Floorplanning and Signal Assignment for Silicon Interposer-based 3D ICs*

Wen-Hao Liu[1,2], Min-Sheng Chang[1], and Ting-Chi Wang[1]
[1]Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan
[2]Block Implementation, ICD, Cadence Design Systems, Taiwan
whliu@cadence.com, noon78621@hotmail.com, tcwang@cs.nthu.edu.tw

***Abstract*– Interposer-based 3D ICs (or known as 2.5D ICs) have been seen as an alternative approach to true 3D stacked ICs, which mount multiple dies on a silicon interposer and route signals between dies by the interconnects in the interposer. However, the floorplan of dies on the interposer and the signal assignment for macro-bumps and TSVs will largely impact the wirelength of the interconnects in a 2.5D IC. Because long interconnects would degrade the performance of 2.5D ICs, the multi-die floorplanning problem and signal assignment problem for 2.5D ICs are critical. This paper presents an enumeration-based algorithm and a network-flow-based algorithm to solve the multi-die floorplanning and signal assignment problems in a 2.5D IC, respectively. Also, to speed up the floorplanning and signal assignment algorithms, several acceleration techniques are proposed. The experimental results reveal that this work can effectively reduce the total wirelength in a 2.5D IC and the acceleration techniques can significantly speed up the proposed algorithms.**

1 Introduction

Two classes of three-dimensional integrated circuits (3D ICs) are under development today. The first one consists of true 3D stacked ICs, each of which is implemented as a vertical stack of active dies using through silicon vias (TSVs) to connect dies down to a package substrate. True 3D ICs have several benefits such as smaller footprint area and higher system performance, but have the issues of thermal dissipation, TSV stress effect, high design complexity, and high manufacturing cost, etc. [1, 2]. As a result, the implementation of true 3D ICs is still difficult.

Another class, which has been seen as an alternative approach to true 3D ICs, comes from interposer-based 3D ICs (or known as 2.5D ICs) [3, 4]. The structure of a 2.5D IC is illustrated in Fig 1(a) which also includes a package substrate and a PCB. Different from true 3D ICs, 2.5D ICs use a silicon interposer as an interface between a package and dies, and mount each die on the silicon interposer. The signals are communicated between dies by the routing wires in the redistribution layers (RDLs) of the silicon interposer. If some signals have to be delivered to other components on the PCB, the signals travel from the micro-bumps of dies to the escaping points at the boundaries of the package. Because the active dies in 2.5D ICs usually have no TSV inside, 2.5D ICs are easier to fabricate than true 3D ICs, making 2.5D ICs become more popular.

Minimizing the wirelength in 2.5D ICs is a critical issue, because long wires will degrade the performance of a 2.5D IC and cause timing violations. Recently, the work of [5] deals with a chip-interposer co-design problem to minimize the wirelength in a 2.5D IC, which plans the locations of dies on an interposer as well as the locations of I/O buffers and macro blocks in each die by a simulated annealing (SA) method. Also, a bipartite-matching-based algorithm is adopted in [5] to assign signals to micro-bumps. The method of [5] can effectively reduce the wirelength among micro-blocks, I/O buffers, and micro-

*This work was supported in part by the Ministry of Science and Technology of Taiwan under Grant No. 103-2220-E-007-001.

DAC '14, June 01 - 05 2014, San Francisco, CA, USA Copyright is held by the owner/author(s). Publication rights licensed to ACM. ACM 978-1-4503-2730-5/14/06…$15.00. http://dx.doi.org/10.1145/2593069.2593142

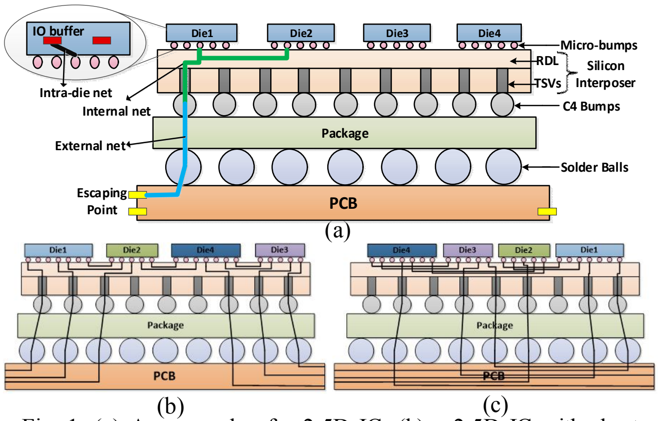


Fig. 1. (a) An example of a 2.5D IC; (b) a 2.5D IC with short interconnects; (c) a 2.5D IC with long interconnects.

bumps. However, it does not consider multi-terminal signals and ignores the signals that have to be delivered to the escaping points. As a result, it may find a result with long wirelength in the PCB level. Figs. 1(b) and 1(c) respectively show the routing results of a 2.5D IC with and without the consideration of wirelength minimization in the PCB level. We can see that the total wirelength in Fig. 1(c) is much longer than that in Fig. 1(b).

This paper assumes that the placement and routing in each die are already finished, so the locations of the I/O buffers in each die are determined. Accordingly, this work focuses on optimizing the multi-die floorplanning and signal assignment in a 2.5D IC to minimize the total wirelength. However, simultaneously planning die locations on an interposer and assigning signals to micro-bumps and TSVs make the problem too complicated, so we break this 2.5D-IC wirelength minimizing problem into a multi-die floorplanning problem and a signal assignment problem (SAP) to reduce the problem complexity. At first, an enumeration-based floorplanning algorithm (EFA) is presented to decide the position of each die on the interposer such that the terminals of each signal are placed close. According to our empirical observation, although the proposed EFA can get a better result than an SA-based floorplanning algorithm, its runtime increases exponentially as the number of dies increases. To overcome the runtime issue, three acceleration techniques for EFA are presented in this paper. After the floorplan of dies is determined, a network-flow-based signal assignment algorithm is proposed to assign signals to macro-bumps and TSVs and minimize the total wirelength in a 2.5D IC. Different from the traditional SAPs [6, 7] with only a single die, the proposed algorithm assigns signals to the micro-bumps of multiple dies and TSVs in the interposer. Also, a window matching method is presented to speed up the proposed signal assignment algorithm.

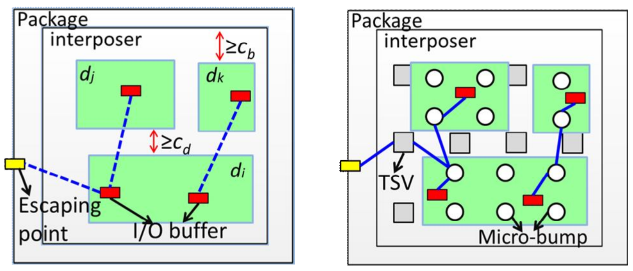
The rest of this paper is organized as follows. Section 2 describes the problem formulations of this paper. Section 3 presents EFA with three acceleration techniques. Section 4 proposes a network-flow-based algorithm to solve the SAP. Finally, section 5 shows the experimental results and Section 6 draws conclusion.

2 Preliminaries

This section first introduces how this work evaluates the wirelength of a 2.5D IC, and then details the problem formulations.

2.1 Wirelength Evaluation

There are two types of signals traveling in a 2.5D IC. The first type of signals communicates only between dies, and the second type of signals is further delivered to the escaping points. In this paper, the



(a) (b)

Fig. 2. (a) A multi-die floorplanning result and (b) a signal assignment result for a 2.5D IC with three dies and two signals.

interconnects to deliver signals in 2.5D ICs are classified into three types of nets: (1) intra-die nets, (2) internal nets and (3) external nets. Fig. 1(a) illustrates these three types of nets. An intra-die net is a two-terminal connection in a die; its two terminals are an I/O buffer and a micro-bump. An internal net is a two-terminal or multi-terminal connection in the interposer; at most one of its terminals is a TSV and the rest of them are micro-bumps. An external net is a two-terminal connection between a TSV and an escaping point at a boundary of the package on the PCB. Note that, similar to the 2.5D-IC shown in [3], we assume each TSV directly attaches a C4 bump and each C4 bump is one-to-one mapped to a solder ball, so an external net has to pass through a C4 bump and a solder ball. For more details, we use an example to illustrate how a signal is delivered by these three types of nets from a die to another die and an escaping point. In Fig. 1(a), a signal in Die1 is delivered from its corresponding I/O buffer to a micro-bump by an intra-die net. Because Die2 requires this signal, an internal net delivers the signal from the micro-bump of Die1 to a micro-bump of Die2. Moreover, in order to deliver the signal to its corresponding escaping point, the internal net also connects a TSV and finally an external net routes the signal from the TSV to the escaping point.

The total wirelength (TWL) of a 2.5D IC is measured by the following function:

$$TWL = \alpha \times WL_D + \beta \times WL_I + \gamma \times WL_E \qquad (1)$$

where WL_D, WL_I, and WL_E denote the total wirelengths of intra-die, internal and external nets, respectively; α, β and γ are user-defined weights to trade off the relative importance among WL_D, WL_I, and WL_E. Note that, in this paper α, β and γ are set to 1 by default, but the proposed algorithms can work well with different settings of α, β and γ. For simplification, this work does not perform routing to get the real routed wirelength for these nets. Instead, the wirelength of each net is measured by the length of its minimum spanning tree (MST), because the MST length of a net has high correlation to its routed wirelength as indicated in [8].

2.2 Problem Formulations

In this work, the 2.5D-IC wirelength minimization problem is divided into a multi-die floorplanning problem and a signal assignment problem. The notations to be used in these two problems are defined below:

- $D = \{d_1, d_2, \ldots\}$ is the set of dies in a 2.5D IC.
- $B_i = \{b_{i,1}, b_{i,2}, \ldots\}$ is the set of I/O buffers in die d_i.
- $M_i = \{m_{i,1}, m_{i,2}, \ldots\}$ is the set of micro-bumps in die d_i.
- M and B respectively denote the union of all M_i's and the union of all B_i's.
- $T = \{t_1, t_2, \ldots\}$ is the set of TSVs in the interposer.
- $E = \{e_1, e_2, \ldots\}$ is the set of escaping points at the boundaries of the package on the PCB.
- $S = \{s_1, s_2, \ldots\}$ is the set of signals.
- $P(s)$ is the set of terminals of signal s. $P(s)$ contains at most an escaping point, and the other terminals in $P(s)$ are I/O buffers, each of which is in a different die.

```
Algorithm Enumeration-based Floorplanning (EFA)
Input: die set $D$, signal set $S$, I/O buffer set $B$, escaping point set
$E$, the dimensions of the interposer;
Output: best floorplan $F_{best}$
1.  $WL_{best} = \infty$
2.  foreach sequence-pair $SP$
3.    foreach combination $R$ of orientations
4.      Floorplan $F \leftarrow$ transform($SP$, $R$, $D$)
5.      Align the center of $F$ to the center of the interposer
6.      If $F$ is illegal
7.        continue;
8.      If $estWL(F, S, B, E) < WL_{best}$
9.        $WL_{best} = estWL(F, S, B, E)$;
10.       $F_{best} \leftarrow F$
11.     end if
12.   end foreach
13. end foreach
14. return $F_{best}$
```

Fig. 3. Pseudo code of EFA.

In this work, the locations of I/O buffers and micro-bumps in each die, the locations of TSVs in the interposer, and the signals of every I/O buffer and escaping point are given. However, the locations of dies on the interposer and the signals of micro-bumps and TSVs are not decided yet. Therefore, this work sequentially solves the following two problems to minimize TWL in Eq. (1).

Multi-die floorplanning Problem: Given D, S, B, E and a fixed-outline silicon interposer, find a floorplan F of all dies in D on the silicon interposer such that no overlap between dies exists. Moreover, the die-to-die and die-to-boundary spacing constraints are imposed and require that the distance between the boundaries of any pair of dies and the distance between the boundaries of a die and the interposer have to be not smaller than technology-dependent parameters c_d and c_b, respectively. These two constraints come from the stress issue during the manufacturing.

Signal Assignment Problem (SAP): Given F, S, B, M, E, and T, assign the signal of each I/O buffer in B_i to a micro-bump in M_i and assign the signal of each escaping point in E to a TSV in T such that no micro-bump or TSV is assigned more than one signal.

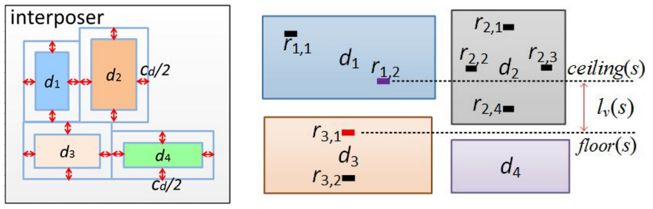
Note that, the micro-bumps and TSVs given in the SAP are the candidate sites for placing micro-bumps and TSVs. If a micro-bump or a TSV is assigned a signal after solving the SAP, it will be fabricated during manufacturing; otherwise, it has no need to be fabricated.

Figs. 2(a) and 2(b) respectively show the results of the multi-die floorplanning and signal assignment problems for an example with three dies and two signals. As shown in Fig. 2(b), after the signals are assigned to TSVs and micro-bumps, the topologies of intra-die nets, internal nets, and external nets are formed, so Eq. (1) can be computed accordingly.

3 Enumeration-based Floorplanning Algorithm

The basic idea of our enumeration-based floorplnning algorithm (EFA) is to enumerate a set of floorplans, and then pick the best floorplan to be the solution for the multi-die floorplanning problem. This work enumerates floorplans by considering every possible sequence pair for all dies in D. Sequence pair [9-11] is a widely-used floorplan representation, and is used in this work to express the up-down or left-right relation between any pair of dies in D. Although an enumeration-based method can explore larger solution space to yield better solution quality, using it to solve the module floorplanning problem for 2D ICs will make runtime unacceptable because the amount of modules in a 2D IC can be from a few hundred to a few thousand. However, a typical 2.5D IC has only a few dies, so the runtime of EFA is relatively affordable to solve the multi-die floorplanning problem.

Fig. 3 shows the pseudo code of EFA. The loop in lines 2-13 enumerates every possible sequence pair, and the inner loop in lines 3-12 enumerates every possible combination of die orientations. Because



(a) (b)

Fig. 4. (a) a floorplan transformed by sequence pair ($d_1d_2d_3d_4$, $d_3d_4d_1d_2$); (b) an example to illustrate how to compute LY_{min}.

die flipping is not allowed in 2.5D ICs and each die can be rotated by 0°, 90°, 180°, and 270°, each die has four orientations. Line 4 transforms the sequence pair and the orientation of each die to a floorplan F. Note that, during the transformation, we temporarily extend the boundaries of each die by $c_d/2$ to guarantee that the floorplan solution can obey the die-to-die spacing constraint. For example, Fig. 4(a) illustrates a floorplan transformed by the sequence pair ($d_1d_2d_3d_4$, $d_3d_4d_1d_2$). For more details about how to transform a sequence pair to a floorplan, please refer to [9-11]. After that, we align the center of F to the center of the interposer at line 5, and then check whether F is **legal** at line 6.

This work regards a floorplan to be legal if the floorplan can fit into the interposer's outline and obey the die-to-die and die-to-boundary spacing constraints; otherwise, it is illegal. When floorplan F is legal, lines 8-11 estimate the total wirelength of F. If the estimated wirelength of F is shorter than WL_{best}, we treat F to be the best solution F_{best} and update WL_{best} at lines 9-10. Finally, the best floorplanning solution is returned at line 14.

The accuracy of the total wirelength estimated by procedure *estWL* in Fig. 3 will influence the result quality of EFA. In order to get more accurate estimation, we have implemented a greedy algorithm (whose details are introduced in Section 5.2) to quickly assign signals to micro-bumps and TSVs and then compute *TWL* by Eq. (1). Although *TWL* can be exactly calculated when the signals of micro-bumps and TSVs are determined, the runtime of EFA is unacceptable because *estWL* is frequently called by EFA and the greedy algorithm is not fast enough. Therefore, we estimate *TWL* by adding up the half-parameter wirelength (HPWL) of $P(s_i)$ for every signal s_i in S. Our experiment reveals that using HPWL to estimate *TWL* can efficiently guide EFA only with a slight quality loss.

Let n denote the number of dies in D. There are $n!^2$ possible sequence pairs and each sequence pair has 4^n different die orientations, so totally $n!^2 4^n$ floorplans can be enumerated by EFA. For each floorplan, estimating its HPWL takes $O(|B|+|E|)$ time where $|B|$ and $|E|$ respectively denote the number of I/O buffers and the number of escaping points. Accordingly, the time complexity of EFA is $O(n!^2 4^n(|B|+|E|))$. Usually, when n is a small constant (e.g., 4), EFA can finish in short time. However, when n increases, the runtime of EFA increases largely. In order to speed up EFA, we develop three acceleration techniques: illegal branch cutting, inferior branch cutting, and die orientation pre-determination.

3.1 Illegal Branch Cutting

Each sequence pair with different die orientations provides at most 4^n different floorplans, in which some floorplans may be illegal. If a sequence pair does not provide any legal floorplans, we can avoid exploring the floorplans of this sequence pair for time saving. Based on this idea, before enumerating the die orientations for a sequence pair *SP*, the illegal branch cutting method identifies from *SP* a floorplan F_{low} by rotating each die to make its height not greater than its width, and a floorplan F_{thin} by rotating each die to make its width not greater than its height. F_{low} and F_{thin} have the least height and the least width among all floorplans of *SP*, respectively. Accordingly, if F_{low} is taller than the interposer or F_{thin} is wider than the interposer, we can claim that *SP* has no legal floorplan because all floorplans of *SP* must be taller or wider than the interposer. Thus, exploring the floorplans of *SP* can be skipped. Note that illegal branch cutting can guarantee that the solution quality does not degrade.

3.2 Inferior Branch Cutting

EFA sequentially explores the floorplans of each sequence pair, and then stores the best floorplan into F_{best}. The basic idea of inferior branch cutting is that if we can predict that all floorplans of a sequence pair are worse than the current F_{best}, we can avoid exploring the floorplans of the sequence pair. Let L_{min} denote a lower bound on wirelength with respect to a sequence pair *SP*, namely the total wirelength of each floorplan of *SP* must be not shorter than L_{min}. Therefore, if L_{min} is longer than the wirelength of F_{best}, we can claim that all floorplans of *SP* are worse than F_{best}. Thus, the exploration of the floorplans of *SP* can be skipped.

Let L_{min} consist of LX_{min} and LY_{min} that respectively denote the horizontal and vertical lower bounds on wirelength with respect to *SP*. Because obtaining a correct lower bound on wirelength is a difficult problem, we use a heuristic method to get approximate LX_{min} and LY_{min}. Without loss of generality, we explain how to estimate LY_{min} below. Since F_{low} is the most compacted floorplan in the vertical direction among all floorplans of *SP*, we estimate LY_{min} by adding up the minimum vertical wirelength of each signal in F_{low}. However, F_{low} can be constructed by *SP* with several different die orientations, so a terminal of a signal may have more than one potential location. For example, in Fig. 4(b), the terminals of signal s are I/O buffers p_1, p_2, and p_3 respectively at dies d_1, d_2, and d_3, in which p_1 (p_3) has two potential locations $r_{1,1}$ and $r_{1,2}$ ($r_{3,1}$ and $r_{3,2}$) because d_1 (d_3) with rotations of 0°and 180° both can form F_{low}, and p_2 has four potential locations $r_{2,1}$, $r_{2,2}$, $r_{2,3}$ and $r_{2,4}$ because d_2 is square and it can form F_{low} by the rotations of 0°, 90°, 180°, and 270°. To handle the case of each terminal with multiple potential locations, the minimum vertical wirelength of signal s, denoted by $l_v(s)$, is computed by the following equations.

$$\begin{aligned} l_v(s) &= \max(ceiling(s) - floor(s), 0) \\ ceiling(s) &= \max\nolimits_{\forall p_i \in P(s)}(\min\nolimits_{\forall r_{i,j} \in R(p_i)}(y(r_{i,j}))) \\ floor(s) &= \min\nolimits_{\forall p_i \in P(s)}(\max\nolimits_{\forall r_{i,j} \in R(p_i)}(y(r_{i,j}))) \end{aligned} \tag{2}$$

where p_i is a terminal of signal s, $R(p_i)$ denotes a set of potential locations of p_i in F_{low}, and $y(r_{i,j})$ denotes the y-coordinate of the potential location $r_{i,j}$. Note that, if p_i is an escaping point, it only has a potential location. Fig. 4(b) shows the ceiling and floor values of signal s computed by Eq. (2). Similarly, LX_{min} can be estimated by adding up $l_h(s)$ for each signal s in F_{thin}, where $l_h(s)$ denotes the minimum horizontal wirelength of s.

EFA accelerated by inferior branch cutting cannot guarantee that the best floorplan still can be obtained since inferior branch cutting prunes solutions based on the approximate LX_{min} and LY_{min}. However, the experiments in Section 5.1 reveal EFA accelerated by inferior branch cutting has no quality loss on our test cases.

3.3 Die Orientation Pre-determination

Although the two proposed branch cutting methods can largely speed up EFA, the runtime of EFA is still considerable when the number of dies is more than 6. To achieve orders-of-magnitude acceleration, the die orientation pre-determination method generates a reference floorplan F_{ref} before EFA. Then, EFA only explores the floorplans whose die orientations are the same as F_{ref}. Namely, the orientation of each die is pre-determined before EFA, so EFA has no need to enumerate all floorplans for each sequence pair with different die orientations and the runtime of EFA is exponentially reduced.

The die orientation pre-determination method is a two-stage packing algorithm that tries every orientation of each die to construct F_{ref} by a greedy manner. Therefore, the orientation of each die in F_{ref} is a good reference to build other floorplans. The first stage of the greedy packing algorithm chooses a best pair of dies and packs them together to form the initial F_{ref}. The second stage iteratively attaches one of unpacked dies to F_{ref} until all dies are packed into F_{ref}. Fig. 5 shows the pseudo code of the greedy packing algorithm.

Lines 2-11 in Fig. 5 are the first stage of the greedy packing algorithm, in which the loop from lines 2 to 11 explores every pair of dies. For each pair (d_i, d_j), the nested loops in lines 3-10, 4-9, and 5-8 enumerate a set of packing results with different contact boundaries

```
Algorithm Greedy Packing
Input: die set $D$, signal set $S$, I/O buffer set $B$, escaping point set
$E$, the dimensions of the interposer;   Output: floorplan $F_{ref}$
1.  // initial packaging
2.  foreach pair($d_i$, $d_j$) of dies in $D$
3.    foreach orientation $r_i$ of $d_i$
4.    foreach orientation $r_j$ of $d_j$
5.    foreach boundary $b$ of $d_i$ to be the contact boundary
6.        Floorplan $F_{pair} = pack(d_i, d_j, r_i, r_j, b)$
7.        $getCost(F_{pair}, S, B, E)$
8.    end foreach
9.    end foreach
10.   end foreach
11. end foreach
12. Pick a $F_{pair}$ with the minimum cost to be $F_{ref}$
13. //incremental packing
14. while (at least a die is not in $F_{ref}$)
15.    foreach die $d$ in $D$ but not in $F_{ref}$
16.    foreach orientation $r$ of $d$
17.    foreach available boundary $b$ of $F_{ref}$
18.        Floorplan $F_{new} = pack(F_{ref}, d, r, b)$
19.        $getCost(F_{new}, S, B, E)$
20.    end foreach
21.    end foreach
22.    end foreach
23.    Pick a $F_{new}$ with the minimum cost to be $F_{ref}$
24. end while
25. return $F_{ref}$
```

Fig. 5. Pseudo code of the greedy packing algorithm.

and different orientations of d_i and d_j. If the contact boundary is the left/right/bottom/top boundary of d_i, we attach the right/left/top/bottom boundary of d_j to the contact boundary and align the center of d_j to the middle of the contact boundary. Line 7 computes the cost for each two-die packing result and then line 12 picks the one with the lowest cost from all two-die packing results to be the initial reference floorplan. To get the cost of a two-die packing result F_{pair}, the procedure *getCost* at line 7 puts F_{pair} to the center of the interposer, and then calculates the total HPWL of all signals in F_{pair}. If F_{pair} is illegal, a very large penalty is added to the cost of F_{pair}.

In the second stage, the loop from lines 14 to 24 iteratively attaches an unpacked die into F_{ref} until all dies are packed in F_{ref}. In each iteration, each unpacked die with an orientation is tentatively attached to an available boundary of F_{ref} to form a new floorplan, where an available boundary means a boundary that has not been attached by other dies. At the end of each iteration, line 23 picks the one with the lowest cost from all new floorplans to be the F_{ref} for the next iteration. Fig. 6 illustrates every possible result of packing die d_3 to F_{ref} that comprises dies d_1 and d_2. Because some packing results have die overlapping, procedure *pack* at line 18 will shift d_3 with the minimal displacement along the vertical or horizontal direction to resolve the overlapping.

The time complexity of the greedy packing algorithm is $O(n^3(|B|+|E|))$, where n is the number of dies. Because of the page limit, the detailed time complexity analysis is skipped. Although the greedy packing algorithm spends extra time to get a reference floorplan to pre-determine the die orientation, the time complexity of EFA can be reduced from $O(n!^2 4^n(|B|+|E|))$ to $O(n!^2(|B|+|E|))$ when the die orientation has been determined.

4 Network-flow-based Signal Assignment

After the floorplan of dies is determined, the next problem is how to solve the signal assignment problem (SAP) for micro-bumps and TSVs. Different from the traditional SAPs [6, 7] with only a single die, the SAP in 2.5D ICs has additional interconnects between different dies, i.e., internal nets. In order to well minimize *TWL* in Eq. (1), we build an MST for each signal as shown in Fig. 2(a), and then solve the SAP based on the topology of each signal's MST to get a signal assignment result with shorter *TWL* (Fig. 2(b)).

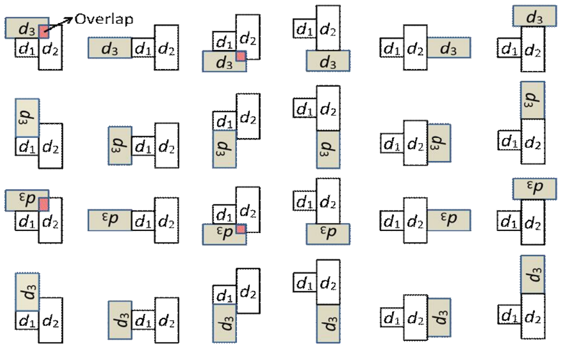


Fig. 6. Every possible result of attaching die d_3 to F_{ref} that comprises dies d_1 and d_2

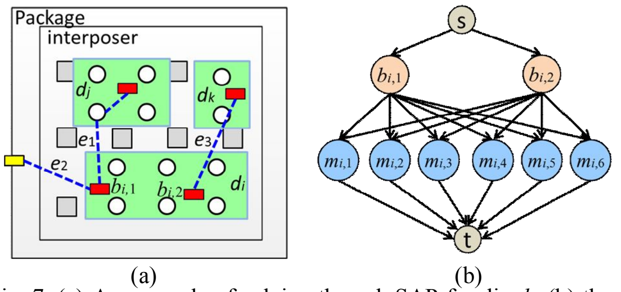


Fig. 7. (a) An example of solving the sub-SAP for die d_i; (b) the graph G_i corresponding to d_i in (a).

This work divides the SAP in a 2.5D IC into several sub problems. We first solve the sub-SAP for micro-bumps die-by-die, and then solve the sub-SAP for TSVs. Each sub-SAP is formulated into a network-flow problem and then a min-cost max-flow (MCMF) algorithm is used to solve the problem. The order of solving the sub-SAPs would affect the solution quality. This work solves the sub-SAP for each die in a decreasing order based on the number of each die's I/O buffers, because we found that this order can yield a better result. Note that, if the sub-SAP of a die is solved, the topology of each signal's MST will be updated immediately. For example, an MST edge originally connects I/O buffers $b_{i,x}$ and $b_{j,y}$; if the signal of $b_{i,x}$ is assigned to the micro-bump $m_{i,z}$, the MST edge will be split into an edge from $b_{i,x}$ to $m_{i,z}$ and an edge from $m_{i,z}$ to $b_{j,y}$. Fig. 2(a) shows the original MSTs before the signal assignment, and Fig. 7(a) shows the updated MSTs after the sub-SAP of die d_j is solved but the sub-SAPs of die d_i and d_k are not solved yet.

4.1 Flow Network and MCMF Algorithm

Without loss of generality, we use the example in Fig. 7(a) to illustrate how to solve the sub-SAP for die d_i by the MCMF algorithm, in which the sub-SAP of die d_j is already solved but the sub-SAP of die d_k is not solved yet. At first, a flow network $G_i=(V_i, E_i)$ is built for d_i. V_i is $\{s, t\} \cup M_i \cup B_i$, where s and t respectively denote the source and sink for the flow network, M_i denotes the set of micro-bumps in d_i, and B_i denotes the set of I/O buffers in d_i. E_i consists of a set of edges from s to each I/O buffer $b_{i,x} \in B_i$, a set of edges from each I/O buffer $b_{i,x} \in B_i$ to each micro-bump $m_{i,y} \in M_i$, and a set of edges from each micro-bump $m_{i,y} \in M_i$ to t. Fig 7(b) shows the flow network G_i corresponding to d_i in Fig. 7(a).

In the flow network G_i, the capacity of each edge is one, the cost of each edge leaving s or entering t is zero, and the cost of the edge from an I/O buffer $b_{i,x}$ to a micro-bump $m_{i,y}$, denoted by $c(b_{i,x}, m_{i,y})$, is formulated as follows:

$$c(b_{i,x}, m_{i,y}) = \alpha \times D(b_{i,x}, m_{i,y}) + \sum\nolimits_{\forall e \in ME(b_{i,x})} WC(m_{i,y}, t_i(e)) \qquad (3)$$

The concept behind Eq. (3) is that the cost of assigning a signal from $b_{i,x}$ to $m_{i,y}$ comprises the wirelength cost from $b_{i,x}$ to $m_{i,y}$ and the total wirelength cost from $m_{i,y}$ to the terminals connected to $b_{i,x}$ by the MST edges. In Eq. (3), $D(b_{i,x}, m_{i,y})$ denotes the distance between $b_{i,x}$ and $m_{i,y}$, $ME(b_{i,x})$ denotes the set of MST edges connecting $b_{i,x}$, $t_i(e)$ denotes the terminal of MST edge e not in d_i, and $WC(m_{i,y}, t_i(e))$ denotes the wirelength cost of connecting $m_{i,y}$ to $t_i(e)$. Note that, $t_i(e)$ could be an I/O buffer, a micro-bump, or an escaping point. In Fig. 7(a), $ME(b_{i,1})=\{e_1, e_2\}$ and $ME(b_{i,2})=\{e_3\}$, and $t_i(e_1)$, $t_i(e_2)$, and $t_i(e_3)$ are a micro-bump in d_j, an escaping point, and an I/O buffer in d_k, respectively. The wirelength cost, $WC(m_{i,y}, t_i(e))$, is formulated below:

$$WC(m_{i,y},t_i(e))=\begin{cases}\beta\times D(m_{i,y},t_i(e)), & \text{if } t_i(e) \text{ is a micro-bump}\\ \min(\alpha,\beta)\times D(m_{i,y},t_i(e)), & \text{if } t_i(e) \text{ is an I/O buffer}\\ \min(\beta,\gamma)\times D(m_{i,y},t_i(e)), & \text{if } t_i(e) \text{ is an escaping point}\end{cases} \quad (4)$$

where $D(m_{i,y}, t_i(e))$ denotes the distance between $m_{i,y}$ and $t_i(e)$, and recall that α, β and γ are the weight parameters defined in Eq. (1) respectively for intra-die nets, internal nets, and external nets. Based on Eq. (4), the wirelength cost between $m_{i,y}$ and $t_i(e)$ will not be overestimated.

After the capacity and cost of each edge in G_i are set, we perform an MCMF algorithm for G_i. If the MCMF algorithm assigns a unit of flow passing through the edge between $b_{i,x}$ and $m_{i,y}$, the signal of $b_{i,x}$ is assigned to $m_{i,y}$.

After the sub-SAP of each die is solved, the sub-SAP for TSVs in the interposer can be solved by the similar manner. We just treat the interposer as a big die, each escaping point as an I/O buffer, and each TSV as a micro-bump. Then, a flow network is built for the interposer and the MCMF algorithm is performed on the graph. Finally, if a unit of flow passes through an edge between an escaping point and a TSV, the signal of the escaping point is assigned to the TSV.

4.2 Window Matching Method to Accelerate

The flow network G_i has edges from every I/O buffer to every micro-bump in die d_i. As a result, the number of edges may be quite large and slow down the MCMF algorithm. In order to reduce the number of edges in G_i, we empirically observed the signal assignment results and found that the signal of an I/O buffer $b_{i,x}$ is usually assigned to one of its neighboring micro-bumps. Therefore, we have no need to treat all micro-bumps as the assignment candidates for $b_{i,x}$, and thus we only need to build edges between $b_{i,x}$ and its neighboring micro-bumps. However, if a set of I/O buffers is more than their assignment candidates, a feasible signal assignment result cannot be found.

To build the edges between I/O buffers and micro-bumps as few as possible and guarantee that a feasible result can be found, we use a window matching method to identify the assignment candidates for each I/O buffer. At first, for each I/O buffer $b_{i,x}$, we create a window $w_{i,x}$ whose center is $b_{i,x}$, and its width and height are $2\times m_{pitch}$, where m_{pitch} denotes the pitch between micro-bumps. Let $B(w_{i,x})$ and $M(w_{i,x})$ respectively denote the numbers of I/O buffers and micro-bumps in $w_{i,x}$. If $M(w_{i,x})-B(w_{i,x})<\lambda$, we iteratively extend each boundary of $w_{i,x}$ by m_{pitch} until $M(w_{i,x})-B(w_{i,x})\geq\lambda$, where λ is a user defined positive constant and is set to zero in this work. After that, for G_i, we only build the edges from each I/O buffer $b_{i,x}$ to the micro-bumps in $w_{i,x}$.

5 Experimental Results

The proposed algorithms were implemented in C/C++ language on a Linux machine with Intel Xeon 2.4GHz and 96G memory. This work adopts the MCMF program provided by the open-source library LEDA [12] to solve the network-flow problem. Due to no available test case for the multi-die floorplanning and signal assignment problems of 2.5D ICs, we generate 9 test cases based on the technology nodes of [3, 4] as follows.

We modify the ISPD08 global routing benchmarks [13] to build our test cases. ISPD08 benchmarks are traditional 2D ICs, so each benchmark contains only a single chip. To transform an ISPD08 benchmark into a test case for 2.5D ICs, we divide the chip in the original ISPD08 benchmark into several pieces by the slicing partitioning, and regard each piece as a die in a 2.5D IC. After that, we expand the width and height of the original benchmark by 10~20% to set the dimensions of the interposer, and then set the frame of a package to enclose the interposer according to the technology node presented in [14]. In addition, we randomly select a set of nets whose pins are in different dies to be the signals traveling in the 2.5D IC, and

Table 1: Test Cases Information.

Testcase	**Bench**	$\|D\|$	$\|S\|$	$\|B\|$	$\|E\|$	$\|T\|$	$\|M\|$
t4s	adaptec1	4	1019	2104	789	2025	61752
t4m	adaptec4	4	4152	8392	1174	8649	261630
t4b	bigblue3	4	11232	22701	1033	10201	308024
t6s	adaptec2	6	1081	2192	639	3481	105950
t6m	bigblue1	6	5945	12848	1162	2025	61752
t6b	adaptec5	6	13072	26314	1192	7140	216688
t8s	adaptec3	8	1036	2114	882	8649	260604
t8m	bigblue2	8	7000	14162	1391	5550	168917
t8b	bigblue4	8	11544	23242	1049	13806	416021

we add escaping points for some signals to create external nets. Based on the design rule mentioned in [4], we set the pitch of micro-bumps in each die and the pitch of TSVs in the interposer to 0.04 *mm* and 0.2 *mm*, respectively. The statistics of our test cases are shown in Table 1, in which the column "Bench" denotes that the test case is generated from which ISPD08 benchmark, and $|D|$, $|S|$, $|B|$, $|E|$, $|T|$, and $|M|$ respectively denote the numbers of dies, signals, I/O buffers, escaping points, TSVs, and micro-bumps in each test case.

5.1 Effectiveness of EFA

Table 2 compares various EFAs to show the effectiveness of the proposed acceleration techniques, in which EFA_{ori}, EFA_{c1}, EFA_{c2}, EFA_{c3}, and EFA_{dop} respectively denote EFA without any acceleration technique, with illegal branch cutting, with inferior branch cutting, with both illegal and inferior branch cuttings, and with die orientation pre-determination. Notably, the two branch cutting methods speed up EFA by skipping the exploration of different die orientations for some sequence pairs. However, when the orientation of each die is pre-determined, each sequence pair has only a floorplan to explore and thus the two branch cutting methods get no benefit for runtime saving. Accordingly, we do not simultaneously adopt die orientation pre-determination and the two branch cuttings to speed up EFA.

In Table 2, TWL denotes the total wirelength (10^6*mm*) computed by Eq. (1) after the SAP is solved by the proposed network-flow-based algorithm, and FT denotes the runtime (sec) of the floorplanning stage. Since the TWLs of EFA_{c1}, EFA_{c2}, and EFA_{c3} are the same as EFA_{ori}, their TWLs are not listed in Table 2. Note that, because EFA_{ori}, EFA_{c1}, EFA_{c2}, and EFA_{c3} spend too much runtime for test cases t8s, t8m, and t8b, we force them to jump out of the floorplanning stage after 12 hours and then return their best found floorplan to the signal assignment stage. Generally speaking, with the same runtime, EFA_{c1}, EFA_{c2} and EFA_{c3} will explore more floorplans than EFA_{ori}, so they may find better solutions than EFA_{ori}. However, unluckily, EFA_{c1}, EFA_{c2} and EFA_{c3} do not find better solutions than EFA_{ori} for t8s, t8m, and t8b in 12 hours. Table 2 reveals that the speedups of EFA_{c1}, EFA_{c2}, and EFA_{c3} increase when the number of dies increases. When the number of dies is 6, EFA_{c1}, EFA_{c2} and EFA_{c3} can averagely achieve 3.94X, 2.37X and 5.52X speedup for EFA_{ori}, respectively. Moreover, although EFA_{dop} would slightly increase TWL, it gets orders-of-magnitude speedup. Table 2 reveals that EFA_{dop} gets 11137X speedup for test cases t6m. Also, for the test cases with 8 dies, the TWLs obtained by EFA_{dop} in 3 hours are much shorter than those obtained by EFA_{ori} in 12 hours. According to the experimental results shown in Table 2, in order to strike a good balance between TWL and runtime, we design a hybrid flow denoted by EFA_{mix} that invokes EFA_{c3} when the number of dies is not greater than 5; otherwise it invokes EFA_{dop}. EFA_{mix} will be used in the following experiments.

5.2 Signal Assignment Results

Table 3 compares three signal assignment algorithms that all read the multi-die floorplanning results generated by EFA_{mix}, in which $MCMF_{ori}$ and $MCMF_{fast}$ respectively denote the proposed network-flow-based algorithm without and with the window matching method for acceleration. In addition, we implemented a greedy signal assignment algorithm for comparison. The greedy algorithm solves the sub-SAP for micro-bumps die-by-die, and then solves the sub-SAP for TSVs in the interposer. When the greedy algorithm solves the sub-SAP for die d_i, it sequentially assigns the signal from each I/O buffer $b_{i,x}$ to

Table 3: Comparison between signal assignment algorithms

Testcase	MCMF$_{ori}$		MCMF$_{fast}$		Greedy	
	TWL(10^6)	AT (s)	TWL(10^6)	AT (s)	TWL(10^6)	AT (s)
t4s	10.41	277.06	10.43	27.27	11.88	0.65
t4m	-	Crash	36.24	375.63	44.88	7.41
t4b	-	> 12hr	63.27	854.56	76.40	17.06
t6s	8.84	234.75	8.84	27.36	11.62	0.82
t6m	30.50	564.45	30.52	81.45	35.64	1.62
t6b	-	> 12hr	55.24	726.73	65.28	12.51
t8s	21.80	861.27	21.80	104.27	26.29	2.54
t8m	33.12	1764.50	33.15	176.45	42.30	4.43
t8b	-	> 12hr	59.20	859.13	67.35	19.15
Ratio	0.999	8.786	1	1	1.208	0.225

Table 4: Comparison between MCMF$_{fast}$ and [5].

Testcase	MCMF$_{fast}$		[5]		[5] + window matching	
	TWL	AT (s)	TWL	AT (s)	TWL	AT (s)
t4s'	3.49	11.43	3.60	82.06	3.70	11.98
t4m'	15.86	233.67	-	Crash	16.65	218.98
t4b'	43.40	659.03	-	> 12hr	45.18	636.6
t6s'	1.89	14.11	1.99	124.64	2.05	14.11
t6m'	19.18	54.02	19.79	412.55	20.26	54.1
t6b'	37.68	513.02	-	> 12hr	40.26	508.85
t8s'	4.53	44.85	4.80	353.05	5.04	43.47
t8m'	13.71	181.16	14.56	1220.72	14.81	131.83
t8b'	37.29	835.03	-	> 12hr	39.87	876.14
Ratio	1		1.05		1.07	

a micro-bump $m_{i,y}$ that has no assigned signal and has the minimum cost of Eq. (3). Finally, the greedy algorithm uses the similar idea to solve the sub-SAP for TSVs in the interposer.

The column "AT" in Table 3 denotes the runtimes of the signal assignment algorithms. Table 3 reveals that MCMF$_{ori}$ cannot obtain the signal assignment results for test cases t4b, t6b, and t8b in 12 hours because these three test cases have much more nets (>10000) than other test cases such that the SAP is too complicated to be solved by the MCMF algorithm. In addition, MCMF$_{ori}$ crashes on test case t4m since the memory is run out by the MCMF algorithm. This implies that the number of edges in the flow network is unaffordable to the MCMF algorithm. In contrast, MCMF$_{fast}$ uses the proposed window matching method to reduce the edges in the flow network, so MCMF$_{fast}$ has no crash problem and can averagely achieve 8.8X faster compared to MCMF$_{ori}$ with only 0.1% increase of TWL. This demonstrates that the proposed window matching method can effectively reduce the number of edges in a flow network to largely speed up the MCMF algorithm. In addition, compared to the greedy algorithm, MCMF$_{fast}$ averagely gets 20.8% shorter wirelength. This implies MCMF$_{fast}$ strikes a good balance between the runtime and quality.

The paper of [5] uses a bipartite matching algorithm to solve the SAP in 2.5D ICs, but the algorithm of [5] does not consider the signal assignment for TSVs and cannot handle the multi-terminal signals. To make the comparison feasible, we modify our test cases by removing the multi-terminal signals and the signals connecting escaping points. Table 4 shows the comparison between MCMF$_{fast}$ and [5]. The results of test cases t4m', t4b', t6b', and t8b' cannot be obtained by [5] in 12 hours because the bipartite matching graphs built by [5] have too many edges such that each problem is too complicate for the bipartite matching algorithm. To reduce the number of edges in each graph, the proposed window matching method is integrated into [5]. Table 4 reveals that the window matching method can largely speed up the algorithm of [5], and MCMF$_{fast}$ can get shorter TWL than [5] without and with the window matching method by 5% and 7%, respectively.

6 Conclusions and Future Works

This paper addresses a multi-die floorplanning problem and a signal assignment problem to shorten the interconnects in 2.5D ICs. An enumeration-based floorplanning algorithm is presented with three acceleration techniques to solve the multi-die floorplanning problem. Moreover, the proposed network-flow-based algorithm can effectively solve the signal assignment problem. To speed up the network-flow-based algorithm, a window matching method is also presented in this paper.

A post-floorplan optimization algorithm is presented in [16] to further reduce the wirelength of the floorplan of a 2D-IC by shifting the components in the floorplan. Extending the algorithm of [16] and integrating it into our flow is our future work.

References

[1] D. H. Kim et al, "A study of Through-Silicon-Via impact on the 3D stacked IC layout," in *Proc. of ICCAD*, pp. 674-680, 2009.

[2] J. Cong et al, "Thermal-aware 3D IC placement via transformation," in *Proc. of ASP-DAC*, pp. 780-785, 2007.

[3] P. Dorsey, "Xilinx stacked silicon interconnect technology delivers breakthrough FPGA capacity, bandwidth, and power Efficiency," Xilinx White Paper: Virtex-7 FPGAs, 2010.

[4] Liam Madden, "Heterogeneous 3-D stacking, can we have the best of both (technology) worlds," in *Proc. of ISPD*, pp. 1-2, 2013

[5] Y.-K. Ho and Y.-W. Chang, "Multiple chip planning for chip-interposer codesign," *in Proc. of DAC*, 2013.

[6] R.-J. Lee and H.-M. Chen, "Row-based area-array I/O design planning in concurrent chip-package design flow," in *Proc. of ASP-DAC*, pp. 837-842, 2011.

[7] H.-C. Lee and Y.-W. Chang, "A chip-package-board co-design methodology," in *Proc. of DAC*, pp. 1082-1087, 2012.

[8] D. J.-H. Huang and A. B. Kahng, "Partitioning-based standard-cell global placement with an exact objective," in *Proc. of ISPD*, pp 18-25, 1997.

[9] H. Murata et al, "Rectangle packing based module placement", in *Proc. ICCAD*, pp. 482-479, 1995.

[10] X. Tang, and D. F. Wong, "FAST-SP: A fast algorithm for block placement based on sequence pair," in *Proc. of ASP-DAC*, pp. 521-526, 2001.

[11] H. Murata, and E. S. Kuh, "Sequence-pair based placement method for hard/soft/pre-placed modules," in *Proc. of ISPD*, pp. 167-172, 1998 .

[12] http://www.algorithmic-solutions.info/leda_manual/manual.html

[13] http://archive.sigda.org/ispd2008/contests/ispd08rc.html

[14] Intel's Packaging Databook Chapter 15: The Chip Scale Package

[15] W.-H. Liu, T.-K. Chien and T.-C. Wang, "Metal Layer Planning for Silicon Interposers with Consideration of Routability and Manufacturing Cost," in *Proc. of DATE*, 2014.

[16] X. Tang et al, "Minimizing wire length in floorplanning," *IEEE TCAD*, 25(9), pp. 1744-1753, 2006.

Table 2: Comparison between the proposed EFA with different acceleration techniques.

Testcase	EFA$_{ori}$			EFA$_{c1}$		EFA$_{c2}$		EFA$_{c3}$		EFA$_{dop}$			
	TWL(10^6)	FT (s)	speedup	FT (s)	speedup	FT (s)	speedup	FT (s)	speedup	TWL(10^6)	WL incr. %	FT (s)	speedup
t4s	10.43	0.75	1	0.52	1.44	0.74	1.01	0.49	1.53	10.44	0.05%	0.72	1.04
t4m	36.24	5.47	1	5.21	1.05	6.01	0.91	4.59	1.19	36.26	0.05%	8.63	0.63
t4b	63.27	37.81	1	36.88	1.03	34.46	1.01	17.48	2.16	63.27	0.00%	19.67	1.92
t6s	8.84	7317.47	1	1350.71	5.42	2208.54	3.31	958.02	7.64	8.84	0.01%	2.87	2549.64
t6m	30.51	7821.50	1	3712.12	2.11	5086.55	1.54	3528.87	2.22	30.52	0.03%	6.88	11136.85
t6b	55.23	10776.70	1	2505.49	4.30	4775.18	2.26	1607.80	6.70	55.24	0.02%	17.31	622.57
t8s	26.27	12 hr	-	12 hr	-	12 hr	-	12 hr	-	21.80	-	11289	-
t8m	39.05	12 hr	-	12 hr	-	12 hr	-	12 hr	-	33.15	-	11146.20	-
t8b	69.06	12 hr	-	12 hr	-	12 hr	-	12 hr	-	59.20	-	11351.70	-
Ratio			1		2.56		1.67		3.57		0.03%		2385.44