Crosstalk-aware TSV-buffer Insertion in 3D IC

Yen-Hao Chen
yhchen@cs.nthu.edu.tw

Po-Chen Huang
s101062551@m101.nthu.edu.tw

Fu-Wei Chen
fwchen@cs.nthu.edu.tw

Allen C.-H. Wu
allenwuuw@gmail.com

TingTing Hwang
tingting@cs.nthu.edu.tw

Department of Computer Science, National Tsing Hua University, Taiwan

2019 32nd IEEE International System-on-Chip Conference (SOCC) 978-1-7281-3483-3/20/$31.00 ©2020 IEEE 10.1109/SOCC46988.2019.1570539111

ABSTRACT

3D integration is one of the promising technologies to alleviate interconnection delay. Implementing 3D IC is to integrate 2D ICs with Through-Silicon Vias (TSVs). For yield consideration, TSVs are bundled together as a TSV block [1]. Regrettably, this placement will result in crosstalk coupling noises in TSV block, which may cause significant timing degradation. Traditionally, buffer sizing is one of the effective methods to solve the problem. However, we have observed that increasing the TSV-buffer size of aggressor TSV will cause serious timing degradation to the victim TSV in 3D than wires in 2D cases. In this paper, we develop a delay model of a victim TSV surrounded by aggressor TSVs with different driving TSV-buffer sizes. Based on the TSV delay model, we propose (1) an ILP (Integer Linear Programming) method, which is able to find the near-optimal solution, and (2) an efficient crosstalk-aware heuristic method for practical use. Our experimental results show that the proposed heuristic method only uses 2.56% (3.05%) more TSV-buffers compared to the optimal ILP solution and achieves on average 32.88% (42.40%) and 18.21% (23.06%) area reduction of area-overheads compared to the conventional greedy [2] and separator sets [3] methods in our 2-tier (4-tier) benchmark circuits.

I. INTRODUCTION

With the shrinking process technology, the interconnection delay becomes the dominant factor of circuit performance. The Three-Dimensional Integrated Circuit (3D IC) is a promising technology to alleviate interconnection delay in VLSI or even larger scale integrations. Among others, Through-Silicon Vias (TSVs) connecting signals in different tiers is the key technology. Long global wires in 2D IC can be shortened by passing through TSVs with 3D integration. Hence, the wire delay and power consumption can significantly be reduced. Moreover, 3D IC technology gives more design flexibility by heterogeneous integration [1].

For yield consideration, TSVs are usually bundled together as a TSV block [1] rather than isolated and spread in the whole plane. In a TSV block, TSVs are placed regularly as an array [1]. Regrettably, this placement will result in serious coupling noises for a victim TSV from surrounding aggressor TSVs. Due to large dimension of a TSV, hence large coupling noise caused by nearby TSVs, signal integrity issues in 3D IC are more critical than 2D IC [4]. The coupling noise may significantly degrade circuit performance and even result in wrong logic functionality [5]. Previous study has shown that coupling effects are from TSV-to-TSV coupling, TSV-to-device coupling, and TSV landing pad to device coupling. Among the above three coupling effects, TSV-to-TSV coupling is the major contributor [6].

Several techniques have been proposed to reduce the TSV-to-TSV coupling effect. In Liu et al. [6] work, TSV spacing, TSV shielding, and buffer inserting are shown to improve signal integrity and timing effectively. Among all these techniques, it is also shown that buffer insertion is the most practical one due to less area overhead. However, no specific buffer sizing method was proposed in these papers. The study in [7] focuses on data coding to avoid transmitting patterns causing crosstalk. However, it often incurs significant overhead of encoder, decoder and extra TSVs.

In this paper, we propose a buffer insertion method to solve the crosstalk problem of TSVs in 3D IC. We observe that a large size of driving buffer of TSV will cause serious timing degradation to neighboring TSV in 3D than wires in 2D. Our method will take mutual effect of TSV-to-TSV coupling noise into consideration. We develop a delay model of victim TSV surrounding by aggressor TSVs considering different sizes of driving buffers. Then, ILP based and crosstalk-aware heuristic buffer insertion methods are proposed to select the most suitable buffer size so that timing constraint is satisfied.

The rest of this paper is organized as follows. In Section II, we introduce our motivation. Section III defines our problem and Section IV presents our derived equation and proposed method. Section V shows our experimental results. Finally, we conclude this paper in Section VI.

II. MOTIVATION

A. *Timing Effect of Up-sizing Buffer*

In the 2D model, simply up-sizing the buffer of victim can effectively solve the timing violation of victim wire due to crosstalk. The up-sizing of victim wire often slightly degrades the performance of its neighboring wires. However, signals in TSV block will affect each other significantly. A sized-buffer of a victim TSV may become a serious aggressor of its neighboring wires.

Our observation on the delay model of victim wire in 2D and victim TSV in 3D shows that the coupling capacitance of two TSVs is much larger than that of two wires. Hence, a victim TSV is more sensitive to the buffer size of its aggressor [4]. We use the following experiment to demonstrate that buffer sizing affects victim wire differently in 2D and 3D cases. We compare the transition delay of victim wire in 2D and victim TSV in 3D with the same length but with different buffer sizes of neighboring aggressor by SPICE simulation. The simulation setup is as follows. The crosstalk models [6], [8] shown in Figure 1 are adopted for wires in 2D and TSVs in 3D. Both lengths of wire and TSV segments are $80um$. (Detailed experimental parameters will be given in Section V.)

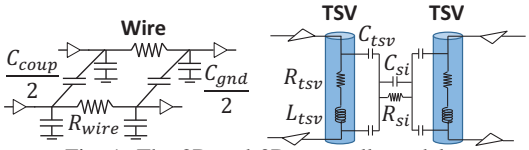

Fig. 1: The 2D and 3D crosstalk models

In our SPICE simulation, we consider the worst case transitions of victim wire in 2D and TSV in 3D, where all aggressor signals are switching in opposite direction to the victim signal. In the 2D case, a victim wire is affected by two adjacent wires in the same planar. In 3D case, we consider four adjacent TSVs as aggressors, because they contribute most of the coupling effect [5]–[8]. We set the buffer size of victim as $8x$ and observe the timing degradation of victim when buffer sizes of aggressors are increased from $2x$ to $8x$. Simulation results are shown in Figure 2. The timing of victim surrounded by aggressors with $2x$ buffer size is set as baseline. From this figure, we observe that when the buffer size of aggressors is $4x$, the timing degradation is 8.77% in 2D case while it is 22.61% in 3D case. When the buffer size is increased to $8x$, the degradations are 14.44% in 2D case and 31.12% in 3D case, respectively. It has shown that the timing degradation of TSV in 3D is much larger than the wire in 2D.

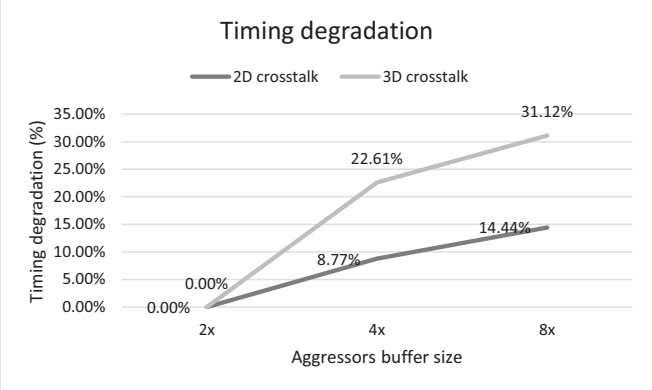

Fig. 2: Timing degradation of different buffer size of aggressors

B. Motivation Example

The study in [6] showed that buffer insertion can reduce path delay significantly by reducing victim driving port impedance in 3D IC. Regrettably, in a TSV block [1], TSVs often affect each other significantly. A victim TSV may become a serious aggressor to adjacent TSVs. Figure 3 shows our observation. Five TSVs with different buffer sizes are shown in Figure 3(a). Let timing constraint be $500ps$, TSV A the most critical TSV with delay $530ps$, TSV B the second critical TSV with delay $495ps$, and others non-critical. If the buffer size of TSV A is increased from $4x$ to $8x$ to meet the timing constraint as shown in Figure 3(b), the delay of TSV A is significantly reduced by $250.33ps$, and TSV A becomes a non-critical TSV with delay $279.67ps$. However, TSV A also becomes a serious aggressor of TSV B. In this case, the timing of TSV B is increased by $8.23ps$, and TSV B becomes the most critical TSV with delay $503.23ps$. Thus, the timing constraint of $500ps$ cannot be satisfied by up-sizing TSV A. A better solution can be taking the coupling effect between TSVs into consideration as shown in Figure 3(c), where we down-size buffer sizes of surrounding TSVs to $2x$. By doing so, the delay of TSV A is decreased by $40.61ps$, and TSV A becomes a non-critical TSV with delay by $489.39ps$. Furthermore, no other critical TSV is introduced, and the timing constraint of the whole circuit is satisfied.

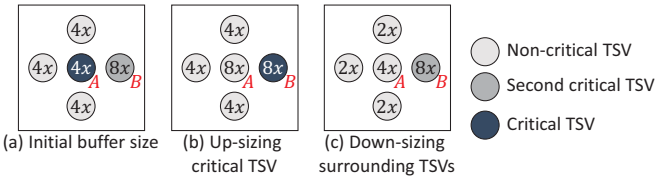

Fig. 3: An example of crosstalk effect on TSVs

III. Design Flow and Problem Formulation of TSV-Buffer Sizing Problem

A. Global Flow

The global flow is shown in Figure 4. Given a circuit, firstly, 3D partitioning and floorplanning are performed to decide the location of blocks. After that, TSV blocks, i.e. array of TSVs [1], are allocated in the white space among blocks, and TSVs are assigned to TSV blocks. Then, unit-sized TSV driving buffers are inserted. After the above steps, our buffer sizing method for timing optimization is performed. If the buffer sizing method cannot find a solution to meet the timing constraint, a timing failure is reported. Otherwise, a circuit satisfying timing constraint is found. In this paper, our problem focuses on the **timing optimization by TSV-buffer sizing** stage.

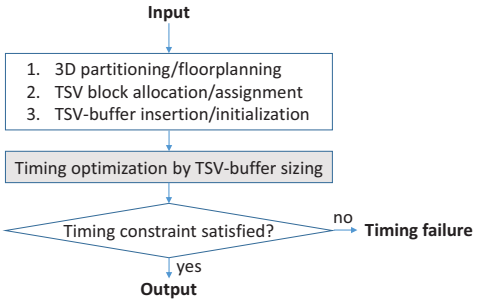

Fig. 4: The global flow diagram

B. Problem Formulation for TSV-buffer Sizing

The input to our problem, i.e. the step of **timing optimization by TSV-buffer sizing**, is a Directed Acyclic Graph (DAG), $G(V, E)$, representing the circuit. Node set V includes all primary inputs PI, primary outputs PO, blocks BLK, and TSVs TSV, i.e. $V = PI + PO + BLK + TSV$. Also, edge set E presents the connecting wires of the circuit. In addition, each node or edge has a corresponding delay value representing I/O pad delay, block delay, TSV delay, or wire delay.

The problem can be formulated as follows. Let the set of possible TSV driving buffer sizes be $SIZE$, and the area of an s-sized TSV-buffer denoted as $area_s$. Also, one-hot encoding is adopted to represent the size of TSV-buffer. Given a TSV t, binary variable n_s^t is one if and only if TSV t uses an s-sized TSV-buffer, otherwise n_s^t is zero, $\forall t \in TSV$. The objective of our problem is to minimize the total TSV-buffer area under timing constraint as shown in Equation (1).

$$\begin{aligned} &\text{Minimize}: \sum_{t \in TSV} \sum_{s \in SIZE} area_s \times n_s^t \\ &\text{s.t. } delay(out) \leq timing\_constraint, \forall out \in PO \end{aligned} \tag{1}$$

The delay of node n is calculated as shown in Equation (2).

$$delay(n) = \max_{p \in pred_n} \{delay(p) + edge\_delay(p,n) + node\_delay(n)\} \quad (2)$$

The $pred_n$ represents all predecessors of n. The $delay(p)$ represents the delay of predecessor, p. The wire delay, $edge\_delay(p,n)$, is proportional to the hamming distance between nodes, i.e. $edge\_delay(p,n) = unit\_wire\_delay \times wire\_length_{p,n}$ The $node\_delay(n)$ represents the I/O pad delay, block delay, and TSV delay. The I/O pad and block delay are given. For simplicity, we assume no I/O pad and block delay in this paper.

When a node, n, is a TSV, the computation of node delay, $delay(n)$, requires to take crosstalk into consideration. To obtain the TSV delay model considering crosstalk effect, we perform a comprehensive SPICE simulation for the TSV delay model discussed in Section II-A. In this paper, we only consider the crosstalk effect caused by the neighboring TSVs, because they contribute the largest portion of crosstalk effect [5]–[8]. Thus, there are four cases, where a victim TSV is surrounded by one, two, three, and four neighboring TSVs (or aggressor TSVs) as shown in Figure 5(a), (b), (c), and (d), respectively. The SPICE simulation results are recorded in a table, $Delay\_Table(size^v, SIZE^a)$. In this table, $size^v$ and $SIZE^a$ represent the buffer size of victim TSV, v, and the set of buffer sizes of aggressor TSVs. Table I shows a demonstration of $Delay\_Table(size^v, SIZE^a)$. Take Figure 5(d) as an example. If a victim TSV, v, with $4x$ buffer size is surrounded by four aggressor TSVs with $2x$, $2x$, $4x$, and $8x$ buffer sizes. The delay of the victim TSV can be referenced by $size^v = 4x$ and $SIZE^a = \{2x, 2x, 4x, 8x\}$ in the table, i.e. $517.52ps$. The size of the $Delay\_Table(size^v, SIZE^a)$ is computed as follows. The number of rows is the total number of possible victim TSV buffer sizes, i.e. $|SIZE|$. Also, let i be the number of aggressor TSVs, where $i \in \{1,2,3,4\}$. Then, the total possible number of combinations of aggressor TSVs is $H_i^{|SIZE|}$. Hence, the total number of columns is $\sum_{i=1}^{4} H_i^{|SIZE|}$. Therefore, the overall size of the $Delay\_Table(size^v, SIZE^a)$ is $|SIZE| \times \sum_{i=1}^{4} H_i^{|SIZE|}$.

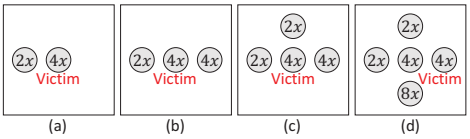

Fig. 5: A victim TSV surrounded by aggressor TSVs

TABLE I: A demonstration of the delay table

	$SIZE^a$						
$size^v$	(ps)	$\{2x\}$	...	$\{2x, 4x\}$	...	$\{2x, 2x, 4x, 8x\}$	...
	$2x$	490.24	...	726.9	...	1285.1	...
	$4x$	221.16	...	313.78	...	517.52	...
	$8x$	112.11	...	146.25	...	223.51	...
	...	...	...	...	...	...	...

IV. TSV-Buffer Sizing Algorithms

This section presents our area optimization methods by TSV-buffer sizing under a timing constraint. Section IV-A shows an Integer Linear Programming (ILP) method, follows by a Crosstalk-Aware Heuristic (CAH) method in Section IV-B.

A. *ILP Method and Linear Delay Model*

The first attempt is to use ILP to find the optimal solutions to the problems formulated in Section III. First, TSV delay in Equation (2) is modeled non-linear as shown in the example of Table I. Hence, we need to develop linear equations to approximate the delay of victim TSVs with minimal deviation from the SPICE simulation results. The linear delay equation of victim TSV v without any aggressor TSV is modeled in Equation (3).

$$TSV\_delay(v) = \sum_{s \in SIZE} resis_s^0 \cdot n_s^v \quad (3)$$

In Equation (3), binary variables, $n_s^v, \forall s \in SIZE$, indicate the buffer size of victim TSV v, in which one and only one of them will be 1 and the others will be 0, i.e. one-hot encoding. In addition, coefficient, $resis_s^0$, represents the resistance delay when using an s-sized buffer. In this case, without any aggressor TSV, $resis_s^0$ can be directly obtained from SPICE simulation without any deviation. On the other hand, the linear delay equation of victim TSV v surrounded by N_aggs aggressor TSVs is modeled in Equation (4), where $N\_aggs \in \{1,2,3,4\}$.

$$\begin{aligned} &TSV\_delay(v) \\ &= \sum_{s \in SIZE} resis_s^{N\_aggs} \cdot n_s^v + \sum_{i=1}^{N\_aggs} \sum_{s \in SIZE} cross_s^{N\_aggs} \cdot n_s^{a_i} \\ &= \sum_{s \in SIZE} resis_s^{N\_aggs} \cdot n_s^v + \sum_{s \in SIZE} cross_s^{N\_aggs} \cdot N\_aggs_s^v \end{aligned} \quad (4)$$

In Equation (4), binary variables, $n_s^{a_i}, \forall s \in SIZE$, indicate the buffer size of aggressor TSV a_i. Similarly, given any aggressor TSV a_i, exactly one of the binary variables, $n_s^{a_i}, \forall s \in SIZE$, will be 1 and the others will be 0. Also, integer $N\_aggs_s^v$ is the total number of aggressor TSVs with s-sized buffers of the victim TSV v, i.e. $N\_aggs_s^v = |\{a_i | n_s^{a_i} = 1\}|$. In addition, the coefficient, $cross_s^{N\_aggs}$, represents crosstalk delay from an aggressor TSV with s-sized buffer, when the victim TSV is surrounded by N_aggs aggressor TSVs. Those linear equations, $TSV\_delay(v)$, will be used by ILP to compute TSV delays.

To find the values of all coefficients $resis_s^{N\_aggs}$ and $cross_s^{N\_aggs}$, we use the linear regression method to approximate the SPICE simulation results with minimal deviation. First, the deviation (or loss function) is defined in Equation (5).

$$\frac{1}{2} \sum_{\substack{size^v \\ SIZE^a}} (TSV\_delay(v) - Delay\_Table(size^v, SIZE^a))^2 \quad (5)$$

Equation (6) shows the equivalent deviation matrix form.

$$\begin{aligned} &\frac{1}{2}(\mathbf{n}^{\mathrm{T}}\mathbf{d} - \mathbf{D})^2, \\ &\text{where } \mathbf{n} = \begin{bmatrix} ... & ... & ... \\ ... & n_{s_1}^v & ... \\ ... & n_{s_2}^v & ... \\ ... & ... & ... \\ ... & N\_aggs_{s_1}^v & ... \\ ... & N\_aggs_{s_2}^v & ... \\ ... & ... & ... \end{bmatrix}, \mathbf{d} = \begin{bmatrix} resis_{s_1}^{N\_aggs} \\ resis_{s_2}^{N\_aggs} \\ ... \\ cross_{s_1}^{N\_aggs} \\ cross_{s_2}^{N\_aggs} \\ ... \end{bmatrix}, \\ &\text{and } \mathbf{D} = \begin{bmatrix} ... \\ Delay\_Table(size^v, SIZE^a) \\ ... \end{bmatrix} \end{aligned} \quad (6)$$

The matrix $\mathbf{n}$ and the vector $\mathbf{D}$ are known constants, while $\mathbf{d}$ is the required vector. Minimal deviation can be easily derived by taking first derivative of Equation (6) w.r.t. vector $\mathbf{d}$ and letting it be zero.

$$\mathbf{d} = (\mathbf{n}\mathbf{n}^{\mathrm{T}})^{-1}\mathbf{n}\mathbf{D} \tag{7}$$

Notice that the delay model of victim TSVs surrounded with different number of aggressor TSVs are separately modeled in different linear systems, i.e. different values of $N\_aggs \in \{1, 2, 3, 4\}$. By this modeling, we compute the coefficients and compare the linear model with the SPICE simulation results. We found that the maximum error rate of the obtained linear equations is only 4.27%. With the above linear delay model, $TSV\_delay(n)$, we can rewrite Equation (1) and (2) into following integer linear equations.

$$\begin{aligned}
&\text{Minimize}: \sum_{t \in TSV} \sum_{s \in SIZE} area_s \times n_s^t \\
&\text{s.t. } timing_constraint \geq delay_n \\
&\geq \begin{cases} delay_p + edge_delay_{p,n} + TSV_delay(n), & \text{if } n \in TSV \\ delay_p + edge_delay_{p,n}, & \text{otherwise} \end{cases} \\
&\geq 0, \forall \text{node } n, \forall p \in pred_n,
\end{aligned} \tag{8}$$

where $delay_n$ represents the delay of node n, $edge\_delay_{p,n}$ the wire delay between nodes p and n. Finally, we can use an ILP solver to find the optimal solution with minimal TSV-buffer area.

B. *Crosstalk-aware Heuristic Method (CAH)*

Although the proposed ILP method can find the optimal solution for the buffer sizing problem. However, ILP is an NP-complete problem. As the number of binary variables (i.e. the number of TSVs in the circuit) increases, the execution time grows exponentially. Also, ILP requires excessive execution time for a circuit when a tight timing constraint is given.

The greedy method [2] is one of the most widely used heuristic method for 2D buffer sizing. It cautiously finds the most critical node to up-size by analysing the delay model of the circuit. Since each buffer size is based on a complete delay analysis, it can easily apply complex delay model considering multiple different physical and chemical effects. However, improving a single most critical node does not necessarily mean the overall critical paths of the circuit can be improved. Another commonly used method for 2D buffer sizing is the min-cut method. Instead of up-sizing a single critical node in each iteration, it applies a min-cut algorithm to find a minimum set of nodes to improve all critical paths at the same time. Usually, a min-cut method may converge faster than the greedy method on a large circuit. More importantly, min-cut methods have a global view on the critical paths which may find a better solution in a general sense. However, conventional min-cut methods do not consider the crosstalk effect of TSVs in 3D IC.

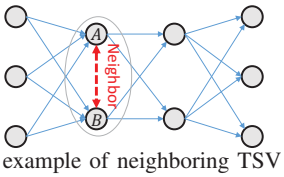

Fig. 6: An example of neighboring TSV selection

With the complex TSV crosstalk effect to be considered in the delay model as shown in Section II-A, it is very difficult to predict the complex TSV crosstalk effect when up-sizing various TSV-buffers simultaneously. In addition, the crosstalk effect is based on the physical location of TSVs, which makes it difficult to model the crosstalk effect into the min-cut cost function. For example, the min-cut method may select A and B two TSVs for up-sizing as shown in Figure 6. If these two TSVs are far apart, the timing and area are correctly modeled and the selection can result in timing reduction. However, if these two TSVs are physically neighboring TSVs, it may even aggravate the crosstalk effect. The timing impact by up-sizing cannot reflect the crosstalk effect.

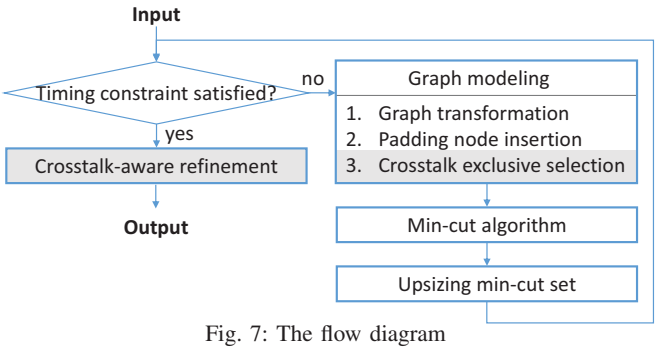

Fig. 7: The flow diagram

With the above observations, we are going to present our method based on the min-cut algorithm to solve the TSV-buffer sizing problem by considering neighboring effect.

Figure 7 shows the flow diagram of our method. First, the circuit is analysed by the crosstalk delay model and checked whether the timing constraint is satisfied. If not, the graph modeling stage will be performed for critical path extraction, which consists of three steps, i.e. graph transformation, padding node insertion [3], and crosstalk exclusive selection. After extracting the critical paths, we can use the min-cut algorithm to find the minimum cut set of the critical paths and up-size TSV-buffers of the cut set to the original circuit. This loop is iterated until the timing constraint of the circuit is satisfied. Lastly, we will perform the crosstalk-aware refinement to further reduce the area, which will be discussed later. The main steps that consider the 3D crosstalk effect are the crosstalk exclusive selection of the graph modeling stage and the crosstalk-aware refinement stage.

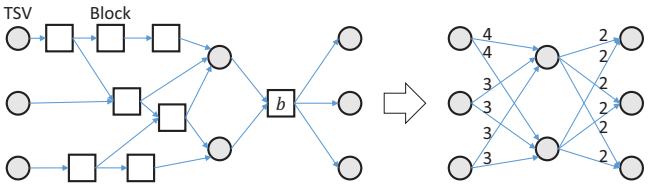

Fig. 8: An example of graph transformation

First, only TSV nodes are valid nodes as buffer sizing candidates. If we directly apply DAG of 3D circuit by the min-cut algorithm, it may find a set with unsizable nodes, i.e. block nodes. Figure 8 shows an example of DAG of 3D circuit on the left hand side where circles are TSV nodes and squares are block nodes. By the figure, the min-cut algorithm will choose block b to up-size, since it is the smallest cut set to optimize the whole circuit. To avoid the min-cut algorithm cutting at block nodes, a DAG transformation is applied. The block nodes in the original graph are removed. Also, a new edge of two TSV nodes is added, if there is a path of block nodes between two TSV nodes. In addition, the delay value of the new edge is the longest path delay of block nodes between the two TSV nodes. Figure 8 shows an example of this representation transformation. Notice that

this graph transformation is still equivalent to the original graph. The total number of nodes is reduced after the transformation. More importantly, each node on the new graph represents a valid buffer sizing candidate, which can easily apply the min-cut algorithm.

The padding node insertion technique proposed by Y. Tamiya [3] is adopted in our method. Y. Tamiya found that using the min-cut method to 2D buffer sizing problem directly might result in suboptimal solution [3]. Thus, they proposed to insert padding nodes on non-critical edges of the circuit and iteratively perform the min-cut algorithm to find several min-cut sets, so called separator sets. By the paper [3], if an edge e has a tail node (the input block) with a larger timing slack than the head node (the output block), then a zero cost padding node is inserted in the edge e.

Algorithm 1 Crosstalk exclusive selection

```
/* Step 1: Select most critical nodes */
selected_V ← ∅
conflicting_V ← ∅
for v : most_critical_path do
    selected_V ← selected_V ∪ {v}
    conflicting_V ← conflicting_V ∪ v.neighbors
end for

/* Step 2: Mutually include critical nodes */
sorted_TSV_list ← sortTSVsBySlack()
for v : sorted_TSV_list do
    if v ∉ conflicting_V then
        selected_V ← selected_V ∪ {v}
        conflicting_V ← conflicting_V ∪ v.neighbors
    end if
end for

/* Step 3: Add corresponding edges and output */
E ← {edge(u, v)|∃edge(u, v), ∀u, v ∈ selected_V}
return graph(selected_V, E)
```

As mentioned previously, crosstalk model has a neighboring effect as shown in Section II-A. The most serious crosstalk effect is from neighboring TSVs [5]–[8]. However, it is unlikely to incorporate such comprehensive timing model into the cost function of the min-cut algorithm, because whether the neighboring nodes are selected or not or which type of buffer is selected is unknown during the cost function computation. Thus, if neighboring TSVs are indeed selected at the same time, it results in serious crosstalk effect and inaccurate cost function of min-cut algorithm. To avoid such scenario, the crosstalk exclusive selection step is proposed. It avoids neighboring TSVs being selected by extracting sub-networks. In each iteration no neighboring TSVs is in the network. Algorithm 1 gives the crosstalk exclusive selection step. There are three steps in the algorithm. First, the nodes on the most critical path are selected and recorded in $selected\_V$. By doing so, the connectivity of primary input and output is guaranteed. Next, nodes that are adjacent **physically** to the selected nodes are conflict nodes which are recorded in $conflict\_V$. In Step 2, we want to include as many nodes into the sub-network as possible. Thus, unselected nodes are sorted by their timing slacks from small to large (from more to less criticality). The most critical nodes that is not physically adjacent to the current sub-network in $selected\_V$ are included. The iteration continues until $sorted\_TSV\_list$ is empty. Finally, the extracted graph is generated by adding the corresponding edges as shown in Step 3.

Once a valid circuit is obtained, we can further reduce area by the following two refinement techniques. First, down-sizing a TSV-buffer may not only increase the delay on current TSV but also reduce the delay of neighboring TSVs as shown in Figure 3. Thus, if a critical path includes some neighboring TSVs, down-sizing one of them may not worsen the delay as expected as the conventional 2D timing model without considering crosstalk effect. To deal with this scenario, we down-size each TSV-buffer and check whether the slack is still positive. If it is positive, then an over sized TSV-buffer is found and can be down-sized to further reduce TSV-buffer area. Second, heuristic algorithms can only find local optimal solutions. A better solution may be found by injecting random permutation. Thus, we up-size (down-size) each TSV-buffer and re-check the over sized buffers till no more area can be reduced.

V. Experimental Results

The proposed method is implemented in C/C++ language on Linux environment under an AMD FX(tm)-8350 processor with 32GB memory. The ILP solver GLPK is used to solve the ILP formulations. The GSRC Hard-Block Benchmark and MCNC Hard-Block Benchmark are used as benchmark circuits. Benchmark circuits are partitioned into 2-tier (4-tier) by a 3D IC floorplan tool [9]. Table II shows the benchmark circuits. In our experiments, technology parameters are based on the 45nm Predictive Technology Model with supply voltage 1V. TSV technology parameters and formulation are referenced from the previous work [5], [6], [10], [11]. The height, radius, and pitch of a TSV are $80um$, $6um$, and $36um$, respectively. The resistance, capacitance, and inductance of TSV are $11.87m\Omega$, $54.85fF$, and $41.1pH$, respectively. The resistance and capacitance of substrate are $79.28k\Omega$ and $15.01fF$. Based on the 45nm Open Cell Library, three buffer sizes are used, i.e. $2x$, $4x$, and $8x$, and their areas are $1.064um^2$, $1.862um^2$, and $3.456um^2$, respectively. In addition, wire resistance R_{wire}, ground capacitance C_{gnd}, and wire coupling capacitance C_{coup} are 152.77Ω, $2.26fF$ and $1.60fF$, respectively, and, from SPICE simulation, the $unit\_wire\_delay$ is $1.1632ps/um$.

TABLE II: Benchmarks

2-tier			4-tier		
name	#blocks	#TSVs	name	#blocks	#TSVs
apte.2tier	9	51	apte.4tier	9	151
xerox.2tier	10	122	xerox.4tier	10	262
hp.2tier	11	48	hp.4tier	11	102
ami33.2tier	33	67	ami33.4tier	33	117
ami49.2tier	49	217	ami49.4tier	49	504
n10.2tier	10	55	n10.4tier	10	128
n30.2tier	30	148	n30.4tier	30	382
n50.2tier	50	223	n50.4tier	50	535
n100.2tier	100	352	n100.4tier	100	760
n200.2tier	200	694	n200.4tier	200	1546
n300.2tier	300	823	n300.4tier	300	1862

Compared to the crosstalk of wires in 2D ICs, which of TSVs in 3D ICs has a more aggregate effect as shown in Figure 2. To confirm the observation, comparison by ILP with and without using the crosstalk model is performed. The experimental result shows that ILP method without using the crosstalk model may underestimate the circuit timing up to 50%. By the above analysis, it is clear that TSV crosstalk effect should be explicitly considered. Thus, in the following experiments, the crosstalk model is used in all methods.

Tables III shows the area comparisons between the Crosstalk-Aware Heuristic (CAH) and the ILP methods. Due to lack of memory space,

TABLE III: The crosstalk-aware heuristic (CAH) v.s. ILP

(um^2)	apte.2tier	xerox.2tier	hp.2tier	ami33.2tier	ami49.2tier	n10.2tier	n30.2tier	n50.2tier	n100.2tier	Avg.
ILP	3.192	2.392	4.786	28.706	31.104	22.326	54.23	55.828	90.916	100.00%
CAH	3.192	2.392	4.786	28.706	32.702	22.326	55.824	62.206	94.112	102.56%

(um^2)	apte.4tier	xerox.4tier	hp.4tier	ami33.4tier	n10.4tier	Avg.
ILP	16.75	23.13	36.678	38.28	80.548	100.00%
CAH	17.548	23.934	36.678	39.076	84.532	103.05%

TABLE IV: Heuristic methods

	ami33.2tier	ami49.2tier	n30.2tier	n50.2tier	n100.2tier	n200.2tier	n300.2tier	Avg.
Greedy [2]	31.894	95.688	62.202	69.38	115.634	252.004	173.858	132.88%
Separator sets [3]	31.894	39.88	59.014	66.988	117.234	231.27	177.85	118.21%
CAH	28.706	32.702	55.824	62.206	94.114	187.418	120.434	100.00%

	ami33.4tier	ami49.4tier	n30.4tier	n50.4tier	n100.4tier	n200.4tier	n300.4tier	Avg.
Greedy [2]	45.452	151.508	161.902	161.088	163.496	267.146	180.246	142.40%
Separator sets [3]	47.046	151.51	143.562	158.696	139.564	171.492	133.21	123.06%
CAH	39.076	147.526	117.258	145.94	97.31	132.42	109.294	100.00%

only 9 out of 11 (2-tier) and 5 out of 11 (4-tier) benchmarks have successfully completed by the ILP solver. The results also show that our proposed crosstalk-aware heuristic performs very effectively, on average, using 2.56% (2-tier) and 3.05% (4-tier) more TSV-buffer area compared to the ILP method.

Since there is no previous work focused on the TVS-buffer sizing to deal with 3D crosstalk effect, we implemented greedy [2] and separator sets [3] methods as references. Tables IV shows the comparisons between three heuristics, namely Greedy, separator sets, and our CAH. We only report the cases with more than 30 blocks to save space. The greedy method each time only selects a single buffer to up-size without considering the global view, which often results in an inferior solution. On the other hand, the separator sets method shows a more stable performance over all benchmarks. Nevertheless, all those two methods do not consider the crosstalk effect. As a result, our proposed method has outperformed the greedy and separator sets methods, on average, 32.88% and 18.21% (42.40% and 23.06%) in TSV-buffer area reductions on 2-tier (4-tier) benchmark circuits.

VI. Conclusions

In this paper, we have observed that TSV driving buffer causes large degradation in 3D designs than in 2D designs. Based on this observation, TSV-buffer sizing methods are proposed for 3D ICs. We presented an ILP method to find the global optimum solution and a runtime efficient crosstalk-aware heuristic method. The results show that the crosstalk-aware heuristic method, on average, only uses 2.56% (3.05%) more TSV-buffer area compared to the ILP method on 2-tier (4-tier) benchmark circuits. Also, on average, it achieves 32.88% and 18.21% (42.40% and 23.06%) less TSV-buffer area compared to the greedy [2] and separator sets [3] methods on 2-tier (4-tier) benchmark circuits. From this study, we have learned that using the conventional 2D buffer sizing methods to solve 3D IC problem will result in excessive buffer area overheads. By taking into account 3D IC crosstalk effects, our proposed methods can provide much effective solutions.

References

[1] A. C. Hsieh, T. Hwang, M. T. Chang, M. H. Tsai, C. M. Tseng, and H. C. Li, "Tsv redundancy: Architecture and design issues in 3d ic," in *2010 Design, Automation Test in Europe Conference Exhibition (DATE 2010)*, pp. 166–171, March 2010.

[2] I. Liu, A. Aziz, D. F. Wong, and H. Zhou, "An efficient buffer insertion algorithm for large networks based on lagrangian relaxation," in *Proceedings of the IEEE International Conference On Computer Design, VLSI in Computers and Processors, ICCD '99, Austin, Texas, USA, October 10-13, 1999*, pp. 210–215, IEEE Computer Society, 1999.

[3] Y. Tamiya, "Performance optimization using separator sets," in *Proceedings of the 1999 IEEE/ACM International Conference on Computer-Aided Design, 1999, San Jose, California, USA, November 7-11, 1999* (J. K. White and E. Sentovich, eds.), pp. 191–194, IEEE Computer Society, 1999.

[4] K. N. Tu, "Reliability challenges in 3d IC packaging technology," *Microelectronics Reliability*, vol. 51, no. 3, pp. 517–523, 2011.

[5] R. Weerasekera, M. Grange, D. Pamunuwa, H. Tenhunen, and L. Zheng, "Compact modelling of through-silicon vias (tsvs) in three-dimensional (3-D) integrated circuits," in *IEEE International Conference on 3D System Integration, 3DIC 2009, San Francisco, California, USA, 28-30 September 2009*, pp. 1–8, IEEE, 2009.

[6] C. Liu, T. Song, J. Cho, J. Kim, J. Kim, and S. K. Lim, "Full-chip tsv-to-tsv coupling analysis and optimization in 3d IC," in *Proceedings of the 48th Design Automation Conference, DAC 2011, San Diego, California, USA, June 5-10, 2011* (L. Stok, N. D. Dutt, and S. Hassoun, eds.), pp. 783–788, ACM, 2011.

[7] R. Kumar and S. P. Khatri, "Crosstalk avoidance codes for 3d VLSI," in *Design, Automation and Test in Europe, DATE 13, Grenoble, France, March 18-22, 2013* (E. Macii, ed.), pp. 1673–1678, EDA Consortium San Jose, CA, USA / ACM DL, 2013.

[8] T. Xiao and M. Marek-Sadowska, "Crosstalk reduction by transistor sizing," in *Proceedings of the 1999 Conference on Asia South Pacific Design Automation, Wanchai, Hong Kong, China, January 18-21, 1999*, pp. 137–140, IEEE Computer Society, 1999.

[9] M. Tsai, T. Wang, and T. T. Hwang, "Through-silicon via planning in 3-d floorplanning," *IEEE Trans. VLSI Syst.*, vol. 19, no. 8, pp. 1448–1457, 2011.

[10] T. Song, C. Liu, Y. Peng, and S. K. Lim, "Full-chip multiple tsv-to-tsv coupling extraction and optimization in 3d ics," in *The 50th Annual Design Automation Conference 2013, DAC '13, Austin, TX, USA, May 29 - June 07, 2013*, pp. 180:1–180:7, ACM, 2013.

[11] J. Cho, E. Song, K. Yoon, J. S. Pak, J. Kim, W. Lee, T. Song, K. Kim, J. Lee, H. Lee, K. Park, S. Yang, M. Suh, K. Byun, and J. Kim, "Modeling and analysis of through-silicon via (tsv) noise coupling and suppression using a guard ring," *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 1, pp. 220–233, Feb 2011.