink. Recently several new TAMs have been proposed and studied such as Test Bus [4], TestRAIL [4], and Addressable Test Ports [5].

In the area of testing embedded core-based systems, most of the work [6], [7] concentrates on viewing test scheduling as a task after TAM is fixed. Most recently, many new DFT (Design-For-Test) techniques [8], [9], [10], [11], [12] have been exploited to address this problem. But none of the above have dealt with implementation.

The current work is based on a graph-theoretic approach to accomplish parallel testing of cores in the SOC under test. The test strategy algorithm is suitable for providing the I/O access and scheduling the tests on the cores available in the SOC. In this paper we present the implementation of our DFT technique with the help of TAM switch.

The rest of the paper is organized as follows. Section II describes the algorithms proposed to carry out the work. In section III, the design and implementation is described. The experimental results are tabulated and discussed in section IV, followed by conclusions on the work carried out in section V and then references made for this work.

## II. Proposed Work

The proposed work is to design and implement a new DFT technique for SOC using TAM switch. The objective is to provide the test designer with a choice between the test hardware and test time in order to implement an efficient complete test solution for the SOCs. For this purpose a test strategy algorithm was developed.

### A. Development of Test Strategy

The test strategy algorithm has been developed for scheduling the tests on various cores in the SOC. The algorithm assumes that every core is to be scan tested, there is no BIST enabled core, several TAM switches that are on different TestRAILS(due to partition of test input pins into smaller rails) may be used for testing cores in parallel, all the cores have a common clock, TAM switch would work for a typical wire load conditions when it simulated or synthesized.

*1) Description of Clustering Algorithm:* A graph for SOC is created considering the connections between the cores. Initially each core is alloted to one cluster and then the deficiency is calculated. A **Cluster** is defined as the combination of cores which can be tested at the same time depending on the number of scan chains and the interconnection between them. The **Deficiency**($d(i)$) of a cluster $C_i$ is defined as difference between highest available test bandwidth of test bandwidth divisions and $\sum$ number of scan chains of core j in the cluster i. All the clusters with positive deficiency are considered eligible for clustering. In order to reduce the test time, the core with least time is chosen from among the neighboring cores which are resulting in the same deficiency. The clustering process continues

until no further clusters can be formed and the deficiency of all the clusters is equal to zero (e.g, for SOC-4 the deficiency calculated for core-0 is $2 - 1 = 1$, core-1 to core-4 is $2 - 2 = 0$, hence five clusters are configured). Thus created clusters are then assigned to appropriate test bandwidth division. The **Test Bandwidth Division** is the partition of the whole test inputs pins and test output pins of the SOC.

*Algorithm 1:* **Clustering**

```
for all cores in SOC
{
 create a graph
 for ascending order of scan chains
 assign bandwidth to all cores
 calculate the deficiency
 while(deficiency > 0)
 {
   for i = 1 to all cores)
   {
       if(scan chains(i) < bandwidth division)
         cluster core i with its neighbor j
         which has min(cores-in(j)) \i^th core in
         cluster j is cores-in(j)\
         else shift the cluster in to other bandwidth
       update-graph
       do final configuration of cluster
   }
 }
}
```

*2) Description of Test Scheduling Algorithm:* The test scheduling algorithm considers all possible test bandwidth divisions such that the total number of test bandwidth divisions at any time is less than or equal to the number of cores. For a specific test bandwidth division, the algorithm considers clustering of cores based on certain rules after initial assignment of cores and TAM switches to test bandwidth. In the initial assignment the number of clusters is equal to the number of cores and each cluster is assigned TAM switch and in turn they are assigned to some test bandwidth. For a particular test bandwidth division, the algorithm would assign bandwidth based on the following 3 cases:

- ◊ If the number of scan chains of a core is equal to the bandwidth division then the core is serviced by that partition.
- ◊ If the number of scan chains of a core is more than the highest bandwidth division possible then the core is initially alloted to the highest test bandwidth division.
- ◊ For all other cases, assign the core to the division that has scan chains nearest(less) with respect to the bandwidth.

Depending on the results of clustering the stitching is taken up. The **stitching** is a process of reducing the number of scan chains for a core so that a switch is able to feed the core with the available test bandwidth. In this process the length of each of the resulting scan chain should be lower than that of maximum length of all scan chains of that core. This step is repeated so as to lessen the number of scan chains as much as possible.

Now these scan chains are partitioned uniformly in the three different alternative test strategies, so as to give rise the number and size of the TAM switches differently for a SOC under Test. **Partitioning** is defined as dividing the scan chains in to groups that are able to be fed to the test rails. The number of partitions is computed with division of the total number of scan chains in that core by the available maximum test bandwidth among all the test bandwidth divisions. This step is used evolve choices by varying the number of partitions leading to different strategies that are possible.The **Objective function** is defined as product of total hardware function and total test time of the SOC. The **Total hardware function** is defined as function of the number and sizes of the TAM switches.

The procedure to compute number of partitions in each of the strategy is explained below:

- ◊ The number of partitions is computed with division of the total number of scan chains of the core by the maximum test bandwidth among the divisions and finding the ceiling value.
- ◊ The TAM switches existing in the same test bandwidth division are combined after partitioning so that their number is reduced their sizes are accordingly computed.
- ◊ Compared to the above two strategies here the partition size has been doubled, thus the number of TAM switches is reduced considerably but at the cost of complexity of each TAM switch which adds up with respect to it's size.

Finally the algorithm for all possible bandwidth divisions the number and sizes of the switches, the total hardware function, minimum objective function are computed. Then configure function is called which generates the details of configuration information of TAM-to-Core connections and their allotments. It also generates the configuration bits required for the TAM switches. Now the generate_rtl_switch function is called to generate the high-level verilog code for the integrated TAM switch. In the C program developed here the pre-computed parameters of switch such as number and size of the switch are passed to a generic verilog design of the switch which is written into a file. Next, reformat function is called. Which takes the pre-computed test vectors as input from a file and rearranges them in accordance with the test bandwidth so that the test vector sets prepared are ready to be fed through the switches to cores depending on the sequence described by the DAG. Create_dag is a function which determines the sequence of testing of the cores in the SOC.

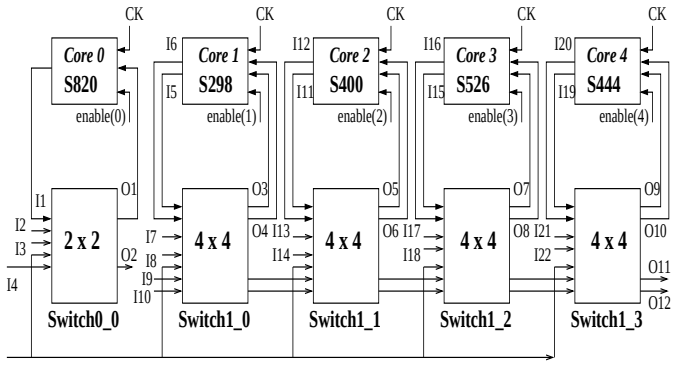
*Algorithm 2:* **Test Scheduling**

```
for all bandwidth divisions
{
  do initial allotment of cores
  while ( deficiency > 0 )
  {
    create cluster
    assigns cluster to a bandwidth
    expand cluster to include neighbors
    calculate deficiency for expansion
      if(deficiency < 0)
      {
        calculate deficiency for new cluster
        break
      }
  }
  while(deficiency < 0)
  {
```

```
    stitching for cores with more scan chains
    calculate the number of partitions for that core
    bandwidth assignment for the partitions
  }
  for all bandwidth divisions
  {
    select and place clusters in
    appropriate bandwidth divisions
  }
  for all clusters
  {
    compute total number of TAM switches
    compute the size of TAM switches
    compute total hardware function
    compute minimum objective function
    compute total testing time
    configure()
    generate_rtl_switch()
    reformat()
    create_dag()
  }
}
```

## III. Design and Implementation of Design-for-Test

The proposed test strategy algorithm is executed on the benchmark SOC and an integrated TAM switch is generated in the process. This TAM switch consists of several optimized switches as required to test all the cores available in the SOC. To illustrate the design and implementation, System-on-Chip 4 in is considered as an example.




| | | | | | |
|---|---|---|---|---|---|
| I1 = tam_in[0] | O1 = tam_out[0] | I7 = Sin[1] | O7 = tam_out[5] | I13 = Sin[2] | I19 = tam_in[7] |
| I2 = Sin[0] | O2 = test_out[0] | I8 = Ctrl_in[1] | O8 = tam_out[6] | I14 = Ctrl_in[2] | I20 = tam_in[8] |
| I3 = Ctrl_in[0] | O3 = tam_out[1] | I9 = test_in[1] | O9 = tam_out[7] | I15 = tam_in[5] | I21 = Sin[4] |
| I4 = test_in[0] | O4 = tam_out[2] | I10 = test_in[2] | O10 = tam_out[8] | I16 = tam_in[6] | I22 = Ctrl_in[4] |
| I5 = tam_in[1] | O5 = tam_out[3] | I11 = tam_in[3] | O11 = test_out[1] | I17 = Sin[3] | |
| I6 = tam_in[2] | O6 = tam_out[4] | I12 = tam_in[4] | O12 = test_out[2] | I18 = Ctrl_in[3] | |

Fig. 1. System Integration for Testing of SOC-4

A High-level(verilog) top module is designed with the net lists of ISCAS'89 benchmark cores corresponding to this SOC4 together with the net lists of integrated TAM switch top module for integrating the system to test. A set of wires are used to connect the cores and TAM switches to complete the connections of the proposed test architecture. Since all the test data is available on hand the hardware is simulatable. Hence it is implemented very effectively.

The *figure 1* shows all the connections for the integration of TAM to cores and vice-versa for this particular SOC.

## IV. Experimental Results

All the experiments were conducted on SOLARIS 2.7 environment in a SUN SPARC ULTRA 10 workstation.

### A. Preparation of Benchmarks

The ISCAS'89 benchmark circuits are considered and studied for converting them into full-scan circuits using SYNOPSYS Design Library. The net list is saved in verilog format after synthesis. The ATPG tool (TetraMax of SYNOPSYS) is used to generate test vectors. For all the circuits listed the test vector sets, test coverage, fault coverage besides the information on the number of scan chains, length of scan chains are generated in Standard Test Interface Language(STIL) format [1] by TetraMax. All the cores tested here are resulting in a 100 % test coverage. These test vector sets can be treated as the precomputed test vectors provided by the core vendors. The results of each core performance are listed in *Table I*.

TABLE I
**Details of Benchmark Cores**

| Core | Scan Chains | Scan Length | Test Vectors | Clocks | Time sec | % Fault Coverage |
|---|---|---|---|---|---|---|
| S27 | 1 | 3 | 4 | 28 | 1.13 | 81.82 |
| S208 | 1 | 8 | 18 | 221 | 1.00 | 88.24 |
| S298 | 2 | 7 | 5 | 75 | 1.02 | 88.24 |
| S344 | 2 | 7, 8 | 5 | 85 | 1.23 | 92.86 |
| S349 | 2 | 7, 8 | 5 | 85 | 1.07 | 92.86 |
| S382 | 2 | 10, 11 | 5 | 115 | 1.10 | 88.24 |
| S386 | 1 | 6 | 11 | 143 | 1.07 | 90.00 |
| S400 | 2 | 10, 11 | 5 | 115 | 1.42 | 88.24 |
| S444 | 2 | 10, 11 | 4 | 92 | 1.23 | 88.24 |
| S510 | 1 | 6 | 37 | 471 | 1.34 | 93.75 |
| S526 | 2 | 10, 11 | 4 | 88 | 1.17 | 88.24 |
| S641 | 2 | 9, 10 | 10 | 210 | 1.03 | 97.01 |
| S713 | 2 | 9, 10 | 9 | 189 | 1.07 | 96.67 |
| S820 | 1 | 5 | 26 | 286 | 1.38 | 95.35 |
| S832 | 1 | 5 | 26 | 286 | 1.38 | 95.35 |
| S1196 | 3 | 6 | 8 | 104 | 1.21 | 89.47 |
| S1238 | 3 | 6 | 8 | 104 | 1.21 | 89.47 |
| S1423 | 11 | 6, 7 | 8 | 136 | 1.37 | 95.83 |
| S1488 | 1 | 6 | 9 | 117 | 1.44 | 93.94 |
| S1494 | 1 | 6 | 11 | 117 | 1.44 | 93.94 |
| S5378 | 22 | 8, 9 | 10 | 190 | 1.77 | 97.35 |

### B. Simulation Results

A post synthesis simulations are carried out on all the SOCs formed using the simulation tool (VCS of SYNOPSYS). Each of the SOC is integrated using the net lists of cores and TAM switches in to a top module. Test benches for each of them are created to run the simulations. For all the SOCs formed the results are presented in the *Table II* .

For example, when SOC4 is tested in four test sessions using our method. It results in a total test time of 862 clock cycles.

TABLE II

**Test Time Comparison**

| SOC No | Session No | Session Core | Session Time | Our Method (#clocks) | Sequential Tesing (#clocks) | % Time Saving |
|---|---|---|---|---|---|---|
| 1 | 1 | 1, 4 | 560 | 51, 560 | 542, 1071 | |
| | 2 | 0, 3 | 492 | 201, 492 | 449, 890 | |
| | 3 | 2 | 229 | 229 | 725 | |
| **Total Test Time** | | | | 1281 | 3677 | 65 |
| 2 | 1 | 0, 5 | 2518 | 43, 2518 | 146, 1610 | |
| | 2 | 4 | 279 | 279 | 944 | |
| | 3 | 3 | 269 | 269 | 738 | |
| | 4 | 1, 2 | 458 | 209, 458 | 292, 542 | |
| **Total Test Time** | | | | 3524 | 4372 | 19 |
| 3 | 1 | 1, 4 | 189 | 157, 189 | 674, 1311 | |
| | 2 | 0, 3 | 367 | 367, 349 | 457, 1140 | |
| | 3 | 2 | 243 | 243 | 859 | |
| **Total Test Time** | | | | 799 | 4441 | 82 |
| 4 | 1 | 0, 4 | 315 | 315, 169 | 413, 1063 | |
| | 2 | 3 | 169 | 169 | 910 | |
| | 3 | 2 | 209 | 209 | 731 | |
| | 4 | 1 | 169 | 169 | 552 | |
| **Total Test Time** | | | | 862 | 3669 | 77 |
| 5 | 1 | 0, 5 | 6402 | 367, 6402 | 629 | |
| | 2 | 4 | 1824 | 1824 | 2307 | |
| | 3 | 3 | 436 | 436 | 1898 | |
| | 4 | 2 | 608 | 608 | 1503 | |
| | 5 | 1 | 754 | 754 | 1110 | |
| **Total Test Time** | | | | 10024 | 7600 | 32* |

*Since test hardware is increased extra time is consumed.

When sequentially tested using a conventional switch the total testing time required is 3669 clock cycles. Hence a 77 % of testing time is saved in our method. In case of SOC-5 since the test hardware overhead(i.e, number of TAM switches used to feed core 5(S5378)) is high it is consuming extra time than what is required. But for core 0 to 4 a significant reduction in test time is observed.

All the SOCs were synthesized to see the areas occupied by the TAM switches used in the process. Concern is to determine what is the overhead due to the area requirements when compared to the conventional switch being used for the testing process. Results are listed in the *Table III*.

## V. Conclusions

We propose a new design-for-test technique for SOC designs catering to the needs of test scheduling, TAM design, test set preparation and test resource placement. The proposed test strategy algorithms allows a trade-off between the optimized test hardware and the test time. The TAM switch results in providing a perfect test access to any core in the SOC.

TABLE III

**Area Overhead Comparison**

| SOC No | Proposed Method | | | Sequential Testing | | |
|---|---|---|---|---|---|---|
| | Total Area | Switch Area | (%) Area Overhead | Total Area | Switch Area | (%) Area Overhead |
| 1 | 367076 | 139698 | 38 | 374120 | 150681 | 40 |
| 2 | 541959 | 262613 | 48 | 558645 | 284548 | 51 |
| 3 | 419272 | 146765 | 35 | 381812 | 113827 | 30 |
| 4 | 370090 | 195371 | 52 | 316022 | 136022 | 43 |
| 5 | 3461293 | 2408291 | 75 | 1541810 | 690417 | 45 |

## References


[1] "IEEE P1500 Web Site," http://grouper.ieee.org/groups/1500/.

[2] Yervant Zorian, Erik Jan Marinissen, and Sujit Dey, "Testing Embedded-Core-Based System Chips," in *IEEE Computer*. June 1999, vol. 32, pp. 52–60, IEEE Computer Society Press.

[3] Yervant Zorian, "Test Requirements for Embedded Core-Based Systems and IEEE P1500," in *Proceedings IEEE International Test Conference (ITC)*, Washington, DC, Nov. 1997, pp. 191–199, IEEE Computer Society Press.

[4] Erik Jan Marinissen, Robert Arendsen, Gerard Bos, Hans Dingemanse, Maurice Lousberg, and Clemens Wouters, "A Structured And Scalable Mechanism for Test Access to Embedded Reusable Cores," in *Proceedings IEEE International Test Conference (ITC)*, Washington, DC, Oct. 1998, pp. 284–293, IEEE Computer Society Press.

[5] Lee Whetsel, "Addressable Test Ports: An Approach to Testing Embedded Cores," in *Proceedings IEEE International Test Conference (ITC)*, Atlantic City, NJ, Sept. 1999, pp. 1055–1064, IEEE Computer Society Press.

[6] D.Bagchi D.R.Chowdhury J.Mukherjee and S.Chattopadhyay, "A Novel Strategy to Test Core Based Designs," in *Proceedings International Conference on VLSI Design*, Bangalore,India, Jan. 2001, pp. 122–127.

[7] K.Chakrabarty, "Test Scheduling for Core-Based Systems Using Mixed-Integer Linear Programming," in *IEEE Transactions on Computer-Aided Design*, Oct. 2000, vol. 19, pp. 1163–1174.

[8] Tomokazu Yoneda and Hideo Fujiwara, "A DFT Method for Core-Based Systems-On-a-Chip based on Consecutive Testability," in *Proceedings IEEE Asian Test Symposium (ATS)*, Nov. 2001, pp. 193–198.

[9] E.Larsson and Z.Peng, "An Integrated Framework for Design and optimization of SOC Test Solutions," in *Journal of Electronic Testing: Theory and Applications(JETTA), Special Issue on Plug-and-Play Test Automation for System-on-a-Chip*, Aug. 2002, vol. 18.

[10] Yu Huang, Wu-Tung Cheng, Chien-Chung Tsai, Omer Samman, Yahya Zaidan, and Sudhakar M.Reddy, "Resource Allocation and Test Scheduling for Concurrent Test of Core-Based SOC Design," in *Proceedings IEEE Asian Test Symposium (ATS)*, Nov. 2001, pp. 361–366.

[11] Subhayu Basu, I. Sengupta, D. Roy Chowdhury, and S. Bhowmik, "An Integrated Approach To Testing Embedded Core and Interconnects Using Test Access Mechanism(TAM) Switch," in *Journal of Electronic Testing: Theory and Applications(JETTA), Special Issue on Plug-and-Play Test Automation for System-on-a-Chip*, Aug. 2002, vol. 18.

[12] C.V. Guru Rao and D. Roy Chowdhury, "Testing of Embedded Core Based Systems Using Linear Cellular Automata," in *Proceedings of Cellular Automata Symposium 2001*, Yokohama National University, Japan, Nov. 2001, number 01-63, pp. 108–113.