Hyper-Graph Based Partitioning to Reduce DFT Cost for Pre-Bond 3D-IC Testing

Amit Kumar Sudhakar M. Reddy
Electrical and Computer Eng. Dept.
University of Iowa
Iowa City, IA 52242, USA

Irith Pomeranz
School of Electrical and Computer Eng.
Purdue University
W. Lafayette, IN, 47907, USA

Bernd Becker
Institute for Computer Sci.
Albert Ludwigs University
Freiburg, 79110, Germany

***Abstract*— 3D IC technology has demonstrated significant performance and power gains over 2D. However, for technology to be viable yield should be increased. Testing a complete 3D IC after stacking leads to an exponential decay in yield. Pre-bond tests are required to insure correct functionality of the die. In this work we propose a hypergraph based biased netlist partitioning scheme scheme for pre-bond testing of individual dies to reduce extra-hardware (flip-flops) required. Further reduction in hardware is achieved by a logic cone based flip-flop sharing scheme. Simulation results on ISCAS89 benchmark circuits and several industrial benchmarks demonstrate the effectiveness of the proposed approach.**

I. Introduction

Through Silicon Via (TSV) based 3D-IC's have several advantages over traditional two dimensional IC's [17], [6], [1]. Fig. 1 shows a typical 3D IC. These advantages can be summarized as:

1) Footprint for the 3D IC is small in comparison to their two dimensional counterparts, resulting in more compact chips.
2) Interconnect length can be drastically reduced leading to better timing performance.
3) Less power consumption is a direct implication of shorter wire length and more functionality packed on the chip itself.
4) Various die layers can be made using different processes, for example in a 3D microprocessor caches can be made using a different technology than the rest of the chip.

However 3D-IC technology is still in its infancy and faces many problems [11], [13], [1]

1) Yield for 3D IC's is very low, resulting in increased cost.
2) There is lack of 3D specific CAD tools. Mostly 2D tools are modified to work on 3D IC designs. Available 3D IC automation is for backend.
3) It is difficult to test a die independently, costly approach of probing is required.
4) Thermal management and heat dissipation is a challenging issue in 3D IC's.

Yield for 3D-IC's is comparatively lower than 2D processes. Main reason for low yield is lack of schemes to test individual dies before bonding. Pre-bond test is therefore important for improving yield. In [8], [9] authors use scan-island based approach for pre-bond testing of dies. However, no CAD algorithm for partitioning was presented. Various circuit blocks were placed on different layers of the design. The above approach has the following drawbacks:

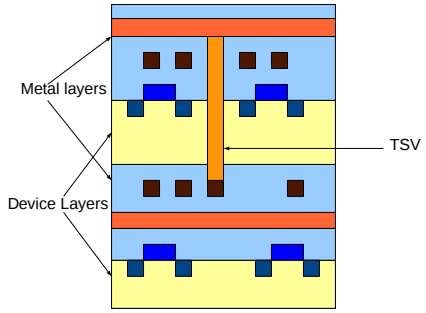

Fig. 1. A TSV based 3D IC.

1) Block based partitioning can result in excessive die area. Various blocks in the circuit can have varying size and architectural partitioning may lead to improper area usage.
2) No formal approach was presented for circuit partitioning [8] to reduce number of inter-die interconnects, which may lead to increased TSV numbers. Since yield for TSV is low and area requirements are very high [17], number of TSV's should be minimized.
3) Large number of additional flip-flops may be required for the incoming and outgoing connections to a die from other dies of the circuit.

In this work we propose a DFT method for pre-bond testing of 3D ICs using a hypergraph [4] based netlist partitioning scheme. Hypergraph based netlist partitioning is a method that has been used for various applications in VLSI design, for example design packaging, HDL synthesis, design optimization and design partitioning [4], [5]. In general the approaches have to be adapted to the problem at hand to work in a sufficiently efficient and effective way. Here, this is done for the first time in the context of partitioning a design to facilitate pre-bond testing of 3D-ICs. We show the efficiency of our approach by experimental evaluations and comparison to the

978-3-9810801-7-9/DATE11/©2011 EDAA

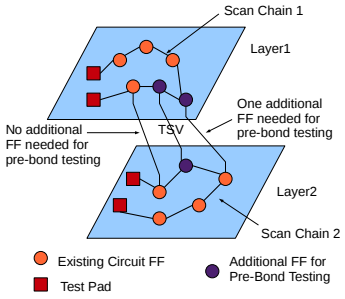

Fig. 2. Scan island based approach for pre-bond testing.

results of applying an earlier partitioning procedure [4]. The main features and advantages of the proposed scheme are summarized below:

1) Proposed partitioning scheme leads to equal cluster/die sizes. All dies have approximately same amount of logic, resulting in saving in terms of area.
2) Partitioning algorithm assigns weights on hyperedges such that cut obtained is biased towards placing a flip-flop at cut, resulting in savings of additional flip-flops required at each connection coming in/out of the die from another die.
3) Further saving of hardware is achieved by sharing flip-flops [3] at the incoming and outgoing connections of a die.

In section II we give the motivation for this work followed by basic definitions. Basic concepts related to hypergraph based netlist partitioning are presented in Section III. The proposed hypergraph based biased netlist partitioning algorithm is presented in Section IV. Section V deals with the flip-flop sharing scheme. In section VI we discuss experimental results obtained. We conclude the paper is section VII.

II. Motivation

Testing of 3D-IC's poses similar kind of problems as of MCM's [10], [9]. Present strategy for testing is to bond and test, resulting in loss of yield. 2D test techniques can be applied for testing after bonding. Pre-bond test needs some tailor-made 3D specific solutions [16], [13], [7]. Major differences between 2D and 3D IC testing are:

1) Primary inputs have full controllability and primary outputs have full observability in 2D IC testing. This is not true for inputs and outputs coming from another module for pre-bonded 3D-IC die.
2) Only method to access pre-bond 3D-IC is using probing. Probing is a costly technique. We are required to reduce the number of probe points which are very expensive in terms of area [11].

Faceside probing is preferred as it does not require wafer to be thinned [9]. We name input and output connections coming into and going out of a die as *pseudo-primary input* and *pseudo-primary output* respectively, in this work.

DEC APLHA 21364 [2] uses a segmentation technique in which the circuit was divided into individual segments with registers at the border. During test mode registers isolate each segment, in normal mode they allow free movement of data from one segment to another. In [2], the entire circuit netlist is partitioned into several smaller circuit blocks called scan islands. Each island houses parallel local scan-chains and can be controlled and observed independently during test mode. Similar architecture for pre-bond 3D IC testing has been proposed in [9]. Fig. 2 shows a scan-island based test architecture, layer1 and layer2 form two different scan-islands (on different dies) and both can be tested independently of each other. Extra flip-flops are added at all the incoming and out-going connections of a die from another die, if a flip-flop is not present on the interconnection. All flip-flops on a die are stitched together to form a scan-chain. Hypergraph based biased netlist partitioning and flip-flop sharing scheme's proposed in this work, use scan-island based architecture. In pre-bond test individual dies forms independent scan-islands. The advantages of scan-island architecture are:

- Pseudo-primary inputs are completely controllable.
- Pseudo-primary outputs are observable.
- Number of test pads required are minimal.

III. Basic Concepts Related to Hypergraph Based Netlist Portitioning

In this section we present some basic concepts and definitions used in this work.

A *Hypergrah* $H = (V, E)$ is a set of vertices V and a set of hyperedges E, where each hyperedge is a subset of the vertex set V [4], [5]. E is the set of m *hyperedges* which are used to represent the nets of the circuit. The set E is a subset of the power set of 2^V of the vertices V in H. A weighted hypergraph has non-negative numeric weights associated with each vertex, each hyperedge, or both. Conventional graphs are a special case of hypergraphs, where all edges are defined by subsets of vertices of size two. Fig. 3 shows a circuit and its corresponding hypergraph.

Hierarchical Clustering: Let

$$C_i : V \longrightarrow 2^V, \quad i = 1, \ldots, l$$

be a family of l mappings where C_i maps the vertices of H into $|V_i|$ clusters $C_{i1}, C_{i2}, \ldots, C_{i|V_i|}$ such that $C_{iu} \cap C_{iv} = \phi$ if $u \neq v$ and $\bigcup_{j=1}^{|Vi|} C_{ij} = V$. The family of mappings mentioned above defines the hierarchical clustering [14] of vertices in H whenever $|V_i| > |V_{i+1}|$ for all $i = 1, \ldots, l-1$.

Coarsened Hypergraph: A coarsened hypergraph [14] $H_i = (V_i, E_i)$ is the set of vertices V_i and hyperedges E_i such that: 1) $v \in V_i$ is a cluster (subset) C_{ij} of vertices from V as defined by the mapping C_i (Note: to simplify notation, v may also be denoted by the corresponding cluster C_{ij}) and 2) a subset e of V_i is a hyperedge of H_i if and only if there exists a hyperedge $e' \in E$ such that e' has nonempty intersection with each cluster represented by the vertices in e.

k-way hypergrah partitioning: A *k-way partitioning* $P^k = \{C_1, C_2, \ldots, C_k\}$ consists of k clusters (subsets of V), $C_1, C_2, \ldots, C_k$ such that $C_1 \cup C_2 \cup \ldots C_k = V$ and $C_i \cap C_j = \phi$

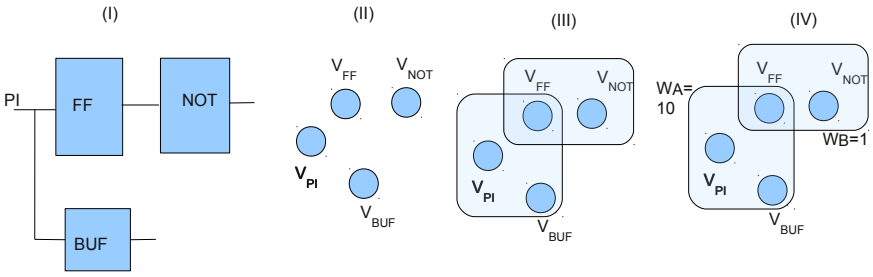

Fig. 3. Transformation of a circuit G in to hypergraph H. (i)Circuit G. (ii)Vertices of hypergraph H. (iii) Hyperedges added to H. (iv) Weight added to hyperedges.

for $1 \leq i \leq k$ and $1 \leq j \leq k$. if k= 2 we refer to P^2 as bipartitioning.

Cutsize or Cost of Cut : The set of hyperedges *cut* by cluster C is given by $E(C)$=$e \in E$ $s.t.$ $0 < |e \cap C| < |e|$, i.e., $e \in E(C)$ if at least one, but not all, of pins of e are in C. The set of nets cut by a partitioning solution P^k can be expressed as $E(P^k)$ = $\bigcup_{h=1}^{k} E(C_i)$ or equivalently $E(P^k)$= $\{e \in E | \exists u, v \in e, h \neq l$ $with$ $u \in C_h$ and $v \in C_l\}$. We say that $|E(P^k)|$ is the *cost* or size of *cut* P^k.

With balance constraints the problem of *k-way hypergraph partitioning* is known to be NP complete [4], [5]. The proof of NP completeness can be derived from mapping the problem instance to subset-sum problem [15]. We propose a partitioning algorithm that attains the dual goal of reducing cut size and biasing the cut, such that majority of nets on the cut are driven by a flip-flop. Any hyperedge e on cut maps into a costly TSV in 3D IC. In this work we focus on bi-partitions, i.e. division of V into two partitions (dies) V_1 and V_2 such that $V_1 \cup V_2 = V$, $V_1 \cap V_2 = \phi$ and $|V_1| \approx |V_2|$.

IV. K-Way Hypergraph Based Biased Netlist Partitioning Algorithm

We now present hypergraph based biased netlist partitioning algorithm in this section. The proposed method is used for partitioning a design across k dies (partitions). For the sake of simplicity we discuss in detail two way partitoning in this work.

A. Algorithm Description

The algorithm can be broadly divided into four parts, transforming circuit netlist to a weighted hypergraph, merging phase followed by a random bisection phase. Later an uncoarsening and a refinement phase are run.

1) Hypergrah Construction from Netlist and Selective Weight Assignment: Hypergraph [4], [5], [14] construction for a circuit netlist is discussed below . An integer index is allocated to each node v and edge e in hypergraph H.

Consider a circuit netlist G with n cells (combinational-gates /buffers /primary-inputs /flip-flops). Allocate an index to every cell G=$(g_1, g_2, \ldots, g_n)$. Every net of circuit G originates at a cell g_i and feeds a given number of say p cells.

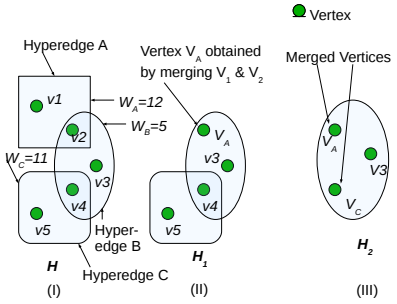

Fig. 4. An illustration of heavy edge merging in hypergraph.

1) For every cell g_i in G there exists a vertex v_i in the hypergraph H. Fig 3(ii) shows the vertices of hypergraph H corresponding to circuit G in Fig 3(i).
2) Every Hyperedge in H corresponds to a net in circuit G. Fig 3(iii) illustrates hypergraph H with hyperedges added. Each hyperedge e_i, corresponds to a net, fed by cell g_i. Index i for hyperedge in e_i in H is the same as cell g_i.
3) A weight of 1 is assigned to hyperedge h_i if g_i is driven by a flip-flop, else, h_i is assigned a weight of $w > 1$. Fig 3(iv) shows weight assignments to hyperedges of H. In our implementation we used $w = 10$.

2) Merging Phase: The initial merging phase is based on hierarchal clustering [4]. Vertices connected with heavier hyperedges are merged first. We randomly select hyperedges for merging if more that one hyperedge has the same weight. Fig. 4 shows an example of heavy hyper-edge based merging scheme. Since hyperedges not driven by a flip-flop are heavier, they have a higher probability of getting merged. Heavy edge merging increases the possibility of a biased bipartition, with more flip-flops on the cut.

Example: Consider a hypergraph H(V, E), shown in Fig. 4(i), with 5 vertices V=$(v_1, v_2, v_3, \ldots, v_5)$ and 3 hyperedges E=(A, B, C), with weights w_A=12, w_B=5, w_C=11, respectively. Heavy hyperedge merging algorithm will merge vertices v_1, v_2 corresponding to hyperedge A (heaviest hyperedge) into vertex v_A, as shown in Fig. 4(ii) to form hypergraph H_1. In the next step vertices's v_4, v_5 corresponding to hyperedge C with weight 11 are merged to form vertex v_C, as illustrated in Fig. 4(iii) to form hypergraph H_2. After merging k

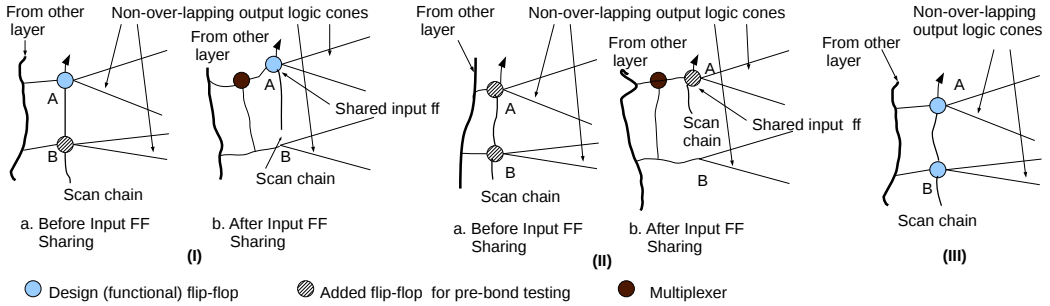

Fig. 5. Various cases in flipflop sharing for two structurally input independent pseudo-primary inputs: (i) Ff sharing with one a flip-flop on cut (ii) Flip-flop sharing with no design flip-flops on cut (iii) No flip-flop sharing required if two design-flops are present on cut.

hyperedges original hypergraph H (corresponding to netlist G) is reduced to a smaller hypergraph H_k. A bisection of the merged hypergraph is created and the solutions are refined using Kernighan-Lin algorithm. Kernighan-Lin (KL) algorithm [4] takes a greedy approach to interchange elements of two partitions to find a better solution i.e. reduce cut size. KL algorithm are known to be quite effective in refining solution of hyperedges with small number of vertices and inefficient for large number of vertices.

3) Random Bisection: A bisection of the hypergraph H_k is obtained for a coarser hypergraph H_k with approximately 200 nodes, such that it has a small cut size, and divides the vertices V into two equal parts [14]. Since this hypergraph has a very small number of vertices, the time to find a partition (of netlist G) using any of the heuristic algorithms tends to be small. Note that it is not useful to find an optimal partition of this hypergraph H_k, as the initial partition will be substantially modified during the refinement phase [5]. In our procedure we derived several cuts and only the best one was further processed.

4) Desplitting and Refinement: During the desplitting phase, we start with merged vertices of the coarser hypergraph H_k (obtained from merging k hyperedges of original hypergraph H). Merged vertices are uncoarsened (un-merged) again to form original hypergraph H (the graph before merging phase) in k successive steps. Only one hyperedge is unmerged in one step. Cut size of merged hypergraph H_k remains same as that of un-merged (original) hypergraph H i.e. circuit netlist G with partitions. All incoming hyperedges to merged vertices were connected to final merged vertex during the merging phase, resulting in the same cut size of hypergraph H_k and H. Cut sizes of intermediate hypergraphs $H_{m-1}, \ldots, H$ (obtained during various steps in demerging phase) are further reduced by use of the Fiduccia-Mattheyses (FM) algorithm [4], [5]. The FM algorithm makes linear time passes to iteratively improve cut sizes by moving each vertex exactly once. FM works by prioritizing moves by gain. A move changes to which partition a particular vertex belongs and the gain is the corresponding change to the cut size. After each vertex is moved, gains for connected modules are updated.

B. Complexity Analysis for Partitioning Algorithm

We take an incremental approach for complexity analysis of hypergrah partitioning algorithm discussed above. Merging phase consists of merging vertices with heaviest hyperedges, which can be done in $O(V)$ times. Heaviest hyperedge can be found in $O(log(E))$. Overall complexity of merging operation is $O(Vlog(E))$. Random Bisection of a merged hypergaph is an $O(V)$ operation. Desplitting and Refinement is the last step. Desplitting takes same time as that for merging. However, refining the partitions created using FM algorithm takes $O(V^2)$. Therefore the overall complexity of the algorithm is of the order of $O(V^2)$.

V. Flip-flop Sharing to Reduce Additional Hardware Requirement

Prime objective of flip-flop sharing scheme is to reduce additional hardware requirements for pre-bond IC testing, without reducing fault coverage. Flip-flop sharing [3] allows two pseudo-outputs or pseudo-inputs to share flip-flops, thus saving on additional scan flip-flops needed for scan-island architecture. Sharing scheme is based on logic cones. In this section, a brief explanation of our proposed flip-flop sharing methodology is presented. In the latter part of this section we present an approximation algorithm[15] for flip-flop sharing.

A. Flip-flop Sharing at Pseudo-Primary Inputs

Structurally Independent Inputs: If there is a path from node a to b then node b is said to be in the output cone of a. Two input nodes a and b are said to be structurally independent if output cones of a and b do not intersect. If two circuit inputs are structurally independent any test vector for a combinational fault does not require to simultaneously satisfy constraints on inputs a and b. Structural independence at inputs allows us to combine inputs a and b with no loss of required controllability. In case inputs a and b are driven by a scan-flop, a single $scan\text{-}flop$ can be $shared$ between both inputs a and b, with no loss in controlabillity and fault-coverage. An extra mux is required for sharing. Our goal is to identify optimal groups of pseudo-primary inputs whose output cones do not overlap. All pseudo-primary inputs with non-overlapping outputs can be connected to the same flip-flop (scan-cell) output.

Example: Referring to Fig 5 (i)*a*, without flip-flop sharing, an additional flip-flop is required at input B for pre-bond scan-

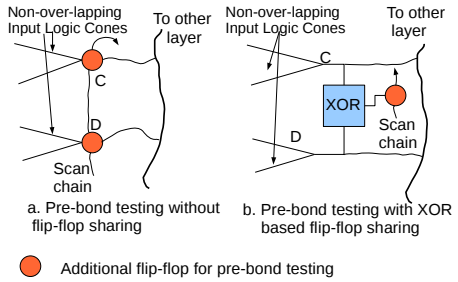

Fig. 6. Flipflop sharing at pseudo-primary outputs.

island based testing. Pseudo-primary input A already has a design flop on cut. Since, inputs A and B have non-overlapping output cones of influence, flip-flops at the input of A and B can be shared. A is a design flip-flop therefore, additional flip-flop for pre-bond testing is not needed at pseudo-primary input B. B can share the flip-flop present at pseudo-primary input A as shown in Fig.5(i) b. Fig. 5(ii) illustrates the case when no design flip-flop is present at both A and B. In Fig. 5(iii) no flip-flop sharing is required as both flip-flops at pseudo-primary inputs A and B are design flip-flops. A and B are flip-flops are required for functional operation of the circuit.

B. Flip-flop Sharing at Pseudo-Primary Outputs

Structurally Independent Outputs: If there is a path from node a to b then node a is said to be in the input cone of b. Two output nodes a and b are said to be observability independent if input cones of a and b do not intersect. In case of pseudo-primary outputs that do not have a flip-flop on the cut we need to add an extra flip-flop as shown in Fig. 6(a). A hardware efficient approach is to add an XOR gate for pairs of two structurally independent outputs, followed by an additional flip-flop as shown in Fig. 6(b). A 2 input XOR guarantees detection of a single fault on its inputs [12]. Assuming single fault model, a 2 input XOR guarantees observability of a fault present in the non-overlapping input cones of a pseudo-primary output. If p is the number of pairs of structurally independent outputs using 2-input XOR reduces additional flip-flop count by p. We can also use an XOR tree connected to $p > 2$ independent outputs to further reduce the need for additional scan-cells.

C. Approximation Algorithm for Flip-flop Sharing at Pseudo-Primary Inputs

Let every pseudo-primary input $I = (i_1, \ldots, i_n)$ be a vertice in $G(I, E)$. An edge e exists between vertices if their output cones do not overlap. Problem of flip-flop sharing reduces to finding cliques in the graph. Flip flops, already present on the cut are first processed. Nodes in a clique can share a flip-flop. An overview of steps in the algorithm $FF\_IP\_SHARE\_APPROX(G)$ are given below.

1) Consider the sub set $I = (i_1, \ldots, i_f)$ of pseudo-primary inputs of graph, with flip-flops on the cut.
2) Find maximal clique at each node i_k such that i_k is a node in the clique. Finding maximal clique, with a given node i_k included in the clique has a complexity of $O(n^2)$, where n is the number of nodes in graph G.
3) Remaining m flip-flops: Divide the vertices of G into $l = m/log(m)$ blocks each containing $log(m)$ vertices (where m is the number of remaining vertices in the graph).
4) For each block try all possible subsets of the vertices's to see if it forms a clique. If it does output it and stop.

$FF\_IP\_SHARE\_APPROX(G)$ has a complexity of $O(n^3)$. A similar procedure can be used to group structurally independent pseudo outputs to feed into XOR tree.

VI. Experimental Results and Discussions

A comprehensive set of experiments were performed on a linux workstation with Intel Xeon $2GHZ$, 8 core processor and 16GB RAM. Experiments were performed on various academic benchmarks and industrial circuits provided by NXP. The proposed partitioning approach creates approximately equally sized partitions with small cut size and large percentage nets on cut were driven by a flip-flop. This leads to low TSV count and less number of additional flip-flops required for pre-bond testing. Hypergraph is constructed directly from the structural circuit. In this work we partition the circuit into two parts, however k-way partitioning can be performed by re-partitioning of already obtained partitions as discussed in [4]. A weight is assigned to each node in the hypergrah. Weights to hyperedges are assigned according to the scheme presented in Section III, a lower weight to hyperedges driven by a flip-flop and a very large weight to other hyperedges.

We define weight ratio, w as the ratio of the weight assigned to nets driven by gates/buffers/primary-inputs and the weight assigned to edges driven by a flip-flop.. After exhaustive experimentation we obsereved that use of $w = 10$ lead to a reduced cut size with large number of flip-flops on the cut as desired. A very high value of w resulted in an increased cut size (thus more TSV's) and low value of w resulted in reduced flip-flop count on cut. The cut size and number of flip-flops on cut, using the proposed scheme, are reported for two values of weight ratio (w) 1 and 10, respectively in Table I. We also compare our results with that by $hmetis$ [4], [5], a hypergraph partitioning package available at http://glaros.dtc.umn.edu/gkhome/metis/hmetis/overview. The allowed imbalance between partitions generated using hmetis was set to 1% and is the same as that used in the proposed procedure.

The proposed netlist partitioning scheme produced high quality bi-partitions, typically more than 80% of nets on the cut were driven by flip-flops. Considering circuit Indus-3, for $w = 10$, cut size=223 and the number of flip-flops on cut=214, i.e. approximately 96% of nets on the cut were driven by design flip-flops. However, using hmetis, a cut size of 273 and 22 flip-flops on the cut were obtained. Compared to hmetis the proposed hypergraph based biased netlist partioning scheme with $w = 10$ resulted in smaller cut sizes in 7 out of 11 circuits. Additionally, as desired, considerably higher portion of the cut by the proposed method includes flip-

TABLE I

CIRCUIT CUT SIZE AND FLIP-FLOPS ON CUT.

Circuit	Size	Proposed Approach					*hmetis*		
		Cut size ($w = 10$)	FFs on Cut ($w = 10$)	Run Time ($w = 10$)	Cut Size ($w = 1$)	FFs on Cut ($w = 1$)	Cut Size	FFs on Cut	Run Time
Indus-1	45K	18	0	6.9s	18	0	39	6	3.3s
Indus-2	141K	457	218	18.2s	477	6	512	7	14.2s
Indus-3	240K	223	214	32.1s	332	29	273	22	27.5s
Indus-4	1000K	198	184	265.9s	206	102	191	96	214s
Indus-5	900K	294	243	197.2s	381	48	306	89	155.8s
Indus-6	3000K	158	62	1035.3s	164	0	154	0	837.3s
s38417	38K	72	59	1.1s	81	47	76	48	1.7s
s15850	15K	29	22	0.4s	44	12	50	16	0.4s
b17	22K	5	1	2.6s	5	1	6	1	2.1s
b18	50K	50	26	10.9s	58	5	44	0	9.5s
b12	12K	39	31	0.1s	46	13	34	19	0.1s

flops for all circuits. Table I shows that runtime for the proposed procedure increases almost linearly with circuit size. The proposed scheme was in general fast and a 3 million gate circuit took approximately only 17 minutes to partition. Runtimes for hmetis are shown in the last column of Table I.

In Table II, results are tabulated for the number of flip-flops that can be reduced by logic cone based flip-flop sharing schemes for various circuits. We observed large number of flip-flops can be shared if the circuit size is large and fewer number of design (functional) flip-flops are present on the cut. Above mentioned trend can be attributed to increased probability of finding non-overlapping logic cones and larger number of candidate flip-flops available for flip-flop sharing. Each shared flip-flop reduces additional flip-flop requirement for pre-bond test by one.

TABLE II

FLIP-FLOPS ON CUT AND NUMBER OF SHARED FLIP-FLOPS.

Circuit	Circuit Size	Cut Size	Num of Shared ffs
Indus-1	45K	18	4
Indus-2	141K	457	39
Indus-3	240K	223	1
Indus-5	900K	294	13
s38417	38K	72	0
s15850	15K	39	1

VII. CONCLUSION

In this paper, a pre-bond 3D IC testing scheme is proposed. This technique uses hypergraph based biased netlist partitioning and flip-flop sharing. Biased netlist partitioning attains a dual goal of keeping cut size low with large percentage of nets on the cut were driven by flip-flops, thus reducing hardware. Logic cone based flip-flop sharing approach allows controllability and observability of pseudo-primary input/output without adding any additional flip-flops. Experimental results shows a significant reduction in the hardware required for scan-island based pre-bond testing.

VIII. ACKNOWLEDGEMENT

Part of the work of AK and SMR was done at the University of Freiburg, Germany, during a stay supported by Von Humboldt Foundation fellowships.

REFERENCES

[1] S. M. Alam, M. Ignatowski, and Y. Xie. Technology, cad tools, and designs for emerging 3d integration technology. *GLS VLSI*, pages 1–2, 2008.

[2] D.K. Bhavsar and R.A. Davies. Scan islands - a scan partitioning architecture and its implementation on the alpha 21364 processor. *VTS*, pages 16 – 21, 2002.

[3] S.-C. Chang, K.-J. Lee, Z.-Z. Wu, and W.-B. Jone. Reducing test application time by scan flip-flops sharing. *Computers and Digital Techniques, IEE Proceedings* -, 147(1):42 –48, jan. 2000.

[4] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar. Multilevel hypergraph partitioning: applications in vlsi domain. *IEEE Trans. VLSI Syst.*, 7(1):69–79, 1999.

[5] G. Karypis and V. Kumar. Multilevel k -way hypergraph partitioning. *DAC*, pages 343–348, 1999.

[6] D. S. Kung and Y. Xie. Introduction: Opportunities and challenges of 3d integration. *IEEE Design & Test of Computers*, 26(5):4–5, 2009.

[7] H. S. Lee and K. Chakrabarty. Test challenges for 3d integrated circuits. *IEEE Design & Test of Computers*, 26(5):26–35, 2009.

[8] D. L. Lewis and H.S. Lee. Testing circuit-partitioned 3d ic designs. In *ISVLSI*, pages 139–144, 2009.

[9] D.L. Lewis and H.S. Lee. A scan island based design enabling prebond testability in die-stacked microprocessors. *ITC*, pages 1 –8, oct. 2007.

[10] E. J. Marinissen. Testing tsv-based three-dimensional stacked ics. *DATE*, 2010.

[11] E. J. Marinissen, D. Y. Lee, J. P. Hayes, C. Sellathamby, B. Moore, S. Slupsky, and L. Pujol. Contactless testing: Possibility or pipe-dream? *DATE*, pages 676–681, 2009.

[12] S. Mitra and K. Sup Kim. X-compact: an efficient response compaction technique. *IEEE Trans. on CAD*, 23(3):421–432, 2004.

[13] B. Noia, S. K. Goel, K. Chakrabarty, E. J. Marinissen, and J. Verbree. Test-architecture optimization for tsv-based 3d stacked ics. *ETS*, pages 24–29, 2010.

[14] M. Ouyang, M. Toulouse, K. Thulasiraman, F. Glover, and J.S. Deogun. Multilevel cooperative search for the circuit/hypergraph partitioning problem. *IEEE Trans. on CAD*, 21(6):685 –693, jun. 2002.

[15] V. Vazirani. Approximation Algorithms. Springer, Germany, 2003.

[16] X. Wu, P. Falkenstern, K. Chakrabarty, and Y. Xie. Scan-chain design and optimization for three-dimensional integrated circuits. *JETC*, 5(2), 2009.

[17] J. Yang, K. Athikulwongse, Y. J. Lee, S. K. Lim, and D.Z. Pan. Tsv stress aware timing analysis with applications to 3d-ic layout optimization. *DAC*, pages 803 –806, jun. 2010.