Placement-Driven Partitioning for Congestion Mitigation in Monolithic 3D IC Designs

Shreepad Panth, *Student Member, IEEE*, Kambiz Samadi, *Member, IEEE*, Yang Du, *Member, IEEE*, and Sung Kyu Lim, *Senior Member, IEEE*

***Abstract*—Monolithic 3D (M3D) is an emerging technology that enables integration density which is orders of magnitude higher than that offered by through-silicon-vias. In this paper, we demonstrate that a modified 2D placement technique coupled with a post-placement partitioning step is sufficient to produce high-quality M3D placement solutions. We also present a commercial router-based monolithic intertier via insertion methodology that improves the routability of M3D ICs. We demonstrate that, unlike in 2D ICs, the routing supply and demand in M3D ICs are not completely independent of each other. We develop a routing demand model for M3D ICs, and use it to develop an $O(N)$ min-overflow partitioner that enhances routability by off-loading demand from one tier to another. This technique reduces the routed wirelength and the power delay product by up to 7.44% and 4.31%, respectively. This allows a two-tier M3D IC to achieve, on average, 19.9% and 11.8% improvement in routed wirelength and power delay product over 2D, even with reduced metal layer usage.**

***Index Terms*—Monolithic 3D (M3D), partitioning, placement, routing congestion.**

I. Introduction

THREE dimensional integrated circuits (3D ICs) have emerged as a promising solution to extend the 2D scaling trajectory predicted by Moore's law. Monolithic 3D IC (M3D) is an emerging technology that enables orders of magnitude higher integration density than through-silicon-via (TSV)-based 3D, due to the extremely small size of the monolithic intertier vias (MIVs). In M3D integration technology, two or more tiers of devices are fabricated sequentially, one on top of another. This eliminates the need for any die alignment, which enables much smaller via sizes. Each MIV has essentially the same size as a regular local via (<100 nm diameter) [1].

The first M3D fabrication technique was to grow an amorphous silicon layer over an existing bottom tier and fabricate TFT transistors on the top tier [2]. Next, attempts were made to crystallize the top silicon by using lasers [3]. The highest quality silicon was achieved by transferring an extremely thin layer of silicon onto the bottom tier through an ion-cut process [1].

There are also a few design works on M3D ICs. Transistor-level M3D ICs, where the pMOS and nMOS are split up into separate tiers were discussed in [4] and [5]. However, this design style requires extensive redesign of standard cells and does not give a 50% footprint reduction. Block-level design, where 2D blocks are floorplanned on to a 3D space have also been studied [6], [7]. This design style does not fully utilize the high integration density offered by M3D. In terms of gate-level solutions, where each logic gate is confined to a given tier, prior work is limited.

In this paper, we focus on two-tier gate-level M3D ICs, and assume that each tier supports as many metal layers as required. The contributions of this paper are as follows.

1) *Placement:* We empirically demonstrate that minor modifications to existing 2D placement engines coupled with a post-placement partitioning step is sufficient to produce high quality M3D IC placement solutions. We demonstrate this fact by using both academic and commercial 2D placement engines.
2) *MIV Insertion:* We present a router-based MIV insertion algorithm that outperforms existing placement-based 3D via insertion techniques.
3) *Routability Modeling:* This is the first paper to study routability issues in gate-level M3D ICs. We develop a probabilistic routing demand model for M3D ICs. We demonstrate that, unlike in 2D ICs, the routing supply is not completely independent of routing demand.
4) *Routability-Driven Partitioning:* Based on our routing demand model, we present an $O(N)$ min-overflow partitioner that reduces the total routing overflow by intelligently assigning cells to tiers.
5) *Application to Other Technologies:* We demonstrate that our proposed techniques are not limited to M3D, and can also be easily applied to other 3D integration technologies where the via sizes are small, such as face-to-face (F2F) bonding.

The remainder of this paper is organized as follows. Section II presents the overall design flow proposed in this paper. Section III then presents our proposed placement technique. Section IV presents our routability modeling and min-overflow partitioner. Section V discusses our router-based MIV insertion methodology. Experimental results are presented in Section VI, and we provide comparisons with state-of-the-art in Section VII. Section VIII concludes this paper.

Manuscript received July 18, 2014; revised October 7, 2014; accepted November 24, 2014. Date of publication January 6, 2015; date of current version March 17, 2015. This work was supported by Qualcomm Research. This paper was recommended by Associate Editor A. Davoodi.

S. Panth and S.-K. Lim are with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: shreepad.panth@gatech.edu).

K. Samadi and Y. Du are with Qualcomm Research, San Diego, CA 92121 USA.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2014.2387827

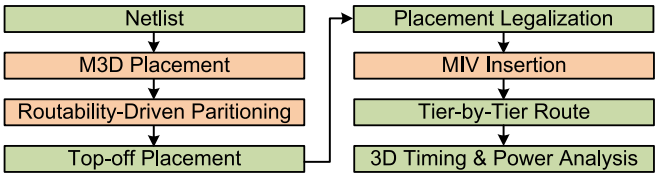

Fig. 1. Design flow used in this paper.

II. Overall Design Flow

A. Problem Formulation

We define the "projected 2D half-perimeter wirelength (HPWL)" as the HPWL of a M3D IC if all the gates are projected onto a single placement layer. We also define the total routing overflow as the sum of routing demand minus routing supply on all global routing edges that are congested.

The problem that we solve can then be stated as: given an initial M3D placement, repartition the gates with minimal change to the projected 2D HPWL, such that the total routing overflow is minimized.

However, this formulation still requires us to have an initial M3D placement. We would, however, like to use existing 2D engines for M3D design. Therefore, we also solve the following problem: generate a 2D design, using minimally modified 2D tools, such that it represents a M3D IC with all the gates projected to a single tier. If we generate such a design, then our partitioning can directly be applied on top of it.

B. Design Flow

In this section, we discuss our proposed design flow, an overview of which is shown in Fig. 1. In this figure, the red boxes indicate steps that will be explained in detail in subsequent sections. We first start with the synthesized netlist and get an initial M3D IC placement result. We then perform routability-driven partitioning, which takes the initial placement solution and repartitions the gates to improve the routed wirelength of the design. A top-off placement step is then performed to make sure that each tier in the M3D IC meets target density requirements. The last step in the placement process is legalization, which snaps the cells to the placement grid. Once the locations of cells are determined, we need to insert MIVs into the whitespace between them. MIVs can then simply be treated as I/Os in each tier, and a tier-by-tier route can be carried out using commercial tools (Cadence Encounter). Finally, we extract parasitics tier-by-tier, create a separate parasitic file to represent MIV parasitics, and feed all this information into Synopsys PrimeTime to obtain 3D timing and power numbers.

III. M3D IC Placement

This section first presents prior work in TSV-based 3D IC placement, and discusses why those approaches are not applicable to M3D ICs. Next, we present our proposed methodology based on modifications to 2D IC tools. Finally, we discuss how to handle preplaced memory macros in a 3D design while still using 2D IC tools.

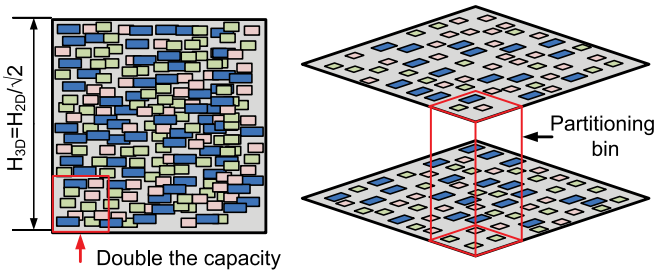

Fig. 2. Placement-aware partitioning. We first use a modified 2D engine to place all the gates into half the area, and then partition it with area balance in each bin.

The M3D gate-level placement problem is similar to the TSV-based problem, except that we do not need to minimize the via count. The first approach to TSV-based 3D placement is folding-based [8]. This takes an existing legal 2D placement, and transforms it to 3D by several folding operations. This approach generates inferior quality solutions [9], and is also not capable of handling preplaced memory. The next method is partitioning-based [10], where the netlist is first partitioned and all tiers are placed simultaneously. Lastly, true 3D placement approaches exist [9], [11], where the HPWL is minimized in the x, y, and z dimensions. However, in M3D ICs, the z-dimension is so small ($1-2\,\mu$m) that attempting to minimize the z HPWL is not really necessary. In addition, all of these engines are geared toward TSV-based 3D, and try to minimize the via count. In this paper, we demonstrate the fact that since monolithic vias are so small, only a minimally modified 2D placement engine suffices, and separate 3D placement engines are not required.

A. Placement-Aware Partitioning

An illustration of our proposed method for a two-tier M3D IC is shown in Fig. 2. If the width and height of a 2D IC are W_{2D} and H_{2D}, respectively, we first define a M3D outline where the width and height are divided by $\sqrt{2}$. This modification leads to exactly half the footprint of a 2D IC. All 2D placement engines have the concept of chip capacity (or target density), which is the maximum number of standard cells that can be placed in a given area. Since we want to fit all the gates into half the area, we simply double the capacity of the chip. Any existing 2D placer can be modified for this purpose, and we choose to implement our own version of KraftWerk2 [12]. Clearly, the HPWL obtained after such a placement represents the HPWL of a M3D IC where all the tiers have been projected onto a single tier—the projected 2D HPWL.

The next step is to partition the gates such that each tier has an equal number of gates, and the deviation from the initial (x, y) location is minimized. An obvious approach to partitioning the gates is a min-cut approach, and modifying the traditional Fiduccia–Mattheyses [13] (FM) min-cut partitioner is straightforward, an overview of which is given below.

First, we define partition bins in a regular fashion. Next, we partition the design so that the cells in a given bin in the modified 2D result remain in the same bin after splitting. As will be

discussed in Section VI-A, the choice of bin size affects solution quality greatly. This is because after partitioning, although each bin in each tier will contain the correct number of cells, these cells may not be distributed uniformly throughout the bin. If the partitioning bin size is much larger than the global placement bin size, we could potentially obtain large areas of extra-dense cell placement and large areas of whitespace. Therefore, top-off placement becomes necessary to obtain an acceptable placement solution that meets target density within each global bin.

Initially, we create a random, area-balanced (within each partition-bin) solution. We define the gain of a cell as the reduction in the cutsize if the cell's tier is changed. A cell is "legal" if moving it does not violate the area-balance constraints within its partition bin. While moving a single cell from one tier to another will not affect the area balance too much, this condition ensures that too many cells are not moved from one tier to another.

Initially, we compute all the cell gains and store them in a bucket structure. We also mark all the cells as "unlocked." Among all legal cells, we pick the one with the highest gain, move it to the other tier, and lock it. Once a cell is moved, only the gains of its neighbors (connected by a net) needs to be updated. This process is continued until all the cells are locked. This is termed a pass. We perform several passes until no more cutsize gains are achieved. Due to the nature of the incremental gain update, this algorithm runs in $O(C)$ time, where C is the number of cells. While the min-cut is straightforward, MIVs are extremely small and there is no real need to perform a min-cut on the netlist. Additional MIVs can be tolerated, if there is good reason to use them. We present a routability-driven partitioner in Section IV that utilizes additional MIVs to reduce routing congestion, and hence, routed wirelength (WL).

Note that while this approach may appear somewhat similar to the local stacking transformation (LST) presented in [8], it is superior in one major aspect—the handling of preplaced memory macros. The LST method obtains the initial (x, y) locations of all the cells by scaling them from a legal 2D placement, and hence has no way to handle preplaced memory macros in a 3D space. Handling them in our method is straightforward, and will be discussed in the following section.

B. Handling Memory Macros

In a M3D design, hard macros such as memory are bound to be preplaced. In this section, we discuss how to handle these memory macros while still leveraging 2D IC tools. Let t_d be the target density required in the final, post-partitioned M3D design, and t'_d be the target density in the modified 2D placement. Consider the preplaced memories in both tiers as shown in Fig. 3(a). First, we project both these tiers onto the same plane as shown in Fig. 3(b). Those regions that have two memories overlapping cannot contain cells in any tier, and hence will have $t'_d = 0$. Those regions that have only one memory can contain cells in the tier where the memory is not placed. To reflect this fact, the target density in those regions will not be doubled, or $t'_d = t_d$, as shown in Fig. 3(c). Finally,

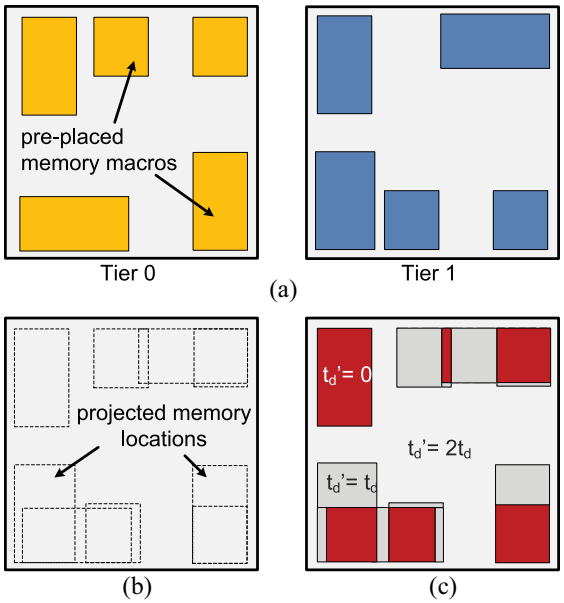

Fig. 3. Handling preplaced memory macros. (a) Initial preplaced locations. (b) Projection of both tiers onto the same plane. (c) Modifying the target density to represent memory locations. t'_d is the target density in the modified 2D placement and t_d is the required target density in the final M3D design.

the regions not containing memory will have cells of both tiers placed, and hence $t'_d = 2t_d$.

Handling these region-specific target density constraints is straightforward in the Kraftwerk placement system. In order to remove overlap between cells, it maintains a supply/demand system of placement space. The chip is divided into fine mesh tiles, and each mesh tile has a supply t_d. Each cell has demand 1 on each mesh tile that it occupies. Solving the Poisson equation of supply minus demand gives the direction and amount to move each cell in order to equalize supply and demand. In this system, we can easily set the supply of each fine mesh tile to be t_d or $2t_d$ depending on our requirement.

The partitioning process can also be modified easily. The regions with memory overlap in both tiers do not have cells, and need not be partitioned. Those cells placed in the regions with a single memory macro are moved to the tier not containing memory. Finally, the regions with cell overlap are partitioned as usual.

IV. Routability-Driven Partitioning

The first step in building a routability-driven partitioner is to estimate the routing congestion in the M3D IC. The routing congestion is measured as the total routing overflow, which is the routing demand minus routing supply on all the global routing edges in the chip. The routing supply is determined from the number and pitch of metal layers, and this section discusses how to determine the 3D routing demand. With this congestion model, we then describe how to repartition the M3D IC to reduce routing congestion.

A. Prior Work

While this is the first work to discuss a M3D routing demand model, this topic has been explored extensively for 2D ICs.

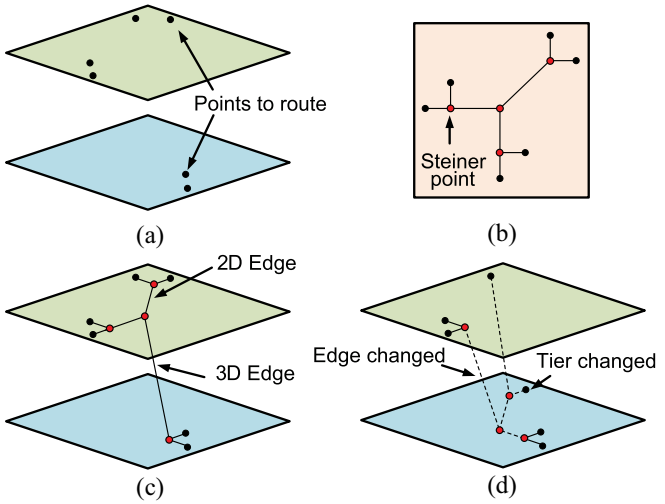

Fig. 4. Construction of a 3D RST. (a) Points to be routed. (b) Project to 2D and construct a 2D rectilinear Steiner minimum tree (RSMT). (c) Expand the 2D RSMT to a 3D RST. (d) If a cell changes tier, the 2D RSMT can be reused.

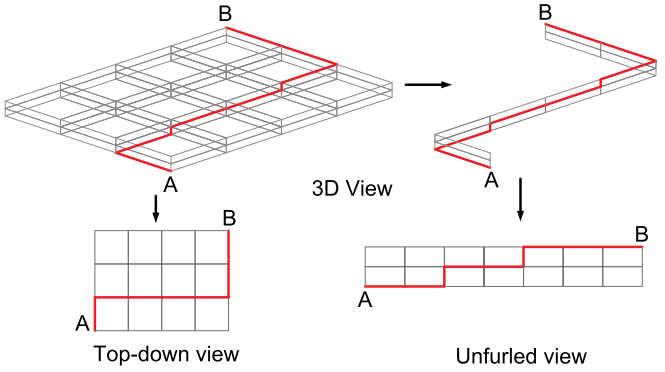

Fig. 5. Legal route from A to B in a $4 \times 3 \times 2$ grid. The top-view is limited to two bends, while the unfurled view can have unlimited bends.

The first approach is a grid-less approach [14], where the demand of a net is assumed to be distributed evenly along all possible Steiner tree combinations. This was extended to consider the differences between horizontal and vertical segments in [15]. These approaches are more suitable for routability-driven placement, not partitioning, as both these papers try to minimize the overlap of the net bounding boxes. The other approach is to first decompose multipin nets into two pin nets, and add each two pin net into the demand estimate. The demand of each two pin net can be estimated either by maze routing [16], rough global (LZ) routing [17], or probabilistically [18]. In this paper, we choose a probabilistic demand model because: 1) it is extremely fast unlike maze routing and 2) the predicted demand numbers are independent of net ordering unlike LZ routing. The first property is necessary as several solutions will be evaluated during partitioning, and the second property is essential for a partitioner as each recompute of the demand of the same two-pin net must yield the same result.

B. Decomposing Multipin Nets Into Two-Pin Nets

In this section, we present our method of decomposing multipin nets into two-pin nets by constructing 3D rectilinear Steiner trees (RSTs). Currently, no tool exists to efficiently compute a 3D RST, so we project the net to 2D, construct a 2D RSMT, and then expand it back to 3D.

Consider the points to be routed as shown in Fig. 4(a). We first project the points to a 2D plane and construct a 2D RSMT using FLUTE [19] [Fig. 4(b)]. Now, while expanding this 2D RSMT to a 3D RST, we already know the tiers of all the fixed points. We then determine the tier of each Steiner point by a majority vote of the tier of all of its neighbors. If any ties are present, we break them in any arbitrary, deterministic manner. A neighbor is defined as any point (Steiner or fixed) that the current Steiner point is connected to. If a neighbor does not have a tier determined yet, it is ignored during the current iteration of the majority vote operation. For example, when we expand the 2D RSMT of Fig. 4(b), the tiers of the three Steiner points that are connected to the fixed points are determined first. They each have two neighbors in one tier, and one undetermined neighbor. Therefore, they all lie in the same tier as the fixed points that they are connected to. Next, the tier of the middle Steiner point can be determined as the top tier as it has two neighbors in the top tier and one in the bottom tier. The resulting 3D RST is shown in Fig. 4(c).

Since we wish to perform move-based partitioning, we need to be able to quickly evaluate the change in the topology if the tier of a particular cell is changed. Since such a change does not change the x- and y-co-ordinate of the cell, the same 2D RSMT can be reused. We change the tier of one cell and show the resulting 3D RST in Fig. 4(d). We redo the expansion from Fig. 4(b), and only the quick majority vote operation needs to be performed on the Steiner points. Note that the Steiner point connected to the cell that has changed tier now has an equal number of neighbors in each tier. This tie can be broken in any deterministic manner, and we choose to always go with the lower tier. Since the middle Steiner point now has two neighbors in the bottom tier and one in the top tier, it is also assigned to the bottom tier.

As seen from this figure, a lot of the routing demand on the top tier is offloaded to the bottom tier, with an unchanged 3D bounding-box. Therefore, to evaluate the change in demand if the tier of a given cell is changed, we need to: 1) redo the majority vote operation for all nets connected to that cell; 2) delete the old topology (rip-up) of the changed two-pin nets from the demand estimate; and 3) add the new topology (reroute) of the changed two-pin nets into the demand estimate. Handling each two-pin net is described next.

C. 3D Demand Model for Two-Pin Nets

We maintain a 3D routing graph for the entire chip. This section assumes that we are looking only at that sub-graph that a given two-pin net spans. Although this paper focuses only on two tier M3D ICs, the model presented in this section is general, and is applicable to any number of tiers.

An example $4 \times 3 \times 2$ three-tier grid is shown in Fig. 5. Assume that the net (A–B) spans a $l \times m \times n$ routing sub-graph.

We wish to compute the probabilistic routing demand contributed by this two-pin net on each edge within this sub-graph. One possible route from A to B is highlighted in red. Many such legal routes exist, and a probabilistic demand model assumes that each legal route is equally probable. Therefore, the key to such a demand model is to correctly identify which routes are legal.

We make two key observations that help us to derive the demand model.

1) If we look at the 3D demand graph from the top-view, each bend represents the usage of a local via. Since current global routers try to minimize the usage of local vias, we are limited to at most two bends (or local vias) in the top view [17], [18].
2) We define a new view called the unfurled view, which unfurls the routing graph along a legal route (refer to Fig. 5).

In such a view, movement along either x or y directions looks the same. In this view, irrespective of the number of bends, the number of MIVs is always the same and equal to exactly $n-1$. For example, in Fig. 5, we always use 2 MIVs to connect A and B, irrespective of the number of bends in the route. Therefore, there are no limits to the number of bends in the unfurled view.

Assuming the above constraints, the total number of routes from A to B is $(l+m) \times^{(l+m+n)} C_n$. First, given the top-view constraint, the sum of all the probabilities along all the edges that look identical in the top-view is given by

$$\sum_{i=1}^{n} P_{(x,x+1),y,i} = \frac{1}{l+m} \times \begin{cases} (l-x), & \text{if } y=0 \\ (x+1), & \text{if } y=m \\ 1, & \text{otherwise.} \end{cases} \quad (1)$$

A similar expression can also be written for all the y edges. Next, in the unfurled view, all edges with the same $(x+y)$ look the same. Therefore, let i represent $(x+y)$. Since there is no limit to the number of bends, the routing probability on any horizontal edge is given by a uniform probability distribution

$$P_{(i,i+1),z} = \frac{{}^{(i+z)}C_i \times^{(l+m+n-i-z-1)} C_{(l+m-i-1)}}{{}^{(l+m+n)}C_n}. \quad (2)$$

Equations (1) and (2) can be combined to give the routing probability on any x-edge in the 3D graph

$$K_{3D} = \frac{{}^{(x+y+z)}C_z \times^{(l+m+n-x-y-z-1)} C_{(l+m-x-y-1)}}{(l+m) \times^{(l+m+n)} C_n}$$

$$P_{(x,x+1),y,z} = K_{3D} \times \begin{cases} (l-x), & \text{if } y=0 \\ (x+1), & \text{if } y=m \\ 1, & \text{otherwise.} \end{cases} \quad (3)$$

A similar expression can also be computed for all the y edges. Once the probabilities of the x and y edges have been computed, the probability on each z edge can be computed by visiting them in turn, and setting the probability to be the sum of the probability on all incoming edges (toward A) minus the sum of the probability on all the outgoing edges (toward B).

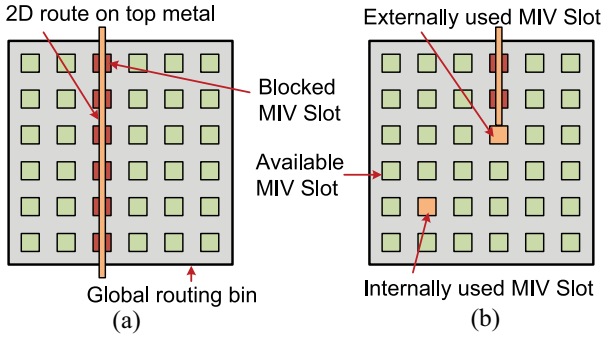

Fig. 6. View of the top metal layer that contains MIV landing pads. (a) 2D wire on the top metal layer blocks potential MIV landing pad slots. (b) If MIVs connect to cells outside the current bin (external), they block other MIVs. If MIVs connect to cells within the current bin (internal), they do not block other potential MIV slots.

D. Interdependent Supply/Demand (IdS) Model

In 2D ICs, there are two types of tracks—x and y. Using an x track does not affect the supply of y tracks, and vice-versa. In M3D ICs, we also need to take into account the number of z tracks available. These z tracks, however, are not independent of the x and y track usage. Assuming that the top metal layer is vertical, we illustrate this fact in Fig. 6. This figure shows the top view of the top metal layer of one global routing bin. The green squares represent potential MIV landing pad sites whose pitch is determined by the pitch of the top metal layer. There are three effects present that we wish to model First, assume that a 2D wire on the top metal layer crosses this bin. As shown in Fig. 6(a), this 2D route blocks potential MIV landing pad sites, and hence reduces the 3D supply. Next, as shown in Fig. 6(b), if a MIV lands on the top metal layer (from the other die), and continues onto a different global routing bin, we term this an externally used MIV slot. Such connections use one MIV slot, but also block others. Finally, if an MIV lands on the top metal layer but connects to a gate within the same bin itself, we term it an internally used MIV. As seen from this figure, it uses one MIV slot but does not block other MIV slots. However, this requires an entire via stack from the top metal to the lowest metal to connect to the cell. This via stack causes via blockages [20], which reduces the 2D supply in the lower metal layers.

Let W_B and H_B be the width and height of the global routing bin. N_{MH} and N_{MV} are the number of horizontal and vertical metal layers, respectively. Let P_{Hi} and P_{Vi} be the pitch of the ith horizontal and vertical metal layer, respectively. Note that we ignore M1 as it is usually used for within-cell routing. Therefore, the "first" metal layer is actually M2. Also, this section assumes that the top metal layer has a preferred vertical direction. The derivation can also easily be carried out if it is horizontal.

If we assume that the top metal pitch is the only factor determining the number of MIV slots, then the number of vertical and horizontal MIV slots are: $N_H = W_B/P_{N_{\text{MV}}}$ and $N_V = H_B/P_{N_{\text{MH}}}$. However, not all these slots are accessible. This is because each metal layer only contributes a finite number of tracks that can connect to MIVs in this bin. The

number of MIV slots can then be given as

$$N_{\text{MIV}} = 2N_H N_{\text{MV}} + 2N_V N_{\text{MH}} - 4N_{\text{MV}} N_{\text{MH}}. \quad (4)$$

This can then be divided into a matrix with N'_H and N'_V effective horizontal and vertical slots. It should be noted that this routing-based constraint on the number of MIVs is far more restrictive than computing the number of MIVs slots available by simply looking at the whitespace available for MIV insertion. It can be shown that even if all the above MIV slots are utilized, it will occupy only 2–3% of the area of a given placement bin.

Next, to determine the number of blocked MIV slots, we first need to determine the number of 2D and 3D routes that use the top metal layer. This requires metal layer assignment, which is a complicated problem. Instead, we make the simplistic assumption that the routes are assigned to metal layers based on the inverse ratio of pitch, i.e., a larger pitch metal will have fewer wires. Let $N_{N,\text{2D},i}$ be the number of 2D routes that cross the north edge on metal layer i. Similar definitions can be made for 3D routes and the east, west, and south edges. Let $N_{N,\text{2D}}$ be the total number of 2D routes crossing the north edge, and P_i be the pitch of the ith metal layer. For each vertical metal layer i, we then get $N_{N,\text{2D},i} = N_{N,\text{2D}}/(P_i . \sum_j (1/P_j))$. We also make the pessimistic assumption that any 2D or 3D wire crossing an edge goes all the way to the center of the bin. The number of blocked MIV slots (assuming the top metal is vertical) can then be given as

$$N_{\text{MIV,Blk}} = 0.5N'_V(N_{N,\text{2D},N_{\text{MV}}} + N_{S,\text{2D},N_{\text{MV}}}) + \left(0.5N'_V - 1\right)\left(N_{N,\text{3D},N_{\text{MV}}} + N_{S,\text{3D},N_{\text{MV}}}\right). \quad (5)$$

The first term in the above equation represents the number of MIV slots blocked by 2D wires and the second-term represents the number of MIV slots blocked by external MIV connections. The actual number of MIV slots can be obtained by subtracting (5) from (4).

The next step is to calculate the 2D supply reduction due to the via blockages introduced by MIV connections. Let $N_{\text{int,3D}}$ be the number of internal MIV connections in this bin. We divide each bin into four quadrants, numbered one through four, in the usual naming convention. The number of vias in the first quadrant, on metal layer i, can then be given as

$$N_{\text{via},1,i} = 0.25N_{\text{int,3D}} + \sum_{j<i} 0.5\left(N_{N,\text{3D},j} + N_{E,\text{3D},j}\right). \quad (6)$$

If $W_{\text{via},i}$ is the width of the via on metal layer i, then the fraction of metal layer i in the first quadrant that is blocked by vias is given as [20]

$$B_{\text{via},1,i} = \sqrt{\frac{N_{\text{via},1,i}(W_{\text{via},i} + 0.5P_i)}{0.25W_B H_B}}. \quad (7)$$

Based on this, the actual 2D supply on the north edge of the routing bin is given as

$$\text{SUP}_N = W_B \sum_{i=1}^{N_{\text{MV}}} (1 - 0.5(B_{\text{via},1,i} + B_{\text{via},2,i}))/P_i. \quad (8)$$

Algorithm 1: Our Min-Overflow Partitioning Heuristic

```
1  Function MinOverflow()
2      demandEstimate →Clear() ;
3      Stage(build) ;
4      Stage(refine) ;
5  Function Stage(type)
6      if (type == build) then
7          ∀n ∈ N : n →valid = false ;
8      Sort N in descending order of bounding-box ;
9      foreach n ∈ N do
10         if (type == build) then
11             demandEstimate →AddRST(n →rst) ;
12             n →valid = true ;
13         FM( n → c_n) ;
```

Similar expressions can then be derived for all the other edges as well.

E. Min-Overflow Partitioning

Given our 3D demand model, we can now perform routability-driven (min-overflow) partitioning. We first perform a min-cut as described in Section III. We then perform min-overflow partitioning on top of this solution. We use total overflow as the metric to be minimized, which is defined as the summation of the overflow on all the 2D and 3D edges in the chip that are congested. The overflow-gain of a cell is then the reduction in the total overflow when its tier is changed, and it is computed by the procedure outlined in Section IV-B.

Let C be the set of all cells and N be the set of all the nets in the design. In the min-cut partitioner, once a cell is moved, only the gains of its neighbors needs to be updated. However, the overflow depends on all nets that use a particular routing edge, not just those connected to this cell. If a cell is moved, it affects several routing edges. Any other net that uses the affected routing edges will now need to have its overflow updated. Since the gain is defined for moving a cell, all cells connected to such nets will also need to have their gain updated. For cells connected to nets with large bounding boxes, up to C cells will need to be updated every time it is moved. This means that maintaining a priority queue with all cells, such as in the default FM algorithm, would lead to a time complexity of $O(C^2)$. This neglects the time necessary to rebuild the queue, which adds a further $O(\log(C))$ complexity. Overall, this would lead to excessively large runtime, making it infeasible. We now present a heuristic that reduces the time complexity significantly.

Our heuristic is shown in Algorithm 1, with the top-level function being MinOverflow(). Initially, the demand estimate is cleared i.e., all nets are removed, and the utilization on each routing edge is set to 0. Next, there are two stages, build and refine, both of which are similar, and handled by the Stage() function. In the build phase, all the nets are initially set to invalid. In both stages, the nets are then sorted by bounding-box. This is because nets with a larger bounding

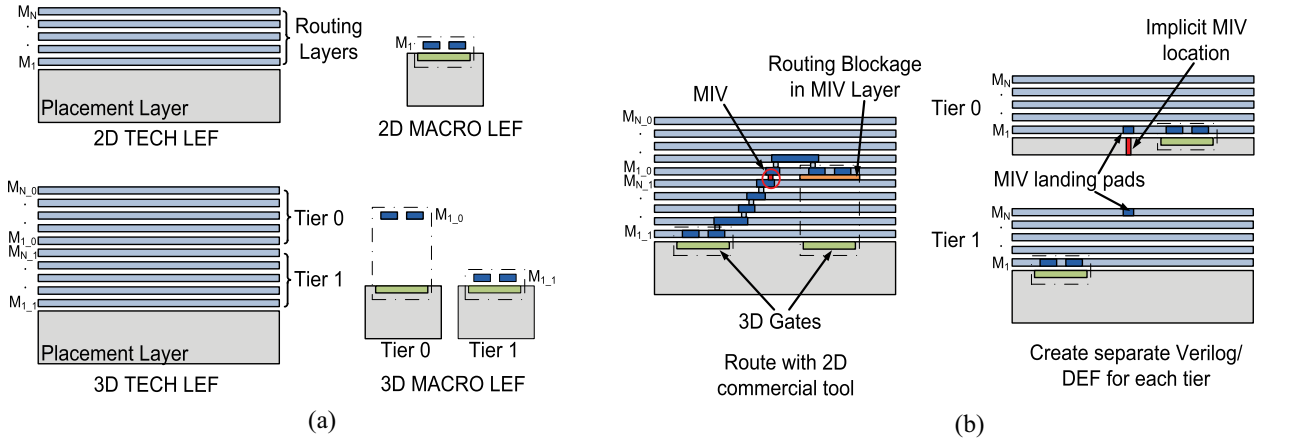

Fig. 7. Overview of our router-based MIV insertion methodology. (a) Technology and macro library exchange format (LEF) are modified to represent a two-tier M3D IC. (b) Structure that is fed into the commercial router, which is then routed. The MIV locations are extracted and separate verilog/design exchange format (DEF) files are created for each tier.

box have a greater impact on the routing graph, and will be processed first. During the build phase, the 3D-RST of the net currently being processed is added into the demand estimate, and the net is set to valid. Next, irrespective of stage, the FM() function (to be described later) is performed on the cells of the current net. Note that in the build phase, the demand estimate does not have all the nets included, only the ones that have been processed so far. This is to avoid any noise introduced by a bad initial random partitioning of the unprocessed nets.

The FM() function is similar to the basic algorithm described in Section III, with a few differences.

1) We use a heap instead of a bucket, as the gains are not integer values.
2) We only look at a subset of cells that belong to a given net.
3) When a cell is moved to another tier, we update the gains of all cells within the current subset.
4) The gain function is the global max-overflow gain, considering all "valid" nets in the design, not just the current net being processed.

The above heuristic adds one net at a time into the demand estimate, maintaining a local optima of the global total overflow after each net is added. Once all the nets are added, we go over each net again to further reduce the overflow. This approach leads to a time complexity of $O(N.(\text{rms}_{N_d})^2)$, where rms_{N_d} is the root-mean-square of the net degrees. This value does not scale much with circuit size, and therefore, our heuristic is more or less linear.

V. Router-Based 3D-Via Insertion

To continue with the P&R flow, we need to perform routing, and then parasitic extraction. However, current routers can only handle 2D ICs, and the usual approach is to split the 3D design into separate designs for each tier, each of which can be routed independently. This requires the locations of the MIVs to be known, so that they can be represented as I/O pins within each tier.

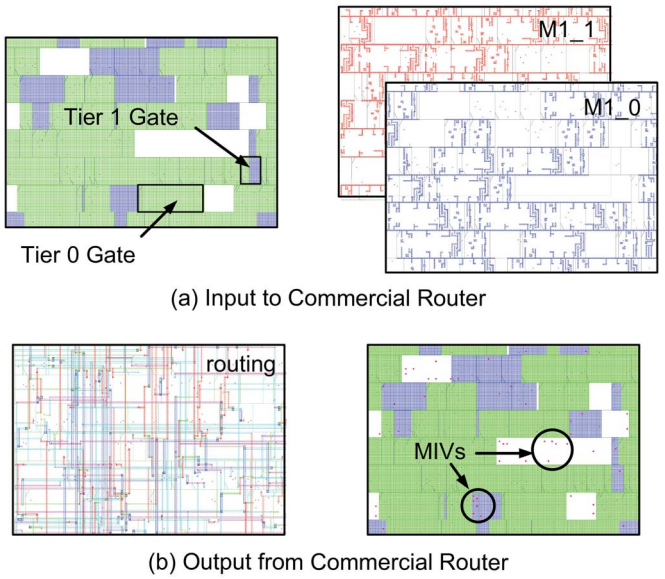

Fig. 8. Screenshots of router-based MIV insertion. (a) All the gates are placed in the same placement layer, but no overlap exists in the routing layers. (b) Result after routing. The MIV locations are highlighted in red.

Once the partition of all cells are finalized, current TSV-based placers perform a TSV and cell co-placement step [9], [10] to determine the via locations. However, MIVs are so small that they can actually be handled by the router, and the only hurdle is the lack of an existing 3D commercial router. In this paper, we present a method to trick existing 2D commercial routers into performing MIV insertion.

An illustration of our approach is shown in Fig. 7. First, all the metal layers in the technology LEF are duplicated to yield a new 3D LEF with twice the number of metal layers. Next, for each standard cell in the LEF file, we define two flavors—one for each tier. The only difference between the two flavors is that their pins are mapped onto different metal layers depending on its tier. Next, each cell in the 3D space is mapped to its appropriate flavor, and forced onto the same placement layer. Note that this will lead to cell overlap in the placement layer, but there will be no overlap in the routing

TABLE I
VARIOUS BENCHMARKS CONSIDERED IN THIS PAPER

Circuit	Clock Period (ns)	#Cells	#Nets	Area (mm^2)		#ML
				Std. Cell	Memory	
mul_64	1.2	21,671	22,399	0.078	0	4
LEON3	0.9	17,419	19,069	0.051	0.034	4
nova	2.3	57,339	60,867	0.179	0.028	6
rca_16	0.4	67,086	75,786	0.263	0	4
aes_128	0.5	133,944	138,861	0.349	0	5
jpeg	1.5	193,988	238,496	0.739	0	4
OS_T2	1.5	316,573	334,374	1.110	0.468	6
fft_256	1.0	488,508	492,499	1.833	0	5

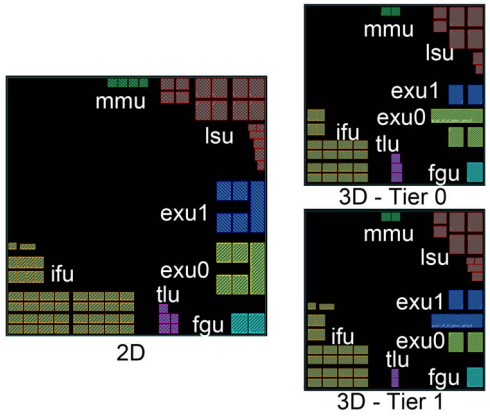

Fig. 9. Manual partitioning of the memories in the OS_T2 benchmark. The memories belonging to each sub-module are partitioned, and placed in a configuration similar to that in 2D.

layers (Fig. 8). We also place routing blockages in the via layer between the two tiers, to prevent MIVs being placed over cells. This entire structure is then fed into an existing commercial router (Cadence Encounter). Once routed, we trace the routing topology to extract the MIV locations, and generate separate verilog/DEF files for each tier.

VI. EXPERIMENTAL RESULTS

We choose eight benchmarks, six of which are from the OpenCores benchmark suite. We also choose two processor designs, the OpenSPARC (OS_T2) and LEON3 cores. These designs vary in size from a few tens of thousands of gates to half a million gates. They are synthesized with a 28 nm cell library, and their statistics are tabulated in Table I. Of these eight designs, three have memory macros, as listed under the memory area column in Table I. In addition to the clock period, number of cells, and number of nets, we also tabulate the minimum number of metal layers with which our 2D placement is routable. This is used as the number of metal layers for both 2D and M3D versions of each design. The footprint area of each design is chosen such that the standard cells have a target density of 70%. All M3D designs are implemented such that they have exactly 0% area overhead compared to its corresponding 2D version, i.e., exactly 50% footprint area, irrespective of MIV count. This condition also ensures that the standard cells in the M3D design have a target density of 70%.

In order to obtain preplaced memory macro locations for 3D, we partition the memory macros architecturally. An example of this for the OS_T2 benchmark is shown in Fig. 9. The 2D design contains several modules such as load-store unit (lsu), instruction-fetch unit (ifu), etc. We allocate roughly half the memories in each module to each tier, and manually place the memories to mimic the 2D placement as close as possible.

We assume the diameter of each MIV to be 100 nm, with a resistance of 2 Ω and a capacitance of 0.1 fF [5]. All routed WL results presented have been both global and detail routed using Cadence Encounter. After routing is complete, we dump parasitics for each tier separately. We then feed the tier netlists and parasitics, as well as the top-level parasitics (for MIVs) into Synopsys PrimeTime, which stitches everything together. We can then obtain the 3D timing and power, from which we obtain power delay product (PDP) numbers. We now conduct several experiments that demonstrate the benefits and scalability of our approach.

A. Impact of Partitioning Bin Size

As discussed in Section III-A, the choice of partition-bin size affects the solution quality greatly. From the perspective of cell displacement, smaller bin sizes are better. However, smaller bin sizes mean more partitioning-bins, which leads to more area-balance constraints the partitioner needs to satisfy. More constraints imply a worse objective function, which means a larger cutsize in the min-cut partitioner. Since we care about routed WL and not just 3D HPWL, more 3D vias mean that we need to find an appropriate whitespace location for more MIVs, which may not always be feasible. Therefore, a smaller bin size may not always lead to lower wirelength. To quantify this effect, we run the min-cut partitioner on all our benchmarks with varying bin sizes, and tabulate results in Table II.

For each benchmark, we evaluate five different bin sizes. We tabulate the MIV count after router-based MIV insertion and the projected 2D HPWL which is the objective function of the top-off placement. As expected, we see that increasing the bin size always reduces the MIV count due to the partitioner having more freedom, and also always increases the projected 2D HPWL as the final (x, y) location of cells deviates more. However, the impact on routed wirelength is mixed, which is due to the trade-off mentioned earlier. We observe that there is a sweet spot in terms of bin size. Increasing the bin size reduces the MIV count, which means that MIV insertion is easier, which reduces the routed wirelength. However, increasing the bin size too much means that the increase in projected 2D HPWL outweighs any benefits obtained from fewer MIVs. This sweet spot is different for different benchmarks, but Table II suggests that a bin size of 10–20 μm works well across a wide range of designs, for this technology. Note that with a different technology, this will bin size will need to change to keep the number of cells per bin a constant. Since sweeping the bin size is not feasible for each new benchmark, we choose a partitioning bin size of 20 μm for all benchmarks, and all subsequent results presented in this paper assume this bin size.

B. Impact of Router-Based MIV Insertion

The conventional method for 3D via insertion is to perform a post-place cell and 3D via co-placement [9], [10].

TABLE II
IMPACT OF PARTITION BIN SIZE ON SOLUTION QUALITY

Bin W (μm)	#MIV ($\times 10^3$)	Proj. 2D HPWL (m)	Routed WL (m)	PDP (mW-ns)
mul_64				
5	15.410 (1.000)	0.315 (1.000)	0.462 (1.000)	35.618 (1.000)
10	8.353 (0.542)	0.315 (1.000)	0.451 (0.975)	34.993 (0.982)
20	5.677 (0.368)	0.321 (1.017)	**0.447 (0.966)**	**34.637 (0.972)**
40	4.731 (0.307)	0.324 (1.027)	0.456 (0.986)	35.226 (0.989)
80	3.503 (0.227)	0.343 (1.088)	0.472 (1.021)	35.342 (0.992)
LEON3				
5	12.508 (1.000)	0.369 (1.000)	0.548 (1.000)	25.927 (1.000)
10	6.793 (0.543)	0.371 (1.005)	0.537 (0.979)	25.600 (0.987)
20	5.772 (0.461)	0.375 (1.016)	**0.529 (0.966)**	**25.510 (0.984)**
40	5.441 (0.435)	0.378 (1.024)	0.533 (0.973)	25.626 (0.988)
80	4.195 (0.335)	0.382 (1.036)	0.534 (0.974)	26.041 (1.004)
nova				
5	44.813 (1.000)	1.276 (1.000)	2.095 (1.000)	68.845 (1.000)
10	25.666 (0.573)	1.278 (1.002)	2.015 (0.962)	68.080 (0.989)
20	22.256 (0.497)	1.291 (1.012)	**1.988 (0.949)**	**68.072 (0.989)**
40	17.074 (0.381)	1.305 (1.023)	1.990 (0.950)	67.389 (0.979)
80	14.348 (0.320)	1.353 (1.060)	1.997 (0.953)	68.443 (0.994)
rca_16				
5	53.380 (1.000)	0.797 (1.000)	1.526 (1.000)	23.915 (1.000)
10	31.835 (0.596)	0.823 (1.033)	**1.503 (0.985)**	**23.762 (0.994)**
20	19.344 (0.362)	0.862 (1.081)	1.531 (1.003)	24.075 (1.007)
40	14.168 (0.265)	0.901 (1.131)	1.546 (1.013)	24.565 (1.027)
80	11.254 (0.211)	0.931 (1.168)	1.568 (1.027)	24.757 (1.035)

Bin W (μm)	#MIV ($\times 10^3$)	Proj. 2D HPWL (m)	Routed WL (m)	PDP (mW-ns)
aes_128				
5	95.439 (1.000)	1.942 (1.000)	3.009 (1.000)	105.165 (1.000)
10	63.759 (0.668)	1.975 (1.017)	**2.953 (0.981)**	**105.053 (0.999)**
20	56.632 (0.593)	2.027 (1.044)	2.991 (0.994)	105.377 (1.002)
40	35.960 (0.377)	2.279 (1.174)	3.197 (1.062)	107.042 (1.018)
80	16.767 (0.176)	2.439 (1.256)	3.341 (1.110)	108.480 (1.032)
jpeg				
5	161.065 (1.000)	3.792 (1.000)	5.406 (1.000)	359.209 (1.000)
10	88.844 (0.552)	3.785 (0.998)	5.326 (0.985)	352.726 (0.982)
20	56.791 (0.353)	3.835 (1.011)	**5.279 (0.977)**	**350.511 (0.976)**
40	47.293 (0.294)	3.904 (1.029)	5.303 (0.981)	351.062 (0.977)
80	35.472 (0.220)	4.143 (1.093)	5.480 (1.014)	355.505 (0.990)
OS_T2				
5	270.770 (1.000)	11.440 (1.000)	-	-
10	149.364 (0.552)	11.626 (1.016)	17.410 (1.000)	520.200 (1.000)
20	129.308 (0.478)	11.645 (1.018)	**17.368 (0.998)**	**517.500 (0.995)**
40	108.171 (0.399)	11.724 (1.025)	17.401 (0.999)	518.100 (0.996)
80	102.429 (0.378)	11.790 (1.031)	17.446 (1.002)	519.900 (0.999)
fft_256				
5	368.223 (1.000)	14.107 (1.000)	-	-
10	227.620 (0.618)	14.117 (1.001)	24.765 (1.000)	775.320 (1.000)
20	164.784 (0.448)	14.349 (1.017)	24.711 (0.998)	767.550 (0.990)
40	145.877 (0.396)	14.486 (1.027)	24.584 (0.993)	755.238 (0.974)
80	130.147 (0.353)	14.494 (1.027)	**24.170 (0.976)**	**752.006 (0.970)**

TABLE III
IMPACT OF ROUTER-BASED MIV INSERTION. ENTRIES MARKED WITH A * ARE UNROUTABLE

Circuit	Placement-driven		Router-driven	
	WL (m)	#MIV ($\times 10^3$)	WL (m)	#MIV ($\times 10^3$)
mul_64	0.530	3.723	0.473	5.677
LEON3	0.628	3.907	0.549	5.772
nova	2.170	13.687	2.031	22.256
rca_16	1.575	11.749	1.535	19.344
aes_128	3.213	35.026	2.988	56.632
jpeg	6.233	24.010	5.304	56.791
OS_T2	21.740*	73.805*	17.469	129.308
fft_256	31.829*	71.272*	25.133	164.784
Geo-Mean	3.348	17.859	2.943	31.489
Norm.	1.000	1.000	0.879	1.763

In this section, we compare our router-based MIV insertion scheme against this approach. For reasons that will be given in Section VI-D, we assume that M3D has one metal layer removed from the top tier.

We perform both placement-driven MIV insertion, as well as our proposed router-driven MIV insertion and tabulate the statistics in Table III. In this table, entries marked with *a** indicate that particular flavor is unroutable, and the wirelength reported is on designs with many thousands of design rule check (DRC) violations. Since reliable parasitic extraction cannot be performed on such designs, we only compare wirelength and MIV count. As observed from this table, the placement-driven MIV insertion often produces results that are unroutable. In those cases that are routable, our router-based MIV insertion improves the routed WL by up to 15%. This is because the placement-based method tends to cluster vias together, leading to large clumps of vias, and large areas without any vias. When routing the placement-based method with the commercial router, we observe no significant congestion during the trial route or global route phase. However, the vias are so small that it becomes difficult to route to them causing huge issues during detailed routing. The router-based method, although it has more MIVs (due to multiple vias inserted per net), spreads them out over the area of the chip, increasing the routability.

C. Impact of Routability-Driven Partitioning

We start with the min-cut solution, and perform routability-driven partitioning with and without the IdS proposed in Section IV-D. We also assume that one metal layer is reduced from the top tier in M3D. We plot the supply, demand, and overflow of the min-cut partition of mul_64 with and without IdS in Fig. 10. From this figure, we see that in the case of IdS, the supply of the MIV layer is reduced due to the demand in the tier 1 top metal, and vice-versa. Clearly, not considering IdS during min-overflow partitioning significantly overestimates the MIV supply. Our results are tabulated in Table IV.

When compared with the min-cut solution, we see that the min-overflow partitioner without IdS can reduce the routed WL by up to 4.30% (mul_64) and the PDP by up to 3.14% (fft_256). On average, the min-overflow partitioner without IdS gives 1.8% and 0.9% better wirelength and PDP respectively. If, however, we consider IdS during partitioning, we get up to a further 3.8% and 2.65% boost in the WL and PDP, respectively. In this case, we can improve the min-cut solution by up to 7.44% with respect to WL and 4.31% with respect to PDP. This takes the average WL and PDP gain over min-cut to 3.4% and 2.2%, respectively. We also observe that the min-overflow solution without IdS underestimates the congestion in the MIV layer, and, on average uses 25.2% more MIVs than the min-cut solution. If IdS is considered

TABLE IV
IMPACT OF ROUTABILITY-DRIVEN PARTITIONING

Circuit	Min-cut			Min-overflow (w/o IdS)			Min-overflow (with IdS)		
	WL (m)	PDP (mW-ns)	#MIV ($\times 10^3$)	WL (m)	PDP (mW-ns)	#MIV ($\times 10^3$)	WL (m)	PDP (mW-ns)	#MIV ($\times 10^3$)
mul_64	0.473	35.520	5.677	0.452	34.913	7.248	0.452	33.119	6.327
LEON3	0.549	25.955	5.772	0.553	26.260	6.691	0.537	25.863	5.846
nova	2.031	69.825	22.256	1.983	68.152	27.517	1.982	67.947	25.055
rca_16	1.535	23.754	19.344	1.507	23.587	25.829	1.491	23.443	23.140
aes_128	2.988	105.473	56.632	2.987	104.058	63.738	2.961	103.978	61.950
jpeg	5.304	351.935	56.791	5.276	357.750	72.432	5.183	349.531	63.109
OS_T2	17.469	522.300	129.308	17.164	517.950	164.863	16.615	509.550	134.988
fft_256	25.133	791.648	164.784	24.184	766.720	222.051	23.263	758.793	180.665
Geo-Mean	2.943	111.253	31.489	2.891	110.224	39.415	2.842	108.476	34.580
Norm.	1.000	1.000	1.000	0.982	0.991	1.252	0.966	0.975	1.098

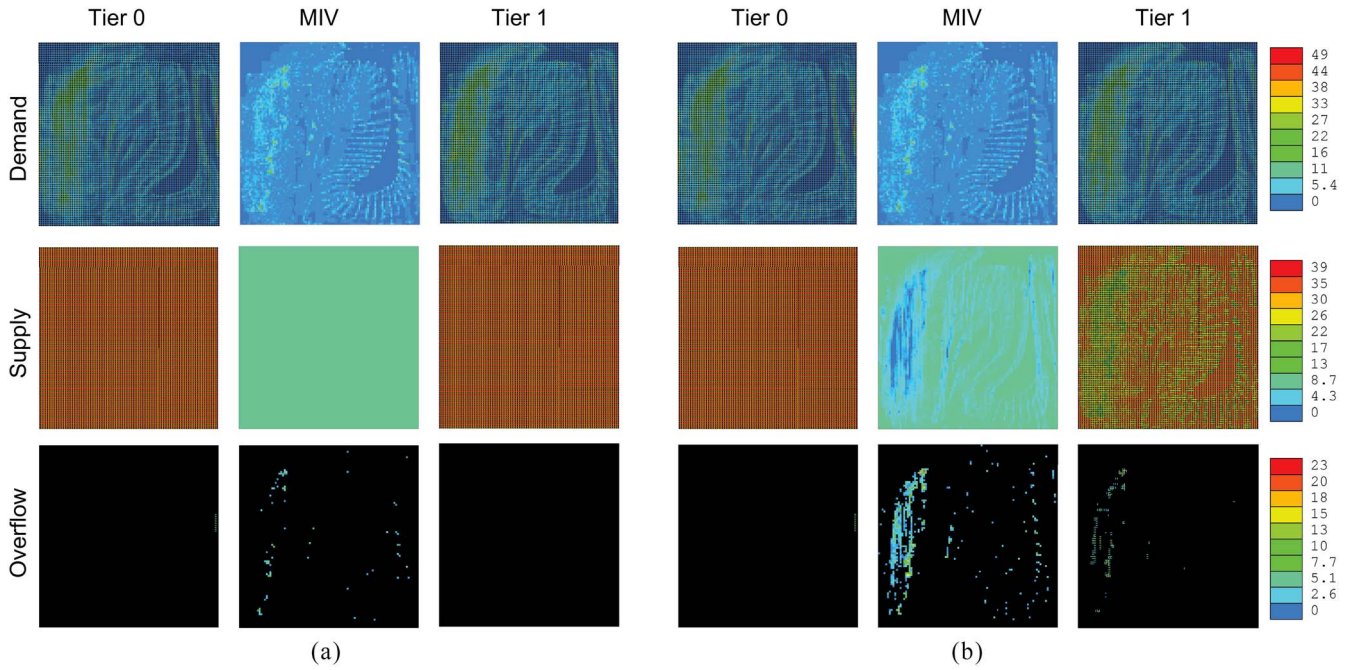

Fig. 10. Supply, demand, and overflow maps of the mul_64 benchmark for min-cut-based partitioning solution. If we consider IdS, we notice a significant reduction in supply in densely wired areas, leading to more overflow. (a) Without IdS. (b) With IdS.

during partitioning, the MIV count increase over min-cut goes down to 9.8%.

D. Reducing Metal Layers in M3D

Cost is one of the primary concerns that needs to be addressed before 3D ICs can be widely adopted. If each tier in a M3D IC uses the same number of metal layers as 2D, the additional cost over 2D is the bonding of the empty silicon wafer. One method to offset the increased cost is to reduce the number of metal layers in 3D, reducing the total cost of the chip.

We now explore reducing the number metal layers in M3D. The default case is when both tiers have the same number of metal layers as 2D (Table I). We term reducing one metal layer from the top-tier alone as Tm1, and reducing one metal layer from each of the top and bottom tiers as Tm1_Bm1. For each of these cases, we perform min-cut partitioning, as well as min-overflow partitioning, with and without IdS. We plot the wirelength and PDP for all these cases in Fig. 11. We also plot the curves for 2D as comparison.

The first thing we observe is that even with a reduced metal count, all designs in M3D are able to be routed with zero DRC violations. These designs were not routable with fewer metal layers in 2D, so the fact that they are now routable indicates that M3D reduces the routing demand significantly. The next thing to note is that, as expected, reducing the metal layer count increases the wirelength and PDP. The magnitude of this increase depends on how congested the initial design is to begin with. We also note that our min-overflow partitioner helps both wirelength and PDP significantly. In many cases, the Tm1 min-overflow (without IdS) result is better than the min-cut with all metal layers. Similarly, the addition of IdS into the partitioner gives a huge WL and PDP benefit. In several cases, we can now reduce two metal layers and still have lower WL than the min-cut case with all metal layers.

E. Application to F2F Bonding

So far, we have discussed the application of our approach to M3D ICs only. However, our approach is general and is applicable to any 3D technology where the via size is so small

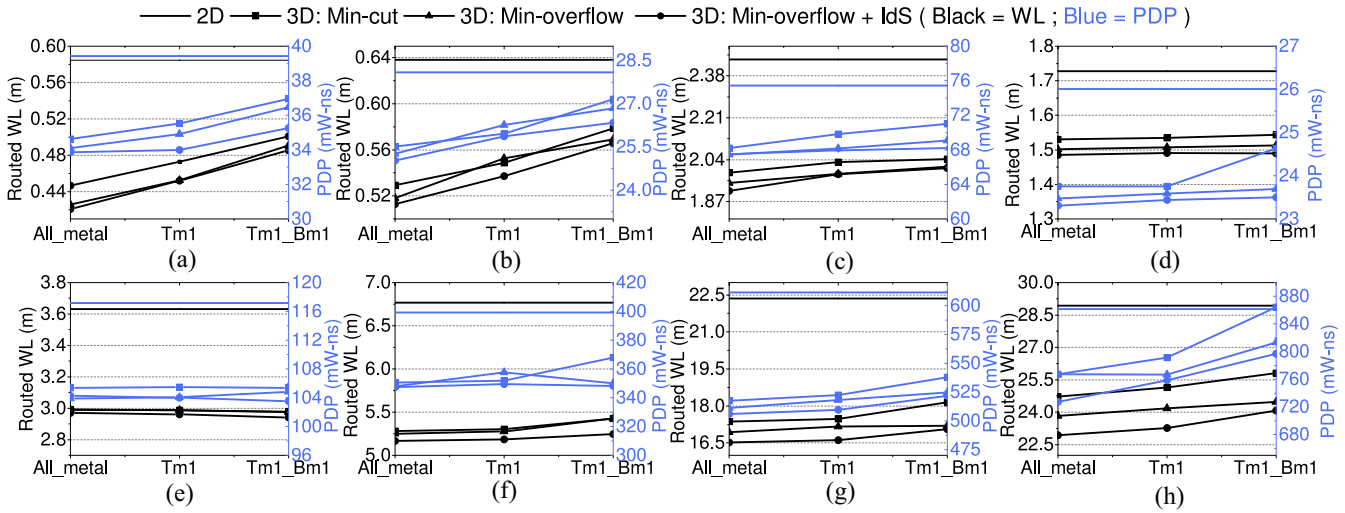

Fig. 11. Impact of reducing the metal layer count. "Tm1" ("Bm1") stands for one metal layer removed from the top (bottom) tier. (a) fp_mul. (b) LEON3. (c) nova. (d) rca_16. (e) aes_128. (f) jpeg. (g) OS_T2. (h) fft_256.

TABLE V
IMPACT OF ROUTABILITY-DRIVEN PARTITIONING FOR F2F DESIGNS

Circuit	Min-cut			Min-overflow (w/o IdS)			Min-overflow (with IdS)		
	WL (m)	PDP (mW-ns)	#F2F ($\times 10^3$)	WL (m)	PDP (mW-ns)	#F2F ($\times 10^3$)	WL (m)	PDP (mW-ns)	#F2F ($\times 10^3$)
mul_64	0.492	35.878	5.294	0.473	35.674	6.948	0.468	35.454	6.444
LEON3	0.597	27.292	5.321	0.587	27.927	6.367	0.582	27.060	5.904
nova	2.091	73.574	20.217	2.037	73.893	25.584	2.028	71.308	23.826
rca_16	1.535	23.754	19.344	1.501	23.680	24.360	1.474	23.384	21.974
aes_128	3.019	111.720	52.812	3.053	112.382	60.330	2.979	108.365	62.202
jpeg	5.377	351.637	52.582	5.307	348.762	68.124	5.193	344.972	60.002
OS_T2	17.783	533.250	115.735	17.484	530.550	153.964	17.024	521.850	130.314
fft_256	25.319	762.764	145.048	24.356	757.786	204.967	23.436	735.297	168.829
Geo-Mean	3.019	113.398	29.095	2.957	113.464	37.083	2.901	110.941	33.629
Norm.	1.000	1.000	1.000	0.980	1.001	1.275	0.961	0.978	1.156

that the placement need not be aware of them. We now discuss how our methodology is applicable to F2F technology that has a different stack-up than the face-to-back style discussed so far.

The placement engine itself need not change. This is because we can place a design as if it was face-to-back, and then mirror the mask of the top tier with the center of the die as the axis of symmetry. We now discuss how to modify our min-overflow algorithm and router-based via insertion steps.

For the min-overflow partitioner, F2F without IdS is identical to the M3D partitioner without IdS. With IdS, only a few changes need to be made. First, the supply in the F2F layer depends on the top metal layer of both tiers, not just the top tier. Therefore, (5) is computed for both tiers separately, and the number of F2F via blockages is the maximum of the two. Next, to calculate the 2D supply reduction, (8) is applied to each tier independently.

For router-based F2F insertion, consider the modified technology LEF file as shown in Fig. 7(a). To represent F2F, all we need to do is to reverse the order of the metal layers of the top die. The stack-up will now be $M_{1\_1}, \ldots, M_{N\_1}$, $M_{N\_0}, \ldots, M_{1\_0}$. Note that, we do not make any additional modifications to the macro LEF file. We also do not have to place any routing blockages over cells, as F2F vias do not occupy silicon space. Finally, while tracing the routing topology, we create the F2F landing pads on the top metal layers of each tier. Each tier can then be routed independently, and the mask of the top tier will be mirrored before fabrication.

We assume F2F vias have a width of 0.5 μm, a resistance of 0.1 Ω, and a capacitance of 0.2 fF. We assume the Tm1 case discussed so far, and tabulate the routed WL and PDP for the min-cut and min-overflow (with and without IdS) in Table V. In this case, we observe that although the min-overflow partitioner without IdS gives an average WL reduction of 2%, the PDP actually goes up slightly. This is due to overestimation of the available F2F supply, and the more accurate partitioner with IdS corrects this issue. We now observe an average reduction of 3.9% and 2.2% in the WL and PDP, respectively.

F. Overall Comparisons

We now compare the WL and PDP numbers of 2D, and the M3D and F2F designs obtained after partitioning with IdS. The results are tabulated in Table VI. From this table, we observe that M3D offers up to a 25.6% WL benefit and 16.6% PDP benefit. On average, M3D offers 19.9% and 11.8% WL and PDP benefit, respectively. In contrast, F2F offers up to 23.8% WL benefit and 14.6% PDP benefit. On average, we obtain 18.2% and 10.1% WL and PDP benefit, respectively.

TABLE VI
OVERALL COMPARISONS

Circuit	2D		3D – MIV		3D – F2F	
	WL (m)	PDP (mW-ns)	WL (m)	PDP (mW-ns)	WL (m)	PDP (mW-ns)
mul_64	0.584	39.432	0.452	33.119	0.468	35.454
LEON3	0.638	28.088	0.537	25.863	0.582	27.060
nova	2.447	75.420	1.982	67.947	2.028	71.308
rca_16	1.727	26.010	1.491	23.443	1.474	23.384
aes_128	3.632	117.148	2.961	103.978	2.979	108.365
jpeg	6.769	399.339	5.183	349.531	5.193	344.972
OS_T2	22.352	611.400	16.615	509.550	17.024	521.850
fft_256	28.922	861.750	23.263	758.793	23.436	735.297
Geo-Mean	3.547	123.338	2.842	108.476	2.901	110.941
Norm.	1.000	1.000	0.801	0.880	0.818	0.899

From this table, we observe that in general, F2F has slightly worse numbers than M3D. This is because of the larger vias sizes (necessitated by die-alignment) and the fact that connecting two gates in 3D requires a stacked via through both tiers. F2F also has other issues not considered here, such as the requirement of being in a regular array, TSV required for I/O connections to the chip, and the nonavailability of flip-chip style packaging.

G. Application to Commercial 2D Engines

Our approach is general, and can be applied to any 2D placement engine easily. In this section, we demonstrate how a commercial engine, Cadence Encounter, can be used to obtain M3D results. As described in Section III, we need to provide region specific target densities of t_d and $2t_d$. However, the commercial tool cannot handle target densities greater than one. Instead, we first halve the area of all the cells in the library, which will place twice the number of cells in a given area as before. Therefore, the requirements of having target densities of t_d and $2t_d$ now directly correspond to regions with target density $0.5t_d$ and t_d. Regions with target density t_d is straightforward, and a target density of $0.5t_d$ can easily be achieved by the use of partial placement blockages, which are supported by any commercial tool. Once the placement is complete, we can blow the cells back up to their original areas, and use our partitioner.

Using this technique to get a modified 2D placement, we apply our min-cut and min-overflow partitioners and tabulate the results for the Tm1 case in Table VII. Since academic tool licenses explicitly prohibit benchmarking against commercial tool results, we normalize all numbers to the 2D results obtained from Cadence Encounter. From this table, we observe similar trends for all the partitioners, and the improvement of 3D versus 2D. We observe that the min-overflow partitioner gives better results than the min-cut partitioner, and adding IdS improves these results further, while reducing the MIV count. Overall, we observe that M3D gives a 17.4% and 10.2% improvement in WL and PDP over 2D, respectively.

VII. COMPARISON WITH EXISTING 3D PLACERS

We compare our placer against two existing techniques, which were primarily developed for TSV-based 3D placement. We first compare our paper with 3D-craft [9] which performs true 3D placement, and next we compare with the partition-then-place approach [10]. We do not compare against another TSV-specific 3D placer [11], because the binary is not publicly available, and hence, we cannot run it ourselves. In addition, Hsu *et al*. [11] only presents absolute 3D WL numbers without providing any 2D baseline number. It is therefore unclear how much of the improvement comes from their 2D engine, and how much from their 3D specific approach. Since our 3D approach can easily incorporate any 2D engine, any engine specific gains in [11] will also carry over to our approach.

A. Comparison With 3D-Craft [9]

Only the binary version of this tool is available, and it does not support a target density driven mode. The cells are preset to always be placed with a target density of 1, or without any whitespace in between them. Such a placement solution will not have any space for router-driven MIV insertion, and hence is inherently not routable. For this reason, we only compare the 3D HPWL in this section. In addition, the binary provided is not capable of handling preplaced hard macros such as memory. Therefore, in this subsection, we only compare the pure-logic designs.

We first run both our placer and 3D-craft with the number of dies set to one to give us a 2D placement. Next, we run both placers with the number of dies set to two, which gives us a 3D placement. We compare only the improvement in HPWL when going to 3D. Our placer is run with a target density of 1 to match the preset setting of 3D-craft. 3D-craft also has a via weight parameter in the cost function (as it is TSV-based), which controls the number of 3D vias. We set this to 0 to make the cost function purely 3D HPWL driven. We tabulate the results of both our placer and 3D-craft in Table VIII.

From this table, we observe that both placement approaches produce comparable wirelength improvements when going to 3D. Since we undertake some steps to minimize the MIV count such as min-cut partitioning, we do not compare the MIV count with 3D-Craft. The benefit of our placer comes not just from comparable improvements in HPWL, but in the fact that any 2D placer can be easily modified and coupled with our partitioner to give high quality results. This was demonstrated in Section VI-G, where we applied our partitioning engine to 2D placement results from a commercial tool.

B. Comparison With Partition-Then-Place [10]

This technique of 3D placement first performs partitioning, and then simultaneous 2D placement of all the tiers while minimizing 3D HPWL. During placement, it looks at all gates in the 3D space, but does not move gates between tiers. Therefore, the initial partition solution is very important, as it greatly affects solution quality. We use our own engine to perform both types of placement, so they have identical 2D numbers. In addition, the utilization of each circuit is set to 70%. Both placement solutions are taken through router-based MIV insertion to get an accurate comparison of routed WL. To generate initial partitions for the partition-then-place approach, we modify an existing multilevel partitioner [21] to give us any target cutsize between min-cut and max-cut.

TABLE VII
IMPACT OF ROUTABILITY-DRIVEN PARTITIONING WITH CADENCE ENCOUNTER AS THE 2D ENGINE

Circuit	2D		3D - Min-cut			3D Min-overflow (w/o IdS)			3D Min-overflow (with IdS)		
	WL	PDP	WL	PDP	#MIV	WL	PDP	#MIV	WL	PDP	#MIV
mul_64	1.000	1.000	0.856	0.931	1.000	0.811	0.908	1.199	0.811	0.901	1.161
LEON3	1.000	1.000	0.894	0.950	1.000	0.870	0.938	1.156	0.867	0.927	1.009
nova	1.000	1.000	0.818	0.884	1.000	0.797	0.873	1.227	0.788	0.865	1.127
rca_16	1.000	1.000	0.894	0.896	1.000	0.875	0.891	1.406	0.861	0.875	1.200
aes_128	1.000	1.000	0.850	0.940	1.000	0.844	0.934	1.081	0.843	0.930	1.028
jpeg	1.000	1.000	0.864	0.921	1.000	0.853	0.920	1.292	0.837	0.914	1.126
OS_T2	1.000	1.000	0.813	0.897	1.000	0.802	0.897	1.261	0.777	0.879	1.050
fft_256	1.000	1.000	0.885	0.951	1.000	0.865	0.897	1.351	0.828	0.897	1.114
Geo-Mean	1.000	1.000	0.858	0.921	1.000	0.839	0.907	1.243	0.826	0.898	1.100

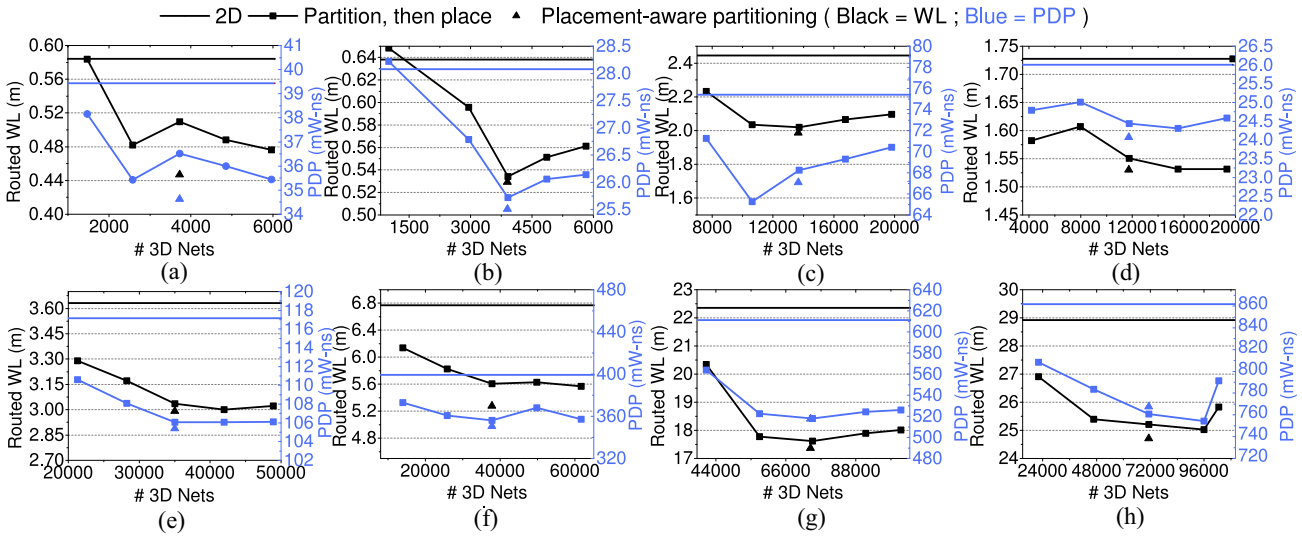

Fig. 12. Comparison of 2D, partition-then-place, and placement-aware partitioning methods. (a) fp_mul. (b) LEON3. (c) nova. (d) rca_16. (e) aes_128. (f) jpeg. (g) OS_T2. (h) fft_256.

TABLE VIII
COMPARISON BETWEEN 3D-CRAFT AND OUR PLACER

Circuit	Our HPWL (m)			3D-Craft HPWL (m)		
	2D	3D	3D/2D	2D	3D	3D/2D
mul_64	0.39	0.30	0.77	0.34	0.27	0.79
rca_16	1.15	0.92	0.79	1.22	0.97	0.80
aes_128	2.61	1.93	0.74	2.52	1.87	0.74
jpeg	4.96	3.70	0.74	5.09	3.78	0.74
fft_256	18.95	13.63	0.72	19.57	13.31	0.68
Geo-Mean	2.56	1.93	0.75	2.54	1.90	0.75
Norm.	1.00	1.00	1.00	0.99	0.99	1.00

First, we run the placement-aware partitioning approach and compute the number of 3D nets it uses. Next, we generate partitions starting from this cutsize, in increments of $\pm 5\%$ of the number of nets. The wirelength and PDP for all approaches are plotted in Fig. 12. From these graphs, we observe that choosing an appropriate cutsize is very important to the solution quality. We also observe that our approach gives the best wirelength, without the need to sweep the cutsize. Since the placer is not timing-driven, we expect, but cannot guarantee that the PDP is the best, and we observe that it is the minimum achieved in most cases, or very close.

VIII. CONCLUSION

In this paper, we have demonstrated that modified 2D placement coupled with a placement-aware partitioning step is sufficient to produce high quality M3D IC placement results. We demonstrate that any 2D placement engine, including commercial ones, can be used with our technique. We have presented a router-based MIV insertion algorithm that makes previously unroutable designs routable. We demonstrate that unlike in 2D ICs, in M3D ICs the supply and demand are not independent. We have developed a M3D demand model, and used it to build a fast $O(N)$ min-overflow partitioning heuristic. We also demonstrate that our approach is equally valid for other fine grained 3D technologies.

REFERENCES

[1] P. Batude *et al.*, "Advances in 3D CMOS sequential integration," in *Proc. IEEE Int. Electron Devices Meeting*, Baltimore, MD, USA, Dec. 2009, pp. 1–4.

[2] S.-M. Jung *et al.*, "The revolutionary and truly 3-dimensional 25F2 SRAM technology with the smallest S3 (stacked single-crystal Si) cell, 0.16um2, and SSTFT (atacked single-crystal thin film transistor) for ultra high density SRAM," in *Proc. IEEE Int. Symp. VLSI Technol.*, Honolulu, HI, USA, Jun. 2004, pp. 228–229.

[3] S.-M. Jung, H. Lim, K. Kwak, and K. Kim, "A 500-MHz DDR high-performance 72-Mb 3D SRAM fabricated with laser-induced epitaxial c-Si growth technology for a stand-alone and embedded memory application," *IEEE Trans. Electron Devices*, vol. 57, no. 2, pp. 474–481, Feb. 2010.

[4] S. Bobba *et al.*, "CELONCEL: Effective design technique for 3D monolithic integration targeting high performance integrated circuits," in *Proc. Asia South Pac. Design Autom. Conf.*, Yokohama, Japan, Jan. 2011, pp. 336–343.

[5] Y.-J. Lee, D. Limbrick, and S. K. Lim, "Power benefit study for ultra-high density transistor-level monolithic 3D ICs," in *Proc. ACM Design Autom. Conf.*, Austin, TX, USA, May 2013, pp. 1–10.
[6] S. Panth, K. Samadi, Y. Du, and S. K. Lim, "High-density integration of functional modules using monolithic 3D-IC technology," in *Proc. Asia South Pac. Design Autom. Conf.*, Yokohama, Japan, Jan. 2013, pp. 681–686.
[7] S. Panth, K. Samadi, Y. Du, and S. K. Lim, "Power-performance study of block-level monolithic 3D-ICs considering inter-tier performance variations," in *Proc. ACM Design Autom. Conf.*, San Francisco, CA, USA, 2014, pp. 62:1–62:6.
[8] J. Cong, G. Luo, J. Wei, and Y. Zhang, "Thermal-aware 3D IC placement via transformation," in *Proc. Asia South Pac. Design Autom. Conf.*, Yokohama, Japan, Jan. 2007, pp. 780–785.
[9] J. Cong and G. Luo, "A multilevel analytical placement for 3D ICs," in *Proc. Asia South Pac. Design Autom. Conf.*, Yokohama, Japan, Jan. 2009, pp. 361–366.
[10] D. Kim, K. Athikulwongse, and S. Lim, "A study of through-silicon-via impact on the 3D stacked IC layout," in *Proc. IEEE Int. Conf. Comput.-Aided Design*, San Jose, CA, USA, Nov. 2009, pp. 674–680.
[11] M. K. Hsu, Y. W. Chang, and V. Balabanov, "TSV-aware analytical placement for 3D IC designs," in *Proc. ACM Design Autom. Conf.*, New York, NY, USA, Jun. 2011, pp. 664–669.
[12] P. Spindler, U. Schlichtmann, and F. Johannes, "Kraftwerk2: A fast force-directed quadratic placement approach using an accurate net model," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 8, pp. 1398–1411, Aug. 2008.
[13] C. M. Fiduccia and R. M. Mattheyses, "A linear-time heuristic for improving network partitions," in *Proc. ACM Design Autom. Conf.*, Las Vegas, NV, USA, Jun. 1982, pp. 175–181.
[14] P. Spindler and F. Johannes, "Fast and accurate routing demand estimation for efficient routability-driven placement," in *Proc. Design Autom. Test Europe*, Nice, France, Apr. 2007, pp. 1–6.
[15] Z.-W. Jiang, B.-Y. Su, and Y.-W. Chang, "Routability-driven analytical placement by net overlapping removal for large-scale mixed-size designs," in *Proc. ACM Design Autom. Conf.*, Anaheim, CA, USA, Jun. 2008, pp. 167–172.
[16] M.-C. Kim, J. Hu, D.-J. Lee, and I. Markov, "A SimPLR method for routability-driven placement," in *Proc. IEEE Int. Conf. Comput.-Aided Design*, San Jose, CA, USA, Nov. 2011, pp. 67–73.
[17] C. Li, M. Xie, C.-K. Koh, J. Cong, and P. Madden, "Routability-driven placement and white space allocation," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 26, no. 5, pp. 858–871, May 2007.
[18] U. Brenner and A. Rohe, "An effective congestion-driven placement framework," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 22, no. 4, pp. 387–394, Apr. 2003.
[19] C. Chu and Y.-C. Wong, "FLUTE: Fast lookup table based rectilinear Steiner minimal tree algorithm for VLSI design," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 1, pp. 70–83, Jan. 2008.
[20] Q. Chen, J. Davis, P. Zarkesh-Ha, and J. Meindl, "A compact physical via blockage model," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 8, no. 6, pp. 689–692, Dec. 2000.
[21] J. Cong and S. K. Lim, "Edge separability-based circuit clustering with application to multilevel circuit partitioning," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 23, no. 3, pp. 346–357, Nov. 2006.

Shreepad Panth (S'11) received the B.S. degree from Anna University, Chennai, India, in 2009, and the M.S. degree from the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA, in 2011, where he is currently pursuing the Ph.D. degree under the supervision of Dr. S. K. Lim.

His current research interests include physical design methodologies for monolithic 3D ICs. He has authored over 20 publications in top conferences and journals.

Mr. Panth was the recipient of the Best Paper Award at Asian Test Symposium (ATS)'12 and the Best Paper Award Nominee at International Symposium on Physical Design (ISPD)'14 and Design Automation Conference (DAC)'14.

Kambiz Samadi (S'04–M'12) received the M.Sc. and Ph.D. degrees from the University of California, San Diego, CA, USA, in 2007 and 2010, respectively.

He joined Qualcomm Research, San Diego, in 2011, where he is currently a Staff Research Engineer, focusing on 3D-IC EDA solutions and 3D-IC architecture-level design space explorations. His current research interests include on-chip interconnection modeling and optimization for system-level design, 3D-IC modeling and optimization, and VLSI design manufacturing interface. He has authored over 25 publications in refereed journals and conferences.

Dr. Samadi was the recipient of the two best paper nominations and a best paper award.

Yang Du (M'96) received the Ph.D. degree from Columbia University, New York, NY, USA, in 1994.

He is currently a Director of Engineering with Qualcomm Research, San Diego, CA, USA, where he leads a team in advanced nano-technology and semiconductor research. His current research interests include emerging semiconductor devices, predictive device, circuit modeling, novel VLSI circuits and architecture, next generation 3D-IC technology and design, emerging 3D-VLSI circuit, architecture and system integration, design automation, advanced thermal modeling, and thermal aware design methodologies. He has held various engineering positions in Analog Devices, Norwood, MA, USA, AMD, Sunnyvale, CA, USA, Motorola, Chicago, IL, USA, and Qualcomm. He has authored/co-authored over 50 patents/patent publications and numerous conference/journal papers in VLSI technology, SPICE modeling, IC design, test, design automation.

Dr. Du was the recipient of the IEEE Subthreshold Microelectronics Conference and served in the Technical Program Committee since 2011. He has also served in the Advisory Committee of IEEE S3S Conference since 2013.

Sung Kyu Lim (S'94–M'00–SM'05) received the B.S., M.S., and Ph.D. degrees from the Computer Science Department, University of California, Los Angeles (UCLA), Los Angeles, CA, USA, in 1994, 1997, and 2000, respectively.

From 2000 to 2001, he was a Post-Doctoral Scholar at UCLA, and a Senior Engineer at Aplus Design Technologies, Inc., Los Angeles. He joined the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA, in 2001, where he is currently a Dan Fielder Professor of Electrical and Computer Engineering. His current research interests include architecture, circuit design, and physical design automation for 3D ICs. His research on 3D IC reliability is featured as a research highlight in the Communication of the ACM in 2014. He has authored the book entitled Practical Problems in VLSI Physical Design Automation (Springer, 2008).

Dr. Lim was the recipient of the National Science Foundation Faculty Early Career Development (CAREER) Award in 2006, the ACM Special Interest Group on Design Automation (SIGDA) Distinguished Service Award in 2008, and the Best Paper Awards from TECHCON'11, TECHCON'12, and ATS'12. His work was nominated for the Best Paper Award at ISPD'06, International Conference on Computer-Aided Design'09, Custom Integrated Circuits Conference'10, DAC'11, DAC'12, International Symposium on Low Power Electronics and Design'12, and DAC'14. He was on the Advisory Board of the ACM SIGDA from 2003 to 2008. He was an Associate Editor of the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION SYSTEMS from 2007 to 2009. He has also been an Associate Editor of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS since 2013. He has served on the Technical Program Committee of several premier conferences in Electronic Design Automation. He was a member of the Design International Technology Working Group of the International Technology Roadmap for Semiconductors. He led the Cross-Center Theme on 3D Integration for the Focus Center Research Program, Semiconductor Research Corporation, from 2010 to 2012.