A Study of Through-Silicon-Via Impact on the 3D Stacked IC Layout[*]

Dae Hyun Kim, Krit Athikulwongse, and Sung Kyu Lim
School of Electrical and Computer Engineering
Georgia Institute of Technology, Atlanta, Georgia
{daehyun, krit, limsk}@ece.gatech.edu

ABSTRACT

Through-Silicon-Via (TSV) is the enabling technology for the fine-grained 3D integration of multiple dies into a single stack. These TSVs occupy non-negligible silicon area because of their sheer size. This significant silicon area occupied by the TSVs and the interconnections made to the TSVs greatly affect area, power, performance, and reliability of 3D IC layouts. Well-managed TSVs alleviate congestion, reduce wirelength, and improve performance, whereas excessive TSVs not only increase the die area, but also have negative impact on many design objectives. In this paper, we study the impact of TSV on various aspects of 3D layouts. We use GDSII layouts of 2D and 3D designs, and thoroughly compare the pros and cons of TSV usage. We propose a new force-directed 3D gate-level placement that efficiently handles TSVs. In addition, we present an algorithm that assigns TSVs to nets to complete routing that involves TSVs. This algorithm, together with our 3D placer, is integrated into a commercial P&R tool to generate fully validated GDSII layouts. Our experiments based on synthesized benchmarks indicate that our algorithms help generate GDSII layouts of 3D designs that are optimized in terms of area, wirelength, and metal layer count.

1. INTRODUCTION

Three-dimensional integrated circuits (3D ICs) are emerging as a natural way to overcome interconnect scaling problems in 2D ICs. 3D ICs benefit from smaller footprint area than 2D ICs and from vertical (z-direction) interconnections between different dies [6, 7]. Small footprint area of 3D ICs allows gates to be placed closer, thereby leading to shorter wirelength than 2D ICs. Vertical interconnections by Through-Silicon-Vias (TSVs) also help shorten wirelength because gates can be placed on top of each other in different dies, eliminating the need of long cross-chip interconnects existing in 2D ICs. This shorter wirelength helps alleviate routing congestion as well as crosstalk and noise problems. Therefore, 3D ICs are expected to replace 2D ICs in the coming future.

[*]This material is based upon the work supported by the National Science Foundation under CAREER Grant No. CCF-0546382, the Center for Circuit and System Solutions (C2S2), and the Interconnect Focus Center (IFC).

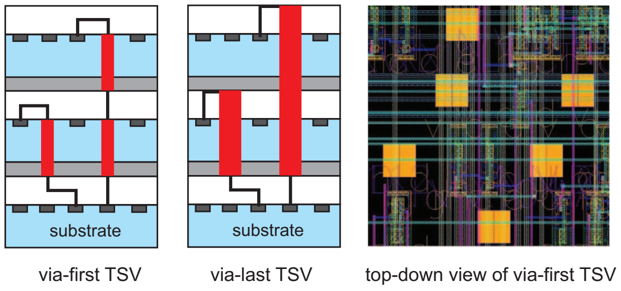


Figure 1: Via-first and via-last TSVs

Although TSVs can alleviate congestion, reduce wirelength, and improve performance, they occupy non-negligible silicon area. Excessive or ill-placed TSVs not only increase die area, but also have negative impact on these objectives in 3D ICs [7]. Therefore, CAD tools for 3D ICs should carefully consider the impact of TSVs during placement and routing. Depending on their type, via-first TSVs interfere with device layer, whereas via-last TSVs interfere with both device and metal layers (see Figure 1). A typical size of via-first TSVs ranges from $1\mu m$ to $5\mu m$, whereas that of via-last TSVs ranges from $5\mu m$ to $20\mu m$ [1]. These TSVs are much larger than wires, local vias, and gates. Thus, care must be taken to consider the impact of TSV usage on the layout of each die in a 3D stack. Most previous works on 3D IC CAD tools [3, 4], however, ignore either the sheer size of TSVs or the fact that TSVs interfere with gates and wires.

The contributions of this paper are as follows: (1) We propose a new force-directed 3D placement algorithm. We also introduce two different TSV handling schemes, namely "TSV-site" and "TSV co-placement". TSV-site scheme places TSVs at regular positions, and then places gates, whereas TSV co-placement scheme places TSVs and gates simultaneously. (2) Since there exist many excellent 2D routers, we show how to use existing 2D routers to complete routing in 3D ICs. While it is easy to use 2D routers in TSV co-placement scheme because TSVs are inserted into the netlist, we need one more step, which is "TSV assignment", to use 2D routers in TSV-site scheme. We solve this TSV assignment problem using minimum spanning tree (MST) or placement-based method. (3) Our placement and TSV assignment algorithms are integrated into a commercial tool. This new tool flow generates GDSII-level 3D layouts that are fully validated. We perform various studies based on these GDSII layouts, and demonstrate how TSVs affect 3D layouts. To the best of our knowledge, this is the first work that studies TSV-related issues based on detailed layout data.

2. PRELIMINARIES

2.1 Maximum Allowable TSV Count

TSVs occupy significant silicon area; however, previous researches on 3D placement and routing did not consider this fact. For example, the authors of [3] used $18,519$ TSVs for ibm01 circuit which has $12,282$ cells. If we assume that the average cell area is $2\mu m^2$ in $45nm$ technology, the total cell area becomes $24,564\mu m^2$. If a TSV occupies $10\mu m^2$, the total TSV area becomes $185,190\mu m^2$, which is $7\times$ bigger than the cell area. Similarly, the authors of [4] used about $15,000$ TSVs for ibm01, which is not a realistic number.

Since we know that the smallest 2D chip area is simply the total cell area, we can compute the maximum TSV count such that the chip area of a 3D IC is less than a pre-defined number. For instance, the maximum TSV count, $N_{\mathrm{TSV}_{\max}}$, based on 2D and 3D chip areas can be calculated by

$$N_{\mathrm{TSV}_{\max}} = (A_{\mathrm{3D}} - A_{\mathrm{2D}})/A_{\mathrm{TSV}}\ , \tag{1}$$

where A_{3D} is the sum of the area of all dies in a 3D IC, A_{2D} is the area when the circuit is designed in 2D, and A_{TSV} is the area required by a TSV. If we apply Equation (1) to ibm01 in $45nm$ technology with $10\mu m^2$ TSV area, and $A_{\mathrm{3D}} = 1.5 \times A_{\mathrm{2D}}$, we get approximately $1,200$ for the maximum TSV count. This result means that the total die area of the 3D IC will be greater than $1.5 \times A_{\mathrm{2D}}$ if we use more than $1,200$ TSVs.

2.2 Wirelength and TSV Count Trade-Off

TSVs help reduce wirelength because long wires in 2D ICs can be shortened by placing cells in a net on top of each other in different dies and connecting them with TSVs. However, TSVs have two negative impacts on the layout. First, they occupy silicon area, and interfere with cells, thereby spreading cells out so that the average distance between cells does not decrease as much as expected [7]. Second, TSVs contribute to routing congestion because they need to be connected to other cells. This impact becomes severe for via-last TSVs [9, 10] because these TSVs go through all metal layers (and device layers plus the bulk), and become routing obstacles. Therefore, designers have to wisely control the TSV usage [8]. In our work, we control the number of TSVs during partitioning as discussed in Section 3.

3. 3D IC DESIGN FLOW

We devised two 3D IC design flows for comparisons in this work, namely TSV co-placement and TSV-site, as shown in Fig. 2. These flows are developed in such a way that we can use existing 2D routing tools while handling TSVs efficiently. By utilizing existing 2D routing tools, we can easily generate GDSII layouts of 3D ICs for in-depth analysis.

Partitioning: In the first stage of both design schemes, cells in the 2D netlist are distributed into N_{die} dies by a modified FM partitioning. During the partitioning, we control the cutsize in order to obtain the desired number of TSVs. The output of this stage is the 3D netlist in which some of the 2D nets (nets having all their cells in a die) of the original design become 3D nets (nets having their cells in different dies). After partitioning is completed, we can compute the minimum number of TSVs to be inserted. Although we can use multiple TSVs for a 3D net to connect cells in two adjacent dies, we use only one TSV for a 3D net between two adjacent dies.

TSV insertion and placement in TSV co-placement scheme: In TSV co-placement scheme, TSVs are added into the 3D netlist

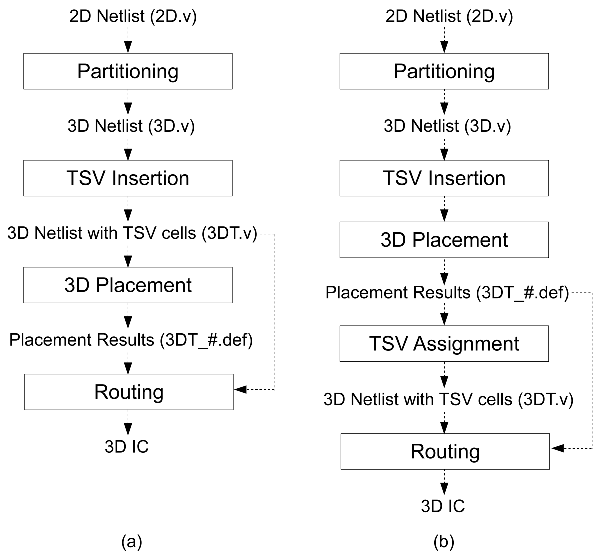


Figure 2: Two 3D IC design flows developed in this paper. (a) TSV co-placement, (b) TSV-site

during TSV insertion stage, and then cells and TSVs are placed simultaneously during 3D placement. Our 3D placer is explained in Section 4. The output of the 3D placer is a DEF file for each die.

TSV insertion and placement in TSV-site scheme: In TSV-site scheme, we pre-place TSVs uniformly on each die in TSV insertion stage, and then place cells in 3D placement stage. During 3D placement, pre-placed TSVs are treated as placement obstacles because there should not be any overlap between a TSV and a cell. An additional stage, TSV assignment, is needed after 3D placement for us to determine which pre-placed TSVs belong to which 3D nets. Then, we update the 3D netlist to reflect the assigned TSVs.

Routing: After we generate the DEF and the netlist files for each die, we use Cadence SoC Encounter [2] to route each die. Routing is done separately for each die because each die has its own netlist and cell positions. To facilitate TSV manipulation by Cadence SoC Encounter, we define a "TSV cell" as if it is a standard cell.

4. 3D PLACEMENT ALGORITHM

The 3D placement algorithm used in this work is based on a force-directed quadratic placement algorithm [11]. We modified the algorithm to place cells and TSVs in 3D.

4.1 Overview of Force-Directed Placement

In quadratic placement, a placement result is computed by minimizing the quadratic wirelength function $\mathit{\Gamma}$, which can be expressed as

$$\mathit{\Gamma} = \mathit{\Gamma}_{\mathrm{x}} + \mathit{\Gamma}_{\mathrm{y}}, \tag{2}$$

where $\mathit{\Gamma}_{\mathrm{x}}$ and $\mathit{\Gamma}_{\mathrm{y}}$ are wirelength along x- and y-axis. Because $\mathit{\Gamma}_{\mathrm{x}}$ and $\mathit{\Gamma}_{\mathrm{y}}$ are independent, they can be separately minimized to obtain the minimum of $\mathit{\Gamma}$. The following description for x-dimension applies similarly to y-dimension. Here, $\mathit{\Gamma}_{\mathrm{x}}$ can be written in a matrix form as

$$\mathit{\Gamma}_{\mathrm{x}} = \frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{C}_{\mathrm{x}}\mathbf{x} + \mathbf{x}^{\mathrm{T}}\mathbf{d}_{\mathrm{x}} + \mathrm{constant}, \tag{3}$$

where $\mathbf{x} = [x_1 \cdots x_N]^{\mathrm{T}}$ is a vector representing the x-position of N cells being placed, $\mathbf{C}_{\mathrm{x}}$ is an $N \times N$ matrix representing the connection among the cells along x-axis, and $\mathbf{d}_{\mathrm{x}} = [d_{\mathrm{x},1} \cdots d_{\mathrm{x},N}]^{\mathrm{T}}$ is a vector representing the connection to fixed pins along x-axis. Element $c_{\mathrm{x},ij}$ of $\mathbf{C}_\mathrm{x}$ matrix is the weight of connection between cell i and cell j, and element $d_{\mathrm{x},i}$ is the negative weighted position of fixed pins connected to cell i. The minimum of $\mathit{\Gamma}_\mathrm{x}$ can be obtained by setting its derivative to zero. Therefore, the cell placement along x-axis is computed by solving

$$\mathbf{C}_\mathrm{x}\mathbf{x} + \mathbf{d}_\mathrm{x} = \mathbf{0}. \tag{4}$$

Quadratic placement can be viewed as an elastic spring system when we treat $\mathit{\Gamma}$ as the total spring energy of the system. Because the derivative of a spring energy is a force, the derivative of $\mathit{\Gamma}_\mathrm{x}$ in Equation (3) can be view as a net force $\mathbf{f}_\mathrm{x}^\mathrm{net}$ as

$$\mathbf{f}_\mathrm{x}^\mathrm{net} = \boldsymbol{\nabla}_\mathrm{x}\mathit{\Gamma}_\mathrm{x} = \mathbf{C}_\mathrm{x}\mathbf{x} + \mathbf{d}_\mathrm{x}, \tag{5}$$

where $\boldsymbol{\nabla}_\mathrm{x} = [\partial/\partial_{x_1} \cdots \partial/\partial_{x_N}]^\mathrm{T}$ is the vector differential operator. At equilibrium, $\mathbf{f}_\mathrm{x}^\mathrm{net}$ is zero, resulting in minimum $\mathit{\Gamma}_\mathrm{x}$, but cells can be crowded in few area of the chip, resulting in high cell overlap.

Move force is density-based force that spreads cells away from high cell density area to low cell density area to reduce cell overlap. Move force in [11] is defined for 2D ICs. We modified it to support cell overlap removal in 3D ICs. Our modification is explained in Section 4.3. Hold force is used to decouple each placement iteration from the previous iteration. It cancels out net force that pulls cells back to the placement in initial iteration, and can be written as

$$\mathbf{f}_\mathrm{x}^\mathrm{hold} = -(\mathbf{C}_\mathrm{x}\mathbf{x}' + \mathbf{d}_\mathrm{x}), \tag{6}$$

where $\mathbf{x}' = [x_1' \cdots x_N']^\mathrm{T}$ is a vector representing the x-position of cells from the previous placement iteration. When no move force is applied, hold force holds cells being placed into their position.

Total force $\mathbf{f}_\mathrm{x}$ is the summation of net force, move force, and hold force. The total force is set to zero,

$$\mathbf{f}_\mathrm{x} = \mathbf{f}_\mathrm{x}^\mathrm{net} + \mathbf{f}_\mathrm{x}^\mathrm{move} + \mathbf{f}_\mathrm{x}^\mathrm{hold} = \mathbf{0}, \tag{7}$$

to get the placement result with minimal wirelength and some cell overlap reduction for each placement iteration.

4.2 Overview of Our 3D Placement Algorithm

Our 3D placement algorithm is divided into three phases: initial placement, global placement, and detail placement.

In the first phase, the initial placement is computed by solving Equation (4). The initial placement result contains high cell overlap, which will be reduced in each global placement iteration in the second phase by introducing move force and hold force in Equation (7), and solving the equation.

Global placement continues until the amount of remaining cell overlap is low. Then, detail placement starts in the third phase to legalize the result from global placement using a greedy algorithm.

4.3 Placing Cells in 3D ICs

It is not possible to extend the 2D force-directed quadratic placement algorithm to 3D placement algorithm simply by adding z-axis variable in Equation (2). The reason is that all the fixed pins in 3D ICs are on the C4-bump side, resulting in placing all the cells at the same z-position, $\mathbf{z} = \mathbf{0}$, in the initial placement [4]. In this work, we extended the force-directed quadratic placement algorithm in [11] by exploiting the fact that cells are already assigned into dies by the partitioner and not moving them across dies during placement. Therefore, we do not include $\mathit{\Gamma}_\mathrm{z}$ into Equation (2), but let the placer focus on wirelength minimization along x- and y-axis.

Move force in [11] is modified to support placing cells in 3D ICs. Because cell overlap on all dies are different, we compute move force for a cell based on the cell overlap of the die on which the cell is being placed.

The placement problem is formulated as a global electrostatic problem by treating cell area as positive charge and chip area as negative charge. The placement density D on die d can be computed by

$$D(x,y)\Big|_{z=d} = D^\mathrm{cell}(x,y)\Big|_{z=d} - D^\mathrm{chip}(x,y)\Big|_{z=d}, \tag{8}$$

where $D^\mathrm{cell}(x,y)\big|_{z=d}$ is the cell density at position (x,y) computed by using only cells being placed on die d, and $D^\mathrm{chip}(x,y)\big|_{z=d}$ is the chip capacity scaled to match total area of cells being placed on the die.

After D is computed, placement potential $\mathit{\Phi}$ can be obtained by solving Poisson's equation

$$\Delta\mathit{\Phi}(x,y)\Big|_{z=d} = -D(x,y)\Big|_{z=d}. \tag{9}$$

The negative gradient of $\mathit{\Phi}$ indicates in which direction and how fast the cell at that position should move. Move force is modeled by connecting cell i to its target point $\mathring{x}_i$ with a spring of spring constant $\mathring{w}_i$. The target point is computed by

$$\mathring{x}_i = x_i' - \frac{\partial}{\partial x}\mathit{\Phi}(x,y)\Big|_{(x_i',y_i'),z=d}, \tag{10}$$

where x_i' is the x-position of cell i being placed on die d from the previous placement iteration. Therefore, for cell i, move force $f_{\mathrm{x},i}^\mathrm{move} = \mathring{w}_i(x_i - \mathring{x}_i)$, where x_i is the x-position of cell i being placed. Move force $\mathbf{f}_\mathrm{x}^\mathrm{move}$ is finally defined for 3D ICs by

$$\mathbf{f}_\mathrm{x}^\mathrm{move} = \mathring{\mathbf{C}}_\mathrm{x}(\mathbf{x} - \mathring{\mathbf{x}}), \tag{11}$$

where $\mathring{\mathbf{C}}_\mathrm{x}$ is a diagonal matrix of $\mathring{w}_i$, $\mathbf{x} = [x_1 \cdots x_N]^\mathrm{T}$ is a vector representing the x-position of N cells being placed, and $\mathring{\mathbf{x}} = [\mathring{x}_1 \cdots \mathring{x}_N]^\mathrm{T}$ is a vector representing the target x-position of the cells.

4.4 Placing TSVs in TSV Co-placement Scheme

In TSV co-placement scheme, we treat a TSV as a cell being placed by our 3D placement algorithm. Therefore, we call it a TSV cell in this subsection, and explicitly call an original cell in the design a gate cell. We modified our 3D placement algorithm to place TSV cells in TSV co-placement scheme. After adding the minimum number of TSV cells into the netlist, the total number of cells being placed is updated. The area of TSV cells is also used to compute $D^\mathrm{cell}(x,y)\big|_{z=d}$ and $D^\mathrm{chip}(x,y)\big|_{z=d}$ in Equation (8). The resulting $\mathbf{x}$ vector obtained from solving Equation (4) and (7) also includes the x-position of TSV cells.

4.5 Net Splitting

During wirelength computation, we use net splitting to compute wirelength more accurately as shown in Fig. 3. Wirelength computation without net splitting is based on the projection of the cell locations in all dies onto a 2D plane. On the other hand, wirelength computation with net splitting is based on the projection of the cell locations in each die onto its own 2D plane. Therefore net splitting during wirelength computation gives us more accurate wirelength estimation. The comparison of these two approaches is presented in Section 6.1.

4.6 Pre-placing TSVs in TSV-site Scheme

In TSV-site scheme, TSVs are pre-placed into placement area before the original cells are placed. Therefore, we treat them as placement obstacles. Although the total number of cells being placed is

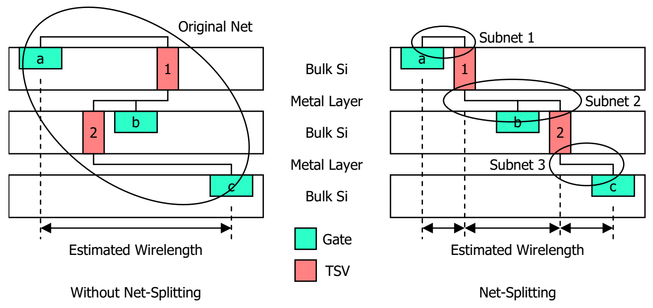


Figure 3: Splitting 3D net into subnets (side view)

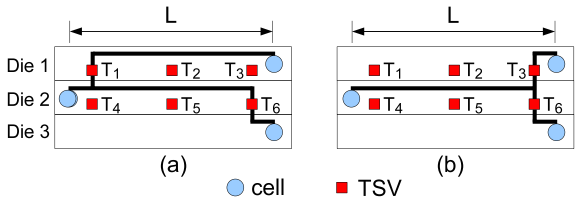


Figure 4: Cost computation for each combination of TSVs in three dies (side view). (a) wirelength = $2L$ for (T_1,T_6), (b) wirelength = L for (T_3,T_6)

not updated, and the resulting $\mathbf{x}$ vector obtained from solving Equation (4) and (7) still includes only the x-position of the original cells in the design, we include the area of pre-placed TSVs when computing $D^{\text{cell}}(x,y)\big|_{z=d}$ and $D^{\text{chip}}(x,y)\big|_{z=d}$ in Equation (8).

TSVs are evenly pre-placed as placement obstacles in rows and columns in this scheme. Placement obstacles can be handled naturally by the mean of placement density in [11]. By including the area of pre-placed TSVs when computing placement density, move force is altered in such a way that it drives cells being placed away from pre-placed TSVs.

5. TSV ASSIGNMENT

TSV assignment problem is to assign 3D nets to TSVs for given sets of dies, 3D nets, placed cells, and placed TSVs so that the total wirelength of 3D nets is minimized. The constraints are: (1) a TSV cannot be assigned to more than one 3D net, and (2) a 3D net should use one TSV between two adjacent dies.

5.1 Optimum Solution for TSV Assignment

The Binary Integer Linear Programming (BILP) formulation to find the optimum solution of TSV assignment for two dies was already shown in [12]. Since the number of binary integer variables in the formula was too big, the authors in [12] introduced heuristic algorithms based on neighborhood search.

If we have more than two dies, and a 3D net spans in more than two dies, combinations of TSVs in different dies should be considered for cost computation (see Figure 4). In Fig. 4 (a), T_1 and T_6 are assigned to the 3D net, and the cost (=wirelength) is approximately $2L$. On the other hand, T_3 and T_6 are assigned to the 3D net in Fig. 4 (b), and the cost is approximately L. Although T_6 is used in both cases, its contributions to the cost are different. Therefore, the cost should be computed for each combination of TSVs.

The optimum solution of TSV assignment for the case of more than two dies is found by the following formulation:

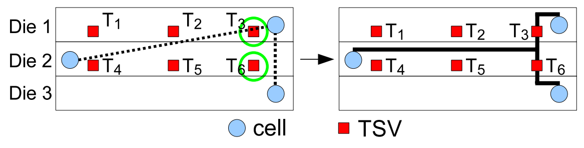


Figure 5: MST-based TSV assignment (side view)

Minimize

$$\sum_{i=1}^{N_{\text{3DNet}}} \sum_{k=1}^{CB_i} \sum_{p=1}^{N_{\text{TSV}}} d_{i,k,p} \cdot x_{i,k,p} \tag{12}$$

Subject to

$$\sum_{k=1}^{CB_i} \sum_{p=1}^{N_{\text{TSV}}} x_{i,k,p} = 1, \quad (i = 1, \cdots, N_{\text{3DNet}}) \tag{13}$$

$$\sum_{i=1}^{N_{\text{3DNet}}} \sum_{k=1}^{CB_i} x_{i,k,p} \le 1 \quad (p = 1, \cdots, N_{\text{TSV}}) \tag{14}$$

where N_{3DNet} is the total number of 3D nets, N_{TSV} is the total number of TSVs, CB_i is the total number of combinations of TSVs for the 3D net H_i, and $d_{i,k,p}$ is the cost when the k-th combination is used for the 3D net H_i. Here, $x_{i,k,p}$ is 1 if (1) the 3D net H_i uses the combination CB_k, and (2) the combination CB_k uses the TSV T_p, and otherwise $x_{i,k,p}$ is 0. Equation (13) denotes that a 3D net uses only one combination, and Equation (14) denotes that a TSV is assigned to at most one 3D net.

The number of variables in this problem is also very big because we have to consider all the possible combinations for all 3D nets. Even if we restrict available TSVs for a 3D net to TSVs inside a small window, the number of combinations is still big. For example, if a 3D net spans in four dies, and the window contains 20 TSVs in each die, $8,000$ combinations are available for the net. Moreover, restriction of window size may result in the infeasibility of BILP. Therefore, we introduce two heuristic algorithms in the next two subsections.

5.2 MST-based TSV Assignment

In this method, we use MST for TSV assignment as shown in Fig. 5. After constructing MST for a 3D net, we choose the nearest TSV to the shortest edge. If we still need TSVs, we search for the next shortest edge, and choose TSVs. In Fig. 5, the shortest edge spans in all the three dies so that the nearest TSV in each die is assigned to the edge.

MST-based TSV assignment is a sequential (net by net) method. Therefore, the order of nets for assignment becomes important because 3D nets assigned at the beginning have more avaiable TSVs. In our method, we sort the 3D nets in the ascending order of bounding-box size because a net which has a large bounding box containing many TSVs inside has more choices for its TSVs.

5.3 Placement-based TSV Assignment

In this method, we solve the assignment problem by 3D placement algorithm. The placed cells, however, become fixed cells at this time, and TSVs become movable cells. The assignment is done in two steps - global and detailed. Fig. 6 shows how TSVs are assigned by 3D placement. The global assignment is done by 3D global placement. During this step, TSVs are placed by force-directed quadratic method regardless of TSV-site locations. After global placement is done, we solve the detailed assignment by cell snapping. In this step, we move each TSV to each TSV-site.

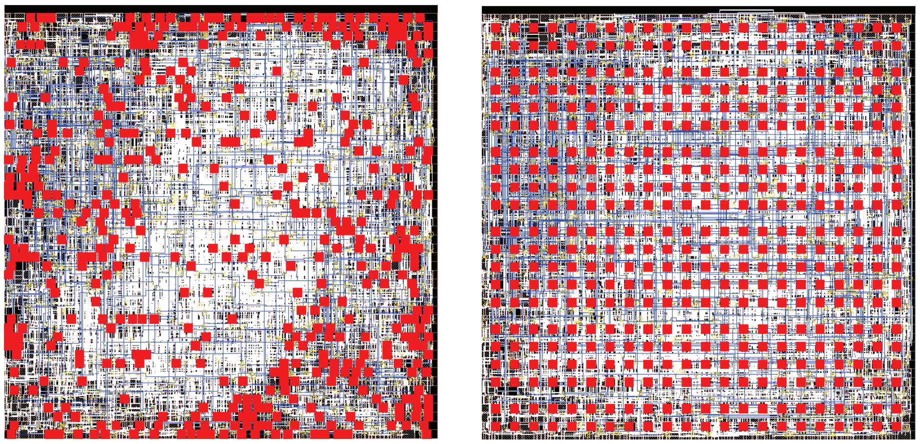


Figure 7: Cadence SoC Encounter snapshot of the bottommost die of Ind2 designed by TSV co-placement and TSV-site methods. Routing for 3D nets are shown in blue.

Table 3: Comparison of wirelength (WL), the minimum number of metal layers (ML), runtime for placement, and total silicon area for 2D and 3D (4 dies) design for IWLS 2005 benchmarks and industrial circuits. Cell occupancy is 80%, and the number of 3D nets was set to be 3% to 5% of the number of total nets during partitioning. The numbers in parentheses are ratios to 2D.

	our 2D				our 3D				
Circuit	WL (μm)	# ML	runtime (s)	Area (μm^2)	WL (μm)	# ML	runtime (s)	Area (μm^2)	# TSVs
Ind 1	397,015 (1.0)	5	85 (1.0)	44,944 (1.0)	399,924 (1.01)	4	93 (1.10)	69,696 (1.55)	1,700
Ind 2	334,648 (1.0)	4	72 (1.0)	44,944 (1.0)	284,340 (0.85)	4	53 (0.73)	58,564 (1.30)	1,302
Ind 3	287,587 (1.0)	4	71 (1.0)	48,841 (1.0)	300,781 (1.05)	4	81 (1.14)	69,696 (1.43)	798
Ind 4	411,993 (1.0)	4	157 (1.0)	63,001 (1.0)	388,315 (0.94)	4	101 (0.64)	80,656 (1.28)	1,016
Ind 5	703,461 (1.0)	5	189 (1.0)	103,684 (1.0)	582,603 (0.83)	4	188 (1.00)	147,456 (1.42)	2,789
ethernet	1,534,386 (1.0)	4	1,289 (1.0)	293,764 (1.0)	1,401,059 (0.91)	4	1,287 (1.00)	341,056 (1.16)	3,866
RISC	1,976,549 (1.0)	4	880 (1.0)	314,721 (1.0)	2,001,986 (1.01)	4	727 (0.83)	386,884 (1.23)	4,438
b18	2,415,867 (1.0)	5	1,459 (1.0)	338,724 (1.0)	2,683,424 (1.11)	4	1,134 (0.78)	495,616 (1.46)	10,404
des_perf	2,445,398 (1.0)	5	1,367 (1.0)	327,184 (1.0)	1,911,731 (0.78)	4	950 (0.69)	386,884 (1.18)	3,856
b19	3,986,586 (1.0)	5	2,642 (1.0)	580,644 (1.0)	3,945,515 (0.99)	4	2,173 (0.82)	712,336 (1.23)	8,497

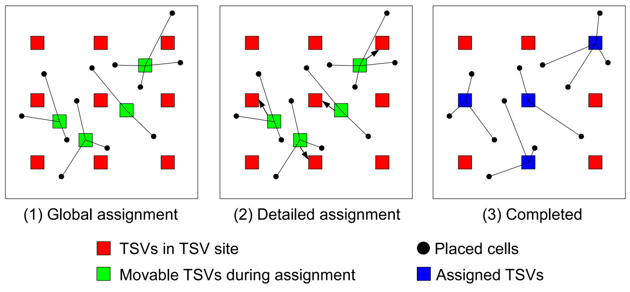


Figure 6: TSV assignment based on 3D Placement (top view)

Table 1: Benchmark Circuits

Circuit	# gates	# TRs	# nets	Profile
Ind 1	16K	137K	12K	Microprocessor
Ind 2	15K	106K	15K	Inverse DCT
Ind 3	16K	134K	16K	Microprocessor
Ind 4	20K	146K	20K	Microprocessor
Ind 5	30K	317K	30K	Arithmetic Unit
ethernet	77K	729K	77K	Ethernet IP Core
RISC	88K	775K	89K	Microprocessor
b18	104K	728K	104K	Microprocessor Cores
des_perf	109K	823K	109K	DES (Data Encryption Standard)
b19	169K	1.29M	169K	Microprocessor Cores

6. EXPERIMENTAL RESULTS

We use IWLS 2005 benchmarks [5] and several industrial circuits which are listed in Table 1. We also used $45nm$ technology for our experiments. TSV cell size is $2.47\mu m \times 2.47\mu m$.

6.1 Net-splitting Results

We first compare wirelengths of our 3D placement using TSV co-placement scheme without net-splitting and with net-splitting. Table 2 shows the wirelength comparison. Although 'without net-splitting' is better for two circuits, 'with net-splitting' is generally better, and the average improvement is 5.59%. The reason that 'with net-splitting' generates shorter wirelength is that it estimates wirelength more accurately in a 3D view so that it makes our placer reduce the total wirelength more efficiently. For the rest of this paper, we use net-splitting for wirelength estimation.

6.2 Wirelength and Runtime Comparison

Table 3 shows wirelength and runtime of our 2D placement and 3D placement results. The wirelength reduction in non-microprocessor

Table 2: Wirelength of our 3D placement with and without net-splitting

Circuit	without net-splitting (μm)	with net-splitting (μm)	Dif.(%)
Ind 1	$444,867$	$408,713$	-8.13%
Ind 2	$309,936$	$288,143$	-7.03%
Ind 3	$305,961$	$308,006$	$+0.67\%$
Ind 4	$405,010$	$393,215$	-2.91%
Ind 5	$658,886$	$584,024$	-11.36%
ethernet	$1,538,792$	$1,406,073$	-8.62%
RISC	$2,225,730$	$2,025,187$	-9.01%
b18	$2,610,358$	$2,683,424$	$+2.80\%$
des_perf	$2,362,977$	$2,199,149$	-6.93%
b19	$4,612,405$	$4,364,694$	-5.37%
		Average	-5.59%

Table 4: Comparison of wirelength of TSV co-placement, TSV-site placement with MST-based TSV assignment, and TSV-site placement with placement-based TSV assignment. The numbers in the parentheses are ratios to TSV co-placement.

Circuit	Wirelength (μm)		
		TSV-site	
	TSV co-placement	MST-based	Placement-based
Ind 2	$284,340$ (1.0)	$310,677$ (1.09)	$312,423$ (1.10)
ethernet	$1,401,059$ (1.0)	$1,513,381$ (1.08)	$1,554,960$ (1.11)
des_perf	$1,911,731$ (1.0)	$2,197,209$ (1.15)	$2,228,375$ (1.17)
	Runtime for assignment (s)		
Ind 2	-	0.08	34
ethernet	-	2.86	188
des_perf	-	1.13	290

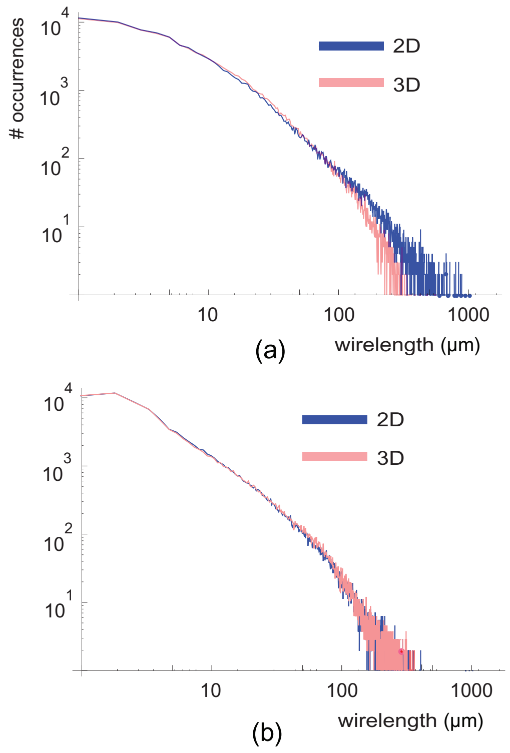


Figure 8: Wirelength distribution of (a) des_perf, where the die width is $572\mu m$ in 2D design and $311\mu m$ in 3D design (4 dies), (b) b19, where the die width is $762\mu m$ in 2D design and $411\mu m$ in 3D design (4 dies).

circuits is 10% to 20% in 3D. However, we could not benefit from 3D design in terms of wirelength for microprocessors.

To figure out the reasons, we plotted the wirelength distributions in Fig. 8 for des_perf which is a non-microprocessor circuit, and for b19 which is a set of microprocessors. As shown in Fig. 8, long interconnections of des_perf in 2D become shorter in 3D. The longest wire of des_perf in 2D design is about $1000\mu m$-long, whereas the longest wire in 3D design is about $320\mu m$-long. This effect obviously comes from smaller footprint area than 2D design and connections in z-direction.

On the other hand, long interconnections of b19 in 2D do not become shorter in 3D. Since we use partitioning as a pre-process for 3D placement, we counted the cut size of the min-cut 4-way partitioning, and the result was that the cut size of des_perf was $1,613(1.47\%)$ out of $109,415$ nets, whereas the cut size of b19 was $253(0.15\%)$ out of $169,470$ nets. This cut size means that b19 is highly modulized so that the total wirelength cannot be reduced much if we use min-cut partitioning.

Runtime of 3D placement is smaller than 2D placement. The reason is that 3D placement results have smaller number of overlaps than 2D placement results because each die in 3D ICs has less number of cells to be placed. Since force-directed quadratic placement algorithm spends a significant portion of its runtime in removing overlaps, having less number of cells in a die improves runtime.

6.3 Metal Layers and Silicon Area Results

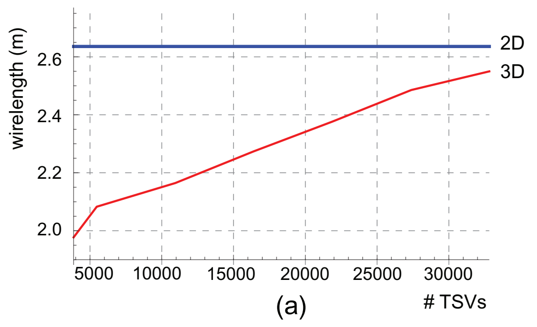
Since 3D design has smaller footprint area than 2D design, and each die has less number of cells, the number of metal layers required for 3D design could be smaller than that for 2D design. Therefore, we tried to find the minimum number of metal layers that led to a successful routing. For fair comparisons, we fixed the cell occupancy (80%), increased the cut size from the minimum in 3D design, and tried routing until we found a successful routing result. Table 3 shows the comparisons of the minimum number of metal layers in 2D and 3D designs. While we could route all the circuits with 4 metal layers in 3D designs, some of the 2D designs could not be routed with 4 metal layers because of congestion (DRC errors). The benefit of the decreased number of metal layers in 3D design comes from TSV insertion which results in the increase of the silicon area. Table 3 also shows how much area in 3D design increased.

6.4 On Wirelength vs # TSVs

Since we use partitioning as a pre-process for 3D placement, we experimented on how the TSV count affects the wirelength reduction in 3D design. Fig. 9 shows the results for des_perf and b19. The wirelength of des_perf in 3D design monotonically increases as the TSV count increases. This result indicates that the additional TSVs do not help wirelength reduction much. They rather increase die area thereby increasing the wirelength. On the other hand, the wirelength of b19 in 3D design generally increases at first as the TSV count increases, but it saturates after all. Although we cannot draw a clear and obvious conclusion on the relationship between wirelength and the number of TSVs from these observations, using too many TSVs will eventually increase the die area, which will result in wirelength increase.

6.5 On Wirelength and Die Area vs # Dies

In this experiment, we vary the number of dies (N_{die}) from 2 to 16, and observe wirelength, die area, and the number of TSVs.



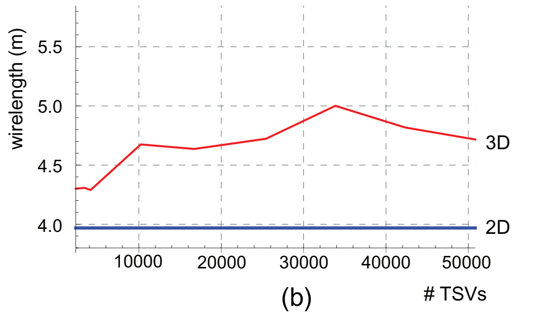


Figure 9: Wirelength vs # TSVs of (a) des_perf, and (b) b19 for 2D and 3D (4 dies) designs

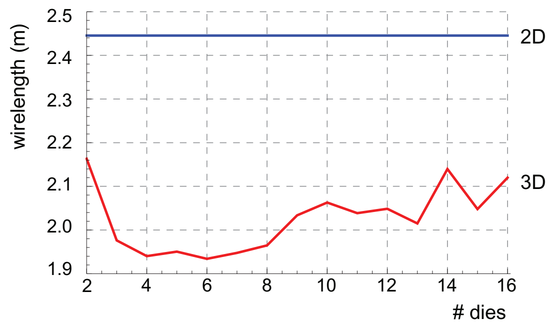


Figure 10: Wirelength vs # dies of des_perf in 3D design

The wirelength of des_perf in 3D design dramatically decreases as $N_{\rm die}$ increases up to 4, then it saturates or slightly goes up as shown in Fig. 10. If we increase $N_{\rm die}$ more, the TSV count and die area will go up as shown in Fig. 11. In other words, increasing $N_{\rm die}$ is helpful at first, but becomes not helpful as $N_{\rm die}$ goes up because 1) the TSV count increases, 2) the increased TSV count leads to the increase of die area, and 3) some of the 2D nets do not need to be 3D nets. This trend may not be applicable to all the 3D designs. However, we observe that using a small number of TSVs is helpful if partitioning is used as a pre-process for 3D placement.

6.6 TSV Co-placement vs TSV-site

Table 4 shows the comparison between TSV co-placement and TSV-site. The wirelength increase of TSV-site placement with MST-based TSV assignment compared to TSV co-placement is 8% to 15%, whereas the wirelength increase of TSV-site placement with placement-based TSV assignment is 10% to 17%. Runtime overhead is a few seconds for MST-based TSV assignment and a few minutes for placement-based TSV assignment. Although TSV co-placement was better than TSV-site with respect to wirelength, TSV-

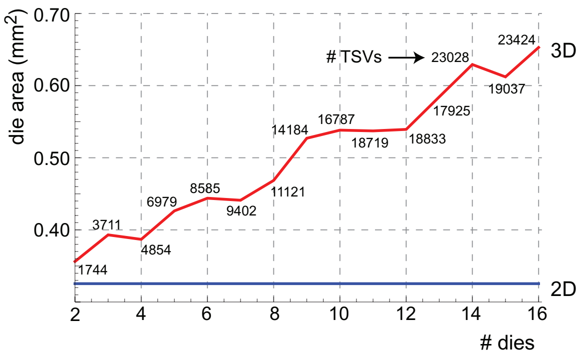


Figure 11: Die area and # TSVs of des_perf in 3D design

site has its own advantages which are "better heat dissipation and stronger package bonding" according to [12].

7. CONCLUSION

In this paper, we studied, and presented how TSVs affect the 3D stacked IC layout. According to our 3D placement and routing results, wirelength reduction in 3D design for non-microprocessor circuits was 10% to 20%, whereas the area overhead caused by TSV insertion was 15% to 30% depending on the number of TSVs. We could also use less number of metal layers for some of the 3D designs. The number of dies and the TSV count also had non-negligible impact on the layout of 3D ICs.

8. REFERENCES

[1] E. Beyne and et al. Through-Silicon Via and Die Stacking Technologies for Microsystems-integration. In *Proc. IEEE Int. Electron Devices Meeting*, 2008.

[2] Cadence. Soc Encounter. http://www.cadence.com.

[3] J. Cong, G. Luo, J. Wei, and Y. Zhang. Thermal-Aware 3D IC Placement Via Transformation. In *Proc. Asia and South Pacific Design Automation Conf.*, 2007.

[4] B. Goplen and S. Sapatnekar. Placement of 3D ICs with Thermal and Interlayer Via Considerations. In *Proc. ACM Design Automation Conf.*, 2007.

[5] IWLS. IWLS 2005 Benchmarks. http://www.iwls.org/iwls2005.

[6] J. W. Joyner, P. Zarkesh-Ha, J. A. Davis, and J. D. Meindl. A Three-Dimensional Stochastic Wire-Length Distribution for Variable Separation of Strata. In *Proc. IEEE Int. Interconnect Technology Conference*, 2000.

[7] D. H. Kim, S. Mukhopadhyay, and S. K. Lim. Through-Silicon-Via Aware Interconnect Prediction and Optimization for 3D Stacked ICs. In *Proc. ACM/IEEE Int. Workshop on System Level Interconnect Prediction*, 2009.

[8] D. H. Kim, S. Mukhopadhyay, and S. K. Lim. TSV-aware Interconnect Length and Power Prediction for 3D Stacked ICs. In *Proc. IEEE Int. Interconnect Technology Conference*, 2009.

[9] J. U. Knickerbocker and et al. Development of next-generation system-on-package (SOP) technology based on silicon carriers with fine-pitch chip interconnection. In *IBM J. Res. Dev. 49(4/5)*, 2005.

[10] J. U. Knickerbocker and et al. Three-dimensional silicon integration. In *IBM J. Res. Dev. 52(6)*, 2008.

[11] P. Spindler, U. Schlichtmann, and F. M. Johannes. Kraftwerk2 - A Fast Force-Directed Quadratic Placement Approach Using an Accurate Net Model. In *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 2008.

[12] H. Yan, Z. Li, Q. Zhou, and X. Hong. Via Assignment Algorithm for Hierarchical 3-D Placement. In *Proc. IEEE Int. Conf. on Communications, Circuits and Systems*, 2005.