TSV and DFT Cost Aware Circuit Partitioning for 3D-SOCs

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-SOC technology has significant performance and power gains over 2D as interconnects can be shortened significantly. To accrue full benefits of reduced interconnect lengths large designs need to be partitioned into several dies.**

In this work we propose a hypergraph based multi-objective circuit partitioning scheme for 3D-SOCs that simultaneously reduces the number of inter die connections, which use through silicon vias (TSVs), and reduces additional DFT logic needed for pre-bond test of dies. An Ordered Block hypergraph partitioning scheme is proposed to achieve these objectives. Experimental results on several industrial circuits demonstrate the effectiveness of the proposed approach.

I. Introduction

3D-SOCs are being considered to achieve higher levels of integration in electronic systems. 3D-SOCs have the following advantages over the currently used 2D-SOCs [8], [9]

1) Shorter interconnect lengths in 3D-Socs reduce signal propagation delays and power consumption.
2) Stacking dies allows mixing of heterogeneous technologies and increasing packaging densities.

In order to fully benefit from the above advantages of 3D-SOCs large designs should be partitioned [5], [9], [17]. However circuit partitioning methods for 2D-SOCs using min-cut k-way partitioning [6] to reduce the number of connections between blocks of the partition cannot be directly used for the following reasons. One reason is that connections between nodes in different dies of 3D-SOCs are made using through silicon vias (TSVs) [1], [4], [16]. The number of TSVs needed to make one connection between two dies with i intervening layers (dies) is (i+1). The number of intervening layers between any two dies is unknown prior to partitioning of a design and mapping the dies to the SOC layers. Thus, circuit partitioning procedures for 3D-SOCs must take into consideration variable TSV costs of inter-die connections. This requirement is illustrated in Fig. 1 using three dies. In Fig. 1 (a) four connections A, B, C and D are illustrated where connections A and B are between dies 1 and 3, connection C is between dies 1 and 2 and connection D is between dies 2 and 3. A total of six TSVs are needed since connections A and B each require 2 TSVs. Fig. 1(b) illustrates a different order of stacking the same dies. By changing the SOC layers to which dies 2 and 3 are mapped, the number of TSVs needed for the four connections in Fig. 1(b) is reduced to 5. Placement

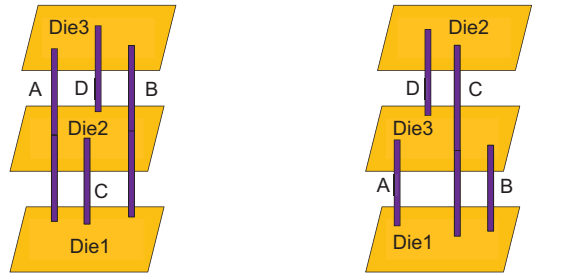

Fig. 1. Different number of TSVs for inter-die connections.

of TSVs may be confined to pre-determined areas of dies to accommodate landing of tester probes [14]. Even in this case minimizing the cost of TSVs is important.

The second reason for the need for 3D-SOC-specific circuit partitioning procedures is the additional design for test (DFT) logic needed for a given partition as discussed next. Yields of 3D-SOCs can be unacceptably low if they are tested after bonding the dies. For this reason pre-bond test of individual dies is necessary. Test of individual dies requires that the inputs to the dies from other dies must be controllable and the die outputs driving connections to the other dies must be observable. Digital circuits are universally tested using scan chains [9], [10], [11], [18]. To facilitate pre-bond test of dies of a 3D-SOC, additional scan cells may be needed to control and observe inter-die connections [10]. For example if a node in a die, say D, not driven by an existing circuit flip-flop drives a connection to another die then an additional scan cell must be added at this node to observe it when testing die D. This is illustrated by the two die example shown in Fig. 2. We assume that each die uses a single scan chain. Connection A does not impose the requirement for any additional scan cell since it is between flip-flops existing in the given circuit. Connection B requires one additional scan cell since the node in die 1 is not a flip-flop in the circuit. Connection C requires two additional scan cells one each in dies 1 and 2 as the connection is between two nodes neither of them being an existing flip-flop.

Another approach for pre-bond testing is based on P-1500 test architecture [2], [11], [12], [13], [14]. Additional wrapper cells are added, instead of scan-cells to provide testability.

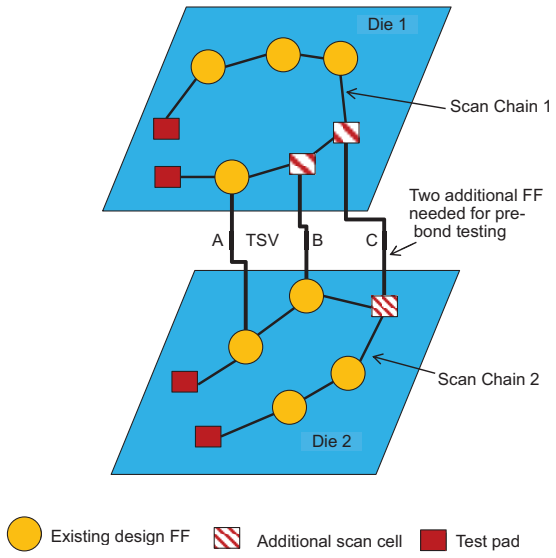

Fig. 2. Additional scan cells for pre-bond testing.

These additional wrapper cells are attached together to form a P1500 complaint wrapper. A P1500 [3], [15] wrapper cell can operate as a flip-flop in functional mode. Any design flip-flop on inter-die connection becomes redundant and can be removed (resulting in hardware saving) if a wrapper cell is added for pre-bond test.

Existing procedures for 3D-SOC circuit partitioning use two-step procedures [5]. In the first step a min-cut k-way partitioning procedure using a hypergraph circuit model [5], [6] is used. In the second step, this is followed by improving the number of TSVs by random and/or directed exchanges of circuit elements between dies. These approaches yield sub-optimal results as we demonstrate in the experimental results section. Recently the use of differential weights of hyperedges of the hypergraph model of circuits was investigated to reduce the number of additional scan cells for pre-bond test of dies [7]. Results were obtained only for the case of two dies for which the differential TSV cost is not an issue. We also use differential hyperedge weights in this work. Another approach for reduction in DFT cost for pre-bond testing is based on logical correlations between inter-die connections [11]. However, TSV cost and cell areas were ignored and results were presented for the case of two dies only [11].

In this work we propose a 3D-SOC circuit partitioning procedure that simultaneously reduces the number of TSVs and additional scan cells. We use dynamically changing weights of hyperedges of a hypergraph model of the circuit to implicitly account for variable TSV cost of inter die connections. Experimental results on several industrial designs show that the proposed partitioning procedure yields significant reductions in the numbers of TSVs as well as requiring a much smaller number of additional scan cells for pre-bond test of dies.

The remainder of the paper is organized as follows. In Section II we review hyergraph model based partitioning of circuits and discuss the requirements on circuit partitioning for 3D-SOCs. We also give a brief review of earlier related works and an overview of the proposed partitioning procedure. In Section III we give details of the proposed partitioning procedure and experimental results are given in Section IV. The paper is concluded in section V.

II. Preliminaries

In this section we review requirements on circuit partitioning for 3D-SOCs. Next we present basics of circuit partitioning using hypergraph models of circuits together with an overview of the proposed procedure discussed in detail in the next section. In Section II(C) we discuss earlier works.

A. Circuit partitioning for 3D-SOCs

Circuit partitioning for 3D-SOCs must achieve similar objectives to that for 2D-SOCs with additional requirements. These are listed below.

1) Partition the circuit into dies of essentially equal areas. Die area includes that for gates and flip-flops of the circuit as well as for routing. Since circuit partitioning is done from net lists, die areas for gates and flip-flops are known but routing area typically needs to be estimated using some heuristics. In this work we use only the area for gates and flip-flops. Additionally area for inter-die connections using TSVs is taken in to account.
2) Minimize the number of TSVs needed for inter-die connections. This is a fundamental additional requirement for 3D-SOC partitions.
3) Minimize the need for additional scan cells for pre-bond test of dies. This is achieved by maximizing the presence of circuit flip-flops on inter-die connections.
4) The pads for all external inputs and outputs to and from the SOC must be on the first (bottom most) die. This is required for SOC fabrication.

B. Hypergraph Based Circuit Partitioning

Typically circuits are modeled by hypergraphs for partitioning [6]. Experience has shown that hypergraph based partitioning is efficient and yields excellent results [6]. We also use hypergraph based partitioning incorporating methods to guide the procedure to achieve the above listed objectives for 3D-SOCs.

A multilevel hypergraph partitioning algorithm [5], [6], [7] has the following steps:

1) Hypergraph Construction: Cells of a circuit are mapped to nodes and circuit nets are mapped to hyperedges of a hypergraph. Weights on hyperedges and nodes are used to guide partitioning procedures to meet desired objectives. We use differential hyperedge weights and dynamically vary the weights to achieve lower TSV and DFT costs. Node weights represent the area of the circuit components such as gates, flip-flops etc.
2) Hypergraph Coarsening: Coarsening reduces the complexity of the problem and run time. A sequence of smaller hypergraphs are constructed via hyperedge merging. Merging of heavy (larger weight) hyperedges

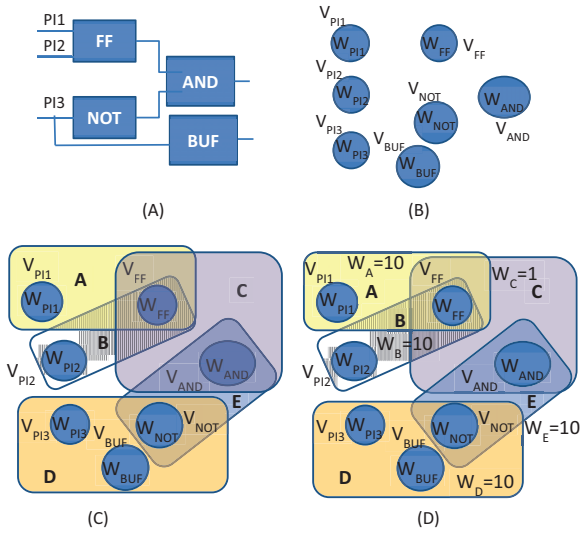

Fig. 3. (A) Circuit (B) Nodes in H corresponds to cells/IO in a circuit. (C) Nets in a circuit maps into hyperedges A, B, C, D, E. (D) Selective weight assignment to hyperedges.

is used in this work. During the hypergraph construction phase, hyperedges corresponding to nets driven by flip-flops are assigned lower weights. Lower weight hyperedges are usually not merged and thus have a higher probability to be on the cut which reduces the DFT cost as discussed earlier.

3) Initial Partition: The coarsened hypergraph is partitioned into blocks randomly. A good initial partition for a coarser hypergraph is usually of the same quality as a partition created on the original graph. Previous partitioning schemes obtain k-way partitions ignoring order of partition blocks [6]. We propose an Ordered Block Partitioning method to generate an initial partition using dynamic hyperedge weight assignments. This approach facilitates reduced TSV costs.
4) De-splitting and Refining: Merged vertices are uncoarsened to form the original hypergraph H. During the uncoarsening phase intermediate hypergraphs are refined to reduce the cut size using the Fiduccia-Mattheyses (FM) algorithm [6]. 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 block a particular vertex belongs and the gain is the corresponding change in the cut cost.

C. Review of Earlier Work

Hypergraph partitioning has its roots in circuit partitioning [6]. An area aware two-step hypergraph 3D-SOC partitioning scheme is proposed in [5]. The scheme presented in [5] has the following drawbacks: 1. DFT cost was ignored. 2. Procedure to reduce TSV count was incorporated late in the procedure during refining phase after an initial k-way partition of the coarsened graph was obtained using a known procedure. The approach proposed in [5] usually results in an increased TSV

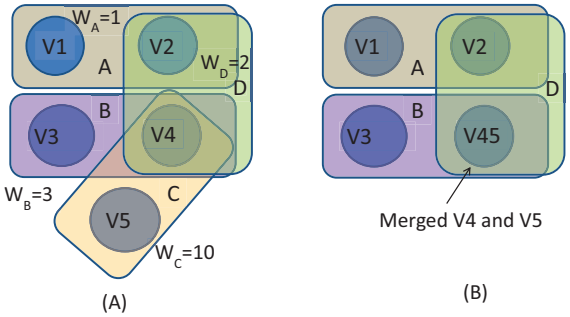

Fig. 4. An illustration of heavy hyperedge merging in hypergraph.

count in comparison to the algorithm proposed in this work. A hypergraph partitioning based approach to reduce DFT cost was proposed in [7]. However, TSV cost and cell areas were not considered and results were presented for the case of 2 dies only [7].

III. TSV Cost Aware Multi-level Netlist Partitioning Algorithm

In the following sections we discuss the details of the proposed circuit partitioning procedure.

A. Selective Weight Assignment Based Hypergraph Construction from Netlist

Consider a circuit netlist N with n cells (combinational-gates /buffers /flip-flops/ PIs/ POs). Fig 3 (A) shows a netlist N. Hypergraph construction for a circuit netlist is discussed below.

1) For every cell i in N there exists a vertex v_i in the hypergraph H. Fig 3 (B) shows the vertices of hypergraph H corresponding to the netlist N. Each v_i is assigned a weight w_i corresponding to its area.
2) Every hyperedge e_j in H corresponds to a net in a circuit netlist N. Fig 3 (C) gives hypergraph H with hyperedges added.
3) Hyperedges corresponding to nets driven by flip-flops are assigned weight w_{ff}. All other hyperedges are assigned a weight of w_g. As in [7] we bias the cut to have more flip-flops on it to reduce the additional DFT cost (scan cells) set $w_{ff} < w_g$.

Fig. 3(D) shows weight assignments for hyperedges in H using w_g=10 and w_{ff}=1. Hyperedges A, B, D, E, F are assigned weight w_g, hyperedge C driven by a flip-flop is assigned a weight of 1.

We assign increasing integer indices to the dies in a 3D-SOC stack from the bottom, with lower most die assigned the index of 1. The number of TSV's required to make a connection between vertices v_i in die p and v_j in die q is $|p-q|$.

B. Coarsening Phase

During the coarsening (merging) phase, a sequence of successively smaller hypergraphs are constructed. As discussed earlier the purpose of coarsening is to create a smaller hypergraph, such that a good partition of the smaller hypergraph is not significantly worse than the partition directly obtained

for the original hypergraph. Coarsening reduces run times as well as improves the quality of the solution, since refining algorithms used later are very effective in refining partitions of smaller hypergraphs but are quite ineffective in refining partitions of hypergraphs with a large number of vertices. We use heavy hyperedge merging. Hyperedges not driven by a flip-flop are heavier and thus have a higher probability of getting merged and placed on a single block of the partition. The light hyperedges are placed on the cut leading to more flip-flops on the cut.

Consider a hypergraph H(V, E_h), shown in Fig. 4(A), with 5 vertices V=(V1, V2, V3, V4, V5) and 4 hyperedges E=(A, B, C, D), with weights w_A=1, w_B=3, w_c=10 and w_D=2, respectively. The heavy hyperedge merging algorithm will merge vertices V4 and V5 corresponding to hyperedge C, the heaviest hyperedge into vertex V45, as shown in Fig. 4(B) to form the hypergraph H_1.

After merging m hyperedges the original hypergraph H is reduced to a smaller hypergraph H_m.

C. Ordered Block Partitioning

After the hypergraph is coarsened to contain about 200 nodes we stop coarsening. At this point we perform an initial partitioning. One way to obtain an initial partitioning is to use the recursive bisection algorithm of [6] to obtain a k-way partition. However, this approach generates partition blocks with no forethought on maintaining the order of the dies that determines the TSV cost of inter die connections.

We propose a method we call an Ordered Block Partitioning. In this approach we extract blocks of a partition corresponding to SOC dies in order starting from the bottom-most die. As we extract blocks we increase the weights of hyperedges from nodes of the hypergraph that have been included in the already extracted blocks to nodes in the remainder of the hypergraph. This allows the partitioning procedure to avoid higher weighted hyperedges to be on the cut, thus achieving lower TSV cost. Next we define some terms used in the description of the proposed procedure.

Recall that a circuit partition divides the nodes of the corresponding hypergraph (cells of the circuit) into disjoint subsets called *blocks* in this work. The union of already extracted blocks is referred to as the *base* and the set of the nodes not in the base is called the *remainder*. If blocks 1 to (i-1) are extracted then the i_{th} block to be extracted (from the remainder) is called the *target block*.

During hypergraph partitioning, user defined constraints are used to determine each block. Common constraints are number of nodes in a block or sum of the weights of nodes in a block. In previous works, [6] constraints are given at the start of an algorithm and not changed (static constraints). In the case of 3D-SOC partitioning, TSV area can only be determined during partitioning. We generate area constraints of target blocks during the execution of the algorithm as discussed in steps 2-5 described next.

The input to the block ordered partitiong algorithm is the node coarsened hypergraph with V_m vertices, the number of

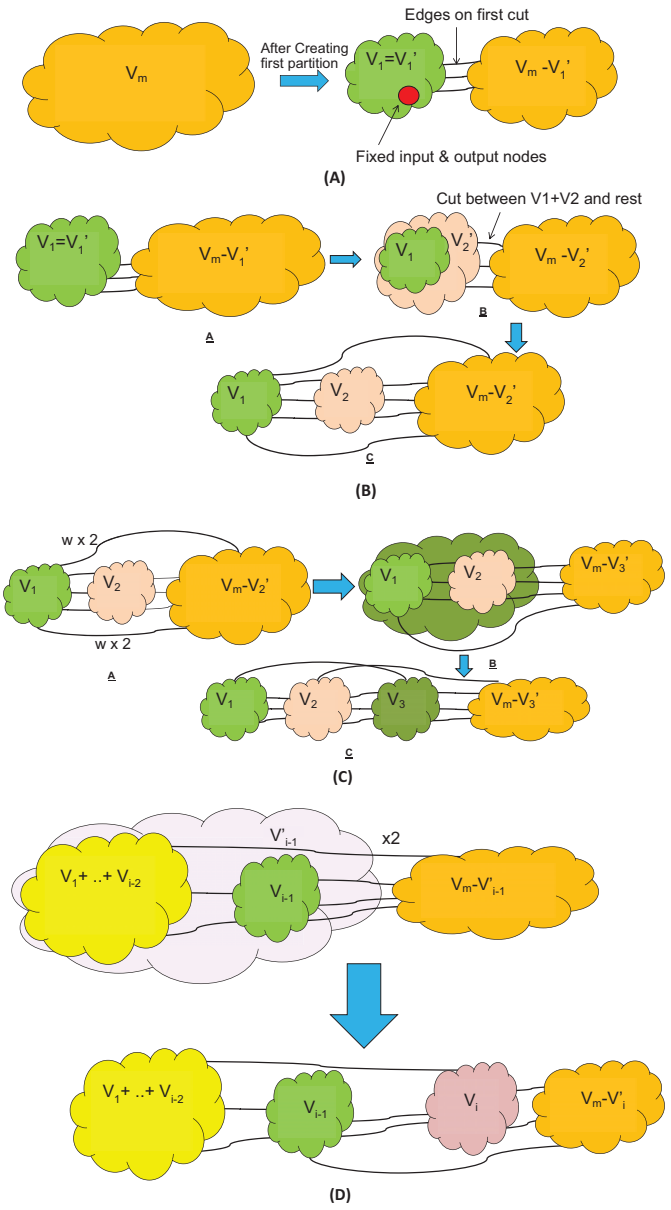

Fig. 5. Illustration of steps in the Ordered Block partitioning procedure.

blocks k, cell areas and IO pad areas. For simplicity we represent a block by the set of nodes V in it. The procedure outputs k extracted blocks of nearly equal area. A_{ci} is the area constraint for target die i. $A_{remainder}$ is the current area occupied by cells in the remainder partition. A_{base} is the current area occupied by cells in the base block and A_{TSVi} is the area occupied by the landing pads of TSVs between base and remainder partitions when i blocks have already been added to the base partition. V_i are nodes assigned to block i.

Main steps of the algorithm are discussed below:

1) Initially remainder = V_m. Base = ϕ.
2) When the target block =1, assign SOC IO pads to the target block. Area constraints for the first block is A_{c1}=($A_{remainder} + A_{IO}$)/k. Randomly add more nodes to block 1 to satisfy the area constraint, followed by refining. Let V_1 be the set of nodes added to block 1.

TABLE I

CIRCUIT CUT SIZE, FLIP-FLOPS ON CUT AND TOTAL TSV COST FOR 4 AND 6 DIES.

Circuit (Size)	# of Dies	w_g,w_{ff}	Ordered Block Partitioning			
			Cut	Addl SC	TSV #	Time (sec)
CKT1 240K	4	10,1	292	433	408	103
		1,1	382	762	486	101
	6	10,1	438	621	792	124
		1,1	416	824	889	129
CKT2 400K	4	10,1	779	1176	937	205
		1,1	1188	2370	1274	193
	6	10,1	922	1429	1162	237
		1,1	1375	2745	1836	214
CKT3 1M	4	10,1	916	1428	1162	427
		1,1	876	1749	1229	394
	6	10,1	1085	1613	1370	425
		1,1	1041	2073	1496	458
CKT4 3M	4	10,1	813	964	1442	1398
		1,1	742	1467	1731	1327
	6	10,1	1168	1429	1738	1501
		1,1	1093	2171	2183	1479

Update base =V_1, remainder = $V_m - V_1$.

Fig. 5(A) illustrates the first step of our algorithm. All IOs are fixed to the first die. Vertices are placed into blocks from the remainder until area constraint is met. Usually multiple random selections are made and the one with the smallest cut is chosen to form V_1. V_1 is further refined (not shown) using Fiduccia-Mattheyses [6] based refining algorithm.

3) When the target block is 2, generate new area constraints. A_{c2}= $(A_{remainder}$ + $A_{TSV1})/k - 1$, where A_{TSV1} is the area of TSVs landing pads on the remainder partition from base partition (block 1). Create a partition V_2' of size $A_{c1} + A_{c2}$, by randomly adding nodes to base followed by refining. Nodes assigned to the new target block are V_2=V_2'- base block, where base block is V_1 at this time. Update the remainder block and base blocks to base = V_2' and remainder = V_m - V_2'.

   Fig. 5(B) illustrates the construction of block 2.

4) For target partition 3 area constraint A_{c3} = $(A_{remiander} + A_{TSV2})/(k - 2)$, where A_{TSV2} is the area of TSVs landing pads on remainder partition from base partition (block 1 and block 2). Multiply the weight of the hyperedges between V_1 and the remainder block by 2. Generate block V_3' of size $A_{c1} + A_{c2} + A_{c3}$ by adding nodes from the remainder to the base V_2' and refine. Assign vertices V_3'-V_1-V_2 to target block 3. Update base and remainder blocks to base = V_3' and remainder = V_m - V_3'.

   Fig. 5(C) illustrates the construction of block 3.

5) In a manner similar to obtaining block 3 in Step 4 above, for the i_{th} block multiply hyperedges between (V_1 +. . .+V_{i-2}) and the remainder by 2. Using new area constraints, form a partition V_i' that includes vertices in (V_1+ . . .+ V_{i-1}). Vertices assigned to die i are V_i = (V_i'-(V_1+ . . .+ V_{i-1})). Update base and remainder blocks.

   Fig. 5(D) illustrates this step.

D. Desplitting and Refinement

Assume that in forming the coarsened hypergraph was obtained by merging m hyperedges of the original hypergraph H. For convenience we call the coarsened hypergraph H_m= (V_m, E_m). In the previous step the procedure created an initial partition of H_m. During the desplitting phase, the merged vertices are uncoarsened (un-merged) to form the original hypergraph H in m successive steps. Only one hyperedge is unmerged in one step. Cut sizes of intermediate hypergraphs obtained by unmerging are further refined by use of the Fiduccia-Mattheyses (FM) algorithm [6].

E. Iterations to Decrease Area Imbalance

It is not possible to estimate the area of TSVs for the partitioned circuit beforehand. For this reason we used an approximation to include the TSV areas in computing the area constraints for dies in Steps 2-5 of Ordered Block Partitions. Ordered Block partitioning method is run repeatedly till user specified bounds for area imbalance are achieved or maximum number of iterations allowed are completed. At the start of each iteration, area constraints for the first die is recalculated based on the result of the previous iteration.

During the first iteration, an estimate of die area is made based on cell area and IO pad area and the area constraints for the dies are calculated during the execution of the partitioning procedure as described before. During subsequent iterations area constraints for the first die for the next (i_{th}) iterations is set to, A^i_{c1} = $(A_{cellArea} + A^{i-1}_{TSV} + A_{IO})/k$, where $A_{cellArea}$ is the total area occupied by all cells of the 3D-SOC, A^{i-1}_{TSV} is the area of the TSVs in $(i-1)_{th}$ iteration and A_{IO} is the area of the input/output pads. Area constraints for the remaining dies are calculated as before as described in Steps 2-5 of Ordered Block partitioning procedure.

TABLE II

COMPARISON OF TSV COUNT AND DIE AREA FOR PROPOSED WORK AND [5].

Circuit (Size)	# of Dies	Approach	Die1 Area	Die2 Area	Die3 Area	Die4 Area	Die5 Area	Total Area	Addl SC	# of TSV's
Bench1 6291	4	OBP	50648	50203	49853	49107	–	199811	473	503
		[5]	49076	52743	52792	52799	–	207411	-	579
Bench2 85013	5	OBP	289924	281263	280026	279235	278970	1409418	106	141
		[5]	278872	285933	282571	285791	277851	1411018	-	157
Bench3 9155	3	OBP	88653	86132	85960	–	–	260745	169	227
		[5]	84409	91065	91071	–	–	266545	-	285

F. Reordering to Reduce TSV Cost

Even though Ordered Block partitioning procedure gives optimal die order most of the time, it does not guarantee optimal order due to its heuristic nature. We attempt reordering of dies after partitioning. All possible permutations of die stacking excluding the bottom most die are generated. Permutation with the lowest TSV cost is chosen after reordering.

IV. EXPERIMENTAL RESULTS AND DISCUSSIONS

A set of experiments using industrial designs were performed on a linux workstation with Intel Xeon $2GHZ$, 8 core processor and 16GB RAM. For the initial weights of the hyperedges we set $w_{ff} = 1$ and $w_g = 10$, where w_{ff} is the weight assigned to hyperedges driven by a flip-flop and w_g is the weight assigned to other hyperedges. After extensive experimentation we observed that use of the above mentioned weight assignment leads to a reduced cut size with large number of flip-flops on the cut as desired, while decreasing overall TSV cost. In Table I we present results for partitioning circuits into 4 and 6 dies (blocks) on four industrial circuits. Cell area, TSV area and I/O pad area for circuits are chosen as in [5]. For each circuit the cut size, number of additional scan-cells needed, the total number of TSVs and the run-time for the proposed Ordered Block partitioning are given, for cases where all hyperedges are assigned weight of 1 and the other that used $w_{ff} = 1$ and $w_g = 10$. The allowed area imbalance between partition blocks was set to 1%.

From Table I we note that in all cases the number of TSVs and the number of additional scan cells needed are smaller for weights $w_g = 10$ and $w_{ff} = 1$ in comparison to $w_g = 1$ and $w_{ff} = 1$.

For example for Ckt2, for weight assignment w_{ff}=1 and w_g=10 and 6 dies, the proposed algorithm achieves cut size of 922, 1429 additional scan cells for pre-bond test and a TSV count of 1162. Cut size, additional scan cells and number of TSVs produced by using hyperedge weight assignment of $w_g = 1$ and $w_{ff} = 1$ are 1375, 2745 and 1836, respectively. Table I shows that runtime for the proposed procedure increases almost linearly with the circuit size. We do not compare the proposed approach to methods described in [11] and [7] as TSV cost and cell areas were not considered in [11], [7] and results were presented for 2 dies only.

Table II gives results for partitioning the circuits used in [5]. These industrial circuits were provided by ITRI, Taiwan, for a design contest in IC/CAD 2009, Taiwan. The results on partitioning for 3D-SOCs reported in [5] and using the proposed Ordered Block Partitioning (OBP) are reported in Table II. From Table II it can be seen that the proposed method produces partitions, requiring on average, about 15% fewer TSVs than those by [5]. Data on the number of additional scan cells is not given for [5] as it was not reported.

V. CONCLUSIONS

In this paper, a TSV and DFT cost driven circuit partitioning for 3D-SOCs was proposed. This procedure called Ordered Block Partitioning uses hypergraph partitioning and achieves substantially reduced TSV costs and DFT costs of 3D-SOCs.

REFERENCES

[1] M. Buttrick and S. Kundu. On testing prebond dies with incomplete clock networks in a 3d ic using dlls. In *DATE, 2011*, pages 1 –6, 2011.

[2] C-C. Chi, E. J. Marinissen, S. K. Goel, and C-W. Wu. Dft architecture for 3d-sics with multiple towers. In *ETS*, pages 51 –56, 2011.

[3] F. DaSilva, Y. Zorian, L. Whetsel, K. Arabi, and R. Kapur. Overview of the ieee p1500 standard. *ITC*, 2003.

[4] T.-Y. Hsueh, H.-H. Yang, W.-C. Wu, and M.C. Chi. A layer prediction method for minimum cost three dimensional integrated circuits. In *IEEE ISQED*, pages 1 –5, 2011.

[5] Yu Cheng Hu, Yin Lin Chung, and Mely Chen Chi. A multilevel multilayer partitioning algorithm for three dimensional integrated circuits. *ISQED*, pages 483 –487, 2010.

[6] 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.

[7] A. Kumar, S.M. Reddy, I. Pomeranz, and B. Becker. Hyper-graph based partitioning to reduce dft cost for pre-bond 3d-ic testing. *DATE, 2011*.

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

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

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

[11] Jia Li and Dong Xiang. Dft optimization for pre-bond testing of 3d-sics containing tsvs. *ICCD*, pages 474 –479, 2010.

[12] C-Y. Lo, Y-T. Hsing, L-M. Denq, and C-W. Wu. Soc test architecture and method for 3-d ics. *IEEE TCAD*, pages 1645 –1649, oct. 2010.

[13] E. J. Marinissen. Testing tsv-based three-dimensional stacked ics. *IEEE, DATE*, pages 1689–1694, 2010.

[14] E.J. Marinissen, C-C. Chi, J. Verbree, and M. Konijnenburg. 3d dft architecture for pre-bond and post-bond testing. In *3DIC Conf., IEEE*, pages 1 –8, 2010.

[15] E.J. Marinissen, J. Verbree, and M. Konijnenburg. A structured and scalable test access architecture for tsv-based 3d stacked ics. In *VTS*, pages 269 –274, 2010.

[16] S. Panth and S.K. Lim. Scan chain and power delivery network synthesis for pre-bond test of 3d ics. *VTS*, pages 26 –31, may 2011.

[17] M. Taouil, S. Hamdioui, K. Beenakker, and E.J. Marinissen. Test cost analysis for 3d die-to-wafer stacking. *ATS*, pages 435 –441, 2010.

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