Test Wrapper Design for 3D System-on-Chip Using Optimized Number of TSVs

Surajit Kumar Roy[1], Sourav Ghosh, Hafizur Rahaman[2] and Chandan Giri[3]
Dept. of Information Technology
Bengal Engineering & Science University
Shibpur, Howrah - 711103, India
Email:[1]suraroy@gmail.com,[2]rahaman_h@yahoo.co.in,[3]chandangiri@gmail.com

***Abstract*—Manufacturing of 3D stacked IC chips has became feasible recently. But testing of these 3D ICs is becoming important in the semiconductor industry. Now a days also embedded core based 3D ICs are equally popular. Hence the increased complexity in the chips becomes a constraint for the test engineers. This paper addresses a 1500-style wrapper optimization in 3D ICs based on *Through Silicon Vias* (TSVs) for vertical interconnects. This work minimizes scan test time with the optimum number of TSVs available for testing. The results are presented based on the ITC'02 SOC test benchmarks. The simulation results show that small number of TSVs are sufficient to test the embedded cores in the SOC.**

***Keywords*-Scan chain; 3D integrated circuits; wrapper design; test access mechanism**

I. Introduction

System-on-Chip (SOC) design & manufacturing paradigm is now used commonly to design embedded systems that comprises a number of cores like logics, memory, processors, radio frequency/photonic devices, microelectromechanical systems in a single chip. But these cores should be tested properly before the embedded system is launched in the market. For testing these cores, the modular testing approach has been adopted, where a test access mechanism (TAM) is to be included on the chip. Test stimuli are transfered to the cores using the chip input pins and test responses are collected through TAM using chip output pins. The modular testing is being done by designing the IEEE Std. 1500 [1] wrapper interface between the I/O pins of the embedded cores and the TAM. In order to reduce the test cost in terms of test time, the test wrapper and TAM optimization technique has been used. Various techniques [2], [3], [4] have been done to optimize the test cost together with several other constraints. All these works are done mainly for two dimensional (2D) integrated circuits.

Three dimensional integration technology using TSV is rapidly emerging that provides numerous benefits over traditional 2D ICs[5]. This integration technology focuses on portraying advances in interconnect technologies. But the integration of embedded cores in multi-layer 3D ICs leads to new test challenges.

A number of 3D integration methods have been proposed in the literature. In this work, we consider the integration method on TSV interconnects as it provides the highest vertical density among the proposed technologies. In this technology, the 3D ICs are fabricated by placing multiple device layers together through wafer or die stacking which are connected via TSVs [6]. Figure 1 provides an example of 3D IC using TSV interconnects for multi die configurations.

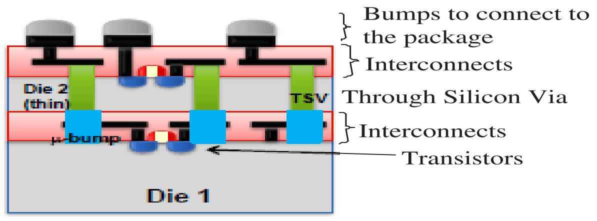

Figure 1. Example 3D ICs using TSV Stack

The design of a 3D SOC can be done at two levels of granularity [7]: a) *Coarse-granularity partitioning*: Here the embedded cores in the SOC is like a 2D design space.

b) *Fine-granularity partitioning*: Here the cores are partitioned in multiple layers [6] that shows significant improvement in performance and overall frequencies. For example, it is shown in [8] that repartitioning of the Intel Core 2 processor across four die-stacked tiers can improve 47.9% increase in overall frequency and 47% performance.

But the distribution of the cores in several layer again provide some extra challenges in testing. As the scan chains of the cores distributed across layers, it reduces the wire length of the scan chains. In [9], it is shown how the scan chain length can be reduced in 3D ICs. In [10], the authors showed the dual optimization of scan chain length(interconnect delay) together with scan shift power by reordering scan chain using Genetic Algorithmic (GA) approach for 3D ICs.

Though it is now feasible for manufacturing the 3D ICs, but the CAD tools for design and testing are not matured yet for commercial use as it needs to optimize several design related constraints. For example, in TSV based 3D ICs number of available TSVs for test access is limited. Hence in this work, we are mainly trying to focus on wrapper design and optimization of cores with optimized number of TSVs.

Several works on wrapper optimization and test infrastructure design for 2D SOCs [2], [3], [4] have been proposed in last few years. Optimization methods include ILP [2], bin packing [11], [4], Genetic Algorithm[4]and heuristic method [3]. None of these have considered the design problems related to 3D technologies.

Some works [12], [13] have been presented recently on testing of 3D ICs. In [7], the authors have made an attempt to solve the problem by using 1500 Std. based test wrapper design for 3D SOCs where they have considered that the wrapper elements are distributed over several layers and the layers are connected vertically through TSVs. The solution is made based on two polynomial-time heuristic algorithm and the results are presented based on the ITC'02 benchmarks.

In this work, we address the wrapper optimization problem for TSV based 3D SOCs which is shown as *NP-hard* problem [2] for 3D ICs. Our objective here is to minimize the scan test time by proper partitioning/allocation of internal scan chains and functional I/Os of the cores in different layers of the SOC so that the total number of TSVs required are optimized. It is assumed that the embedded cores have full scan and the scan chains are hard.

The sequel of the paper is organized as follows. Section II describes the problem definition. Algorithms for 3D wrapper chain design using optimized TSV have been discussed in Section III. Section IV discusses about the Genetic Algorithm formulation for obtaining optimal partitioning of the core elements. Experimental results are discussed in Section V and finally Section VI draws the conclusion.

II. Problem formulation

The main objective of the wrapper optimization is to minimize the test time of each core. The problem that has been considered in this paper is as follows: *Given a core and its different test parameters like number of functional inputs, number of functional outputs, number of scan chains, length of each of the scan chains, the TAM width, determine the optimal partitioning of the core elements in different layer and the design of the wrapper scan chain such that the length of the longest wrapper scan chain is minimized and the total number of TSVs required are optimized.*

In this paper we have used Genetic Algorithm (GA) formulation to get the optimized partitioning of the core elements so that the total number of TSVs required is optimized. The experimental result on ITC'02 benchmark SOCs shows that the total number of TSVs required is less than the heuristics presented earlier in [7].

III. 3D Wrapper Chain Design Using Optimized TSVs

Wrapper design is used to make the interconnections between the wrapper cells, the core internal scan chains, and the TAM plugs. We define this activity as a wrapper chain design and the elements that make this wrapper chain known as wrapper chain items.

Algorithm 1: Subroutine Createscanchainset [2]

```
   Input : TAM_width, internal scan chains
   Output: Wrapper chains containing the internal scan chains
1  begin
2      Sort the scan chains in descending order of length;
3      for each scan chain $l$ do
4          Find a wrapper scan chain $S_{max}$ with current maximum length;
5          Find wrapper scan chain $S_{min}$ with current minimum length;
6          Assign $l$ to wrapper scan chain $S$ such that
           $length(S_{max} - length(S) + length(l))$ is minimum;
7      end
8  end
```

Algorithm 2: Subroutine req_input_outputcell

```
   Input : Set S, functional I/Os, TAM_width
   Output: Complete designed wrapper chain
1  begin
2      $c = \sum length(S_i) + n_i + n_o$;
3      $avg = \lfloor (c/TAM\_width) \rfloor$;
4      for ($k \leftarrow 1$ to $TAM\_width$) do
5          if ($avg > length(S_i)$) then
6          begin
7              $reqn_i = \lfloor ((avg\text{-}length(S_i)/2) \rfloor$;
8              $reqn_o = avg\text{-}(reqn_i + length(S_i))$;
9              if $reqn_i \leq n_i \&\& reqn_o \leq n_o$ then assign $reqn_i$ and
               $reqn_o$ to wrapper chain set;
10             if $reqn_i > n_i \&\& reqn_o \leq n_o$ then assign $n_i$ and
               remaining available output cells to make scan chain length and
               $avg$ length equal;
11             if $reqn_i \leq n_i \&\& reqn_o > n_o$ then assign $n_o$ and
               remaining available input cells to make scan chain length and
               $avg$ length equal;
12             if $reqn_i > n_i \&\& reqn_o > n_o$ then assign $n_i$ and $n_o$ to
               wrapper chain;
13             Assign remaining wrapper I/O cells to last wrapper chain;
14         end
15     end
16 end
```

The algorithms which are used as a sub-routine to design the wrapper chain is presented in Algorithm 1 [2] and Algorithm 2. The algorithms are explained with an example in the following section.

A. Illustrative example for creating wrapper chains

Example 1: Consider the distribution of the wrapper chain items for an embedded core in different layers is shown in Fig. 2. The TAM width is taken as 4. So the number of wrapper chains is to be created also 4. The example shows that the core consists of 8 internal scan chains of length 8 each and 16 input cells and 16 output cells. The internal scan chains of the core are partitioned among 4 wrapper chains. Table I shows the distribution of the internal scan chains over wrapper chains using the *createscanchainset()* (Algorithm 1) [2]. The algorithm 1 works as follows:
The internal scan chains are sorted in descending order of length. Then for each internal scan chain l select a wrapper chain (WC_i) with *minimum* length and assign l to that wrapper chain. In this way, the set $S = \{\{8,8\},\{8,8\},\{8,8\},\{8,8\}\}$ is obtained, where each wrapper scan chain contains two internal scan chains of equal lengths.

$length(S_i)$	$R_i =$ $avg - length(S_i)$	$Reqn_i =$ $\lfloor R_i/2 \rfloor$	$Reqn_o =$ $R_i - Reqn_i$	$n_i =$ $n_i - Reqn_i$	$n_o =$ $n_o - Reqn_o$
16	24 - 16 = 8	8/2 = 4	8-4 = 4	16 - 4 = 12	16 -4 =12
16	24 - 16 = 8	8/2 = 4	8-4 = 4	12 - 4 = 8	12-4 =8
16	24 - 16 = 8	8/2 = 4	8-4 = 4	8 - 4 = 4	8 -4 =4
16	24 - 16 = 8	8/2 = 4	8-4 = 4	4 - 4 = 0	4 -4 =0

Table II
NUMBER OF WRAPPER I/O CELLS ASSIGNED TO DIFFERENT WRAPPER CHAINS

WC_1	WC_2	WC_3	WC_4
8	8	8	8
8	8	8	8 + 8
8	8	8+8	8 + 8
8	8+8	8 + 8	8+8
8+8	8+8	8+8	8+8

Table I
INTERNAL SCAN CHAIN DISTRIBUTION OVER WRAPPER CHAIN USING ALGORITHM 1

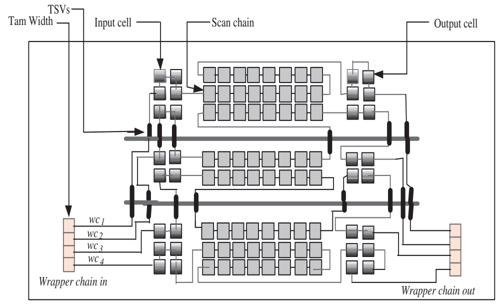

Figure 2. Conceptual design of wrapper chain and calculation of number of TSVs required

In order to complete the design, it is necessary to connect input cells and output cells of the core with these wrapper chains. Hence it is required to calculate the number of input and output cells for each wrapper chain to make the chains almost of equal length. Wrapper cells are distributed among 4 wrapper chains using *req_input_outputcell()* algorithm and the working of the algorithm is explained using the Table II. In algorithm 2, average length of each wrapper chain is calculated using the Eqn.(1).

$$Average(avg) = \lfloor \sum length(S_i) + (n_i + n_o)/TAM\_width \rfloor \quad (1)$$

In Eqn.(1) $length(S_i)$ is the length of the i^{th} wrapper chain and n_i, n_o is the total number of input cells and output cells respectively.

In Table II R_i denotes the number of more wrapper elements to be added to S_i to get the avg value. $Reqn_i$ and $Reqn_o$ are the number of input cells and output cells to be added to S_i to make the length equal with avg length. Thus the resulting wrapper set is obtained as $WC = \{\{4, \{8, 8\}, 4\}, \{4, \{8, 8\}, 4\}, \{4, \{8, 8\}, 4\}, \{4, \{8, 8\}, 4\}\}$.

layer_incell	Number of input cells for layer L
layer_outcell	Number of output cells for layer L
layer_sc	Number(LEN) of internal scan chains for layer L
sc_length[1...LEN]	Length of the internal scan chains

Table III
LAYER DATA STRUCTURE

Here all the wrapper chains are designed in a balanced way. Balanced wrapper scan chains are those that are equal in length to each other.

The next section describes how the number of TSVs are calculated for this particular configuration of the core A.

B. Calculating the number of TSVs

The wrapper chain items are not in the same layer. Thus in order to complete the wrapper chain design, it is necessary to connect wrapper chain items present in the different layers using vertical interconnect known as TSV. As all of the chip pins are at the lowest layer, it necessitates that wrapper chains begin and end on the lowest layer. This effects the total number of TSVs. Consider the first wrapper chain WC_1 (Fig.2). Here all its elements are located in layer number three. So the wrapper chain begins at layer 1, connects all the wrapper elements in the layer 3 and again ends at the layer 1. So to design this wrapper chain, number of TSVs required are 4. This calculation is being done by using the data structure presented in Table III. The number of TSVs are calculated as $\sum(currentlayer_i - nextlayer_i))$, where $currentlayer_i$ determines whether any wrapper elements are present in the layer currently being considered and the $nextlayer_i$ indicates the layer number which has already considered for the wrapper chain i. Similarly for the other wrapper chains, the number of TSVs required can also be easily calculated. In the given example in Fig. 2, total number of TSVs required are 12 when TSVs that are internal to scan chains are counted. The required TSVs are 10 for the same, if TSVs internal to scan chains are not counted.

The data structure presented in Table III is created after random partitioning of the input wrapper cells , internal scan chains and output wrapper cells into different layers which will initialize object in each layer data structure. The random partitioning of the core elements is obtained using Genetic Algorithm (GA) which is discussed in the next section.

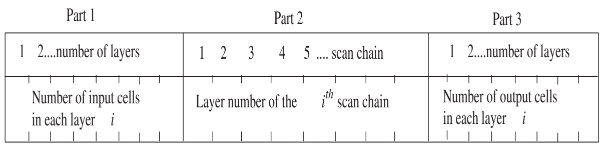
Part 1
Part 2
Part 3
1 2....number of layers
1 2 3 4 5 .... scan chain
1 2....number of layers
Number of input cells in each layer i
Layer number of the i^{th} scan chain
Number of output cells in each layer i

Figure 3. Structure of the chromosome

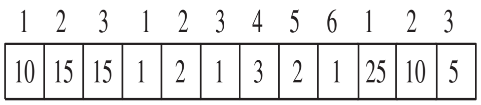

Figure 4. Example chromosome

IV. Genetic Algorithm Formulation for 3D Wrapper Optimization with Optimized Number of TSVs

In case of fine-granularity partitioning internal scan chains of a core can be distributed over the different layers. TSVs internal to a scan chain are not counted as TSV constraints. We have assumed that the lowest layer of the IC is layer 1, followed by layers 2, 3, etc. This section describes how the core wrapper elements are partitioned into the different layers of the 3D SOCs.

A. Basics of GA

Genetic algorithm [14] is a stochastic search and optimization algorithm. Instead of working with a single solution, the algorithm starts searching a random set of chromosome called the initial population. Each chromosome is assigned a fitness score that is directly related to the objective function of the optimization problem. The basic genetic algorithm works as follows:

Step 1: Randomly generate an initial population of fixed size.

Step 2: Define selection criteria for reproduction.

Step 3: Compute fitness for each chromosome.

Step 4: Produce offspring by mutation and crossover operation.

Step 5: Repeat Step 2, 3, and 4 until a satisfactory solution is obtained.

The next section discusses about how the concerned problem is mapped into genetic space and its different constituents.

B. Genotype

The genotype represents a chromosome that gives a solution of the problem. For the current problem a chromosome is consisting of three parts. Part 1 determines the number of input cells in each layer, Part 2 represents the assignment of internal scan chains in different layers and Part 3 determines the distribution of output cells in the layers. Each part of the chromosome is encoded using integer number. Part 1 of the chromosome is encoded as $< p_1, p_2, ..., p_l >$, where p_i ($1 \leq i \leq l$, l is the number of layers) determines that p number of input cells are assigned to the layer number i. Similarly for the Part 3 also. For Part 2, the chromosome is encoded as $< L_1, L_2, L_3, ..., L_n >$, where L_i ($1 \leq i \leq n$, n is the number of internal scan chains) indicates that the i^{th} scan chain is assigned to the layer number L. According to the encoding policy it is clearly indicated that the size of the Part 1 and Part 3 depends on the number of layers whereas the size of the Part 2 of the chromosome depends on the number of internal scan chains present in the particular core of the SOC. The Fig. 3 shows the structure of the chromosome and its various parts. One typical chromosome is shown in Fig. 4.

Example 2: Let the number of input pins and output pins of a core are 40 each and total number of internal scan chains are 6 with lengths $12, 12, 12, 12, 24, 24$ respectively. Let the randomly distribution of the core elements are obtained from the chromosome as shown in Fig. 4 that gives a solution for our given problem. The chromosome presented in Fig. 4 conveys the following informations: a) First part(Part 1) gives the information about the number of input cell in each layer. Here number of layer is 3, thus it is indexed as 1 to 3. Therefore, 2^{nd} entry of Part 1 says that second layer consists of 15 input pins of the core. Similarly, it is clear that first and third layer consists of input cells 10 and 15 respectively.

b) Part 2 is indexed from 1 to 6 as total number of internal scan chains present is 6. First entry in the Part 2 denotes the internal scan chain number 1 which is in first layer (Layer 1). Similarly, other entries define the layer numbers for each of the internal scan chains.

c) The chromosome also provides the total number of internal scan chains in each layer after distribution. From the given example it can be said that 1^{st} , 2^{nd} and 6^{th} internal scan chains are present in first layer, that means Layer 1 contains 3 internal scan chains. Similarly, Layer 2 contains 2 scan chains and Layer 3 contains only one scan chain.

d) Part 3 of the chromosome provides the number of output pins of the core present in each layer after partitioning as discussed in Part 1.

C. Crossover

The crossover operator tends to enable the evolutionary process to move towards‘promising’ regions of the search space. In this operation randomly selected two chromosomes are participated and a single point crossover operation is used. As the chromosome consists of three parts (as shown in Fig. 3), crossover operation is applied separately for each part of the chromosome. The crossover operation for the Part 2 is shown in the Fig. 5. Child chromosomes are obtained after the crossover operation. But it is necessary to keep in mind that readjustment of the values for Part 1 and Part 3 is required if the total number of input and output pins in the

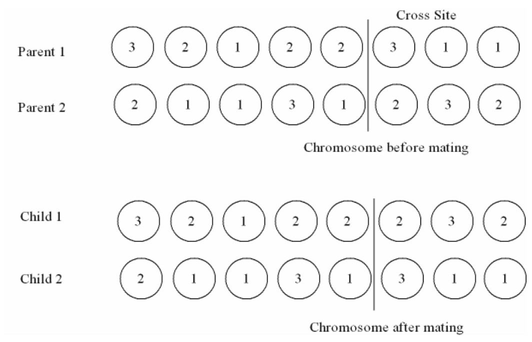

Figure 5. Crossover operation for the Part 2 of the chromosome

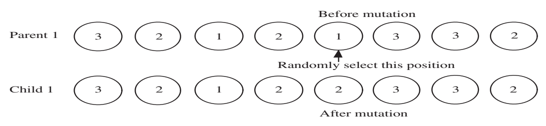

Figure 6. Mutation operation

chromosome is exceeded to the total number of input/output pins of the core. But no readjustment operation is required for Part 2 of the chromosome after doing the crossover operation.

D. Mutation

This operator is introduced for avoiding premature convergence to local optima by randomly sampling new point in the search space. Mutation operation is also used separately for each part of the chromosome. In our formulation we have used the single point mutation operation as shown in Fig. 6 for the Part 2.

E. Fitness function

The fitness value of each of the chromosome is considered as the number of TSVs required to design the wrapper chains for the corresponding core. This is calculated for each of the chromosome to get the optimized number of TSVs as discussed in Section III.

F. Evolution Process

The size of the initial population depends on the number of core elements present in an embedded cores. The size is ranging form 100-500. To get the new generation the chromosomes are sorted according to the fitness value and the best 20% chromosomes are copied directly to the next generation. The remaining 80% chromosomes are created using crossover and mutation operation. In our case, 70% chromosomes are created using crossover operation and the remaining 10% chromosomes are created using mutation operation. This process is continued for 100 generations to reach an optimal solution.

V. Experimental Results

A. Experimental Setup

The experiments are performed on the ITC'02 SOC test benchmarks. For simulation, the algorithms are implemented in C++ language and run on a Intel Core 2 Duo processor having 1GB RAM. In this paper we have presented our experimental results for the core number 7 of d281, core number 4 of p93791 and core number 5 of h953. Number of layers for the SOC chip considered for each case of the simulation is 3.

B. Results

The simulation results for the core 7 of SOC d281 is presented in Table IV and the range of TAM widths are considered in between 2 to 6. Table V shows the results for core 4 of SOC p93791 for the range of TAM width between 2 to 8 and Table VI is shown for core 5 of h953 benchmark. For each of the table, column 1 lists the TAM width (W), column 2 lists the shortest wrapper length obtained by using our algorithm, column 3 shows the shortest wrapper length as in [7]. Column 4 shows the number of TSVs required to design the wrapper, where it counts the number of TSVs for internal scan chains (column marked with WSC) and column 5 lists the number of TSVs required where TSVs for internal scan chains are not counted (column marked with SC). Column 6 lists the number of TSVs required present in [7] for the shortest wrapper length. Column 7 and 8 compare the simulation time taken by our algorithm and with [7].

The Table IV, Table V and Table VI show that for each case of the wrapper chain design using our algorithm, the number of TSVs required are very less than the results presented in [7], even if the TSVs for internal scan chains are counted. The length of the shortest wrapper chain obtained using our algorithm is less when the number of TAM width is increased. This is due to mainly the optimal partitioning of the core elements over wrapper chains. For example in case of core 4 of p93791 when the TAM width is 8, the length of the shortest wrapper chain is 19, whereas the length of the shortest wrapper chain presented in [7] is 25. Simulation time taken for our case is smaller for the core 7 of d281 for the given TAM widths than the simulation times presented in [7]. But for the case of Core 4 of p93791 the simulation time taken in our case is less for higher TAM widths compared to [7]. Finally we can say that the algorithms presented optimum number of TSVs required to design the wrapper chains for various TAM width. This is because the number of TSVs obtained here is optimized based on the optimal partitioning of the core elements using GA.

VI. Conclusion

In this paper, we have presented a GA based optimization technique for minimizing test time by reducing the wrapper chain length for 3D core-based SOCs. Our algorithm optimized the number of TSVs required to design the wrapper

SOC Benchmark (d281) Core 7							
W	**Shortest WC Length**	**Shortest WC Length** [7]	**No. of TSVs**		**No. of TSVs max** [7]	**CPU time (sec)** (our)	**CPU time (sec)** [7]
			WSC	**SC**			
2	1064	1064	8	7	18	72	188
3	709	710	8	5	18	59	220
4	532	532	8	6	18	90	423
5	425	426	8	5	22	52	1937
6	354	355	8	6	22	56	3230

Table IV
RESULTS ON WRAPPER OPTIMIZATION FOR CORE 7 OF D281 WITH 3 LAYERS

SOC Benchmark (p93791) Core 4							
W	**Shortest WC Length**	**Shortest WC Length** [7]	**No. of TSVs**		**No. of TSVs max** [7]	**CPU time (sec)** (our)	**CPU time (sec)** [7]
			WSC	**SC**			
2	76	77	8	4	18	75	≈ 0
3	50	51	8	5	18	63	1.8
4	38	39	8	5	18	68	5
5	30	35	6	5	20	68	7
6	25	28	6	5	20	75	19
7	21	30	6	5	20	73	52
8	19	25	4	4	20	82	199

Table V
RESULTS ON WRAPPER OPTIMIZATION FOR CORE 4 OF P93791 WITH 3 LAYERS

SOC Benchmark (h953) Core 5							
W	**Shortest WC Length**	**Shortest WC Length** [7]	**No. of TSVs**		**No. of TSVs max** [7]	**CPU time (sec)** (our)	**CPU time (sec)** [7]
			WSC	**SC**			
2	257	258	4	4	16	52	≈ 0
3	121	241	4	4	16	306	≈ 0
4	128	129	4	4	16	53	≈ 0
5	32	129	8	8	16	44	≈ 0
6	16	129	8	8	16	92	≈ 0
7	10	129	8	8	16	77	7
8	8	129	8	8	16	63	28

Table VI
RESULTS ON WRAPPER OPTIMIZATION FOR CORE 5 OF H953 WITH 3 LAYERS

chains with the constraints of TSVs by optimal partitioning of the core elements in several layers of the SOC. Results on ITC'02 benchmark show that the results are obtained within a considerable time and the number of TSVs required are less compared to the heuristic method presented earlier.

REFERENCES

[1] "IEEE Std. 1500: IEEE Standard Testability method for Embedded Core-based Integrated Circuits," IEEE Press, 2005.

[2] V. Iyengar, K. Chakrabarty, and E. J. Marinissen, "Test Wrapper and Test Access Mechanism Co-Optimization for System-on-Chip," *Journal of Electronic Testing: Theory and Applications(JETTA)*, vol. 18, pp. 213–230, 2002.

[3] S. K. Goel and E. J. Marinissen, "SOC Test Architecture Design for Efficient Utilization of Test Bandwidth," *ACM Trans. Design Automation of Electronic Systems*, vol. 8, no. 4, pp. 399–429, 2003.

[4] C. Giri, S. Sarkar, and S. Chattopadhyaya, "A Genetic Algorithm Based Heuristic Technique for Power Constrained Test Scheduling in Core-based SOCs," in *Proc. IEEE Intl. Conf. on IFIP VLSI-SOC*, 2007, pp. 320–323.

[5] W. R. Davis, J. Wilson, S. Mick, J. Xu, H. Hua, C. Mineo, A. M. Sule, M. Steer, and P. D. Franzon, "Demistifying 3D ICs: the Pros and Cons of Going Vertical," *IEEE Design and Test of Computers*, vol. 22, no. 6, pp. 498–510, 2005.

[6] Y. Xie, G. H. Loh, B. Black, and K. Bernstein, "Design Space Exploration for 3D Architectures," *ACM Journal of Emerging Technology and and Computer Systems*, vol. 2, no. 2, pp. 65–103, 2006.

[7] B. Noia, K. Chakrabarty, and Y. Xie, "Test-Wrapper Optimization for Embedded Cores in TSV-Based Three-Dimensional SOCs," in *IEEE International Conference on Computer Design*, 2009, pp. 70–77.

[8] K. Puttaswamy and G. H. Loh, "Thermal herding: Microarchitecture techniques for controlling hotspots in high performance 3d integrated processors," in *IEEE High Performance Computer Architecture*, 2007, pp. 193–204.

[9] X. Wu, P. Falkenstern, and Y. Xie, "Scan Chain Design and Optimization for 3D ICs," *ACM Journal of Emerging Technology and and Computer Systems*, vol. 5, no. 9, July 2009.

[10] C. Giri, S. K. Roy, B. Banerjee, and H. Rahaman, "Scan Chain Design Targeting Dual Power and Delay Optimization for 3D Integrated Circuits," in *Proc. IEEE Intl. Conf. on Advances in Computing, Control, and Telecommunication Technologies*, India, 2009, pp. 845–849.

[11] Y. Huang, S. M. 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 *Proc. International Test Conference*, 2002, pp. 74–82.

[12] X. Wu, Y. Chen, K. Chakrabarty, and Y. Xie, "Test Acess Mechanism Optimization for Core-based Three-dimensional SOCs," in *IEEE International Conference on Computer Design*, 2008, pp. 212–218.

[13] L. Jiang, L. Huang, and Q. Xu, "Test Architecture Design and Optimization for Three-dimensional SOCs," in *Proc. Design, Automation and Test in Europe*, 2009, pp. 220–225.

[14] D. E. Goldberg, "Genetic Algorithms in Search, Optimization, and Machine Learning," Addison-Wesley, 1989.