Optimizing Test Wrapper for Embedded Cores using TSV based 3D SOCs

Surajit Kumar Roy[1], Chandan Giri[2], Sourav Ghosh[3] and Hafizur Rahaman[4]
Dept. of Information Technology
Bengal Engineering & Science University
Shibpur, Howrah - 711103, India
Email:{[1]suraroy,[3]sourav.of.25}@gmail.com, [2]chandan@it.becs.ac.in and [4]rahaman_h@yahoo.co.in

***Abstract*—Core based three-dimensional(3D) integrated circuits (ICs) design is an emerging field of semiconductor industry that promises greater number of devices on chip, increased performance and reduced power consumption. But due to scaling in technology features these chips are more complex and hence testing of these 3D ICs is a challenging task. This paper follows a P1500-style wrapper design for 3D ICs using through silicon vias (TSVs) for testing purpose. It is assumed that the core elements are distributed over several layers of the ICs. As the number of available TSVs are limited due to small chip area, this work is intended to design balanced wrapper chains using available TSVs. In this work we have proposed a polynomial time algorithm of $O(N)$ to design the test wrapper. The results are presented based on the ITC'02 SOC test benchmarks and compared with prior work. Obtained results show that our algorithm provides better utilization of TSVs compared to the work presented in [1].**

***Keywords*-3D integrated circuits, test access mechanism, wrapper chain, through silicon via**

I. Introduction

System-on-Chip (SOC) is common for today's integrated circuits. SOC designers can integrate different types of embedded cores such as ROM, DSP processor, combinational logic blocks, finite state machines (FSM), ALU, multipliers, comparators etc. on a single chip. Each of these cores need to be tested individually after manufacturing of the chip. For this reason modular test approach has been adopted[2]. To test the core using modular testing approach, Test Access Mechanism (TAM) and wrapper are introduced. TAM carries test vectors from the test source to any module under test and transport module output responses to the test sink. A typical test source provides the test vector to the core. Test sink is an on chip signature analyzer or an output response comparison circuit. The modular testing is being done by designing IEEE standard 1500 [3] wrapper. The test wrapper is a thin cell around the core that connects the core terminals to the rest of the chip and TAM, switch a core between functional mode to test mode and vice versa. The wrapper design and TAM optimization have great impact on SOC testing, because they directly affect the core testing time.

Due to the increasing demand of high performance, high density portable hand held application, electronic system's design has shifted from 2D design space to 3D design space. The 3D ICs are built by stacking multiple device layers and interconnecting them using vertical interconnects known as TSV as shown in Fig. 1. TSV offers the greatest vertical density and therefore is the most promising one among all the vertical interconnect technologies.

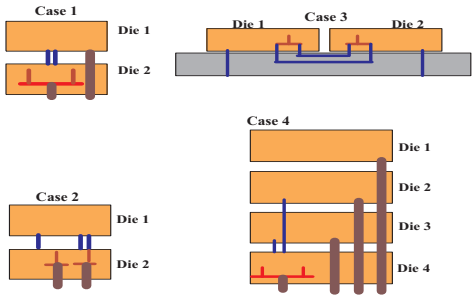

Figure 1. Example 3D ICs

Major potential benefit of vertical integration is reduction in the total length of wiring required for a given system configuration. The wire length reduction alone can minimize the interconnect energy and propagation delay by 51% and 54% respectively[4] at the 45nm technology node. The main hurdle for 3D integration is poor thermal conductivity and heat dissipation which results temperature rise due to the high power density.

The design of a 3D IC can be done at two levels of granularity [1]: a) *Coarse-granularity partitioning*: Here the embedded cores in the SOC is like a 2D design space.
b) *Fine-granularity partitioning*: cores are partitioned in multiple layers [5] that shows significant improvement in performance and overall frequencies. For example, it is shown in [6] that repartitioning of the Intel Core 2 processor across four die-stacked tiers can improve 47.9% increase in overall frequency and 47% performance.

Manufacturing of 3D ICs are now possible, but the support of CAD tools for design and testing are not available for commercial use as it needs to optimize several design related constraints. For example, in TSV based 3D ICs number

978-0-7695-4447-2/11 $26.00 © 2011 IEEE
DOI 10.1109/ISVLSI.2011.33

of available TSVs for test access is limited. Hence in this work, we are trying to focus on wrapper design and optimization of cores with available number of TSVs for vertical interconnects taking as a constraint. It is known that the test application time of a core is dependent on longest wrapper chain because it determines the time needed to load and read out test data. So during the wrapper chain design it is also one of the consideration to design the balanced wrapper so that test time can be minimized.

The rest of the paper is organized as follows. Section II discusses about the previous works done related to testing of 3D SOCs. Section III describes the problem formulation. Proposed algorithm for solving the problem is presented in Section IV. An illustrative example of proposed solution is discussed in Section V. SectionVI presents the experimental results based on ITC'02 benchmark circuits. Finally, Section VII draws the conclusion.

II. Previous Works on Wrapper Design

Previous works on wrapper optimization and test infrastructure design for 2D SOCs [2], [7], [8] have been proposed in last few years. Optimization methods include ILP [2], bin packing [9], [8], Genetic Algorithm[8] and heuristic method [7]. None of these have considered the design problems related to 3D technologies.

There are few limited work on 3D SOC testing. Some works have been feasible for testing of 3D ICs. TAM design for 3D SOC is provided in [10] where integer linear programming(ILP) is used to divide the TAM wires into several test buses and assign the cores to test buses to minimize the test time under TSV constraint. Lewis et al.[11] have presented a scan-island based design to enable pre-bond test method for the testability of die-stacked microprocessors. The authors in [12] addressed optimization of scan chain length along with scan power by reordering the scan chain using Genetic Algorithmic approach for 3D SOC. Jiang et al. [13] proposed simulated annealing (SA) based algorithms to optimize the pre-bond and post-bond test time of 3D ICs. A heuristic method to reduce weighted test cost with constraints on test pin width in pre-bond and post bond are discussed in [14]. In [15], authors have presented an optimization method to minimize the test time for a 3D-SIC, either for the final stack test or for any number of multiple test insertions during bonding. In [16], the authors have made an attempt to design 1500 Std. based test wrapper for 3D SOCs. The authors have modeled 1500-style wrapper optimization in 3D ICs using optimum number of TSVs available for testing.

III. Problem Formulation

The main problem of the wrapper optimization is to minimize the test time of each core. The problem can be elaborated as follows: *Given a core for 3D IC that consists of different functional parameter such as number of functional inputs, number of functional outputs, set of scan chains, length of each of the scan chains and TAM width, determine (a) the distribution of the core elements over several layers of 3D IC for a number of wrapper chains and (b) connects them through the maximum available number of TSVs (TSV_{max}) for a core such that the length of the longest wrapper chain is minimized.*

To solve this problem which is *NP-Hard* in nature[2], we have proposed one heuristic approach that try to distribute the core wrapper elements optimally over several layers of the IC so that the length of the longest wrapper chains is minimized in polynomial time of $O(N)$, where N is the total number of core wrapper elements. The following section IV describes the proposed algorithm with one illustrative example.

IV. Proposed Methodology

The goal of this heuristic method is to distribute the core wrapper elements over different layers of IC into a number of wrapper chains using 3D design constraint as maximum number of available TSVs (TSV_{max}). Scan chains can present in multiple layers and hence the scan-in and scan-out also can present at different layers. The lowest layer of the IC is layer number 1, followed by 2, 3 and so on up to L, the maximum number of layer present in the IC. Basic elements for designing the wrapper of a core are functional inputs, functional outputs and internal scan chains. So we are given the followings: a set of core wrapper elements $E = \{E_1, E_2, ..., E_{x+y+n}\}$, TAM width W and maximum number of available TSVs (TSV_{max}), which are used for routing of the wrapper chains. Here x is the number of functional inputs, y is the number of functional outputs and n is the number of internal scan chains of the core. During the design of the wrapper chain it is considered that TSVs internal to the scan chains are not counted against TSV constraint as assumed in [1].

The following sections discusses about the proposed algorithm and the data structures used to implement the algorithm and the complexity of the algorithm.

A. Data Structures

The proposed algorithm 1 have used two data structures named *Wrapper* and *Element*. Data structure *Element* is presented in Table I contains the information about each and every wrapper elements E_i of the list E. The information about the wrapper elements include the type of element (whether I/O wrapper cell or scan chain), layer number in which it is placed and the length of this wrapper element. These three different information are stored in the variable $Type$, $Layer$ and $Length$ as shown in column 1 of Table I.

Data structure *Wrapper* (shown in Table II) maintains

Type	Type of the i^{th} element
Layer	Layer number of the i^{th} element
Length	Length of the i^{th} element

Table I
Element DATA STRUCTURE

No_of_tsv	Total number of TSVs required to connect all the elements for a particular wrapper chain
No_of_element	Total number of element for a particular wrapper chain
Wrapper_length	Length of a wrapper chain
***incell**	Point to the list of input wrapper cells
***scanchain**	Point to the list of scan chains
***outcell**	Point to the list of wrapper output cells

Table II
Wrapper DATA STRUCTURE

three different lists using three variables corresponding to each type of wrapper elements E_i. Information about the number of TSVs required at any instant of time (specifically before insertion of an wrapper element in to the wrapper chain) of designing the wrapper is stored in the variable *No_of_tsv*. Two other variables named *wrapper_length* and *No_of_element* are used to hold the length of the wrapper chain and number of different types of wrapper elements contains by the wrapper chain. The length of wrapper chain before starting of the algorithm are initialized with 0.

B. Heuristic Algorithm

The algorithm **Createchain()** (as shown in Algorithm 1) starts execution by initializing *Element* and *Wrapper* data structures as discussed. The algorithm tries to pick up one wrapper element and assign layer number to it randomly where the chosen element is placed. But for every wrapper chain we have calculated the required number wrapper I/O cells where both the wrapper input and output cells are distributed almost equally among the wrapper chains. This is done because the assignment of wrapper I/O cells directly affect the number of TSVs required for a chain. When we have added an wrapper input or output cell in a wrapper chain during design we have checked with respect to the apriori calculated I/O wrapper cells. If the I/O wrapper cell requirement is satisfied for the current chain then only selected wrapper element is assigned otherwise we have found out another wrapper chain where the wrapper I/O cell requirement is satisfied. When we assign first wrapper element to any one of the wrapper chain, it is assumed that this must be placed at any layer except the lowest. Wrapper elements are assigned to the wrapper chain in clockwise fashion. For example, if four wrapper chains are to be designed then order will be $W_1 \rightarrow W_2 \rightarrow W_3 \rightarrow W_4$. Each time we have inserted an wrapper element into a wrapper chain, number of TSVs required for that wrapper chain is calculated. We keep on inserting element into the same wrapper chain until no extra TSV is required to connect this new element. When all the available TSVs are utilized we have assumed that rest of the elements will be placed in the lowest layer. This method will be continued until all the elements are assigned. Here we have not counted TSVs that are internal between two scan chains as the assumption made in [1].

Algorithm 1: CreateChain

Input : Test set E, Maximum available TSV (TSV_{max}), Number of layers (L), TAM_width (W)
Output: Designed wrapper chain

begin
  Initialize $k = 0$, $TSV\_count = 0$, $tflag = 0$, $pretsv = 0$, $posttsv = 0$, set $V = \phi$, $layer\_number(E_i \in E,$ for all i$) = -1$;
  For each wrapper chain W_k determine wrapper I/O cells requirements R_k, where $k = 1, 2, ..., W$;
  while $E \neq \phi$ **do**
  begin
    if *($pretsv \neq posttsv \| tflag == 1$)* **then**
    $k = (k + 1)\%W$;
    Pick up an element $E_i \in (E - V)$ if available;
    if *($TSV\_count == TSV_{max}$)* **then**
    $tflag = 1$;
    Set $layer\_number(E_i \in (E - V) = 1$;
    if *($length(W_k) == 0$)* **then**
    set $layer\_number(E_i) \in \{2, 3, ..., L\}$;
    else set $layer\_number(E_i) \in \{1, 2, 3, ..., L\}$;
    if *(type_of(E_i)==I/O wrapper cell)* **then**
    begin
      if *($no\_of\_I/O\_wrapper\_cell_k + 1 \leq R_k$)* **then**
      set $r = k$;
      else
      begin
        Find next wrapper chain $W_r (r \neq k)$ that has $No\_of\_I/O\_wrapper\_cell_k + 1 \leq R_k$;
      end
      $No\_of\_I/O\_wrapper\_cell_r+ = 1$;
    end
    if *(type_of(E_i) = scan chain)* **then**
    set $r = k$;
    Add E_i to W_k;
    $No\_of\_elements+ = 1$;
    $pretsv = No\_of\_TSV$;
    $posttsv =$ Required TSV for W_r after insertion of E_i;
    $No\_of\_tsv_r = posttsv$;
    $wrapper\_length(W_r)+ = length(E_i)$;
    $V = V \bigcup E_i$;
    $E = E - E_i$;
    $k = r$;
  end
end

C. Analysis of the Algorithm

The Createchain() algorithm will attempt to place all N wrapper elements in W wrapper chains. According to the algorithm, for every wrapper elements to be placed in

a wrapper chain we have checked whether the wrapper chain satisfies its wrapper I/O cell requirement or not. In the best-case scenario, every time the wrapper I/O cell requirement of the wrapper chain will satisfy and the element is placed in the current chain. Thus the best case complexity of the algorithm is $O(N)$. Hence the Createchain() runs in polynomial time.

If we pick up an I/O wrapper cell and the I/O wrapper cell requirement of the current wrapper chain is not satisfied, then we have to find out the another wrapper chain that satisfies the required condition. In the worst case scenario, we have to search $(W-1)$ wrapper chains to find out the desired wrapper chain where the currently picked wrapper I/O cell has to be assigned. According to the algorithm, for the scan chains no condition checking has been done. Thus for remaining $(N-n)$ elements, which are the wrapper I/O cells, I/O cell requirement condition has to be checked. So the worst case complexity of the algorithm is $O(N+(W-1)*(N-n))$. But the number of elements in a core is much higher than the number of wrapper chains to be designed. Hence $W << N$ and the algorithm has worst case complexity $O(N)$.

In the average case, we assume that on an average half of the wrapper I/O cells satisfy the cell requirement for the current wrapper chain. Therefore on an average $(N-n)/2$ wrapper I/O cells do not satisfy the criteria for the current wrapper chains. Thus in the average case the complexity of the algorithm is $O(N+(N-n)/2*(W-1))$. As said previously $W << N$, the average case complexity can be written as $O(N)$.

V. Illustrative Example

Consider a core with 10 functional inputs (x), 10 functional outputs(y) and 9 internal scan chains(n) of length 8 each. The TAM width is considered 4 and the number of maximum available TSVs are 12. Hence 28 ($x+y+n$) wrapper elements are to be distributed over 4 wrapper chains. According to the algorithm discussed in Section IV we have calculated apriori the required number of I/O wrapper cells for each of the wrapper chains. So, after distribution of all the wrapper I/O cells over 4 wrapper chains, each chain contains 3, 3, 2 and 2 number of wrapper input cells respectively. Similarly, the number of output wrapper cells for 4 wrapper chains are 2, 2, 3, and 3 respectively. During the distribution of wrapper I/O cells over different layers to satisfy TSV constraint, we have checked with the calculated I/O wrapper cells. This is because number of wrapper I/O cells directly affects the number of TSVs. So we have tried to distribute equally the wrapper I/O cells among the wrapper chains.

Now we have two different sets $E=\{E_1,E_2,...,E_{28}\}$ and $W=\{W_1,W_2,W_3,W_4\}$. During execution, the algorithm

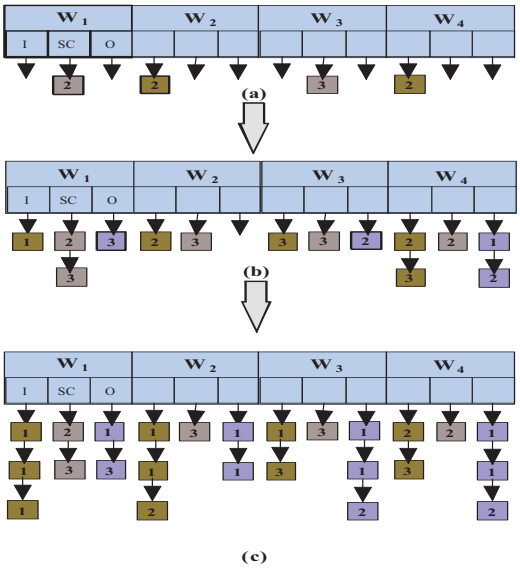

Figure 2. Example of wrapper design

picks randomly different wrapper elements E_i from the set E. Suppose we have chosen the first element E_i of type internal scan chain and it is placed in the layer number 2. It is assigned to the first wrapper chain W_1 as shown in Fig. 2(a). As all of the chip pins are at the lowest layer, it is necessary that all the wrapper chains begin and end at the lowest layer. Hence, after placement of the internal scan chain at layer number 2 for W_1, number of TSVs required is 2. According to the proposed algorithm the next wrapper element that will be picked from E, must be assigned to the next wrapper chain except W_1 that will satisfy the condition of apriori calculated required wrapper I/O cells. Assume that the next wrapper element picked is a wrapper input cell and it is placed in layer number 2. Hence, the wrapper chain number W_2 will be the next wrapper chain where the currently picked input wrapper cell will be assigned. So number of TSVs required for this wrapper W_2 are 2. In the same way, the wrapper elements are assigned to the other two wrapper chains W_3 and W_4. As shown in Fig. 2(a), at this point each wrapper chain contains exactly one wrapper element and total number of TSVs required up to this stage is 8. This is the completion of first cycle of assigning wrapper elements to every wrapper chain. In a clockwise fashion the next element will be assigned to the wrapper chain W_1.

In the next stage, it is assumed that the next wrapper element picked is a wrapper input cell and placed at layer 1 as shown in Fig. 2(b). So no extra TSV is required for the assigned wrapper input cell. According to the algorithm, as the number of TSVs before and after the assignment is same, as shown in Fig. 2(b), the next wrapper element picked is

again added to the same wrapper chain W_1. In this way, we have assigned one internal scan chain and one wrapper output cell both in layer number 3 (as in Fig. 2(b)). Hence, after the second cycle, total number of TSVs required are 12, which is equal to the number of available TSVs. In this stage, as we have reached to the maximum available TSV limit, the remaining wrapper elements are assigned to lowest layer to make the TSV count within limit and final configurations of the wrapper is shown in Fig. 2(c). In our proposed algorithm it is assumed that the TSVs internal to the scan chains are not counted as in [1].

VI. Experimental Results

The proposed algorithm is implemented in C++ language and executed on a Intel Core 2 Duo processor having 1GB RAM. The simulation results are provided based on the cores from ITC'02 SOC test benchmarks. We have shown our experimental results for the core number 7 of d281 and core number 4 and 13 of p93791. Number of layers for the SOC chip considered for each case of the simulation is 3.

A. Results

The simulation results for the core 7 of SOC d281 is presented in Table III for the range of TAM width between 2 to 6. Table IV and Table V represents the results for core 4 and core 13 of SOC p93791 respectively for the range of TAM width between 2 to 8. Column 1 of each table lists the TAM width, Column 2 represents the maximum umber of available TSVs (TSV_{max}). Column 3 shows the total number of TSVs utilized for wrapper design using the proposed algorithm. Length of the shortest and longest wrapper chain is shown in Column number 4 and 6 respectively obtained according to our algorithm. Column number 5 compares with the length of the shortest wrapper length as given in [1]. Column number 7 and 8 compares the CPU running time of the proposed algorithm and the algorithm presented in [1]. It is seen from the tables that our proposed algorithm utilizes the available number TSVs efficiently and the time required for designing the wrapper chain is less than [1] for all of the cases that supports the complexity of our algorithm. It is seen form the Table III, Table IV and Table V that for lower number of wrapper chains some TSVs are still unused. Hence, for lower number of wrapper chains where more number of I/O wrapper cells has to be connected we can design that wrapper chain even with less number of TSVs. Though in [1] authors have proposed polynomial time ($O(N^3)$ and $O(N^2)$) heuristic algorithms to design the wrapper for ITC'02 SOC cores that span over multi-layer, no comparison is made with respect to the longest wrapper chain length obtained by us. This is because all the results presented in [1] are in terms of the shortest wrapper length of the core. Again, it is seen from the Table III, Table IV and Table V that for most of the cases when the number of available TSVs are less the algorithm proposed in [1] is not able to design the wrapper chain which is not the case arise according to our proposed algorithm. So, we can conclude that, the proposed algorithm is better than [1] with respect to the following reasons: (a) Complexity is very less, (b) can design the wrapper for even less number of available TSVs and (c) efficient utilization of the TSVs.

TAM Width	Max TSV	Req TSV	Shortest Wrapper Length	Shortest Wrapper Length [1]	Longest Wrapper Length	CPU Time (Min) (Our)	CPU Time (Min) [1]
2	12	10	969	N/A	1159	0.11	≈ 0.00
	14	10	809	N/A	1319	0.11	≈ 0.00
	16	10	905	1633	1223	0.11	0.22
	18	10	905	1064	1223	0.15	1.42
	22	12	1033	1064	1095	0.13	2.03
3	12	12	626	N/A	784	0.13	≈0.00
	14	14	592	N/A	846	0.11	≈0.00
	16	16	560	1633	814	0.13	0.25
	18	15	656	710	816	0.13	3.67
	22	15	592	710	878	0.13	6.73
4	12	12	468	N/A	629	0.11	≈0.00
	14	14	437	N/A	626	0.1	0.02
	16	16	434	1633	596	0.16	0.30
	18	18	405	532	724	0.11	7.05
	22	22	404	532	789	0.08	16.17
5	12	12	394	N/A	458	0.13	0.02
	14	14	394	N/A	458	0.16	0.05
	16	16	392	1633	458	0.16	0.37
	18	18	362	470	490	.11	11.98
	22	22	330	426	618	0.1	32.28
6	12	12	279	N/A	409	0.1	0.02
	14	14	279	N/A	439	0.1	0.07
	16	16	309	1633	409	0.13	0.47
	18	18	279	459	441	0.13	16.75
	22	22	279	355	473	0.13	53.83

Table III
Results for core 7 of D281

TAM Width	Max TSV	Req TSV	Shortest Wrapper Length	Shortest Wrapper Length [1]	Longest Wrapper Length	CPU Time (Min) (Our)	CPU Time (Min) [1]
2	12	10	72	N/A	81	0.1	≈0.00
	13	11	59	N/A	94	0.13	≈0.00
	14	10	56	89	97	0.1	≈0.00
	18	11	76	77	77	0.15	≈0.00
	20	11	67	77	86	0.15	≈0.00
3	12	12	43	N/A	58	0.11	≈0.00
	13	13	40	N/A	69	0.1	≈ 0.00
	14	14	50	84	52	0.08	≈ 0.00
	18	14	46	51	55	0.15	0.03
	20	14	50	51	52	0.08	0.03
4	12	12	29	N/A	60	0.11	≈ 0.00
	13	13	20	N/A	55	0.1	≈0.00
	14	14	32	79	45	0.1	0.02
	18	18	20	39	56	0.06	0.07
	20	19	25	39	63	0.15	0.08
5	12	12	13	N/A	48	0.1	≈0.00
	13	13	22	N/A	32	0.08	≈0.00
	14	14	13	79	42	0.06	0.02
	18	18	14	37	56	0.06	0.12
	20	20	13	35	51	0.15	0.12
6	12	12	17	N/A	33	0.08	≈0.00
	13	13	17	N/A	37	0.13	≈ 0.00
	14	14	11	79	37	0.1	0.03
	18	18	11	31	38	0.08	0.18
	20	20	16	28	40	0.1	0.32
7	12	12	16	N/A	27	0.15	≈0.00
	13	13	16	N/A	32	0.13	0.02
	14	14	11	79	35	0.11	0.07
	18	18	16	30	35	0.06	0.43
	20	20	15	30	35	0.06	0.87
8	12	12	10	N/A	25	0.1	≈0.00
	13	13	13	N/A	25	0.1	0.08
	14	14	9	79	29	0.11	0.18
	18	18	10	30	40	0.1	1.37
	20	20	10	25	30	0.15	3.32

Table IV
Results for core 4 of p93791

VII. Conclusion

In this work we have proposed an polynomial time ($O(N)$) heuristic algorithm to minimize the test time by reducing the wrapper chain length for 3D core-based SOCs.

TAM Width	Max TSV	Req TSV	Shortest Wrapper Length	Shortest Wrapper Length [1]	Longest Wrapper Length	CPU Time (Min) (Our)	CPU Time (Min) [1]
2	18	11	3712	N/A	5957	0.13	≈0.00
	20	11	4443	N/A	5226	0.13	≈ 0.00
	22	11	4006	4875	5663	0.11	0.03
	24	10	4379	4835	5290	0.11	0.12
	34	10	4044	4835	5625	0.1	0.18
3	18	16	1793	N/A	3943	0.1	0.02
	20	15	2014	3371	4001	0.11	0.05
	22	15	3038	3328	3526	0.15	0.12
	24	15	2805	3257	3762	0.13	0.30
	34	16	2053	3226	4328	0.1	1.02
4	18	18	1885	N/A	3360	0.1	0.05
	20	19	1104	N/A	3273	0.1	0.12
	22	20	1733	2598	3072	0.13	0.25
	24	20	898	2477	3260	0.15	0.70
	34	21	1783	2432	3309	0.11	3.37
5	18	18	1303	N/A	2305	0.11	0.08
	20	20	801	N/A	3371	0.11	0.20
	22	22	812	2520	3596	0.15	0.50
	24	23	1123	2042	3087	0.1	1.10
	34	26	1244	1963	2914	0.1	8.58
6	18	18	840	N/A	2105	0.11	0.15
	20	20	842	N/A	2556	0.13	0.37
	22	22	840	2520	2612	0.1	0.88
	24	24	1092	1820	2107	.11	1.83
	34	30	654	1645	2338	0.1	22.07
7	18	18	837	N/A	1713	0.11	0.42
	20	20	849	N/A	2517	0.1	0.97
	22	22	619	2520	3358	0.1	2.18
	24	24	1011	1820	1712	0.11	4.83
	34	32	459	1426	2128	0.1	89.73
8	18	18	846	N/A	1916	0.1	1.47
	20	20	224	N/A	1895	0.1	3.68
	22	22	628	2520	2354	0.11	8.80
	24	24	834	1820	1677	0.1	20.07
	34	34	604	1242	2354	0.13	535.35

Table V
RESULTS FOR CORE 13 OF P93791

We have distributed the core wrapper elements over different wrapper chains and calculated the required number of TSVs. Our goal was to design the balanced wrapper chains so that the length of the longest wrapper chain is minimum using the available number of TSVs for a particular core. Results are presented on the cores of ITC'02 SOC benchmarks and it shows that the proposed algorithm design the wrapper chain within considerable time which is much less than the algorithm presented in [1] and efficiently utilize the available TSVs. Future work includes the modification of the proposed algorithm to design more balanced wrapper chains so that testing time can be reduced further.

ACKNOWLEDGMENT

Part of this work is sponsored by the Department of Science & Technology, Govt. of West Bengal, India, under the Grant of VLSI Design Project.

REFERENCES

[1] 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.

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

[4] K. Banerjee, S. J. Souri, P. Kapur, and K. C. Saraswat, "3-D ICS: A Novel Chip Design for Improving deep-submicrometer Interconnect Performance and System-on-Chip Integration," *Proceedings of the IEEE*, vol. 89, no. 5, pp. 602–633, May 2001.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] D. L. Lewis and H.-H. S. Lee, "A scan-Island Based Design Enabling Prebond Testability in Die-Stacked Microprocessors," in *Proc. International Test Conference*, 2007, pp. 1–8.

[12] 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.

[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] L. Jiang, Q. Xu, K. Chakrabarty, and T. Mak, "Layout-driven Test-architecture Design and Optimization for 3D SoCs Under Pre-bond Test-pin-count Constraint," in *IEEE International Conference on Computer Design*, 2009, pp. 191–196.

[15] B. Noia, K. Chakrabarty, , and E. J. Marinissen, "Optimization Methods for Post-Bond Die-Internal/External Testing in 3D Stacked ICs," in *Proc. International Test Conference*, 2010.

[16] S. K. Roy, S. Ghosh, H. Rahaman, and C. Giri, "Test Wrapper Design for 3D System-on-Chip Using Optimized Number of TSVs," in *IEEE Intl. Symposium on Electronic System Design*, Orissa, Bhubaneswar, India, December 2010.