Performance Aware Partitioning for 3D-SOCs

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

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

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

***Abstract*— Through silicon vias (TSVs) have a significant impact on the area and timing performance of a 3D-SOC. Performance aware design partitioning is required to reduce delay by placing gates on critical paths close to each other on the same or adjacent dies in the 3D-SOC. Also, pre-bond test of dies in the 3D-SOC is required to insure correct functionality of the dies before bonding. Additional design for test (DFT) logic for pre-bond test also depends on the design partitions used. In this work we propose a hypergraph based multi-objective netlist partitioning scheme to improve timing performace of 3D-SOC while keeping additional DFT cost low. Accurate interconnect estimation models are incorporated during partitioning to reduce delay and interconnect length variations across dies. Results on ISCAS89 and ITC99 benchmark circuits demonstrate the improved timing performance and reduced DFT cost using the proposed approach.**

I. Introduction

In 3D-SOC design flow a circuit is partitioned across multiple dies in the very first step in the flow as illustrated in Fig. 1. Vertical interconnections by Through-Silicon-Vias (TSVs) [1], [2] are used for inter-die connections in 3D-SOCs. TSV's also help shorten interconnect length are stacked on top of each other. Partitioning procedure used determine the number of TSV's and interconnect length [1], [3]. TSVs have non-negligible size, their increased numbers can have other indirect impacts such as routing congestion, delay as well as crosstalk and noise problems [1], [4], [5], [6], [7]. In some cases this may result in 3D-SOC performance worse than its 2D counterpart. Unfortunately, good interconnect length prediction is only available after floorplanning and placement stages [5]. However, it will be a prohibitively expensive iterative process to incorporate routing within the optimization process of an early stage [4], [5], [8]. A more realistic approach is to apply some fast yet accurate techniques to estimate the physical interconnect information and use it during circuit partitioning. It is imperative to account for timing during the partitioning process to allow for early interconnect planning. All nodes within the same critical paths should be kept close by, preferably placing them on the same die. In Fig. 2, the highlighted critical path in Fig. 2(B) performs better than in Fig. 2(A), as majority of nodes in the critical path are on the same die. In Fig 2(A) four TSV's are required in comparison to two in Fig. 2(B).

Another major issue with 3D-SOC design is low yield [9], [10]. Pre-bond test of individual dies is required to unsure adequate yield of 3D-SOCs [9]. Test of individual dies requires additional DFT logic consisting of additional scan flip-flops

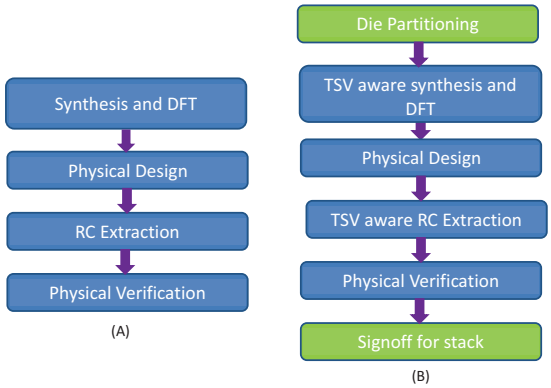

Fig. 1. (A) Flow for 2-D SOC. (B) 3D-SOC flow.

[3].

We have recently proposed a hypergraph based circuit partitioning to reduce the number of additional scan flip-flops needed [3], [9]. However, interconnect length and circuit delays were not considered. Authors in [7] introduced the concept of TSV aware placement, however, initial partitioning phase was not interconnect aware, resulting in performance degradation and the DFT cost was also not considered.

In this work we propose a hypergraph based 3D-SOC multi-objective performance aware circuit partitioning procedure that simultaneously reduces additional flip-flops needed for DFT, the number of TSVs, while improving timing performance by keeping interconnect length approximately the same on each die and assigning nodes on critical path gates to the same or neighbouring dies during partitioning. A novel interconnect length based weight assignment is proposed for nodes, resulting in reduced total interconnect variation among dies. Additional hyperedges with very high weights are added for K longest critical paths under consideration to reduce circuit delay. We use the selective hyperedge weight assignment proposed in [9] to reduce additional flip-flops required for DFT.

The remainder of the paper is organized as follows. In Section II we review the requirements on circuit partitioning for 3D-SOCs. 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. Circuit Partitioning for 3D-IC's

Objectives that need to be satisfied by any partitioning scheme for 3D-SOCs are [2], [3], [7]:

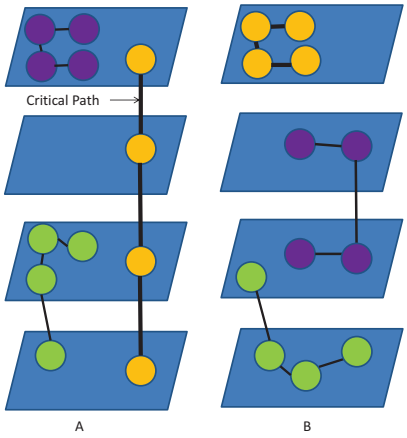

Fig. 2. Critical path aware partitioning.

1) Assign nodes in the critical paths to same die if possible to improve timing performance.
2) Bias the partition cut to have design flip-flops on the cut [3], [9], it reduces the requirement for additional scan-cells required for pre-bond testing.
3) Partition the circuit into dies of approximately equal interconnect length and area.
4) Number of connections between non-adjacent dies should be reduced. This results in the reduction in TSV's as the number of TSV's required for a inter-die connection is equal to the difference between the die indexes [2].

III. Hypergraph Based Partitioning Algorithm

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

1) Initial Construction: Our method of hypergraph construction from netlist is similar to that described in [2], [3], [9], however weights assigned to nodes and hyperedges are different. Consider a circuit netlist N with n cells (combinational-gates /buffers /flip-flops/ PIs/ POs). Overview of hypergraph construction method is: (1) For every cell i in N there exists a vertex v_i in the hypergraph H. (2) Every hyperedge e_j in H corresponds to a net in a circuit netlist N. (3) Weights corresponding to wirelength are assigned to hyperedges using a interconnect estimation method, which are moved to nodes using a weight translocation scheme. Hyperedge weights are deleted after translocation. (4) Hyperedges are reassigned weight using a selective weight assignment scheme to reduce DFT cost [9]. (5) Additional hyperedges are added for K longest critical paths.

A. Weight Translocation Based Hypergraph Construction from Netlist

Typically for hypergraph based circuit partitioning, gates are represented by nodes and nets represent hyperedges. Both hyperedges and nodes can be assigned weight. We deal with two hyperedge weight sets:

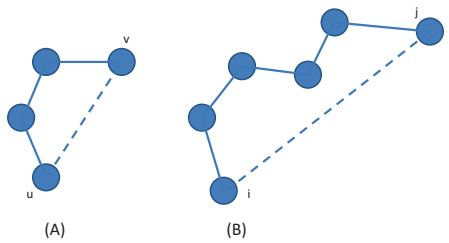

Fig. 3. Intrinsic shortest path length ($ISPL$) calculation (A) $ISPL(u,v) = 3$ (B) $ISPL(i,j) = 5$.

1) Reduction in number of additional scan-cells required for pre-bond test is made possible by using selective weight assignments for hyperedge.
2) Interconnect length for a given die needs to be estimated interconnect is a property of nets (hyperedges).

Since two different weight sets can not be simultaneously assigned to hyperedges, we introduce the concept of weight translocation,

Weights corresponding to interconnect length are first assigned to hyperedges and shifted to nodes after weight translocation.

1) Intrinsic Shortest Path Based Interconnect Length Calculations: One popular technique for interconnect estimation is by using Rents rule, which involves a tedious pre-processing step [5] and do not directly operate on hypergraphs, making it infeasible for our purpose. We use interconnect estimation technique from [5]. We start with some basic definitions: *Restricted Shortest Path:* Given a hypergraph H and a hyperedge r, the length of the restricted shortest path between two nodes u and v ($(u,v) \in r$), is the minimum number of hyperedges in a path connecting u and v in hypergraph H_{mod} =(G, E-r). Since hypergraphs contain multi-pin nets, we set the weight of a hyperedge e_k as $|e_k|$ (where $|e_k|$ is number of nodes in hyperedge e_k)

Example: Consider two hypergraphs in Fig 3(a) and 3(b) any good placement algorithm will result in $wirelength(u,v) < wirelenght(i,j)$, corresponding restricted shortest paths for pair (v,j)=3 and (i,j) =5. For purpose of illustration, each hyperedge has upto 2 nodes (as in a graph), therefore an initial weight of $2/2$=1.

Intrinsic Restricted Shortest Path: Intrinsic restricted shortest path for a hyperedge S is defined as the maximum of restricted shortest paths over all possible node pairs constituting hyperedge S.

For every hyperedge calculate the Intrinsic Shortest Path Length ($ISPL$) for all node pairs and re-assign corresponding $ISPL$ values as hyperedge weights.

2) Hyperedge Weight Translocation to Nodes: Since, a hyperedge can have only one weight, we translocate weights assigned to hyperedges to node weights. Prior to this nodes were not assigned any weights. Now for partitioning algorithm another objective is to assign equal total sum of node weights to each die to have less variation of total interconnect length on each die. During weight translocation, weights of a hypergraph is divided equally among all its nodes. If a node is connected

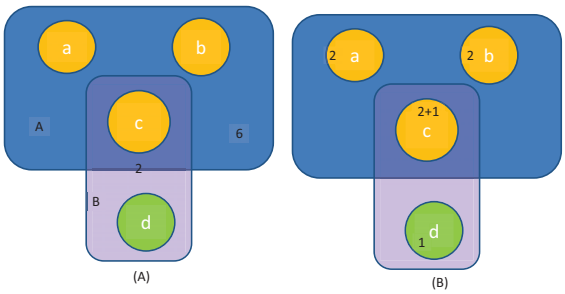

Fig. 4. An illustration of hypergraph weight translocation (A) Weights on hyperedges. (B) Hyperedge weights translocated to nodes.

by multiple hyperedges, weight of a node is sum total of contributions of all its hyperedges. All weights corresponding to hyperedges are deleted after weight translocation.

Example Consider a hypergraph in Fig. 4(A) with two hyperedges A & B. Weights of A and B are w_A=6 and w_B=2. Hyperedge A contains nodes a, b, c and hyperedge B contains nodes c, d. Hyperedge A makes a weight contribution of $6/$(number of nodes in hyperedge A =3), i.e, 2 to nodes it contains a, b, c and hyperedge B makes a weight contribution of $2/2$=1 to nodes c, d. Weights at individual nodes is sum of weight contribution by all hyperedges. Weights of nodes a, b, c, d after weight translocation are 2, 2, 3 , 1 as shown in Fig. 4(B).

3) Hyperedge Weight Assignment to Save Additional Flip-flop Required for DFT: After translocation and deletion of hyperedge weights corresponding to interconnect length , hyperedges are assigned weights according to gate type hyperedge is driven by.

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 [9] we bias the cut to have more flip-flops on it to reduce the additional DFT cost (scan cells) set $w_{ff} < w_g$.

4) Addition of Hyperedges to Improve Delay Performance: We follow a three pronged strategy for delay minimization, reducing interconnect length by partitioning circuit into dies with approximately equal interconnect length, reduction in TSV count and keeping nodes in the K critical path to be on the same die. Longest K critical paths are generated using a commercial tool. Critical paths are sorted in descending order of delay. An additional hyperedge is added for every critical path, which comprise of all the nodes in the corresponding critical path. Very high weight is assigned to this additional hyperedge.

Since all the nodes in a critical paths are connected by a higher weight hyperedge possibility of these hyperedges being on the cut is very small and they are usually on the same die. Delay is dependent on die ordering as TSV number depends on ordering and TSVs have much higher value for capacitance.

B. Merging Phase

The initial merging phase is based on hierarchical clustering [6], [9]. Vertices connected with heavier hyperedges are merged first. We randomly select edges for merging if more that one edge has the same weight. Since hyperedges not driven by a flip-flop are heavier, they have a higher probability of getting merged. Heavy hyperedge merging increases the possibility of a biased cut, with more flip-flops on the cut.

C. TSV Aware Initial Partitioning

Our initial partitioning approach is based on OBD partitioning scheme described in [3]. An initial partitioning 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_m in to k dies with approximately same interconnect length (node weight). 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. Traditional hypergraph partitioning approaches ignore die order of partitions, since TSV cost is determined by ordering of partitions, 3D SOC partitioning approach should include die order as one of the parameters.

Some basic definitions used are described now. The union of already extracted dies is referred to as the base and the set of the nodes not in the base is called the remainder. If dies 1 to $(i-1)$ are extracted then the i_{th} block to be extracted (from the remainder) is called the target die (or partition). Sum of all node weights (corresponding to interconnect length) is W_N, sum of weights of all node assigned to die i is W_i.

Main steps of the algorithm are discussed below:

1) In the beginning remainder = V_m. Base = ϕ.
2) Assign all IO pads to the target die 1. Randomly form partition 1 to satisfy the weight constraint of W_N/k. Update the base and remainder partitions, if V_1 be the set of nodes assigned to die 1, now base =V_1, remainder = $V_m - V_1$. Usually multiple random selections are made to form V_1, out of which the best one is chosen for further processing.
3) To assign i_{th} die multiply hyperedges between (V_1 +. . .+V_{i-2}) and the remainder by 2. Using new node weight 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 partitions.

D. Desplitting and Refinement

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. Cut sizes of intermediate hypergraphs $H_{m1} \ldots H$ (obtained during various steps in demerging phase) are further reduced by use of the Fiduccia-Mattheyses (FM) algorithm .

IV. Experimental Results

Experiments were performed on two larger ISCAS-89 and two ITC-99 benchmark circuits. The final manuscript will report on a larger set of circuits. We used 45nm NanGate libraries [7] and 0.5μm edge square TSVs with parameters

TABLE I

COMPARISON OF RESULTS.

Circuit	# of Dies	Proposed Approach				Random Partitioning			
		Delay (ps)	# of TSVs	Addl. Scan FFs	Area per Die(μm^2)	Delay (ps)	# of TSVs	Addl. Scan FFs	Area per Die(μm^2)
s5378	4	126	38	41	1197	163	268	517	1256
	6	159	61	79	886	216	383	755	987
	8	194	114	137	587	241	582	1096	681
	1(2D)	186	na	na	4219	na	na	na	na
s13207	4	447	103	165	3798	511	274	483	3840
	6	491	162	194	2715	610	435	815	2807
	8	568	249	305	2179	641	726	1331	2303
	1(2D)	605	na	na	13970	na	na	na	na
b22	4	531	95	86	5182	679	358	701	5329
	6	485	146	162	3741	638	620	1117	3881
	8	672	287	320	2939	794	924	1692	3106
	1(2D)	653	na	na	19239	na	na	na	na
b21	4	381	138	171	4609	474	394	746	4778
	6	457	185	206	3215	518	578	1036	3401
	8	490	273	311	2627	563	731	1328	2785
	1(2D)	475	na	na	17125	na	na	na	na

described in [7]. Place and route using a commercial tool followed by 3D timing optimization was used. Based on the timing analysis result and the target clock frequency, the tool scales target delays of 3D paths and creates a timing constraint file for each die, since each die has its own netlist and timing constraint file. We performed timing optimization for each die separately. We iterate this timing optimization process several times until no further timing improvement is possible. Similar timing optimization for 2D version of each design was done.

In Table I we compare the results obtained for 2D design implementation, 3D implementation with 4, 6 an 8 dies using the proposed partitioning approach and a 3D design implementation using random partitioning. After the circuit name in the first column we give the design used followed by the maximum circuit delay, the number of TSVs, the number of additional flip-flops for pre-bond test and the average size of the dies for the proposed and random partitioning. We also give the maximum circuit delay and the die size for 2D design. The experimental results show the advantages of the proposed 3D partitioning approach, 3D implementation using the proposed approach is up to 32% faster than its 2D counterpart. It can also be seen that the size of the dies using N layers in 3D is approximately area of the corresponding 2D die divided by N. This indicates that the additional area for TSVs is minimal. As noted earlier [11] as the number of dies stacked in a 3D implementation increases the maximum circuit delay increase even beyond that for the 2D counterpart. Also the number of TSVs increases. Thus one has to be conservative in the number of layers used in 3D design to obtain best performance.

V. CONCLUSIONS

In this paper we propose a performance aware 3D SOC partitioning scheme. This technique uses hypergraph based interconnect aware netlist partitioning method. Proposed weight translocation based partitioning approach attains multiple objectives of keeping interconnect length approximately same across dies, reducing delay and reduction of the number of additional flip-flops required for pre-bond testing. Experimental results show a significant reduction in the delay in comparison to 2D implementation and the number of additional flip-flops required for pre-bond testing is simultaneously minimized.

REFERENCES

[1] S. M. Alam, M. Ignatowski, and Y. Xie, "Technology, cad tools, and designs for emerging 3d integration technology," in *Proc. GLSVLSI*, pp. 1–2, 2008.

[2] Y. Hu, Y. L. Chung, and M. Chen-Chi, "A multilevel multilayer partitioning algorithm for three dimensional integrated circuits," in *Proc. ISQED*, pp. 483 –487, 2010.

[3] A. Kumar, S. M. Reddy, I. Pomeranz, and B. Becker, "Tsv and dft cost aware circuit partitioning for 3d-socs," in *Proc. ISQED*, pp. 21–26, 2012.

[4] C. Ababei, N. Selvakkumaran, K. Bazargan, and G. Karypis, "Multi-objective circuit partitioning for cutsize and path-based delay minimization," in *Proc. ICCAD*, pp. 181–185, 2002.

[5] A. Kahng and S. Reda, "Intrinsic shortest path length: a new, accurate a priori wirelength estimator," in *Proc. ICCAD*, pp. 173 – 180, 2005.

[6] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar, "Multilevel hypergraph partitioning applications in vlsi domain," in *Trans. TVLSI*, pp. 69–79, 1999.

[7] D. Kim, S. Kim, and S. K. Lim, "Impact of nano-scale through-silicon vias on the quality of today and future 3d ic designs," in *Proc. SLIP*, pp. 1–8, 2011.

[8] F. Chang, S. Chen, R. Tsay, and W. Mak, "Cut-demand based routing resource allocation and consolidation for routability enhancement," in *Proc. ASPDAC*, pp. 533–538, 2011.

[9] A. Kumar, S. M. Reddy, I. Pomeranz, and B. Becker, "Hyper-graph based partitioning to reduce dft cost for pre-bond 3d-ic testing," in *Proc. DATE*, pp. 1–6, 2011.

[10] H. S. Lee and K. Chakrabarty, "Test challenges for 3d integrated circuits," in *Design & Test of Computers*, pp. 26–35, 2009.

[11] M. Pathak, Y. Lee, T. Moon, and S. K. Lim, "Through-silicon-via management during 3d physical design: when to add and how many?," in *Proc. ICCAD*, pp. 387–394, 2010.