Whitespace-Aware TSV Arrangement in 3D Clock Tree Synthesis

Xin Li, Wulong Liu, Haixiao Du, Yu Wang, Yuchun Ma, Huazhong Yang
Tsinghua National Laboratory for Information Science and Technology (TNList)
EE Dept., Tsinghua Univ., Beijing, China
Email: yu-wang@mail.tsinghua.edu.cn

***Abstract*—Through-silicon via (TSV) could provide vertical connections between different dies in three-dimensional integrated circuits (3D ICs), but the significant silicon area occupied by TSVs may bring great challenge to designers in 3D clock tree synthesis (CTS) because only few whitespace blocks can be used for clock TSVs after floorplan and placement. Unlike most of the published previous works that ignore whitespace, this paper for the first time proposes a whitespace-aware TSV arrangement algorithm in 3D CTS. The algorithm consists of three stages: sink pre-clustering, whitespace-aware three-dimensional method of means and medians (3D-MMM) topology generation and deferred-merge embedding (DME) merging segment reconstruction. We also present a TSV whitespace-aware 3D CTS flow. Experiment results show that our proposed algorithm is more practical and efficient, the average skew and power can be reduced by 49.2% and 1.9% respectively, compared to the traditional 3D-MMM based CTS method with TSV moving adjustment.**

Keywords—Clock tree synthesis; 3D ICs; Whitespace; TSV arrangement

I. INTRODUCTION

As CMOS process technology continuously scaling down, through-silicon-via (TSV) based three-dimensional integrated circuits (3D ICs) have drawn much more attention recently. With the help of 3D technology we can reduce global wirelength, alleviate congestion and improve performance. Moreover, 3D technology gives much more design flexibility by heterogeneous integration [1].

For a 3D stacked IC, a single clock tree distributes through the entire stacks and connects all the clock sinks on different dies by TSVs. Despite the gains mentioned before, TSVs could also cause some serious problems. Under current technologies, TSVs are very huge compared to gates and memory cells [2], as a result, a large number of TSVs will consume significant silicon areas and degrade the yield and reliability of the final chip. Furthermore, TSVs are usually placed in the whitespace among macro blocks or cells, a bad arrangement of TSVs may incur longer wirelength since the available TSV might be far away from its connected cells. Nowadays, Intellectual Property (IP) and Standard cell based design has been extensively used to reduce design cost, but after floorplan and placement, only few whitespace blocks are reserved for clock TSVs [3]. Fig. 1 indicates that without the consideration of TSV whitespace during 3D clock tree synthesis (CTS), TSV moving is necessary to ensure that each TSV is located inside the whitespace and it would incur longer wirelength and lead to potential skew increase.

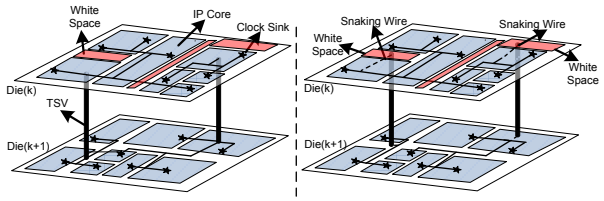

Fig. 1. 3D CTS without whitespace-aware TSV arrangement. (a) TSVs are not located in whitespace after an initial design. (b) Moving TSVs into whitespace may incur longer wirelength and lead to potential skew increase, so a whitespace-aware TSV arrangement algorithm is necessary.

A. Previous Work

Many literatures spring up in the past few years in the field of 3D CTS. Zhao et al. generated a 3D clock tree taken into consideration the number of TSVs by using TSV bound between adjacent dies in their three-dimensional method of means and medians (3D-MMM) algorithm [4]. The basic idea is to recursively divide the given sink set into two subsets until each sink belongs to its own set. The current division is based on the current TSV bound, which is also divided according to the ratio of the estimated number of TSV in each subset. The 3D-MMM-ext algorithm in [5] give the optimal number of TSVs so as to minimize the overall power consumption. Kim et al. proposed MMM-3D algorithm in [6], which uses a designer specified parameter ρ ($0 \le \rho \le 1$) to control the partition direction, if the half perimeter wirelength of a subset is smaller than $\rho \cdot L$ (where L is the half perimeter wirelength of all the sinks), z-cut is executed then. They also proposed a solution called ZCTE-3D to the zero skew clock tree embedding problem, which can give the best TSV allocation and placement result for a given tree topology. These top-down methods could control TSV counts but are not able to accurately predict TSV locations. In [7], Zhao et al. for the first time solved a practical 3D clock routing problem which stems from all kinds of TSV-induced obstacles, such as P/G, signal and clock TSV. They developed a TSV-induced obstacle-aware deferred-merge embedding (DME) method to construct a buffered clock tree which can avoid those obstacles with the help of newly defined merging segments. However, too many obstacles exits and only few whitespace blocks are reserved for clock TSVs after floorplan and placement in IP and Standard cell based designs. Long wire snaking is inevitable in such scenarios and a new whitespace-aware algorithm is necessary.

This work was supported by National Science and Technology Major Project (2010ZX01030-001-001-04, 2011ZX01035-001-001-002), National Natural Science Foundation of China (No. 61261160501, 61028006, 61076035), and Tsinghua University Initiative Scientific Research Program.

B. Our Contributions

As mentioned before, TSV count and location is crucial and only few whitespace is available for clock TSVs during 3D CTS. None of the existing methods still work efficiently on this problem. In this paper, we propose a whitespace-aware TSV arrangement algorithm in 3D CTS. The main contributions are summarized as follows:

(1) To the best of our knowledge, this is **the first work** that arrange clock TSVs according to whitespace in 3D CTS. We formulate the whitespace-aware TSV arrangement problem in 3D CTS and propose a practical and efficient algorithm to deal with it. Furthermore, we propose a whitespace-aware 3D CTS flow in Section III.

(2) The proposed algorithm is made up of three stages: a distance-aware **sink pre-clustering** algorithm, which makes sinks distribute closer to adjacent whitespace; an extended version of the 3D-MMM clock tree topology generation algorithm named as **TSV whitespace-aware 3D-MMM** (**TWA-3D-MMM** for short), which ensure that each sink set contains whitespace; and a **DME merging segment reconstruction** algorithm, which will bring convenience to routing and TSV arrangement.

(3) We investigate the relation between whitespace area, TSV number and the main CTS elements such as power, skew, slew rate by comparing our method to traditional 3D-MMM based one with TSV moving adjustment. We find that our method exceed the other one and could achieve an average skew and power reduction by **49.2%** and **1.9%** respectively. Our method is able to take account of both power and skew in all cases.

II. PRELIMINARIES AND PROBLEM FORMULATION

A. Electrical Model of 3D Clock Network

Die: For a *N*-die stack 3D clock design, we number the die as Die(0), Die(1),…, Die(*N*-1) in a top-down manner, the die on which clock source located is named as the source die. For simplicity, we set the clock source on Die(0) in this paper.

TSV: TSV between nonadjacent dies is considered as several TSVs between adjacent dies. Note that we do not consider TSV-to-TSV coupling effect in this work.

TSV whitespace block: With current technologies, the diameter of TSV is very huge compared to gates and memory cells, so there are few whitespace blocks reserved for TSVs before CTS. TSV whitespace exists between blocks like IP cores and they can be modeled as discrete whitespace blocks. In a *N*-die F2B stack case as shown in Fig. 2, TSVs between Die(*k*) and Die(*k*+1) are only restricted by the whitespace on Die(*k*) [8]. Notice that TSV whitespace on the last die, i.e. Die(*N*-1) is meaningless. For simplicity, TSV whitespace (blocks) is referred to as whitespace (blocks) hereafter.

B. Problem Formulation

The formal definition of *whitespace-aware TSV arrangement problem in 3D clock tree synthesis* is as follows: Given some whitespace blocks *W*, a set of clock sinks *S* on each die, a TSV bound B_{TSV} and a slew rate bound B_S, the objective is to construct a single clock tree such that 1) the number of clock TSV, i.e. #TSV≤B_{TSV}; 2) each clock TSV is located in the whitespace blocks without overlap; 3) clock slew rate is under B_S; 4) clock skew and clock power are minimized.

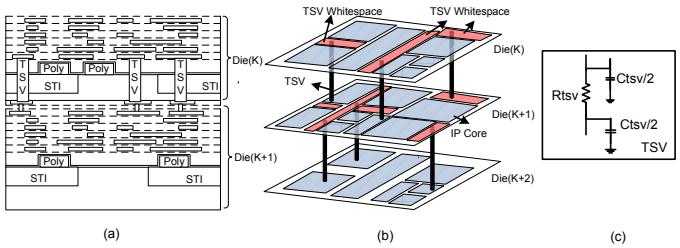

Fig. 2. Models. (a) F2B stack. (b) TSV between Die(k) and Die(k+1) is only restricted by the whitespace blocks on Die(k). (c) TSV model.

III. ALGORITHM

A. Overview of Our Proposed Method

As mentioned before, there are three stages in our whitespace-aware TSV arrangement method. In the sink pre-clustering stage, sinks far away from their related whitespace are clustered to merge subtrees, only the root node of the subtree is reserved and will play as a new "sink" after that. In the TWA-3D-MMM clock tree topology generation stage, we extend the 3D-MMM method by judging whether the current x/y-cut between multiple dies is appropriate, considering whitespace to ensure that each sink set contains whitespace. In the DME merging segment reconstruction stage, we modify the merging segment of the internal nodes having TSVs with the consideration of TSV geometries and whitespace occupation, which would bring convenience to detail routing and TSV arrangement. By leveraging a slew-aware buffering stage we further present a whitespace-aware 3D CTS flow in Fig. 3. The time complexity of our proposed method is O(mn) where n and m are the number of clock sinks and whitespace blocks respectively.

B. Sink Pre-clustering

While whitespace is small and limited by widely distributed clock sink, there may be a long distance between sinks and whitespace. In this case, arranging TSVs in whitespace would lead to wirelength overhead and potential skew increase.

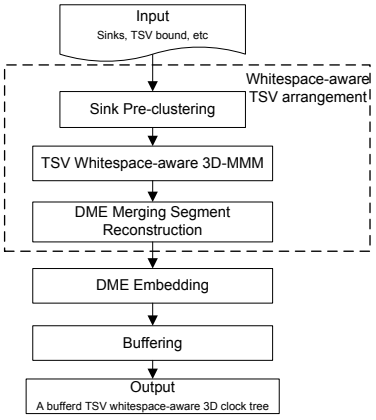

Fig. 3. The proposed whitespace-aware 3D CTS flow

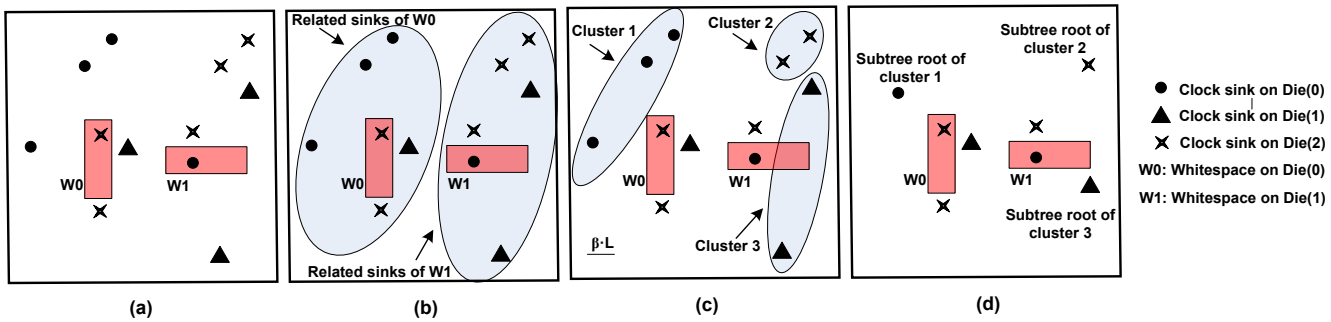

Fig. 4. Sink pre-clustering illustration. (a) Before pre-clustering. (b) Arranging sinks to their related whitespace blocks according to distance. (c) For each whitespace block, generate clusters for its related sinks those are more than $\beta \cdot L$ far away from the whitespace block and on the same die. (d) Subtree roots of the clusters are reserved as new sinks while other nodes in the cluster are neglected.

To solve this problem, an intuitive method is to make sinks distribute closer to whitespace, which is called sink pre-clustering. Firstly, we put all whitespace blocks from different dies on a plane and name it as a whitespace set. Secondly, for each die, we calculate the minimal distance from each sink to the whitespace set through an exhaustive search and arrange the sink to one of the whitespace blocks. Thirdly, we use a designer specified parameter β to control sink pre-clustering. For each die, sinks that have a longer distance from their related whitespace block than the value $\beta \cdot L$ (where L is the half perimeter wirelength of the die) need to be clustered. For each sink cluster, we generate a subtree by using classical method of means and medians (MMM) [9] and DME [10] for clock tree topology generation and detail routing. The root of the subtree is assumed as a new "sink" with its latency and downstream capacitance as input delay and capacitive load, while all the original sinks in the cluster are neglected. After pre-clustering, the sink set contains non-clustered ones and cluster roots. Fig. 4 illustrates the algorithm.

C. TSV Whitespace-Aware 3D-MMM

The basic idea of the famous 3D-MMM algorithm is to recursively divide the given sink set and related TSV bound into two subsets until each sink belongs to its own set. TSVs are necessary when merging nodes on different dies. The algorithm tends to use as many TSVs as the giving bound, but in face of whitespace, this division may cause serious problems. In Fig. 5(a), under current y-cut, sink s1 and s2 from different dies are divided into a subset with no whitespace in it, so a TSV is inserted and moved into the nearest whitespace, which leads to longer wirelength.

To deal with this problem, we modify the 3D-MMM algorithm and extend it to the TWA-3D-MMM algorithm by judging whether the current x/y-cut between multiple dies is appropriate considering whitespace. The pseudo code of the proposed TWA-3D-MMM method is shown in Fig. 6. In line 2, we initial the subset *S1* and *S2*. In line 3, 4, if the current sink set contains only one node, which means it is a sink itself, then return. If not, we execute x/y-cut and divide the current sink set and TSV bound into two subsets when sinks in the current set are distributed on different dies. Then we come to the most important judging procedure (line 11) in our algorithm:

Assuming sink set S is divided into two subsets $S1$ $\{s_{11}, s_{12}, \ldots, s_{1i}\}$ and $S2$ $\{s_{21}, s_{22}, \ldots, s_{2j}\}$ under current x/y-cut, and the maximum and minimum die number of sinks in $S1$ and $S2$ are d_{max1}, d_{min1}, d_{max2}, d_{min2} respectively. In multiple die case $d_{max1} \neq d_{min1}$ and $d_{max2} \neq d_{min2}$. For subset $S1$, all sinks have to be connected, which means TSVs are needed between adjacent dies from Die(d_{min1}) to Die(d_{max1}), so subset $S1$ should contain whitespace on Die(d_{min1}), Die(d_{min1+1}), … , Die(d_{max1-1}), and so does subset $S2$. If one of the subsets does not meet the whitespace conditions, the current cut is canceled and marked to be z-cut. Fig. 5(c) and (d) present a judging example, which would usually happen near the leaf level of the clock tree.

D. DME Merging Segment Reconstruction

There are two phases in the classical DME clock routing method: (1) a bottom-up phase computes all feasible locations for the roots of recursively merged subtrees, saved as related merging segments, and (2) a top-down phase then resolve the exact embedding of these internal nodes [10]. For those internal nodes with TSVs, their related merging segments need to be reconstructed and settled in whitespace.

With the consideration of TSV geometries and whitespace occupation, such as in Fig. 7, firstly we divide the whitespace into many small squares according to the TSV keep-out zone, then we choose the midpoint of the whitespace square with the smallest distance to the merging segment, set it to be the new merging segment, update the delay and downstream

```
TSV Whitespace-aware 3D-MMM Topology Generation (TWA-3D-MMM)
Input: clock sinks, TSV bound, TSV whitespace, cutDirection
Output: a rooted 3D clock tree topology

1:  TWA-3D-MMM (sinkset S, TSV bound B, Whitespace blocks W, cutDirectic
2:      S1 and S2 = subset of S, B1 and B2 = TSV bound of S1 and S2; // initia
3:      if (|S| = 1) then                                   // sink node
4:          return root(S);
5:      else if (B != 1 or stack(S) = 1) then
6:          if (C = x(or y)-cut) then
7:              x(or y)-cut(S, S1, S2);                     // execute current cut
8:              C = y(or x)-cut;                            // change cut directio
9:              Find B1, B2, such that B1 + B2 = B;         // divide TSV bound
10:         if (B != 1) then                                // judging procedure
11:             if (S1 or S2 does not meet the whitespace condition) then
12:                 cancel current cut;
13:                 C = z-cut;
14:                 B = 1;
15:         if (B = 1 and stack(S) > 1) then                // execute z-cut
16:             z-cut(S, S1, S2);
17:             B1 = B2 = 1;
18:     root(S1) = TWA-3D-MMM(S1, B1, C);
19:     root(S2) = TWA-3D-MMM(S2, B2, C);
20:     leftChild(root(S)) = root(S1);
21:     rightChild(root(S)) = root(S2);
22:     return root(S);
```

Fig. 6. Pseudo code of our TWA-3D-MMM

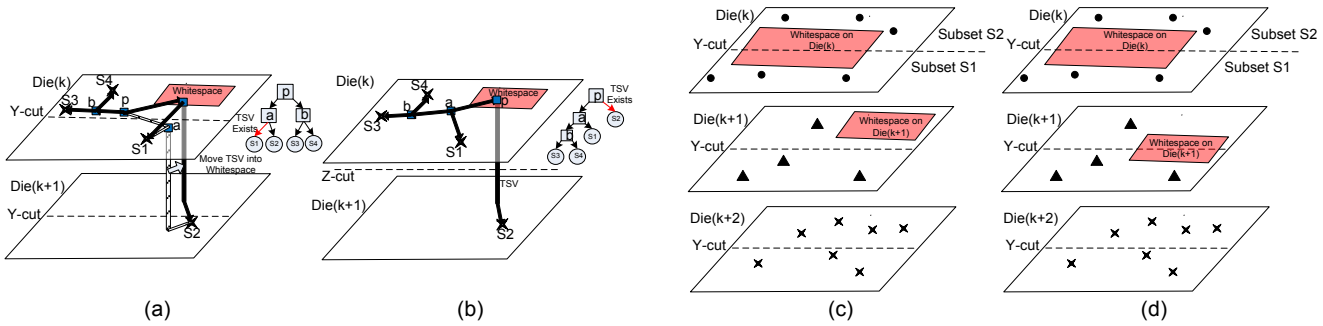

Fig. 5. Comparison of 3D-MMM and our TWA-3D-MMM clock tree topology generation method. (a) When executing current y-cut, there is no whitespace in sink subset {*s1*, *s2*}, so the TSV related parent node a of sink *s1*, *s2* is initially arranged outside whitespace and should be moved to the nearest whitespace place, which would incur longer wirelength and lead to potential skew increase. (b) Since there is no whitespace in sink subset {*s1*, *s2*} ,we change current cut to z-cut, so the TSV is arranged into whitespace without longer wirelength.(c) When judging current y-cut, subset *S1* has no whitespace in Die(*k*+1), so current cut is cancel and change to z-cut. (d) Both subset *S1* and *S2* has whitespace in Die(*k*) and Die(*k*+1), so current cut is successfully.

capacitance of it, and make sure one of the child node might have to do some wire snaking. This movement would lead to intolerable wirelength increase, but fortunately, with the help of sink pre-clustering and TWA-3D-MMM, merging segments will be closer to whitespace. Once a whitespace square is used, it will be recorded as unusable. After merging segment reconstruction, we can execute the DME top-down embedding and get the clock routing result.

E. *Slew-Aware Buffering*

Clock slew rate control is of great importance for high-speed clock design, because a large clock slew rate may cause a setup/hold violation. To ensure the clock signal slew rate, we add a buffering stage to our whitespace-aware 3D CTS flow. Two kinds of buffers are inserted: clock buffers and TSV-buffers [11]. Clock buffers are inserted along the wire to control latency and slew rate, while TSV-buffers are inserted just at each internal node for pre-bond testability. Different from the existing 3D design, which focused on slew-aware buffer insertion during the bottom-up embedding procedure of DME [5, 11, 12], our slew-aware buffering is performed after clock routing for the following reasons: 1) it is easy to achieve with an O(n) time complexity; 2) the buffer delay may change under different supply voltage, so exact zero skew numerical buffer solution during bottom-up embedding procedure of DME under one supply voltage may change under another. In our slew-aware buffering algorithm, clock buffers are added along the clock paths so that the downstream capacitance of each buffer is limited to the bounding condition, which is denoted as CMAX in literature [5]. Long snaking wire paths also need to be buffered. After initial buffer insertion, we insert redundant buffers at the sink node to make sure the buffer numbers from clock source to sinks are balanced. Then, we reduce the buffer number in a bottom-up merging method, i.e. two buffers at each child node could be replaced one buffer at the parent node.

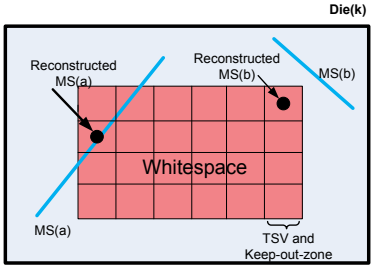

Fig. 7. DME merging segment reconstruction

IV. EXPERIMENTAL RESULTS

A. *Experimental Setup*

We implement our proposed method using C++ programming language on Linux environment with 3GHz processor and 4GB memory. We take use of ISPD 2009 clock network synthesis contest benchmark [13] and 2-die stack for simplicity. For each benchmark, sinks are evenly distributed on each die and whitespace blocks are randomly generated between sinks. In our experiments, we use technology parameters based on the 45nm Predictive Technology Model [14]. The parasitic resistance and capacitance of unit wire length are 0.1 Ohm/um and 0.2fF/um, respectively. The TSV parasitic is assigned to be R_{TSV} = 0.035Ohm and C_{TSV} = 15.48fF, TSV diameter with keep-out-zone = 7.41um [7]. The buffer parameters are: input cap = 35fF, resistance = 61.2Ohm, output parasitic cap = 80fF. The clock frequency is 2GHz, with the supply voltage 1.2V. Note that the runtime of our algorithm is within seconds for all benchmarks.

In SPICE simulation [15], wire and TSVs are segmented by π models. Clock slew rate is defined as the transition time from 10% to 90% of clock signal at each sink and buffer input. The simulated clock slew rate tolerance is 100ps. Although using I, V curve from SPICE simulation to estimate power would be more accurate, it is much more easier and faster to use the CV^2f equation to estimate average power [13]. As a result, we will use total interconnect and buffer capacitance as a measurement of power. The power, skew and slew are reported in fF, ps and ps respectively.

B. *Result Analysis*

1) *Impact of TSV Whitespace Area*

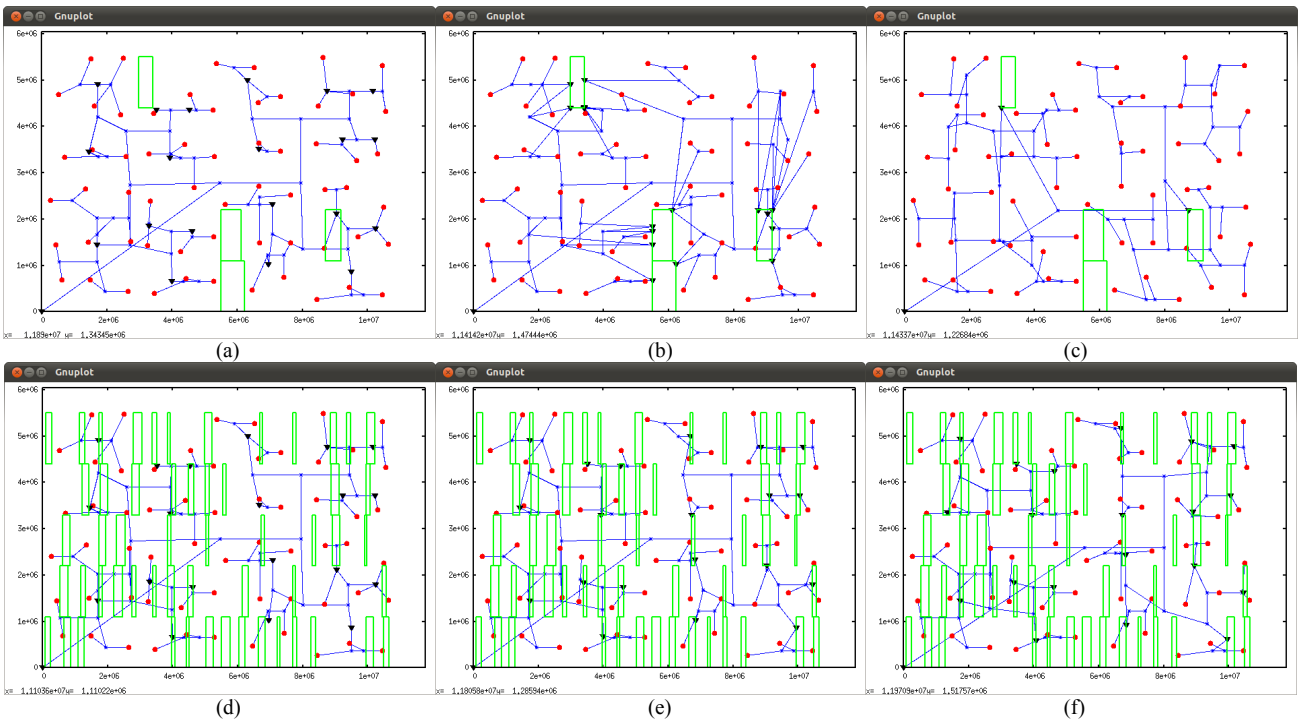

Fig. 8. Clock solution under different whitespace area. Sinks and TSVs are denoted as red points and black triangles respectively, while green rectangles represent whitespace blocks. (a) #block = 4, 3D-MMM-DBM solution before TSV moving, (b) after TSV moving, with longer wirelength; and (c) ours; (d) #block = 55, 3D-MMM-DBM solution before TSV moving, (e) after TSV moving, with longer wirelength, and (f) ours.

To investigate the impact of the whitespace area on clock power consumption and skew, we construct and simulate the entire 3D clock tree by our proposed method on benchmark ispd09f11. We also use the 3D-MMM and DME routing and buffering algorithm for another solution as a contrast. To deal with the whitespace problem, we simply move the internal nodes with their related TSV into the nearest whitespace area, which will significantly increase wirelength. The solution for comparison is named as **3D-MMM-DBM** hereafter.

In Table I, we can observe that the 3D-MMM-DBM method is strongly influenced by the whitespace area. When the area of whitespace is small, such as in Fig. 8 (a) and (b), TSVs have to be moved for a long distance, which would lead to power and skew increase and cause slew violations, although it shows comparable good performance before TSV moving. This long wirelength induced by TSV moving is significantly reduced when whitespace is widely distributed through the whole die area with more TSV arrangement choices, as shown in Fig. 8 (d) and (e). We also present our solutions in Fig. 8 (c) and (f), each TSV is located in the whitespace as expected, with good skew and power results according to Table I. In other words, our method can provide more stable solutions as long as TSV whitespace exists.

2) Exhaustive Search Results For TSV Bound

After exhaustively sweeping the TSV bound from 1 to 50 we observe that in Fig. 9, as the TSV bound increase little by little, the 3D-MMM-DBM solutions suffer from severe power and skew problems, while our method shows consistent good results. This is not difficult to imagine because a larger TSV bound means more TSV moving adjustment, which may worsen clock latency unbalance.

3) β of Pre-clustering

As illustrated in section III.B, β plays an important role in cluster generation. Actually, there exists a β_{max} beyond which pre-clustering is meaningless. This is easy to understand because a big β means that none of the sinks is to be clustered. We can find the secondary longest distance from the sinks to their related whitespace blocks and calculate β_{max}. A sweeping result in Fig. 10 teaches us that we have to do pre-clustering very carefully because a bad choice of β would unnecessarily cluster too many sinks, and affect topology and routing results. Comparatively, β from 90% to 99% of β_{max} would be better.

4) Skew and Power Results

We also compare our method with 3D-MMM-DBM on other benchmarks of ISPD09 contest, the whitespace area is set

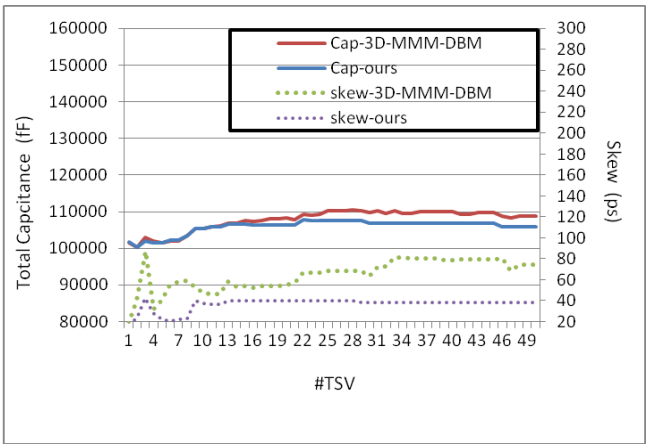

Fig. 9. Skew and power trends for ispd09f11 based on exhaustive search within the TSV range [1, 50] for both 3D-MMM-DBM and our method.

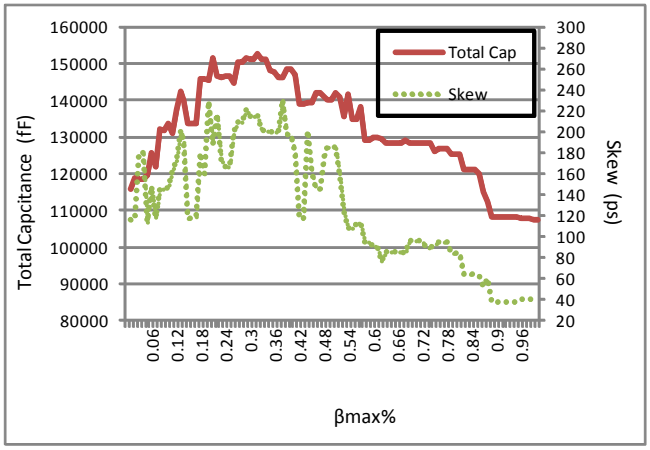

Fig. 10. Clock skew and power trends for ispd09f11 based on exhaustive search within pre-clustering parameter β range (0, 1] β_{max}.

to be around 10% of the whole die area with more than 10 blocks. The results are shown in Table II. We have no slew violation while 3D-MMM-DBM does, and we can reach an average skew reduction of 49.2% and an average power reduction of 1.9% respectively. Note that all the TSVs' locations are restricted to the whitespace blocks, longer wire sometimes is nearly unavoidable, but our method can take account of both power and performance in all cases. We will consider more cases such like multiple dies, uneven sink distribution, and explore the dependency of the whitespace and #TSV with the power reduction in our future work.

V. CONCLUIONS

In this paper, we formulate the whitespace-aware TSV arrangement problem in 3D CTS and propose a practical and efficient algorithm to deal with it. The algorithm consists of three stages: sink pre-clustering, TWA-3D-MMM topology generation, and DME merging segment reconstruction. We also propose a whitespace-aware 3D CTS flow. Experiment results show that our method is more practical and efficient, compared to the traditional 3D-MMM based one with TSV moving adjustment.

REFERENCES

[1] Y. Xie, G. Loh, B. Black, and K. Bernstein, "Design space exploration for 3d architectures," ACM Journal on Emerging Technologies in Computing Systems, vol. 2, no. 2, pp. 65–103, April 2006.

[2] M. Pathak, Y.J. Lee, T. Moon, and S.K. Lim, "Throuth-Silicon-Via management during 3D physical design: when to add and how many?", in ICCAD, November 2010, pp. 387-394.

[3] M.K. Hsu, Y.W. Chang, and V. Balabanov, "TSV-aware analytical placement for 3D IC design", in DAC, June 2011, pp. 664-669.

[4] J. Minz, X. Zhao, and S. K. Lim, "Buffered clock tree synthesis for 3D ICs under thermal variations", in ASPDAC, March 2008, pp. 504-509.

[5] X. Zhao, J. Minz, and S.K. Lim, "Low-power and reliable clock network design for Through-Silicon Via (TSV) based 3D ICs", IEEE Transactions on Components, Packaging and Manufacturing Technology, vol. 1, no. 2, pp. 247-259, February 2011.

[6] T.Y. Kim and T. Kim, "Clock tree embedding for 3D ICs," in ASPDAC, January 2010, pp.486-491.

[7] X. Zhao and S.K. Lim, "Through-Silicon-Via-induced obstacle-aware clock tree synthesis for 3D ICs", in ASPDAC, Feb. 2012, pp. 347-352.

[8] M.C. Tsai, T.H. Wang, and T.T. Huang, "Through-Silicon Via planning in 3-D floorplanning", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 19, no. 8, pp. 1448-1457, August 2011.

[9] M.A.B. Jackson, A. Srinivasan, and E.S. Kuh, "Clock routing for high-performance ICs", in DAC, June 1990, pp. 573-579.

[10] T.H. Chao, Y.C. Hsu, J.M. Ho, and A.B. Kahng, "Zero skew clock routing with minimum wirelength", IEEE Transactions on Circuits and Systems II, vol. 39, no.11, pp. 799-814, November 1992.

[11] X. Zhao, D.L. Lewis, H.H.S. Lee, and S.K. Lim, "Low-power clock tree design for pre-bond testing of 3-D stacked ICs", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 30, no. 5, pp. 732-745.

[12] F.W. Chen, and T.T. Hwang, "Clock tree synthesis with methodology of re-use in 3D IC", in DAC, June 2012, pp. 1094-1099.

[13] ISPD 2009 Clock Network Synthesis Contest benchmark. http://ispd.cc/contests/09/ispd09cts.html

[14] Predictive Technology Model. http://ptm.asu.edu/

[15] NGSPICE http://ngspice.sourceforge.net/

TABLE II. IMPACT OF DIFFERENT TSV BOUND ON DIFFERENT BENCHMARKS BETWEEN 3D-MMM-DBM AND OUR METHOD.

Benchmark #block Area (%)	TSV Bound	3D-MMM-DBM			Our method		
		Skew (ps)	*Power (fF)*	*Slew Vio*	*Skew (ps)*	*Power (fF)*	*Slew Vio*
ispd09f11 #block=16 10.57%	1	18.4	101623	N	17.5	101683	N
	10	47.2	105288	N	37.6	105372	N
	20	55.1	108225	N	40.1	106261	N
ispd09f12 #block=15 9.66%	1	21.9	96097	N	21.9	96195	N
	10	78.8	98646	N	27.0	97233	N
	20	63.9	100512	N	32.3	98291	N
ispd09f21 #block=15 9.97%	1	26.4	103378	N	27.2	103363	N
	10	149	106397	Y	48.1	101492	N
	20	196	117995	Y	46.4	104012	N
ispd09f22 #block=12 7.36%	1	29.9	81961	N	25.4	82052	N
	10	70.5	78994	N	40.2	80224	N
	20	67.6	82225	N	55.1	82355	N
Average	/	68.7	98445	/	34.9	96544	/

TABLE I. IMPACT OF DIFFERENT WHITESPACE AREA ON #TSV, SKEW, POWER AND SLEW BETWEEN 3D-MMM-DBM METHOD AND OUR PROPOSED METHOD (TSV BOUND IS SET TO BE 20, #BLOCK AND #TSV MEANS THE NUMBER OF WHITESPACE BLOCKS AND TSVS, VIO MEANS SLEW VIOLATION).

#block (area %)	3D-MMM-DBM before TSV moving				3D-MMM-DBM after TSV moving				Our method			
	#TSV	*Skew(ps)*	*Power(fF)*	*Vio*	*#TSV*	*Skew(ps)*	*Power(fF)*	*Vio*	*#TSV*	*Skew(ps)*	*Power(fF)*	*Vio*
4 (4.11%)	20	23.805	102920.519	N	20	175.784	123422.868	Y	2	28.575	101368.446	N
9 (5.64%)	20	23.805	102920.519	N	20	148.983	115802.053	Y	9	41.282	108316.412	N
16 (10.03%)	20	23.805	102920.519	N	20	55.135	108224.054	N	14	40.088	106261.184	N
29 (12.86%)	20	23.805	102920.519	N	20	83.057	106533.812	N	18	32.448	106496.828	N
36 (14.47%)	20	23.805	102920.519	N	20	40.938	104757.979	N	19	26.410	106789.130	N
55 (16.23%)	20	23.805	102920.519	N	20	33.221	103869.300	N	20	32.013	104096.017	N
131 (19.79%)	20	23.805	102920.519	N	20	22.977	103480.543	N	20	25.334	104190.917	N