Metal Layer Planning for Silicon Interposers with Consideration of Routability and Manufacturing Cost*

Wen-Hao Liu, Tzu-Kai Chien, and Ting-Chi Wang
Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan
dnoldnol@gmail.com, cakeboy1029@gmail.com, tcwang@cs.nthu.edu.tw

***Abstract* – A 2.5D IC provides a silicon interposer to integrate multiple dies into a package, which not only offers better performance than 2D ICs but also has lower manufacturing complexity than true 3D ICs. In an interposer, routing wires connect signals between dies or route signals from dies to the package substrate. The number of metal layers in an interposer is one of the critical factors to affect the routability and manufacturing cost of the 2.5D IC. Thus, how to achieve 100% routing completion rate in an interposer using a minimum number of metal layers plays a key role for the success of a 2.5D IC. This paper presents a global-routing-based metal layer planner called VGR to identify a minimal number of metal layers for an interposer with consideration of routability and manufacturing cost. Also, VGR can identify a good stacking order of the horizontal and vertical layers in an interposer such that the routing solution in the interposer costs fewer vias. To our best knowledge, this paper is the first study to solve the metal layer planning problem for silicon interposers.**

1 Introduction

Two classes of 3D ICs are under development today. The first one consists of true 3D ICs [1], 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. However, stacked dies cannot easily dissipate heat, plus TSVs used in active dies have their own performance and production issues, making the implementation of true 3D ICs more problematic and difficult.

Another class, which has been seen as an alternative approach to true 3D ICs, comes from silicon interposer-based 2.5D ICs. A 2.5D IC places active dies on a silicon interposer, which in turn is placed on a package substrate. Only the interposer has TSVs, while the active dies (except stacked memory dies where heat and power distribution issues are less critical) do not have any TSV. Besides, the interposer does not contain any active transistors but interconnects and decoupling capacitors only. A successful 2.5D IC that is in volume production today is Xilinx's Virtex-7 2000T FPGA device [2, 3], in which four FPGA dies (28-*nm* process node) are mounted on top of a low-risk and high-yield interposer (65-*nm* process node) by flipping them, and the via-first technique is adopted in which each TSV is attached to the bottom metal layer of the interposer. The metal layers and TSVs in the interposer provide high-bandwidth and low-latency interconnects that connect to each die using micro-bumps and to the package substrate by C4 bumps. An example of an interposer-based 2.5D IC is shown in Fig.1.

To our best knowledge, the size of an interposer can be up to 30×30 mm^2 [4]. Due to the large size, the mask cost for each metal layer in an interposer is higher. Thus, how to achieve 100% routing completion rate in an interposer using a minimum number of metal layers plays a key role for the success of a 2.5D IC. In this paper, we assume that the floorplan of dies on an interposer is given and the signals are already assigned to the micro-bumps and TSVs. We study how to plan a proper number of metal layers for an interposer such that each signal can be successfully routed by a global router in the interposer without causing any overflow. This is a challenging problem. If the number of metal layers is not planned enough for the interposer, a router may find no feasible routing result. On the other hand, planning surplus metal layers to the interposer can resolve the routability issue but needs higher manufacturing cost. Moreover, the metal layer planning problem may not be solved only once in a design flow. If it can be solved by a tool fast enough, the tool may be frequently invoked in the design flow to help evaluate different floorplans of dies such that a floorplan inducing low manufacturing cost and good routability to the interposer can be determined. Thus, our goal is to solve the metal layer planning problem efficiently.

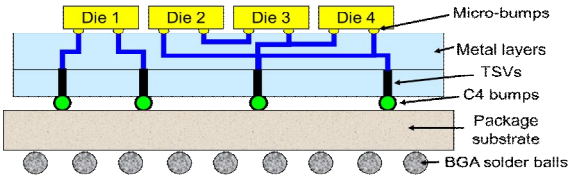

Fig. 1. Example of an interposer-based 2.5D IC

We formulate the metal layer planning problem as a variable-layer global routing (VLGR) problem. Different from a typical 2D-IC global routing problem [5] in which a fixed amount of routing layers is given as the input, the VLGR problem asks to find an overflow-free routing result that requires as few metal layers as possible. To solve this problem, we develop a variable-layer global router **VGR** that can find a layer configuration (explained later) to strike a good balance between manufacturing cost and routability. Also, a panel-based evaluation method is presented to evaluate the routability of layer configurations. Finally, VGR can plan a good stacking order for the horizontal and vertical layers such that its routing result costs fewer vias. To our best knowledge, this paper is the first study to solve the metal layer planning problem for silicon interposers

The rest of this paper is organized as follows. Section 2 formulates the VLGR problem. Section 3 presents the proposed global router VGR. Section 4 reports the experimental results. Finally, conclusions are drawn in Section 5.

2 Preliminaries

This section first reviews the background of global routing, and then describes how to model the metal layer planning problem into a VLGR problem.

2.1 Background of Global Routing Problem

In the typical global routing problem, the given k-layer chip is partitioned into a 3D array of uniform g-cells (Fig. 2(a)), and then the array of g-cells is modeled by a 3D grid graph $G^k(V^k, E^k)$ (Fig. 2(b)), where V^k denotes the set of g-cells, and E^k refers to the set of grid edges (g-edges). Each g-edge is termed by the proximity of the related g-cells to its two end nodes. The capacity ($c(e)$) of a g-edge e indicates the number of routing tracks that can legally cross the abutting boundary of g-cells. The number of wires that pass through g-edge e is called the demand ($d(e)$) of e. The overflow of e is defined as $max(0, d(e)–c(e))$. The goal of global routing is to find the routing paths to connect the pins of each net in $G^k(V^k, E^k)$, and reduce overflows and then wirelength.

*This work was supported in part by the National Science Council of Taiwan under Grant No. NSC-102-2220-E-007-012.

978-3-9815370-2-4/DATE14/©2014 EDAA

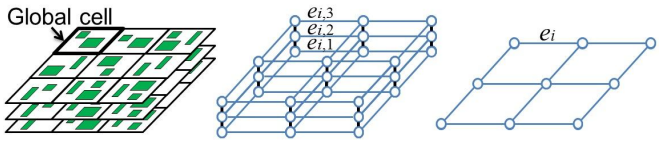

Fig. 2. (a) a 3-layer chip that is partitioned into an array of g-cells (b) a 3D grid graph (c) a 2D grid graph.

Because directly performing global routing on the 3D grid graph is time-consuming, most global routers [6-12] compress $G^k(V^k, E^k)$ into a 2D grid graph $G(V, E)$ (Fig. 2(c)), then solve the 2D global routing problem to get a 2D routing result, and finally layer assignment techniques [12, 13] are adopted to map the 2D routing result to $G^k(V^k, E^k)$ to obtain a 3D result. In the 2D grid graph $G(V, E)$, the capacity of each g-edge in E is obtained by accumulating the capacities of its corresponding edges in E^k. For example, the capacity of e_i in Fig. 2(c) is obtained by adding up $c(e_{i,1})$, $c(e_{i,2})$ and $c(e_{i,3})$ of Fig. 2(b).

2.2 Variable-layer Global Routing Problem

The notations used in the VLGR problem are introduced as follows.

- (L_h, L_v) denotes a layer configuration, where L_h and L_v are the numbers of metal layers with horizontal and vertical routing directions, respectively. In this paper, we call a layer with horizontal/vertical routing direction a horizontal/vertical layer.
- t_h and t_v denote the capacities of each horizontal and vertical g-edges in $G^k(V^k, E^k)$, respectively. In this paper, we assume that the wire width and wire spacing on each layer are uniform for simplification, so the capacity of a horizontal/vertical g-edge on each horizontal/vertical layer is a constant.
- $G(V, E)$ is a 2D grid graph modeled from an interposer. The routing resources of every metal layer in the interposer are aggregated into G. The capacities of each horizontal and vertical g-edges in $G(V, E)$ are $t_h \times L_h$ and $t_v \times L_v$, respectively.
- N is the set of nets to be connected in the interposer, where each net consists of a collection of pins. As shown in Fig. 1, in an interposer, a set of pins corresponding to micro-bumps is located on the top metal layer and a set of pins corresponding to TSVs is located on the bottom metal layer.
- Alternate stacking constraint: Mostly existing manufacturing solutions request that the horizontal layers and vertical layers have to alternately stack to reduce coupling effect between layers, so the difference between L_h and L_v is at most 1.

The VLGR problem is defined as follows: Given $G(V, E)$, t_h, t_v, and N, find a layer configuration (L_h, L_v) such that N can be globally routed on $G(V, E)$ without any overflow and $|L_h - L_v| \le 1$ is satisfied to obey the alternate stacking constraint. The objective is to minimize the following cost function.

$$\lambda \times (L_h + L_v) + TWL_N \qquad (1)$$

where λ is a user-defined constant and TWL_N denotes the total routing wirelength of all nets in N. In this work, we set λ to a very large constant to make (L_h+L_v) dominate Eq. (1).

The most intuitive method to solve the VLGR problem is to run a global router to test several layer configurations. The one with the minimum layer number and including an overflow-free routing result is the solution to this problem. However, this method is inefficient because it has no better way to know which layer configurations need to test, and the router has to route from scratch in every test. In contrast, the proposed VGR can identify a set of configurations that are worth to test, and the routing result of VGR can be incrementally updated for each test to save time. Moreover, judging whether a layer configuration has an overflow-free result is challenging. A global router may spend hours to evaluate a hard-to-route layer configuration, and then report this configuration may have no overflow-free result since the router is unable to get it. In contrast, the proposed VGR can accurately and fast evaluate the routability of a configuration using a panel-based method. We detail the proposed VGR in the next section.

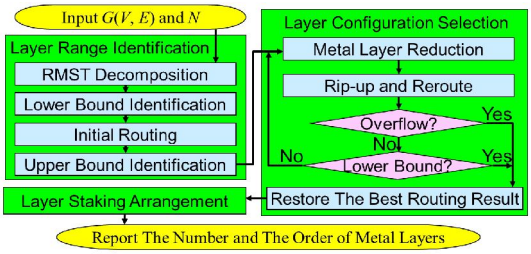

Fig. 3. Flow of VGR.

3 The Proposed VGR

Fig. 3 shows the flow of VGR that consists of three phases: layer range identification, layer configuration selection, and layer stacking arrangement. To easily explain the flow of VGR, we classify layer configurations into two types. If a layer configuration contains at least an overflow-free routing result, the layer configuration is defined to be **routable**. If it is impossible to identify an overflow-free result for a layer configuration by VGR, the layer configuration is defined to be **unroutable**. Fast and accurately judging that a layer configuration is routable or unroutable is the key for the success of VGR.

The layer range identification phase first uses an analytical method to get a lower-bound layer configuration (LB_h, LB_v) such that any layer configuration (L_h, L_v) is unroutable if $L_h < LB_h$ and $L_v < LB_v$. Next, this phase generates an initial routing result based on (LB_h, LB_v), and then analyzes the routing result to get an upper-bound layer configuration (UB_h, UB_v) and guarantee that any layer configuration (L_h, L_v) is routable if $L_h \ge UB_h$ and $L_v \ge UB_v$. After the lower and upper bounds are identified, a layer range can be formed. Namely, a layer configuration (L_h, L_v) is in the layer range if $LB_h \le L_h \le UB_h$ and $LB_v \le L_v \le UB_v$; each layer configuration except the upper-bound one may need to be examined whether it is routable. In order to avoid too many layer configurations included in the layer range, this phase attempts to identify a tighter lower and upper bounds.

The second phase explores each layer configuration in the layer range to test whether it is routable, and then selects a routable one that obeys the alternate stacking constraint and has the minimum cost for Eq. (1) to be the best solution of the VLGR problem. This phase maintains a global routing result and updates the result incrementally for each tested layer configuration to see whether an overflow-free routing result can be obtained.

The final phase decides the stacking order for the best layer configuration identified by the previous phase. The stacking order of horizontal layers and vertical layers in the interposer will impact the via count in its routing result.

3.1 Layer Range Identification

To identify the lower-bound layer configuration of the layer range, we first have to know what situation must cause overflows. At first, VGR decomposes each net in N into two-pin subnets by a rectilinear minimum spanning tree (RMST) algorithm. For each g-cell u, if a two-pin subnet has a terminal in u and another terminal not in u, this two-pin subnet is defined to be a *global segment* of u. If the number of global segments of u exceeds the total capacity of the g-edges connecting to u, overflows must happen. Accordingly, we want to identify a layer configuration (LB_h, LB_v)

to ensure that $|LB_h - LB_v| \le 1$ and the number of global segments of each g-cell is not greater than the total capacity of its adjacent g-edges while (LB_h, LB_v) is as tight as possible. The approach to obtain (LB_h, LB_v) is detailed as follows.

Let $|S(u)|$ denote the number of global segments of g-cell u. The layer configuration ($LB(u)_h$, $LB(u)_v$) ensures that $|S(u)|$ is not greater than the total capacity of the g-edges connecting to u and makes $|LB(u)_h - LB(u)_v| \le 1$. We can find ($LB(u)_h$, $LB(u)_v$) for each g-cell u in $G(V, E)$ by the following equations.

$$(LB(u)_h, LB(u)_v) = \begin{cases} (l_u,\ l_u) & \text{if} \quad l_u \times (c(u)_h + c(u)_v) = |S(u)| \\ (l_u',\ l_u) & \text{else if } l_u' \times c(u)_h + l_u \times c(u)_v \ge |S(u)| \\ (l_u,\ l_u') & \text{else if } l_u \times c(u)_h + l_u' \times c(u)_v \ge |S(u)| \\ (l_u',\ l_u') & \text{otherwise} \end{cases}$$

$$l_u = \left\lfloor \left(\frac{|S(u)|}{c(u)_h + c(u)_v} \right) \right\rfloor, \quad l_u' = l_u + 1, \qquad (2)$$

where $c(u)_h$ and $c(u)_v$ respectively denote the total capacities of the horizontal and vertical g-edges connecting to u. For example, if a g-cell u is connected by two horizontal g-edges and two vertical g-edges, $c(u)_h$ and $c(u)_v$ will be $t_h \times 2$ and $t_v \times 2$, respectively. Note that $c(u)_h$ and $c(u)_v$ would vary when u is located on the boundaries or corners of $G(V, E)$. In order to make the lower bound tighter, the one with the maximum $LB(u)_h + LB(u)_v$ among all g-cells is selected to be (LB_h, LB_v).

After LB_h and LB_v are obtained, we respectively initialize the capacities of the horizontal and vertical g-edges in $G(V, E)$ to $t_h \times LB_h$ and $t_v \times LB_v$, and then invoke the initial routing stage to get an initial routing result. In the initial routing stage, each global segment is routed by L-shaped pattern routing first. If a global segment has overflows, the segment will be rerouted by monotonic routing. After that, to identify the upper-bound layer configuration of the layer range, VGR analyzes the initial routing result to get a layer configuration (UB_h, UB_v) as tight as possible and under which the initial routing result can be overflow-free. For example, if a horizontal g-edge e_i has the maximum demand $d(e_i)$ among all horizontal g-edges, UB_h is set to be $\lceil d(e_i)/t_h \rceil$ to ensure that no horizontal g-edge has overflows; similarly, UB_v is set to be $\lceil d(e_j)/t_v \rceil$ where g-edge e_j has the maximum demand among all vertical g-edges. If $|UB_h - UB_v| > 1$, we increase the smaller one of UB_h and UB_v to make $|UB_h - UB_v| = 1$ and obey the alternate stacking constraint. Since a larger maximum demand of grid edges would result in larger UB_h or UB_v, the initial routing stage attempts to reduce the maximum demand on grid edges to make the upper bound tighter.

3.2 Layer Configuration Selection

After the layer range is obtained, we have tried three different methods to find the best layer configuration in the range. The first method explores the layer configurations from the lower-bound one to the upper-bound one in the range to find a routable layer configuration with the minimum layer number. It initially performs rip-up and reroute (R&R) for the initial routing result under layer configuration (LB_h, LB_v). Then, if an overflow-free result cannot be obtained, the first method repeatedly adds one more metal layer and then performs R&R again until a routable layer configuration is found. However, to recognize that a layer configuration is unroutable, one has to perform R&R under the configuration with very long runtime until overflow reduction is stuck. Thus, the first method may explore several unroutable configurations and be time-consuming. Due to the same reason, the second method, binary search, may also be time-consuming if it explores several unroutable configurations before finding the routable one. Accordingly, this work adopts the third method to find the best layer configuration, which explores the layer configurations from the upper to lower bounds in the layer range. The third method is much faster than the other two methods since it explores at most one unroutable configuration.

At the beginning of the layer configuration selection phase in Fig. 3, we set $L_h = UB_h$ and $L_v = UB_v$. Then, VGR reduces either L_h or L_v by 1 in the metal layer reduction stage, which may induce overflows in the initial overflow-free routing result since the capacities of one half of g-edges in $G(V, E)$ decrease. After that, the R&R stage iteratively reroutes the global segments in the initial routing result to intend obtaining an overflow-free routing result again. In the R&R stage, the efficient routing algorithms presented in [11] are used to quickly reduce overflows. For more details of the R&R stage, please refer to [11]. If an overflow-free result is obtained, the cost of the current routing result is calculated via Eq. (1) and the current result is treated as a feasible solution to save into a solution pool. The R&R stage and metal layer reduction stage are alternately performed until either an overflow-free result cannot be obtained or (L_h, L_v) already reaches the lower bound of the layer range. Finally, VGR decides the values of L_h and L_v by choosing a feasible solution with the minimum cost from the solution pool.

The primary issue in this phase is how to get a good reduction order for L_h and L_v in the metal layer reduction stage because different reduction orders would impact the solution quality. For example, our experiments reveal that (L_h=3, L_v=3) is a routable layer configuration for test case Sb1 (more detailed experimental results will be shown in Section 4). If a metal layer is removed, an overflow-free routing result can be obtained again for (L_h=3, L_v=2) but not for (L_h=2, L_v=3). This implies that (L_h=2, L_v=3) is a dead end. If VGR falls into this dead end, VGR would treat (L_h=3, L_v=3) as the best layer configuration. However, (L_h=3, L_v=2) is better.

In this phase, the R&R stage is iteratively launched to detect whether the explored layer configurations are routable. Each launch of the R&R stage only partially updates the routing result generated by the previous launch, so each launch of the R&R stage can be done in short time except for the last launch. The last launch has to spend long time to recognize that a layer configuration is unroutable, so the runtime of the last launched R&R stage almost dominates the total runtime of VGR. Thus, if we can detect a configuration that is unroutable earlier, we can avoid the last launch of the R&R stage to save runtime.

Section 3.2.1 presents a routability evaluation method to guide the metal layer reduction and Section 3.2.2 presents an early termination scheme to skip the last launch of the R&R stage.

3.2.1 Metal Layer Reduction

When L_h is not equal to L_v, we always reduce the larger one of L_h and L_v in order to obey the alternate stacking constraint. However, when L_h is equal to L_v, reducing which one of L_h and L_v becomes a problem. To handle the case where only one of (L_h–1, L_v) and (L_h, L_v–1) is routable, this work presents a metal layer reduction strategy to avoid VGR falling into the unroutable one.

Given an overflow-free result S under layer configuration (L_h, L_v), this stage would judge which one of (L_h–1, L_v) and (L_h, L_v–1) offers better routability first, and then moves to the one with better routability. A configuration with better routability means that feeding S into the R&R stage under the configuration can get a routing result with fewer overflows and shorter wirelength.

At the beginning of this stage, we tentatively reduce L_h by 1 from (L_h, L_v) and get a congestion map C_h from S under the layer configuration (L_h–1, L_v), and also tentatively reduce L_v by 1 from (L_h, L_v) and get a congestion map C_v from S under the layer configuration (L_h, L_v–1). Figs. 4(a) and 4(b) respectively show

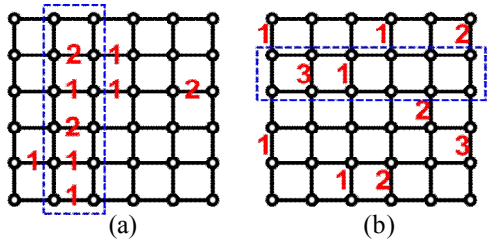

Fig. 4. Congestion maps of (a) (L_h–1, L_v) (b) (L_h, L_v–1).

examples of the overflow distribution in C_h and C_v, in which the numbers next to g-edges denote the overflows induced by the corresponding layer reduction. By evaluating the routability of C_h and C_v, we can know between (L_h–1, L_v) and (L_h, L_v–1) which offers better routability. However, the problem here is how to accurately evaluate the routability of C_h and C_v. We have tried to use the well-known congestion evaluation metrics such as total overflow [5], ACE [14] and WCI [15] metrics to evaluate the routability of C_h and C_v, but these metrics sometimes mislead VGR to choose the wrong layer configuration. Based on our observation, we found that the major reason for the misleading is that these metrics do not consider the congestion distribution. For instance, although Fig. 4(b) has higher total overflow and maximum overflow than those in Fig. 4(a), the overflows in Fig. 4(a) are more concentrated such that they may worsen routability. In addition, the paper in [11] indicates that the congestions in real designs often range horizontally or vertically as a wall, and the routing is difficult as the congestion wall is long and thick. Thus, we present a panel-based routability evaluation method to consider the congestion distribution. Notably, a panel is ether a column or a row in the grid graph, in which a column comprises a set of horizontal g-edges and a row comprises a set of vertical g-edges. The dotted boxes in Fig. 4(a) and 4(b) highlight the most congested column in C_h, and most congested row in C_v, respectively.

We found that a layer configuration is routable or not usually depends on the most congested row or column in its congestion map. Thus, the panel-based routability evaluation method is designed based on this finding. The panel-based evaluation method first calculates a routing difficulty score for each column and each row of the evaluated congestion map, and then adds up the maximum score of rows and the maximum score of columns to get the routing difficulty score for the congestion map. A layer configuration with a higher routing difficulty score for its congestion map means that obtaining an overflow-free result under this configuration is more difficult.

Without loss of generality, we use an example in Fig. 5 to illustrate how the proposed evaluation method calculates the routing difficulty score for a row. Fig. 5(a) shows the value of $d(e)$–$c(e)$ for each vertical g-edge e in the row, in which the g-edges with a positive value are called overflowed edges while the g-edges with a negative value are called slack edges. The panel-based routability evaluation method for a row iteratively visits each vertical g-edge from right to left, and moves overflows from overflowed edges to the nearest slack edges to remove overflow. After each move, the value of $d(e)$–$c(e)$ of the overflowed edge decreases by 1 and the value of the slack edge increases by 1. Fig. 5(b) shows the updated graph after moving a unit of overflow from e_3 to e_2. If the moving distance from an overflowed edge to its nearest left slack edge is the same as that to its nearest right slack edge, the overflow is moved to the right one as shown in Fig. 5(c). The routing difficulty score for a row consists of the total moving cost and the surplus penalty. The cost of moving a unit of

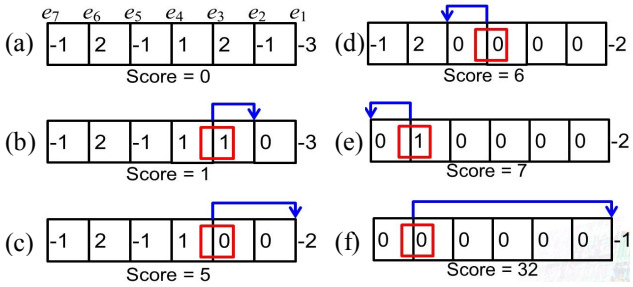

Fig. 5. Example of using the proposed panel-based routability evaluation method to score a row.

overflow to a slack edge is the square of its moving distance (explained later). If a row has not enough slack edges to eliminate all overflows, the remaining overflows are defined to be **surplus overflows**. The surplus penalty is obtained by multiplying the amount of surplus overflows by a very large constant. Figs. 5(d)-5(f) show the subsequent steps to calculate the routing difficulty score for the row, and the final score is shown in Fig. 5(f). The reason why we set the moving cost to the square of the moving distance is to emphasize that moving a unit of overflow further is more difficult. For example, a net crossing a congestion wall induces an overflow. If the overflow is in the middle of the congestion wall, the net may need to detour a lot to bypass the congestion wall. To capture this routing behavior, when overflow is in the middle of the congestion wall, our square cost function can give this situation a higher cost. Moreover, based on this scoring method, a panel with a long congestion wall will have a high routing difficulty score.

As mentioned above, when L_h is not equal to L_v, we can directly reduce the larger one of L_h and L_v by 1 because of the alternate stacking constraint. However, in this case, we still use the proposed method to evaluate the routability of (L_h–1, L_v) when L_h>L_v or (L_h, L_v–1) when L_v>L_h since the evaluation result can help us to decide whether the layer configuration selection phase can be early terminated for time saving. The reason will be detailed later.

3.2.2 Early Termination Scheme

Let layer configurations (L_h–1, L_v) and (L_h, L_v–1) are treated as the children of (L_h, L_v). During the layer configuration selection phase, if the children of (L_h, L_v) are both unroutable, we can claim that reducing a layer from (L_h, L_v) is meaningless since overflow-free results cannot be found in its children. In this work, we predict that a layer configuration is unroutable if the configuration has surplus overflows. Typically, a panel with surplus overflows means that the panel has not enough routing resource for nets to pass through. When a design has panels with surplus overflows, the design is mostly unroutable as indicated in [16]. Therefore, if the panel-based routability evaluation method reports that the children of (L_h, L_v) both have surplus overflows, the layer configuration selection phase can be early terminated for time saving.

Please note that the early termination scheme is a heuristic method based on our observation, so we cannot guarantee its prediction is always correct. However, our experiment shows that if the termination condition is triggered but we let VGR continue the R&R stage under the next layer configuration, VGR always cannot identify an overflow-free routing result.

3.3 Layer Stacking Arrangement

The stacking order of horizontal and vertical layers would impact the via count in the routing result. Due to the alternate stacking constraint, once the routing direction of the bottom metal

TABLE 1 THE LAYER CONFIGURATION SELECTION PHASE

iterations	Sb1				
	(L_h, L_v)	TOF_{init}	TOF_{end}	TWL	Acc. CPU
1	(5, 6)	114	0	9769354	14.97
2	(5, 5)	240	0	9769510	15.62
3	(4, 5)	635	0	9769734	16.27
4	(4, 4)	3949	0	9770573	17.01
5	(3, 4)	6371	0	9774261	18.45
6	(3, 3)	49316	0	9811629	21.13
7	(3, 2)	367773	0	10558189	56.90
8	(2, 2)	605434	212949	11920380	705.90

layer is determined, the stacking order of all horizontal and vertical layers is also determined. Thus, the goal of this stage is to decide the routing direction of the bottom layer.

Assume S is the 2D global routing result and (L_h, L_v) is the best layer configuration obtained by the previous phase such that $|L_h–L_v|\leq 1$. When $L_h–L_v=1$, the bottom layer has to be the horizontal layer to satisfy the alternate stacking constraint; similarly, when $L_v–L_h=1$, the bottom layer has to be the vertical layer. However, when $L_h=L_v$, this phase needs to determine how to stack layers for minimizing via count. At first, this phase builds two possible layer stacking structures for layer configuration (L_h, L_v), i.e., their bottom layers are horizontal and vertical, respectively. Next, two 3D grid graphs are built according to these two layer stacking structures, and then the fast greedy layer assignment algorithm presented in [12] is invoked to map the 2D result S to these two 3D grid graphs to get two 3D routing results. Finally, we choose the stacking order that offers fewer via count in its 3D result to be our final solution.

4 Experimental Results

The proposed router VGR is implemented in C/C++ and tested on a quad-core 2.4 GHz Intel Xeon-based Linux server with 16GB memory. Because no existing work has addressed our problem and no test case is available, we modified the placement solutions of Ripple [17] released by ISPD11 placement contest to be our global routing test cases. The statistics like the numbers of nets and pins for these test cases are available in [18].

These test cases are originally for 2D ICs, but they can capture the characteristics of interposers after our slight modification. Each of these test cases originally has a specified number of metal layers; we have removed this information to treat the number of metal layers as an undecided variable. Since interposers under our study do not contain any active transistors, we have removed the big macros from these test cases. Moreover, because pins of nets in interposers are mostly located on the top metal layer and some are located on the bottom layer, we randomly assign 90% pins to the top layer and 10% pins to the bottom layer. Finally, we assume that the wire width and wire spacing on each metal layer are uniform, and set t_h=15 and t_v=15.

4.1 Behavior of VGR

To better understand the behavior of VGR, Table 1 shows the global routing result after each iteration of the layer configuration selection phase for test case Sb1, in which the "iterations" column shows the iteration count, the "(L_h, L_v)" column shows the explored layer configuration in each iteration, the "TOF_{init}" and "TOF_{end}" columns respectively show the total overflow before and after the R&R stage in each iteration, the "TWL" and "Acc. CPU" columns respectively show the total wirelength of the routing result and the accumulated runtime after each iteration. Note that, the runtime unit is **second** in every table of this paper, and TWL does not include via count.

TABLE 2 SOLUTIONS IDENTIFIED BY VGR

	(LB_h, LB_v)	(UL_h, UL_v)	(L_h, L_v)	BD	CPU_1	CPU_2
Sb1	(2, 1)	(6, 6)	(3, 2)	H	705.90	56.20
Sb2	(2, 1)	(6, 5)	(3, 3)	V	599.61	-
Sb4	(2, 2)	(6, 7)	(3, 2)	H	635.27	49.68
Sb5	(1, 2)	(7, 8)	(3, 3)	V	137.44	-
Sb10	(2, 2)	(8, 9)	(2, 3)	V	2430.94	225.06
Sb12	(1, 2)	(6, 7)	(3, 4)	V	985.10	63.56
Sb15	(2, 1)	(8, 7)	(3, 3)	V	726.02	46.33
Sb18	(2, 1)	(7, 8)	(3, 3)	V	988.01	54.17

For the results shown in Table 1, the early termination scheme is not invoked, and each R&R stage stops when either an overflow-free routing result is obtained or overflow reduction is stuck. Because the routing result in the 8th iteration for Sb1 has overflows, the layer configuration selection phase terminates and then reports that the configuration (L_h=3, L_v=2) identified by iteration 7 is the best solution. To compare different routability evaluation metrics, we have tried to use total overflow metric [5], ACE metric [14] and WCI metric [15] to guide metal layer reduction. However, when these metrics are used in the metal layer reduction stage, VGR falls into layer configuration (L_h=2, L_v=3) and then fails to identify an overflow-free result. As a result, (L_h=3, L_v=3) is treated to be the best layer configuration which in fact is inferior to (L_h=3, L_v=2) reported by VGR.

Table 1 also indicates that the runtime of the last iteration dominates the total runtime of VGR. The reason is that the last launch of the R&R stage would struggle for the insufficient routing resources until overflow reduction is stuck and then finally give up. Table 2 shows the solutions identified by VGR for every test case and the effectiveness of the proposed early termination scheme, in which the "(LB_h, LB_v)" and "(UB_h, UB_v)" columns show the lower bound and the upper bound of the layer range, respectively; the "(L_h, L_v)" column shows the best layer configuration identified by the layer configuration selection phase; the "BD" column shows the routing direction of the bottom layer identified by the layer stacking arrangement phase; the "CPU_1" and "CPU_2" columns respectively show the total runtime of VGR without and with early termination scheme. Table 2 reveals that the early termination scheme can reduce the runtime of VGR by avoiding the last launched R&R stage for 6 out of 8 test cases. For Sb2 and Sb5, since the layer configurations (L_h=3, L_v=2) and (L_h=2, L_v=3) are unroutable to VGR but their congestion levels are not worse enough to trigger the termination condition, VGR cannot be early terminated. Averagely, VGR with the early termination scheme can achieve 10.8X speedup compared to that without the scheme.

4.2 Evaluation by NCTU-GR 2.0

To see how effective and efficient VGR is, we adopt a state-of-the-art global router NCTU-GR 2.0 [7] to route our test cases. Because NCTU-GR 2.0 needs a fixed amount of routing layers as the input, we manually set different layer configurations to run it. NCTU-GR 2.0 is a public global router that is selected to be the evaluation tool in DAC12 and ICCAD12 placement contests, and it can be download from [18]. In our experiments, NCTU-GR 2.0 is run with the default parameter values and terminates when an overflow-free routing result is identified or overflows cannot be reduced anymore.

For each test case, we first perform NCTU-GR 2.0 under the layer configuration identified by VGR. Then, we perform NCTU-GR 2.0 under the configurations with fewer layers or different stacking orders to see whether a better overflow-free result can be obtained. In Table 3, the configurations identified by VGR are highlighted to be boldface. Table 3 shows that VGR found the

routable configuration with the minimum number of metal layers as NCTU-GR 2.0 did for all test cases but Sb2. Notably, even if a test case has an unroutable and a routable layer configurations with the same layer number, VGR still can hit the routable one (see the routing results of Sb1, Sb4, Sb10, and Sb12 in Table 3). This implies that the proposed penal-based routability evaluation scheme well guide VGR to choose the layer configuration with better routability. In addition, the routing results of Sb5, Sb15 and Sb18 reveal that a configuration with different layer stacking orders would have 6% via count difference on average, and VGR can identify the stacking order with fewer via count.

Table 3 also shows that using NCTU-GR 2.0 to solve the metal layer planning problem is time-consuming. For example, VGR can examine several layer configurations to see whether they are routable and then identify an overflow-free result for Sb1 in 56 seconds, while NCTU-GR 2.0 costs 555 seconds for Sb1 to test only the layer configuration (L_h=3, L_v=2). Moreover, if we use NCTU-GR 2.0 to find the best layer configuration for Sb1, we also need to run NCTU-GR 2.0 under the layer configuration (L_h=2, L_v=2) to make sure that it cannot get an overflow-free result, before we can claim the layer configuration (L_h=3, L_v=2) is the best one. For this approach, NCTU-GR 2.0 additionally spends more than 5 hours on the layer configuration (L_h=2, L_v=2); the overall runtime of this approach is clearly much longer than VGR. Thus, due to its fast runtime, VGR is suitable for guiding the floorplanning of dies on an interposer to select a floorplan that offers good routability and low manufacturing cost to the interposer.

NCTU-GR 2.0 can identify an overflow-free result for Sb2 under layer configurations (L_h=3, L_v=2) and (L_h=2, L_v=3), explaining why VGR does not trigger the early termination for Sb2 in Table 2. The proposed early termination scheme believes that Sb2 can be further reduced by one more metal layer from (L_h=3, L_v=3). However, VGR cannot find the overflow-free results under (L_h=3, L_v=2) and (L_h=2, L_v=3) for Sb2. This implies that VGR is fast but still has room for improving its ability on solving overflows. Similarly, the early termination is not triggered for Sb5 in Table 2, because the layer configuration (L_h=3, L_v=2) seems potentially routable. Table 3 shows that the overflow value of Sb5 under layer configuration (L_h=3, L_v=2) is relatively small, only 72. If a router more powerful than NCTU-GR 2.0 is used, an overflow-free result may be achievable.

5 Conclusions

This work studies a metal layer planning problem for silicon interposers and presents a variable-layer global router called VGR to solve the problem. Extensive experiments are also conducted to show the effectiveness and efficiency of VGR.

REFERENCES

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

[2] K. Saban, "Xilinx stacked silicon interconnect technology delivers breakthrough FPGA capacity, bandwidth, and power efficiency," Xilinx white paper: Virtex-7 FPGAs, 2012.

[3] S. K. Goel, "Test challenges in designing complex 3D chips: what is on the horizon for EDA industry," in *Proc. ICCAD*, designer track, 2012.

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

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

[6] Y.-J. Chang *et al.,* "NTHU-Route 2.0: a fast and stable global router," in *Proc. ICCAD,* pp. 338-343, 2008.

[7] W.-H. Liu *et al.*, "NCTU-GR 2.0: multithreaded collision-aware global routing with bounded-length maze routing," *IEEE TCAD*, 32(5), pp. 709-722, 2013.

[8] J. Hu *et al.,* "Completing high-quality global routes," in *Proc. ISPD*, pp. 35-41, 2010.

[9] Y. Xu and C. Chu, "MGR: Multi-level global router," in *Proc. ICCAD*, pp.250-255, 2011.

[10] H. Shojaei *et al.*, "Congestion analysis for global routing via integer programming," in *Proc. ICCAD*, pp. 256-262, 2011.

[11] W.-H. Liu *et al.*, "A fast maze-free routing congestion estimator with hybrid unilateral monotonic routing," in *Proc. ICCAD*, 2012.

[12] K.-R. Dai *et al.*, "NCTU-GR: efficient simulated evolution-based rerouting and congestion-relaxed layer assignment on 3-D global routing," *IEEE TVLSI*, 20(3), pp. 459-472, 2012.

[13] T.-H. Lee and T.-C. Wang, "Congestion-constrained layer assignment for via minimization in global routing," *IEEE TCAD*, 27(9), 1643 – 1656, 2008.

[14] Y. Wei *et al.*, "GLARE: global and local wiring aware routability evaluation," in *Proc. DAC*, pp. 768-773, 2012.

[15] Z. Li *et al.,* "Guiding a physical design closure system to produce easier-to-route designs with more predictable timing," in *Proc. DAC*, pp.465-470, 2012.

[16] F.-Y. Chang *et al.*, "Cut-demand based routing resource allocation and consolidation for routability enhancement," in *Proc. ASPDAC*, pp. 533–538, 2011.

[17] X. He *et al.*, "Ripple: an effective routability-driven placer by iterative cell movement", in *Proc. ICCAD*, pp. 74–79, 2011.

[18] N. Viswanathan *et al.*, "The ISPD-2011 routability-driven placement contest and benchmark suite," in *Proc. ISPD,* pp. 141-146, 2011.

[19] http://cs.nctu.edu.tw/~whliu/NCTU-GR.htm

TABLE 3 EVALUATION BY NCTU-GR 2.0 WITH MANUALLY SETTING LAYER CONFIGURATIONS

	(L_h, L_v)	BD	TOF	TWL	Via	CPU (sec)		(L_h, L_v)	BD	TOF	TWL	Via	CPU (sec)
Sb1	**(3, 2)**	**H**	0	10307170	2762358	554.76	**Sb10**	**(2, 3)**	**V**	0	21232438	4250411	2533.06
	(2, 3)	V	88	10683774	3053022	4494.53		(3, 2)	H	174092	25270391	5550146	56993.40
	(2, 2)	H	235788	12110388	3487853	18767.40		(2, 2)	V	751160	26238917	6393232	76964.40
Sb2	**(3, 3)**	**V**	0	23866253	3702285	300.97	**Sb12**	**(3, 4)**	**V**	0	14265111	6358440	288.35
	(3, 3)	H	0	23866253	3998310	302.48		(4, 3)	H	127438	16037032	7736763	19196.20
	(2, 3)	V	0	25440297	4307114	6160.31		(3, 3)	V	644250	16762196	7720731	25096.30
	(3, 2)	H	0	28081643	4708266	17584.30							
	(2, 2)	V	783388	31189623	6209005	108835.00							
Sb4	**(3, 2)**	**H**	0	7607579	1867402	272.06	**Sb15**	**(3, 3)**	**V**	0	11436621	3958450	105.31
	(2, 3)	V	266	8542881	2204187	2648.39		(3, 3)	H	0	11436621	4171147	105.30
	(2, 2)	H	133736	9276576	2874626	12256.20		(2, 3)	V	391636	13327950	5557724	16025.30
								(3, 2)	H	448410	13841743	5596124	15234.4
Sb5	**(3, 3)**	**V**	0	12273345	2651563	98.01	**Sb18**	**(3, 3)**	**V**	0	8324746	2579202	231.95
	(3, 3)	H	0	12273345	2839029	98.01		(3, 3)	H	0	8324746	2725576	230.02
	(2, 3)	V	4514	13538815	2890550	2483.04		(2, 3)	V	116738	9953287	3596022	14801.8
	(3, 2)	H	72	13047867	2628774	1324.11		(3, 2)	H	395336	9910307	3646525	15615.40