On Legalization of Die Bonding Bumps and Pads for 3-D ICs

Yen-Hsiang Huang, Sai Pentapati, Anthony Agnesina, Moritz Brunion, and Sung Kyu Lim, *Fellow, IEEE*

***Abstract*—As state-of-the-art 3-D IC place-and-route flows were designed with older technology nodes and aggressive bonding pitch assumptions, they introduce an unacceptable number of 3-D via overlap violations during routing in real-world scenarios. Specifically, when dealing with higher via pitch to wire size ratios using more advanced technology nodes than they were designed for, these flows struggle to comply with width and spacing rules. In this article, we propose a novel 3-D via legalization stage and a subsequent refinement stage during routing to address this issue. Two independent via legalization methods are introduced: a force-based algorithm and a bipartite-matching algorithm with Bayesian optimization. Our two legalization methods, along with the refinement stage, are compatible with various process nodes, bonding technologies, and partitioning styles. By implementing the modified 3-D routing with the proposed legalizers, we successfully eliminate all 3-D via overlap violations while minimizing the impact on performance, power, or area.**

***Index Terms*—3-D integrated circuits, 3-D routing, face-to-face (F2F) bonding, routing, via legalization.**

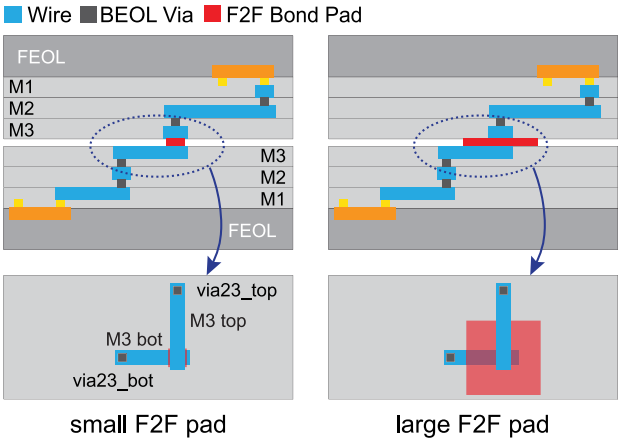


Fig. 1. Two cases of F2F designs with different bond pad sizes. (a) Small pads, (b) large pads. The corresponding top-down views are shown at the bottom.

I. Introduction

3-D IC design has been gaining the limelight in the integrated circuit industry in recent years. For instance, two of the main IC manufacturers, Intel and TSMC, both have their own technologies for 3-D IC manufacturing. Intel utilizes micro-bumping in their Foveros technology to improve an SoC's cost and power consumption with block-level 3-D IC design implementation [1]. Similarly, TSMC's hybrid bonding technology has also been put into mass production. AMD's Ryzen V-Cache 3-D IC uses hybrid bonding to achieve a large L3 cache size of over 96 MB and significantly improve system performance in gaming [2].

Furthermore, finer partitioning at L2 or L1 cache levels can extend these system-level benefits to the architecture. However, as the connection density increases, the via pitch is required to be smaller, such as 1–10 μm. Macro-3-D [3] is designed to optimize these partitioning styles, taking into account the increasingly critical connectivity between the tiers.

Pin-3-D [4] is another state-of-the-art place-and-route (PnR) flow specialized in Monolithic 3-D (M3D) ICs, which require the highest 3-D bandwidth of any partitioning type and, consequently, the smallest 3-D via pitch of around 0.1 μm [5].

All pseudo-3-D flows, including the most recent Macro-3D and Pin-3D, have a defect in the routing stage due to a misassumption about the 3-D via pitch size. Since these flows are first designed for M3D integration and extended to 3-D Face-to-Face (F2F) wafer bonding afterward, they inherit the assumption of F2F bond pad pitch in the order of 0.1–1 μm. However, current research suggests that submicron pitch values for 3-D wafer bonding pads are not easily achievable and can present yield and manufacturability issues [5], [6], [7].

In [3] and [4], using the 28 nm process node along with a small pitch obscures the placement problem of the 3-D via, as they have a larger size in real-world scenarios. Fig. 1 shows how 3-D net routing is impacted when a realistic F2F bond pad pitch is used. A more detailed analysis of this phenomenon with actual design implementations will be presented in Section II. In short, this problem is a consequence of the significantly larger pitch value of the via layer connecting the two 3-D ICs and the relatively smaller connecting metals. Additionally, as metal pitch and overall footprint shrink with advancements in process technology nodes, cut spacing violations will occur even at smaller 3-D via pitch values. A more detailed discussion of this effect can be found in Section II-B.

Manuscript received 13 May 2023; revised 23 November 2023 and 2 February 2024; accepted 6 March 2024. Date of publication 28 March 2024; date of current version 22 August 2024. This work was supported in part by the Semiconductor Research Corporation through the JUMP 2.0 Center Program (CHIMES 3136.002), and in part by the Ministry of Trade, Industry and Energy of South Korea under Grant 1415187652 and Grant RS-2023-00234159. This article was recommended by Associate Editor V. Pavlidis. *(Corresponding author: Yen-Hsiang Huang.)*

Yen-Hsiang Huang, Sai Pentapati, and Sung Kyu Lim are with the ECE Department, Georgia Tech, Atlanta, GA 30332 USA (e-mail: yhhuang@gatech.edu).

Anthony Agnesina is with NVIDIA, Austin, TX 78717 USA.

Moritz Brunion is with the University of Bremen, 28359 Bremen, Germany.

Digital Object Identifier 10.1109/TCAD.2024.3382835

TABLE I
TERMINOLOGIES USED IN THIS ARTICLE

3D via	Vias connecting the two metal layers to be bonded using micro-bumps or hybrid bond pads. We call them "vias" instead of "bumps" or "pads" because they are added during routing.
Cut Spacing	Edge-to-Edge spacing for a via (cut) = minimum cut spacing across all four via edges.
Pitch	Min required Center-Center distance (=width + spacing) as defined by the technology.
Cut Distance	Center to Center distance between two cuts. Here, L∞-norm (Chebyshev distance) is used as the distance measurement. Given two vias centered at (x_1, y_1) and (x_2, y_2), the cut distance or L∞-norm is $\max(\lvert x_2 - x_1\rvert, \lvert y_2 - y_1\rvert)$.
Cut Overlap	When Cut Distance < Pitch, we call the vias/cuts to be overlapping. This is when cut spacing violation occurs between the two vias.

This article is an extension of [8], which struggles when addressing 3-D via overlaps in logic-on-logic designs. In contrast to [8], we successfully manage logic-on-logic designs using both force-based and matching-based methods in this article, and also eliminate all remaining violations in memory-on-logic designs with both methods. We also substantially enhance the runtime and robustness of the force-based method without compromising overall routing quality. Specifically, the force-based method in [8] could take up to three days to converge on designs with high-density 3-D vias, whereas our improved method reduces the runtime to less than five minutes for all test cases. We achieve this primarily through two approaches: 1) a significantly improved force-based legalizer and 2) a new refinement stage compatible with both methods. Furthermore, we also modify the flow to improve the 3-D via legalization process, as the legalization stage presented in [8] is disrupted by the commercial tool in two ways: 1) auto-snapping vias to undesired locations and 2) generating new overlaps due to rerouting or post-route optimization.

II. ISSUE ANALYSIS

This section discusses the main issue tackled in this article, and Table I provides the terminologies used throughout.[1]

A. Issue Explanation and Severity

Memory-on-Logic partitioning, such as Macro-3D [3], partitions the application processor into two tiers: 1) memory tier and 2) logic tier. The memory tier consists of L1 and L2 data cache, while the logic tier contains all other components, such as logic blocks, L1 instruction cache, and additional caches. The pitch size of bumps or pads (3-D vias) connecting these two tiers could vary independently from the technology node, based on the methods used in 3-D manufacturing. As shown in Table II, the number of cut overlaps rapidly increases as the hybrid bond pitch ranges from 1 to 10 μm [5]. The via utilization, presented in the via_{util} column, shows the percentage of vias used compared to the maximum number possible for vias. It is important to note that the calculation of via utilization only considers physical capacity and does not take into account timing or manufacturability impacts. Our results indicate that via utilization should be maintained below 40%; otherwise, achieving a valid legalized result becomes infeasible.

[1]A special attention is to be paid on how we treat micro-bumps or hybrid bond pads as "vias."

TABLE II
3-D VIA OVERLAPS GENERATED BY STATE-OF-THE-ART 3-D FLOWS. THE 3-D DESIGNS WITH MACRO-3D AND PIN-3D HAVE 3K AND 50K 3-D VIAS, RESPECTIVELY

Macro-3D [3] Memory-on-Logic			Pin-3D [4] Logic-on-Logic		
Pitch	# overlaps	via_{util}	Pitch	# overlaps	via_{util}
1 µm	0	0.3%	0.1 µm	0	0.3%
2 µm	2	1.1%	0.2 µm	0	1.3%
5 µm	1410	7.2%	0.5 µm	0	8.9%
10 µm	11080	28.2%	1.0 µm	5315	32.2%

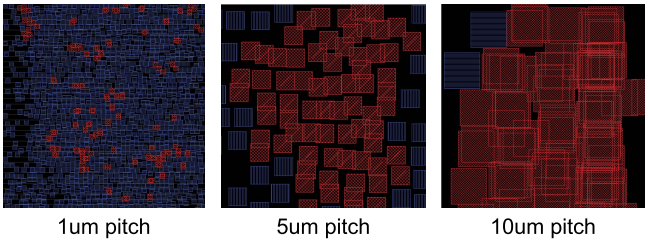


Fig. 2. Via overlaps (shown in red) at various pitch values.

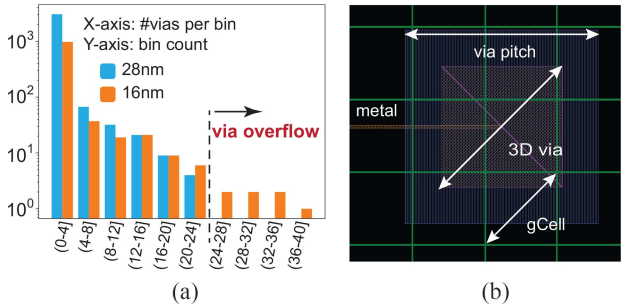


Fig. 3. (a) Via distribution of a design in two different process nodes. Each bin is 25 μm × 25 μm, (b) gCell grid (in green), 3-D via, and an M6 metal layer in a 28 nm design.

Logic-on-Logic partitioning, such as Pin-3D [4], can implement the application processor up to the CPU level (without L2 cache). Since the partitioning is done at a finer level, the number of connections between the two tiers is much higher than those in memory-on-logic partitioning. Therefore, the supported via pitch values are an order of magnitude smaller than in memory-on-logic designs to keep via utilization practical.

Similar to other flows, Pin-3D has the issue of inserting 3-D vias that violate cut spacing or cut short design rules, which is inherited from Compact-2D [9], as Pin-3D is built on top of it. Fig. 2 shows that as the 3-D via pitch increases, both the number and intensity of the violations increase.

Fig. 3(a) shows the significant differences in via densities between two cases that use the same design and partitioning but different technology nodes. Both cases use a 3-D via pitch size of 5 μm, and due to the same design and partitioning, they both contain $\sim$ 1200 3-D vias. With a bin size of 25 μm × 25 μm, the maximum via capacity of one bin is 25. However,

TABLE III
COMPARING BEOL DIMENSIONS IN THE 28 AND 16 nm NODES. THE METAL LAYER (MX) IS DIRECTLY BENEATH THE 3-D VIA. IN OUR EXPERIMENTS, WE HAVE TESTED 20.0 AND 10.0 μM FOR MICRO-BUMPING; WHILE FOR HYBRID-BOND, WE HAVE TESTED 5.0 AND 1.0 μM

Metric	28 nm	16 nm	unit
Mx pitch	0.10	0.08	µm
Via pitch below Mx	0.10	0.08	µm
3D Via pitch (Micro-Bumping)	20.0 / 10.0	10.0	µm
3D Via pitch (Hybrid-Bond)	5.0 / 1.0	5.0 / 1.0	µm
gCell width	1.48	1.08	µm

in the case using a 16 nm technology node, several bins have significantly more vias than their capacity, which is shown as via overflow in Fig. 3(a).

B. Causes of the Issue

The root cause of the issue is how commercial routers operate. Commercial routers divide the routing problem into three main stages: 1) global routing; 2) track assignment; and 3)detail routing. In the global routing stage, the entire footprint is divided into global cell (gCell) grids. Nets are then assigned to these grids according to the capacity of the grids. It is much easier to do this than directly assigning nets to tracks. Afterwards, in detail routing, the router assigns each net to tracks within the gCells to generate the exact physical routing solution.

In general, 10–15 tracks of lower metal layers will be assigned to one gCell. When assigning the nets to gCells, factors such as track utilization, wire length, delay estimations, and via cost are taken into consideration. However, during those calculation, commercial routing engines (Innovus/IC Compiler II) failed to correctly estimated the cost of 3-D vias because of the size difference of 3-D and regular vias, for which the commercial routers are originally developed. Fig. 3(b) defines the various routing dimensions, and Table III gives their values for 28 nm and 16 nm commercial process nodes. As shown in Table III, a 3-D vias can range from 10× to 200× larger than a regular via. Hence, many violations can occur during track assignment. As previously mentioned, the 3-D via pitch size can vary independently from the technology node. When technology nodes advance, the decreased gCell grid size, combined with the same 3-D via pitch, would result in an increased violation count during routing. Similarly, for a given process node, using a larger 3-D via pitch could lead to a worse via overlap situation.

C. Related Works

1) Similarity and Difference Between via and Cell Legalization: Via legalization during the routing stage has some difference but also shares some similarity compared to the legalization stage in cell placement.

Similar to the cell legalization, the major goal of via legalization is to minimize displacement during the process, due to the subsequent engineering change order (ECO) routing stage. This minimization of displacement is crucial in via legalization. If a via has larger than expected displacement, the ECO router may disregard its new location, resulting in the creation of new vias and additional overlaps.

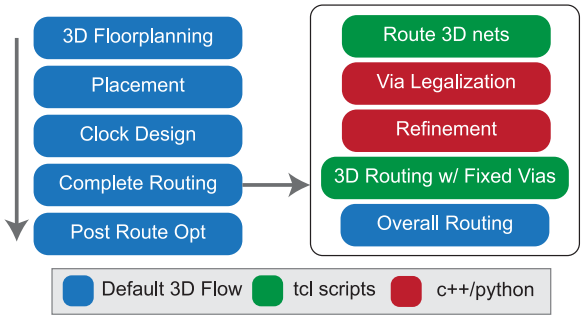


Fig. 4. Typical 3-D IC design flow, and our modifications to the routing stage for via legalization.

However, unlike cell legalization, there are no rows for vias. In the placement stage, the goal is to place cells within rows, whereas via legalization has greater flexibility to place vias.

2) Related Works in Placement: Force-directed method is a popular algorithm for placement in [10], [11], and [12]. Traditional force-based algorithms for global placement move cells to an equilibrium position by solving for the forces on each object. For example, a repelling force moves cells away from each other, while an attractive force allows for wire length minimization and spreads cells toward low-density areas.

Our force-directed method discussed in Section IV serves a different role compared to the force-directed method in global placement. In global placement, the main goal is to spread cells so that an adequate initial solution is provided for subsequent cell legalization and detailed placement. In contrast, our force-directed method aims to simultaneously spread and legalize all vias. Due to this difference, directly applying the same algorithm would yield undesired results. A detailed comparison between these approaches will be discussed in Section IV-A.

Bipartite matching is another popular technique used in cell placement, but in the detailed placement stage. In the approach of [13], bipartite matching is utilized in the detailed placement stage to assign exchangeable cells to legal positions. It also uses a windowing technique to resolve the high complexity and reduce runtime.

III. PROPOSED 3-D ROUTING FLOW

In the state-of-the-art 3-D flows [3], [4], [9], to achieve better routing quality, the implementation environment contains the entire metal layer stack, including all tiers, rather than applying a die-by-die implementation. To remove all via overlaps, the routing stage is modified by adding a via legalization stage within it, as shown in Fig. 4.

In our modified routing stage, the 3-D nets are routed first to obtain the initial placement of the 3-D vias. This initial solution for vias is expected to have many overlaps due to the problems mentioned in Section II-B. Then, with the initial placement of the vias, our legalizer generates the final locations of the vias using one of the two methods discussed in Sections IV and V. Finally, the final locations of the vias are imported into a commercial PnR tool. For example, by using the *editMove* command in Cadence Innovus to move the vias to the derived locations. Note that this step moves the vias regardless of the overall net routing.

To move vias to the desired locations, auto-snapping functions in the tools should be turned off, for example, by using *setEditMode -via_auto_snap true* in Cadence Innovus. These functions automatically snap vias to the intersection of tracks or grid. Since the vias we legalized here are interdie vias, their location should not consider track or grid inside dies.

To prevent unexpected behavior, all related functions should be disabled when moving vias using commercial tools.

The moved vias are fixed in their new locations to prevent the commercial router from moving them and creating new overlaps. Then, the 3-D nets are rerouted using the commercial router with the ECO function enabled. By turning on the ECO function, the router will try to use the new locations of the vias first. However, even with the ECO function on, the router is still capable of discarding the new locations and adding in new vias. Afterwards, to have proper connectivity, nets in all other layers are also be routed with an additional cut layer blockage on the 3-D via layer added. This is because any 3-D vias overlaps created during this step will be difficult to removed later.

Although our legalizer successfully removes all overlaps, new overlaps may still be introduced during the reroute stage following legalization or during the postroute optimization stage. These new overlaps are primarily caused by newly inserted vias in either stage, as the router discard the via location provided by our legalizer. To eliminate these new overlaps, we perform another legalization after the final postroute optimization, followed by an ECO-rerouting stage. It is crucial to mention that during the subsequent ECO-rerouting, there remains some possibility of introducing new vias and, consequently, new overlaps. Therefore, we repeat the legalization and ECO routing until no more overlaps exist. Since the two methods introduced in Sections IV and V are both timing-driven and utilize timing weights calculated from net slacks, as per (2), and slacks may be affected by rerouting, it is necessary to update the slacks and timing weights at the beginning of each iteration. According to our experience, all the violation could be solved within ten iterations with reasonable via utilization (<40%). Cases with via utilization >40% may lead to converged number of overlaps that do not converge, even with more iterations. Because in each iteration, the likelihood of the router discarding the new via locations provided by the legalizer is related to the routability of the via at the new location. When via utilization exceeds 40%, the router tends to discard a significant portion of the new locations, resulting in a large number of via being replaced to an invalid location. This can lead to the creation of even more overlaps than those just resolved by the legalizer, causing a vicious cycle. Table IV presents an example with 37.2% via utilization, which heavily relies on legalization after Post Route Optimization. Note that this example serves as the worst-case scenario among our results. It takes six iterations, and each iteration is around 0.6× to 1.5× of the runtime of the Pin-3D PnR. For other cases, such as memory-on-logic designs shown in Table VIII, the process generally completes within one iteration.

TABLE IV
AN EXAMPLE OF THE OVERLAP COUNT AT THE END OF EACH ITERATION. WITH A 37.2% VIA UTILIZATION, THIS CASE HEAVILY RELIES ON LEGALIZATION AFTER POST ROUTE OPT. THIS EXAMPLE IS CIRCUIT AP5 IN TABLE X, WHICH USES THE BIPARTITE LEGALIZER AND SERVES AS THE WORST-CASE SCENARIO AMONG OUR RESULTS

Stage		#overlap	total run time (s)
Pseudo-3D (Compact-2D)		-	14404
Pin3D PnR		-	22755
Pin3D Post Route Opt		-	11666
Legalization after Post Route Opt Iteration#	0 (right after Post Route Opt)	425	15149
	1	23	15798
	2	9	22651
	3	6	35524
	4	4	14495
	5	3	32101
	6	0	-

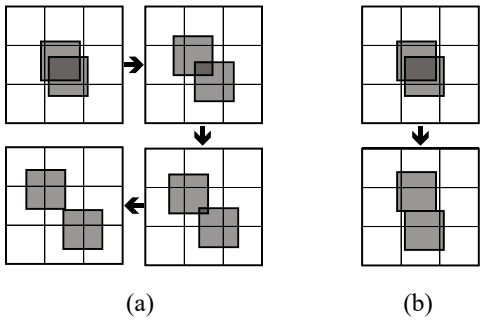


Fig. 5. Comparison between the force-base method used in [8] and one-directional force-based method used in this article. The diagonal structure in (a) negatively affects runtime, space utilization, and overall timing performance. (b) One directional force-based method.

IV. FORCE-BASED VIA LEGALIZATION

We propose a force-directed method to remove the overlaps in the 3-D via layer based on literature [10], [11], [12]. Our flow is a numerical approach by incrementally moving the vias.

A. Comparison With Force-Based Algorithm in [8]

Force-based algorithm in global placement is often used to spread cells to lower the density of cells in certain areas. To do this, the algorithm applies repulsive forces in both directions for each overlapping cells pair.

In our previous work [8], we attempted to borrow this idea. However, the method used in [8] presents challenges when applied to via legalization. Specifically, in the implementation from [8], the repulsive force exerted on an overlapping via pair along one direction is negatively correlated with the distance between them in that direction.

Consequently, the force along the less overlapping direction will be weaker than the force along the other direction. Due to this characteristic, the solver tends to eventually legalize overlapping via pairs into diagonal locations. Fig. 5(a) shows an example of two overlapping vias being gradually placed into diagonal locations when using the force-method in [8].

Diagonal structures adversely affect space efficiency. When a via pair is placed in diagonal positions, it becomes challenging for the solver to position other vias in the nearby area, unless at another diagonal location. Consequently, the solver uses only about 50% of the available area at most, making the solving process cumbersome and time-consuming. Additionally, this leads to larger-than-expected displacements. Fig. 6(a) illustrates an example of using the force-based

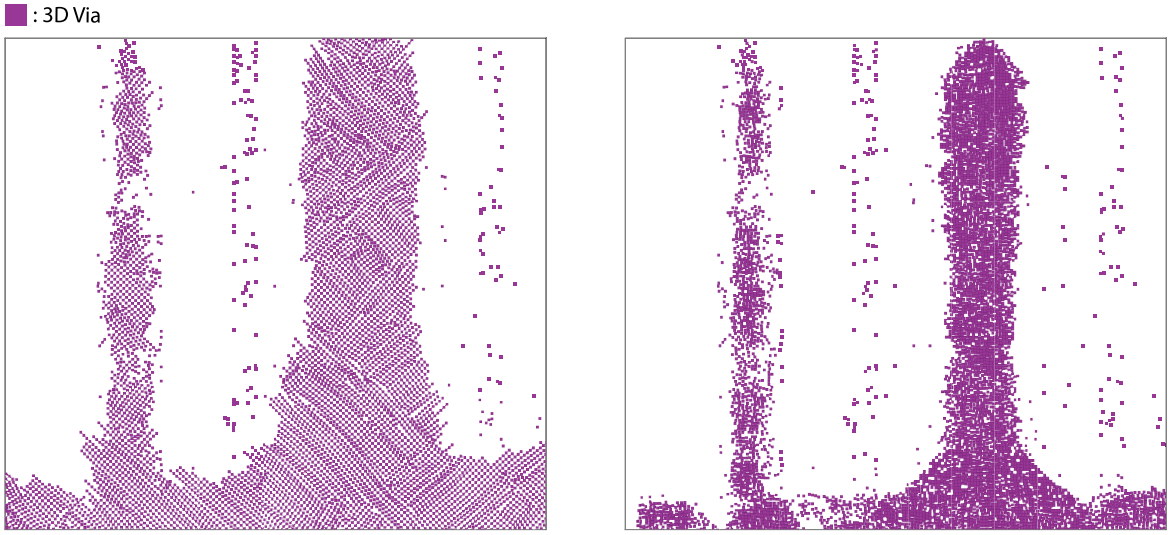


Fig. 6. Comparison of results using force-directed methods in our previous work and this article. The diagonal structure resulting from (a) leads to a less compact via arrangement compared to (b).

method in [8] in via legalization. The diagonal structure creates numerous gaps among vias, resulting in a loose final arrangement and significant overall displacement. Excessive displacements may cause the tool to disregard the via locations provided by our external solver, leading to the creation of additional vias and overlaps. To address this issue, our force-based method moves a via along the direction with less overlap with another via, instead of applying bidirectional force. Fig. 5(b) demonstrates a brief example of how our algorithm legalizes an overlapping via pair. In our research, we found that the one-directional approach significantly outperforms the traditional force-directed method by reducing average displacement by 54.3% on average.

B. Algorithm

The force-based legalizer starts with an initial solution from the commercial router, bypassing the violation fixing step, as this task is done with the force solver. The router optimizes for various design considerations such as wire length, timing criticality, congestion, and many other metrics in the initial via placement to find a good routing solution. After the 3-D net routing, the following actions are performed in each iteration of the force-based solver to remove the overlaps.

1) One-Directional Repulsive Force: We have adapted the force-based algorithm in [8] to apply a one-directional repulsive force when legalizing an overlapping via pair. This approach offers two advantages: 1) it improves space utilization and 2) it reduces runtime due to smaller displacement.

When legalizing an overlapping via pair, the solver selects the direction with less overlap and applies force on both vias to minimize overall displacement. The force is given by

$$F = \frac{\text{pitch} - \text{dis}(u, v)}{2} \tag{1}$$

where $\text{dis}(u, v)$ represents the distance between via u and v. It is also common for a via v to have multiple overlaps. After calculating all the forces corresponding to every via overlapping with v, in each direction, the solver applies the force with the largest value. The rationale behind selecting the largest-valued force is that resolving the most significant overlap may also lead to the legalization of other overlapping vias in the process.

2) Extension to Timing Driven: Restricting the legalization displacement of 3-D vias on the clock signal is crucial to minimize possible PPA degradation. Moreover, vias associated with unconstrained nets connected to, for example, TIE cells, are not as critical as the other vias. Therefore, we propose to weigh the matching cost by the timing-criticality of the connected net based on the net type and static timing analysis. We employ a standard additional net/via weight factor used extensively in timing-driven placement [14]. Per via v, we extract the worst timing path through v and define the weight based on the obtained slack and data arrival time as

$$w(v) = \begin{cases} 2^{\alpha}, & \text{if clock net} \\ \epsilon \ll 1, & \text{if unconstrained net} \\ 1, & \text{if slack}(v) \geq 0 \\ \left(1 - \frac{\text{slack}(v)}{\text{arrival}(v)}\right)^{\alpha}, & \text{otherwise} \end{cases} \tag{2}$$

where α is the criticality exponent (=2 in our experiments). We integrate the weight $w(v)$ for each via v into the force-based legalizer through two steps: First, we legalize all overlapping clock via pairs using the one-directional force method described in Section IV-B1, regardless of any overlaps with signal net vias. After legalizing all clock net vias, we fix their positions by updating their weight to infinity. Second, during the subsequent signal net via legalization, we apply the weight function in (2) to assign a weight to each signal net via. The weighted force applied to a via v when overlapping with another via u is given as

$$F = \frac{w(u)}{w(u) + w(v)} \times \frac{\text{pitch} - \text{dis}(u, v)}{2} \tag{3}$$

where $w(v)$ and $w(u)$ represent the weights of via v and u, respectively. Equation (2) and (3) ensures that the moving distance for constrained nets with negative slack is less than that for any other nets, as any movement could worsen the net

TABLE V
EXAMPLE OF THE TIMING-DRIVEN IMPACT ON CIRCUIT NP1 IN TABLE X

	WNS (ns)	TNS (ns)	Eff Freq GHz)	runtime (s)
w/o timing-driven weight	-0.695	-762.7	0.76	29
w/ timing-driven weight	-0.634	-763.1	0.79	33

slack. The rationale behind dividing the force-based method into two steps is to minimize the displacement of clock vias. Any movement of clock vias affects the clock nets and can potentially worsen the worst negative slack (WNS), leading to poor final timing performance. Furthermore, since clock vias constitute a very small proportion of the total number of vias, legalizing them first does not significantly affect the overall displacement. Consequently, legalizing clock and signal vias separately proves to be a more effective strategy for eliminating overlaps while minimizing the impact on the clock net.

Table V exemplifies the impact of timing-driven weight. We have found that the timing-driven approach is extremely helpful in circuits with high via utilization rates, such as case NP1. In this case, using the timing-driven weight improves the WNS by 9%, while the runtime increases by only 12%.

3) Implementation: To accelerate the force-based method, we employ a data structure similar to the one used in the matching-based method. The entire die is partitioned into grids, with each tile having the same size and shape as a via pitch. A via v belongs to a tile t if the left-bottom vertex of v is located within t. We use a 2-D matrix to store the tiles. To further reduce the runtime for moving vias, we store all vias belonging to a tile t within a doubly linked list instead of an array. This data structure enables the legalizer to quickly identify any overlapping vias for a given via v by only examining the neighboring nine tiles, including the tile t to which v belongs. Additionally, the runtime for moving a via is reduced to $\mathcal{O}(1)$ due to the efficient nature of erasing and adding elements in doubly linked lists.

The legalizer runs iteratively until all overlaps are removed. In each iteration, only vias with overlaps are considered and their forces are calculated, while the others remain stationary. After calculating the forces, the locations of all vias are updated simultaneously. The location of a via v is updated as follows:

$$v.d = v.d + \gamma \times F_d(v), \;\; d \in \{x, y\}. \tag{4}$$

The parameter γ controls the spreading speed of vias. Setting γ to 1 positions a pair of overlapping vias adjacent to each other after legalization, assuming there are no other vias interacting with them. However, if multiple pairs of overlapping vias affect each other, the situation may become more complicated.

4) Runtime and Displacement Optimization: Although a small γ is beneficial for minimizing displacement, it also makes the solver more challenging and time-consuming to resolve overlapping problems with high via density. After applying a small γ for several iterations initially, we can assume that most of the remaining overlaps occur in areas with high via density. Therefore, after several iterations with a small γ, the legalizer switches to a larger γ to expedite the legalization process in high-density areas. After several iterations using a larger γ, it is better to switch back to a small γ to reduce displacement since the density has decreased. Alternating between small and large γ values can significantly improve runtime without greatly impacting displacement. We use the following function to determine γ in iteration i:

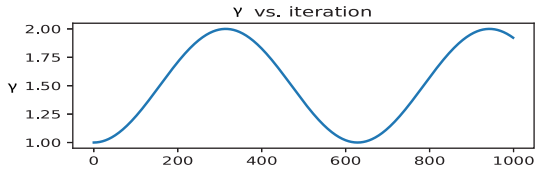


Fig. 7. γ is updated through each iteration. This figure uses the default setting.

$$\gamma = -0.5(\cos(\omega \times i) - 1)\left(\gamma_{\text{upper}} - \gamma_{\text{lower}}\right) + \gamma_{\text{lower}} \tag{5}$$

where ω controls the updating speed of γ, and γ_{upper}, γ_{lower} represent the user-defined upper and lower bounds of γ. By default, ω is set to 0.01, while γ_{upper} and γ_{lower} are set to 2 and 1, respectively. Fig. 7 shows how γ is updated using the default settings.

V. MATCHING-BASED VIA LEGALIZATION

While force-based legalization is a more traditional method, it cannot produce a viable solution if the copper pads or bumps are to be arranged on a uniform grid to yield a regular manufacturing bonding pitch. In such cases, we cast the legalization problem into a combinatorial optimization problem of assigning vias to a grid spaced uniformly with the via pitch. The grid intersections are legal via placement points. Starting from an initial solution where vias overlap, we find a legal via assignment that minimizes the total displacement by solving a minimum weighted bipartite matching problem. Even in cases where vias do not need to be assigned on a grid, using a grid is beneficial when the 3-D via manufacturing grid differs from that of the design or improves the manufacturability of 3-D vias/bond pads.

However, due to a large number of vias and grid points to assign to, it is computationally and runtime-wise only possible to solve the problem by reducing its complexity. Therefore, we propose a windowing technique tuned using Bayesian optimization to reach feasible and close-to-optimal solutions.

A. Algorithm

We see legalizing vias to the manufacturing grid while minimizing a cost metric (here, the total via displacement) as an assignment problem on a bipartite graph. Our goal is to uniquely match the set of vias $\mathcal{S}_V$ to the set of grid points $\mathcal{S}_G$, where the cost of matching a particular via v to a particular grid point g is proportional to their Manhattan distance D: $c_{v,g} \propto D(v, g) = |v_x - g_x| + |v_y - g_y| \in \mathbb{R} \cup \{\infty\}$, where (x, y) correspond to the 2-D locations of the points in the via layer. Typically, this is an unbalanced problem as $\text{card}(\mathcal{S}_V) < \text{card}(\mathcal{S}_G)$, which adds complexity. However, we transform it into a balanced one by adding enough dummy vias with zero displacement cost to any grid point.

Formally, the goal is to find the matching M minimizing $\sum c_{v,g} \;\forall (v, g) \in M$. To solve this minimum cost (weight)

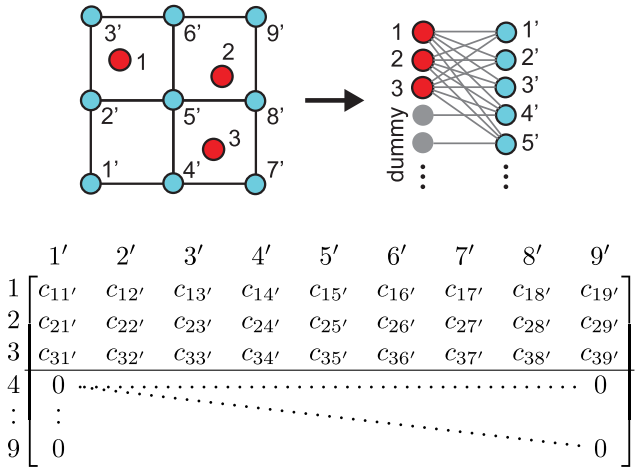


Fig. 8. Our high-level grid assignment formulation. Vias (in red) and manufacturing grid points (in blue) are transformed into a bipartite graph, whose pairwise distances form the weight matrix, input to the LAP solver.

Algorithm 1: Windowed Bipartite Matching Algorithm

Data: (x, y) locations of 3D vias from routed design; floorplan boundary; horizontal/vertical pitches of 3D via manufacturing grid; window definition;
Result: A manufacturing grid via assignment minimizing the total timing-driven displacement cost;
for $w \in$ *Windows* **do**
1. Query vias $\in w$ and build grid in that window;
2. Compute pairwise distances, and multiply with pre-computed timing weights to obtain the cost matrix;
3. Solve the LAP with the shortest augmenting path algorithm [15];
4. Apply the assignment solution: update locations of vias and recompute query matrix;

perfect matching problem, we rewrite it as a linear assignment problem (LAP) as follows:

$$\min \sum_{v,g} c_{v,g} x_{v,g}, \quad \text{s.t.} \quad \sum_{g} x_{v,g} = 1, \quad v \in \mathcal{S}_V$$
$$\sum_{v} x_{v,g} = 1, \quad g \in \mathcal{S}_G$$
$$x_{v,g} \geq 0, \quad v \in \mathcal{S}_V, \quad g \in \mathcal{S}_G \quad (6)$$

where $x_{v,g} = 1$ if $(v, g) \in M$ and 0 otherwise. We solve this problem using the shortest augmenting path algorithm [15]. Fig. 8 depicts the transformation of the geometric problem to LAP represented in a matrix form,[2] input to the shortest augmenting path algorithm.

1) Extension to Timing Driven: We have also integrates the weight function in 2 into our matching-based legalizer. After calculating each via's weight, the new LAP formulation is then updated to use $c_{v,g} = w(v) \cdot D(v, g)$.

2) 2-D Windowing: The shortest augmenting path algorithm has a time complexity of $\mathcal{O}(\max(\text{card}(\mathcal{S}_V), \text{card}(\mathcal{S}_G))^3)$.

[2]Other representations, like adjacency lists for sparse matrices, could enhance runtime and reduce memory usage in certain scenarios. However, these were not explored in this article as the bipartite solver's runtime is relatively minor in the overall process. For details, see Tables XI and XII.

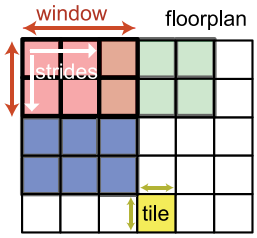


Fig. 9. Window-based 2-D floorplan grids. In each window, the grid assignment problem is solved optimally. There are multiple legal via locations (grid points) within each tile.

TABLE VI
SIX WINDOWING PARAMETERS TUNED WITH BAYESIAN OPTIMIZATION. THE 3-D VIA PITCHES ARE NOTED AS p_x, p_y. WINDOW AND STRIDING PARAMETERS UNITS ARE SET IN TILE WIDTH AND HEIGHT

Name	Type	Range	Default Value
Tile width (um)	float	[5 p_x, 100 p_x]	30 p_x
Tile height (um)	float	[5 p_y, 100 p_y]	30 p_y
Window size x	int	[3, 10]	3
Window size y	int	[3, 10]	3
Horizontal stride	int	[1,3]	2
Vertical stride	int	[1,3]	2

Moreover, the space complexity of the problem is dominated by the size of the cost matrix of $\text{card}(\mathcal{S}_V) \times \text{card}(\mathcal{S}_G) \times 8$, where 8 is the number of bytes to encode *float64* weights. Thus, even with modern machines, the required memory can easily exceed the RAM capabilities.

Therefore, we propose a tiling/windowing method for 2-D floorplan/space partitioning to reduce matrix size and solve the LAP locally in each window, following Algorithm 1. This window is slid over the 2-D floorplan, similar to a 2-D convolution filter. First, we partition the whole floorplan canvas into small tiles. A window is then defined as a rectangle of tiles. Each window will likely contain less than a few thousand vias or grid points for appropriate window sizes, making the problem tractable as we only build the cost matrix and grid points locally. Then, we update the via locations for each window based on the found assignment. Moreover, to counteract the nonoptimality introduced by the solutions being only optimal inside the given window, we do not fix the vias after they have been assigned and use striding to allow reassignment of previously derived via locations if it reduces the total displacement.

B. Parameter Tuning With Bayesian Optimization

The quality of the assignment significantly depends on the values of the windowing parameters presented in Table VI. These correspond to the window configuration in Fig. 9. For example, for a small problem size, the window can be defined to include the entire floorplan, and an optimal solution can be found directly. However, these parameters must be tuned for more complex problems to obtain near-optimal solutions within a reasonable runtime. This objective is realized in the maximization of the following:

$$f(p) = w_C \tanh \frac{C_0}{C(p)} + w_D \tanh \frac{D_{\max_0}}{D_{\max}(p)} + \tanh \frac{T_0}{T(p)} \quad (7)$$

TABLE VII
EXAMPLES OF DISPLACEMENT METRICS BEFORE AND AFTER 10 BAYESIAN OPTIMIZATION ITERATIONS. AP1 IS USED IN THE EXPERIMENT, WHICH HAS AROUND 3K VIAS ON A 5 μM PITCH GRID, WITH AROUND 45K GRID POINTS. WEIGHTS $w_C = 20$, AND $w_D = 10$

Parameters / Metric	Before Tuning	After Tuning
Tile width (um)	30.0 p_x	50.5 p_x
Tile height (um)	30.0 p_y	58.5 p_y
Window size x	3	4
Window size y	3	5
Horizontal stride	2	3
Vertical stride	2	2
Total cost	9338.8	9157.0 (−1.9 %)
Maximum displacement	29.8	19.5 (−34.6 %)
Runtime (sec.)	0.91	1.03

where p denotes the parameter settings, C denotes the total timing-driven displacement cost, $D_{\max}$ is the maximal displacement, and T is the runtime. We integrate the maximal displacement to reflect the maximum deviation from the router's initial decision. We set $f(p) = 0$ if the LAP solver crashes due to a runtime exception from the inability to allocate enough memory for the cost matrix or shortest path algorithm. The reference values subscripted with 0 are set based on the default parameter values. The application of the tanh is to squash the differently scaled metrics into $[-1, 1]$ and make them comparable. The weights of each component can be set to realize different tradeoffs of optimality versus speed.

To maximize this objective, we use Bayesian optimization [16]. The Bayesian algorithm sequentially queries the function f and builds a surrogate function interpolating the evaluations. We use the Gaussian process as a surrogate family with a squared exponential kernel. Moreover, we use the upper confidence bound (UCB) acquisition function to pick the next candidate query point. After multiple iterations, we report and use the assignment that maximized the presented objective function. Table VII shows the positive effect of the tuning on the maximal displacement.

1) Implementation: We implement the flow in Python, based on Numpy vectorized features, and accelerate the cost matrix calculation with multithreading and SIMD through Numba just-in-time compilation. Moreover, to speed up the query of points in a given window, rather than use traditional 2-D spatial query data structures, such as quadtrees or KD-trees, we store the indexes of the list of vias in a 2-D matrix Q where $Q[i][j] = \{vias \in \text{tile}(i, j)\}$. Using this matrix Q to query points is much faster than KD-trees due to the regular memory accesses. Moreover, the matrix is quickly updated locally whenever the via locations are changed. In addition, the Bayesian optimization is done using a Python library [17].

VI. REFINEMENT

We have developed a refinement stage to further reduce displacement, which can be used after either matching-based or force-based legalization, as long as all overlaps have been eliminated. The refinement stage consists of two steps: 1) via swapping and 2) via relocation. In our study, we discovered that our refinement technique significantly reduces both the average and maximum displacement for the force-based and matching-based methods. With the force-based method, we observed a 40.1% reduction in average displacement and a 35.8% reduction in maximum displacement. Meanwhile, for the matching-based method, the average displacement was reduced by a impressive 75.6% and the maximum displacement by 22.8%.

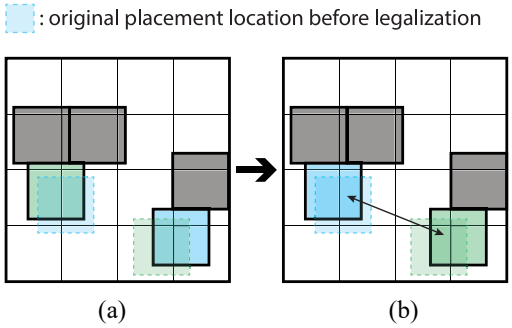


Fig. 10. Via swapping example. (a) Initial output from legalizer. (b) After via swapping.

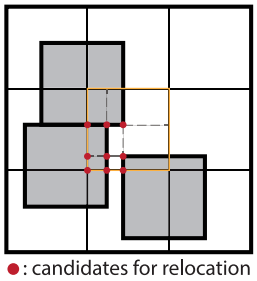


Fig. 11. Via relocation example. The solver searches for a location to place a via within the centered tile. The solver compares the displacements of the locations marked with red dots and selects the one with the least displacement.

A. Via Swapping

Once a fully legal via solution is found, we swap vias with others that are closer to the original placement location before legalization, as long as the swapping reduces the overall displacement. When applying via swapping for via v, the solver examines the area surrounding v's original location to determine if there is any other via u that results in a positive gain after swapping. The gain of swapping v and u is calculated by

$$\begin{aligned}\text{gain} = {} & w(v)\text{dis}(u.\text{loc} - v.\text{loc}_0) + w(u)\text{dis}(v.\text{loc} - u.\text{loc}_0) \\ & - w(v)\text{dis}(v.\text{loc} - v.\text{loc}_0) - w(u)\text{dis}(u.\text{loc} - u.\text{loc}_0).\end{aligned} \tag{8}$$

Fig. 10 demonstrates an example of via swapping. The solver swaps the blue and green vias because doing so reduces the overall displacement. Concerning time complexity, since a tile—the same size of a via—can only accommodate one via in a legalized solution, the total complexity of the search process is $O(n \times d_{\max}^2)$, where n is the total number of vias, and $d_{\max}$ means the maximum displacement given in a legalized solution. Also because of testing whether swapping a pair of vias can result in better displacement is $O(1)$, the total time complexity of via swapping is $O(n \times d_{\max}^2)$.

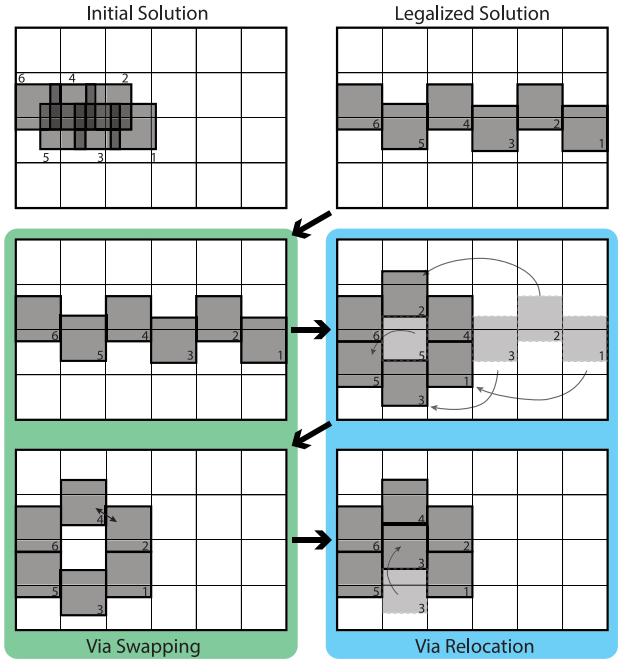


Fig. 12. Via refinement overall run example. In this case, the solver performs two rounds of refinements, each comprising a via swapping step and a via relocation step. Note that during the first via swapping step, no vias are swapped.

B. Via Relocation

When applying via relocation for via v, the solver searches the tiles near v's original location to determine if there is any available space that can accommodate v without creating any new overlaps. The solver starts from the original tile of v and expands outward. When searching in a tile, all the intersections of the tile's border and the extension lines of nearby vias' edges are considered as candidates for locating v. The candidate with the least displacement is selected to place v. Since the refinement runs only on a valid solution without any overlaps, each neighboring tile will contain at most one via, resulting in a maximum of eight nearby vias. Given that the number of total nearby vias is constant, the complexity of searching a tile for one via is $O(1)$. Combining with the complexity of the search process mentioned in Section VI-A, the total time complexity of via relocation is also $O(n \times d_{\max}^2)$.

Fig. 11 demonstrates an example of via relocation. The solver is searching for a location to place a via within the centered tile. All the red dots are candidates to be considered, and the location with the least displacement will be selected.

C. Overall Run

We perform several rounds of refinements until no more effective refinements can be done.

Fig. 12 demonstrates an example of a complete refinement stage. With the given initial solution from the commercial tool and the legalized solution provided by our solver, the refinement stage performs several rounds of optimization. In each round, for each signal net via, the legalizer attempts one via swapping step followed by one via relocation step. The searching area for both via swapping and relocation is determined by the user, which is by default the area not farther than 10× pitch size. We refine vias with larger to smaller weighted displacement, where the weight of each via is also calculated using (2). In this example, the solver performs two rounds of refinements.

VII. Results and Analysis

A. Experimental Setup

1) Technology Setup: In the experiment, two PDKs are used to evaluate the efficiency and applicability of our legalizer: 1) a 28 nm node and 2) a 16 nm node. In addition to the technology nodes, we also test two bonding styles with different 3-D via pitch size, including a micro-bump-based 3-D IC with 20 or 10 μm pitches and a hybrid-bond-based 3-D IC with 5 or 1 μm pitches.

2) Place/Route Flows: The hybrid bond flows are implemented using Macro-3D and Pin-3D flows for memory-on-logic and logic-on-logic partitioning, respectively, as discussed in Section II. Different from the hybrid bond flows, the micro-bump 3-D ICs are designed using a die-by-die flow, with bump locations preassigned during the floorplanning stage. Due to the absence of initial via locations, we initially place the via at the center of macro pins connected by each 3-D net. Additionally, the center of macro pins for each 3-D net is used as the reference point to minimize displacement.

3) Partitioning Types and Benchmark: For the memory-on-logic partitioning, the experiments use the following circuits with Macro-3D flow.

1) AP1 is a dual-core application processor with an L2 cache of size 512 kB. Implemented with the 28 nm node, its memory tier contains both L1 and L2 cache.
2) AP2 is a single-core processor with 1 MB of L2 and is also implemented in the 28 nm node. Different from AP1, the memory tier only contains L2 cache, and the cut-size is $\sim$ 1500.
3) AP3 is similar to AP2 but implemented in the 16 nm PDK. Moreover, it has a smaller L2 cache of 512 kB due to the different scaling factors of the memory and logic cells.

For the logic-on-logic partitioning, AP4 and AP5 are used in the experiments. AP4 and AP5 are modified versions of AP1 and AP2, with their L2 cache removed. Both AP4 and AP5 are implemented in the 28 nm node using the Pin-3D flow.

B. Memory-on-Logic With Hybrid Bonding

The results of the memory-on-logic designs using hybrid bonding are presented in Table VIII, where hybrid bonding pitch is 5 μm, with both width and spacing at 2.5 μm. Results using the methods from [8] are also included in our result tables. Unlike [8], the results for the legalizers presented in this article all utilize the flags mentioned in Section III. This approach provides a clearer illustration of the contributions beyond the commercial tool itself, focusing on the enhancement of the algorithm and flow. Additionally, we have observed that the baseline results (marked as "none") differ from those in [8], despite there being no additional flags involved. We attribute this discrepancy to the version update of Cadence Innovus, as we used a different version than that used in [8].

Note that the number of vias may vary with each technique because the router might add new vias during rerouting after via legalization, or during the post-route optimization

TABLE VIII
VIA LEGALIZATION PPA RESULTS OF MEMORY-ON-LOGIC 3-D IC WITH 5-μM PITCH HYBRID BONDING

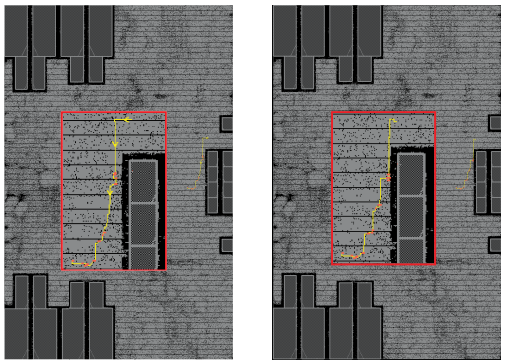
Implementation Details		Physical Stats			Timing and Power			Legalizer Stats			
Circuit	Legalizer	Area	core density	WL	WNS	TNS	Power	#Vias(via$_{util}$)	#Viols	d_{max}	d_{avg}
Units		mm^2	%	m	ns	ns	mW			µm	µm
28 nm AP1 @1.25GHz	None	1.11	77	11.45	-0.113	-150.9	677.2	5053(11.7%)	3780	-	-
	Force [8]	1.11	77	11.67	-0.114	-118.2	680.8	3279(7.3%)	9	30.6	4.2
	Bipartite [8]	1.11	77	11.67	-0.111	-127.0	679.3	3276(7.3%)	2	20.0	3.0
	Force	1.11	77	11.67	-0.123	-173.2	680.9	3275(7.3%)	0	22.3	1.8
	Bipartite	1.11	77	11.67	-0.112	-149.1	680.0	3276(7.3%)	0	19.5	1.6
28 nm AP2 @1.25GHz	None	1.89	70	18.78	-0.253	-693.7	880.1	2062(4.0%)	2222	-	-
	Force [8]	1.89	70	18.92	-0.265	-623.6	870.9	1338(1.8%)	5	21.7	3.4
	Bipartite [8]	1.89	70	18.87	-0.298	-671.8	876.9	1344(1.8%)	2	9.4	2.6
	Force	1.89	70	18.83	-0.292	-674.1	874.8	1346(1.8%)	0	15.0	≈0.0
	Bipartite	1.89	70	18.83	-0.340	-703.5	881.7	1346(1.8%)	0	4.7	≈0.0
16 nm AP3 @1.60GHz	None	0.605	71	9.41	-0.004	-0.029	843.6	1277(5.2%)	2314	-	-
	Force [8]	0.605	71	9.46	-0.008	-0.227	843.0	1424(5.8%)	412	30.7	1.9
	Bipartite [8]	0.605	71	9.44	-0.009	-0.046	843.1	1359(5.7%)	296	21.5	0.8
	Force	0.605	71	9.42	-0.005	-0.026	846.6	1492(5.9%)	0/2*	2.6	≈0.0
	Bipartite	0.605	71	9.42	-0.000	-0.000	845.4	1354(5.7%)	0/2*	28.2	≈0.0

stage. Additionally, the number of legalization iterations following post-route optimization can also affect the total number of vias, as new vias might be added during this stage.

As shown in Table VIII, all of the 3-D via overlap violations in all cases are eliminated by both force-based and bipartite matching-based legalizers. The only exception is AP3, where the legalizer cannot naturally eliminate all violations, leaving two via overlap pairs. However, this issue can be resolved by simply removing one via and then performing ECO routing. In Table XI, we regard this process as a single iteration of legalization. In most cases, we can observe that the resulting number of 3-D vias using both legalizers is smaller than that in the nonlegalized flow. This is because the modified flow in Fig. 4 does not allow non-3D nets to have any 3-D vias. Furthermore, the overall wire length is barely affected, as memory-on-logic partitioning only contains a few 3-D nets. The total power is also only slightly affected for the same reason.

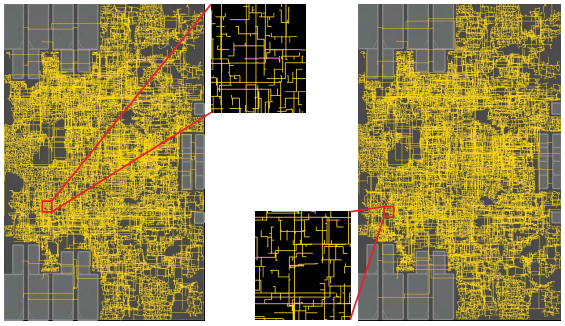
The critical timing paths highlighted in Fig. 13 show similarities between cases with and without legalization. As shown, both paths are across the same hierarchies, indicating that the legalization process does not create new timing bottlenecks. Likewise, given the similar pattern with and without legalization in Fig. 14, the clock tree is only slightly affected, without impacting the overall design qualities.

1) Variations in the via Assignment Pattern: Fig. 15 shows via placement results for different legalization methods. As shown, the force-directed method result in Fig. 15(b) are more scattered than that of the matching-based method in Fig. 15(c). The displacement metrics in Table VIII also support this observation. In addition to smaller displacement and better PPA, the bipartite matching approach offers another advantage related to manufacturing. Since the via is aligned to a wider grid, it simplifies the via alignment during manufacturing. On the other hand, the force-based legalization requires more precise alignment. Although the grid-based method has a smaller solution space, it produces better results in terms of displacement. This is attributed to the optimality of the bipartite matching algorithm, in contrast to the heuristic approach of the force-directed method.



(a) (b)

Fig. 13. Critical paths in case AP1. Routing patterns are similar. (a) Before legalization. (b) After legalization.



(a) (b)

Fig. 14. Clock tree in case AP1. Routing patterns are similar. (a) Before legalization. (b) After legalization.

2) Comparison to [8]: In Table VIII, we show the differences in memory-on-logic design between the previous work [8] and this article. The most noticeable difference is that in this work, all via overlaps are reduced in the

TABLE IX
MEMORY-ON-LOGIC 3-D IC WITH MICRO-BUMPING. WE USE 20 μM PITCH FOR AP2 BENCHMARK, AND 10 μM FOR AP1, AP3 DUE TO THE SMALLER FOOTPRINTS. THERE ARE NO PRELEGALIZATION RESULTS HERE, AS THE MICRO-BUMPS ARE FIXED DURING FLOORPLANNING

Implementation Details				Physical Stats			Timing and Power			Legalizer Stats			
Node	Circuit	Legalizer	Freq.	Area	WL	via_{util}	WNS	TNS	Power	#Vias	#Viols	d_{max}	d_{avg}
	Units		GHz	mm^2	m	%	ns	ns	mW			µm	µm
28 nm	AP1	Greedy	1.25	1.11	11.56	53	-0.118	-25.15	796.6	2957	0	601.0	127.5
		Bipartite	1.25	1.11	11.41	53	-0.014	-5.04	794.4	2957	0	385.1	68.1
28 nm	AP2	Greedy	1.25	2.18	20.06	43	-0.323	-1336.4	912.1	1187	0	435.7	160.6
		Bipartite	1.25	2.18	19.92	43	-0.295	-1237.6	907.9	1187	0	198.1	79.3
16 nm	AP3	Greedy	1.60	0.61	10.39	39	-0.155	-881.7	1043	1169	0	326.6	121.1
		Bipartite	1.60	0.61	9.92	39	-0.140	-835.8	1006	1169	0	102.2	44.7

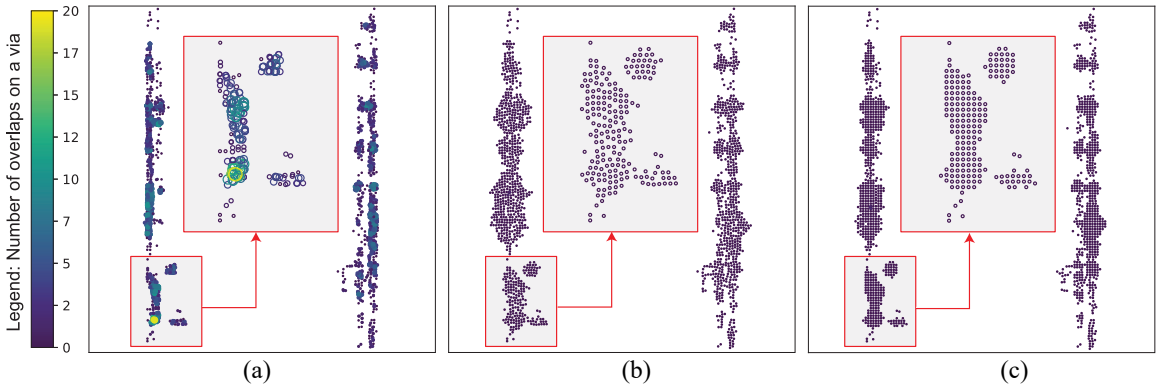


Fig. 15. Via assignment under the three different legalization methods for the AP3 benchmark implemented with 16 nm node. The size and color of each via represent the number of overlaps. The gray rectangle shows the zoom-in on a few vias. (a) No via legalization. (b) Force-based legalization. (c) Matching-based legalization.

memory-on-logic case, even for the resolvable case AP3 in [8], with a bit of help from manually removing a single redundant 3-D via as aforementioned. For cases AP1 and AP2, the result table in [8] indicates no via overlap with the older version of the Cadence Innovus. However, following a software update, some via overlaps are not fully removed using either the force-based or the bipartite-matching method from [8]. These can be addressed by the methods presented in this work, due to the significantly lower average and maximum displacements. With lower displacements, the likelihood of the commercial tool disregarding our recommended via locations is reduced, thereby decreasing the potential for new overlaps after rerouting and post-route optimization. We have observed timing degradation with the legalization method proposed in this work, especially in the AP1 and AP2 cases. Since the legalization after post-route optimization only reroutes a few nets and has a negligible impact on timing, we attribute this degradation to the impact of turning off the auto-snap function mentioned in Section III.

3) Runtime Analysis: In Table XI, we present the runtime of various techniques, including those from our previous work [8]. It is important to note that runtime may vary depending on the configuration and status of the machine used. Consequently, we also provide a breakdown of the runtime for different stages in the default Macro-3D flow to serve as a reference for runtime analysis. As shown in Table XI, the runtime of our legalizer and the refinement runtime overhead is negligible compared to the routing stage of Macro-3D. Regarding the legalization iteration count after post-route optimization, all of our memory-on-logic results can be resolved within one iteration of legalization. The runtime overhead of each iteration is similar to that of Macro-3D routing, because the runtime of the legalization stage is dominated by the rerouting stage.

C. Memory-on-Logic With Micro-Bumping

Table IX shows the results of the three designs implemented with the micro bump bonding assumption. As micro bump bonding flows generally require vias to be preplaced on a custom grid and are not placed by the router, a force-based legalizer cannot be applied. Here we compare the assignment of bumps using bipartite grid assignment with a simple priority greedy algorithm based on timing order. Due to the smaller footprints, a bumping pitch of 20 μm is used for the AP2 benchmark and a 10 μm pitch for AP1 and AP3.

A greedy approach creates large displacements for bumps with the lowest assignment priority and is reflected in the max displacement values in Table IX. The optimal placement with a bipartite matching solution gives much better results, even considering the timing. Compared to greedy solutions, we see a significant improvement in the WNS and TNS with the bipartite matching assignment, showing our proposed solution's robustness to hybrid-bonding and micro-bumping 3-D designs.

D. Logic-on-Logic With Hybrid Bonding

The logic-on-logic via legalization results are presented in Table X, with a 1 μm 3-D via pitch size. We observe that our legalizers can effectively handle the high via count in

TABLE X
VIA LEGALIZATION PPA RESULTS OF LOGIC-ON-LOGIC 3-D IC WITH 1 μM PITCH HYBRID BONDING. FOR NP1, AS STATES IN SECTION III, THE HIGH VIA UTILIZATION RATE (> 40%) WILL CAUSE A VICIOUS CYCLE, AND HENCE THE VIA OVERLAPPING CANNOT BE TOTAL REMOVED

Implementation Details		Physical Stats			Timing and Power			Legalizer Stats			
Circuit	Legalizer	Area	core density	WL	WNS	TNS	Power	#Vias(via$_{util}$)	#Viols	d_{max}	d_{avg}
Units		mm^2	%	m	ns	ns	mW			µm	µm
28 nm AP4 @1.50GHz	None	0.25	77	3.612	-0.021	-0.4	343.3	55710(21.9%)	19165	-	-
	Force [8]	0.25	77	3.607	-0.097	-12.8	344.3	52997(20.9%)	83	10.7	1.3
	Bipartite [8]	0.25	77	3.587	-0.089	-10.7	343.4	52824(20.8%)	25	6.1	0.6
	Force	0.25	77	3.601	-0.070	-7.2	344.8	53106(21.0%)	0	5.2	0.1
	Bipartite	0.25	77	3.602	-0.067	-6.0	344.0	52817(21.1%)	0	6.0	0.5
28 nm AP5 @1.50GHz	None	0.59	71	8.97	-0.023	-8.8	865.6	212076(36.0%)	20735	-	-
	Force [8]	0.59	71	10.36	-0.144	-457.0	918.6	220105(37.3%)	1487	40.1	6.4
	Bipartite [8]	0.59	71	9.00	-0.034	-31.0	870.0	219673(37.3%)	168	4.6	0.5
	Force	0.59	71	8.96	-0.052	-74.7	873.9	219632(37.2%)	0	8.9	0.1
	Bipartite	0.59	71	8.95	-0.046	-75.3	873.0	219721(37.2%)	0	2.5	0.1
16 nm NP1 @1.60GHz	None	0.12	74	2.57	-0.465	-313.3	390.7	59997(51.0%)	43547	-	-
	Force [8]	0.12	74	3.02	-0.715	-1011.7	406.8	63899(54.3%)	72494	34.5	7.8
	Bipartite [8]	0.12	74	2.75	-1.186	-922.2	399.3	56407(47.9%)	100922	27.2	1.0
	Force	0.12	74	2.68	-0.634	-763.1	394.8	51028(42.5%)	37411	31.5	1.1
	Bipartite	0.12	74	2.65	-0.649	-669.9	394.4	50369(43.4%)	38408	25.6	0.8

TABLE XI
RUNTIME COMPARISON FOR MEMORY-ON-LOGIC 3-D IC WITH 5-μM PITCH HYBRID BONDING. FOR AP3, 2 OVERLAPPING CANNOT BE RESOLVED BY EITHER LEGALIZER. HOWEVER, THEY CAN BE ADDRESSED BY SIMPLY REMOVING ONE VIA FOLLOWED BY AN ECO ROUTING

Implementation Details		Legalizer Stats							Reference
Circuit	Legalizer	#Vias(via$_{util}$)	#Viols	d_{max}	d_{avg}	Legalizer runtime	refinement overhead	#iter after Post Route Opt	Runtime of Default Macro-3D flow
Units				µm	µm	s	s		s
28 nm AP1 @1.25GHz	None	5053(11.7%)	3780	-	-	-	-	-	(Before Routing) 20257
	Force [8]	3279(7.3%)	9	30.6	4.2	<1	-	-	+
	Bipartite [8]	3276(7.3%)	2	20.0	3.0	<1	-	-	(Routing) 2176
	Force	3275(7.3%)	0	22.3	1.8	<1	<1	0	+
	Bipartite	3276(7.3%)	0	19.5	1.6	<1	<1	0	(Post Route Opt) 3244
28 nm AP2 @1.25GHz	None	2062(4.0%)	2222	-	-	-	-	-	(Before Routing) 7511
	Force [8]	1338(1.8%)	5	21.7	3.4	<1	-	-	+
	Bipartite [8]	1344(1.8%)	2	9.4	2.6	<1	-	-	(Routing) 3446
	Force	1346(1.8%)	0	15.0	≈0.0	<1	<1	0	+
	Bipartite	1346(1.8%)	0	4.7	≈0.0	<1	<1	1	(Post Route Opt) 4072
16 nm AP3 @1.60GHz	None	1277(5.2%)	2314	-	-	-	-	-	(Before Routing) 13451
	Force [8]	1424(5.8%)	412	30.7	1.9	<1	-	-	+
	Bipartite [8]	1359(5.7%)	296	21.5	0.8	<1	-	-	(Routing) 3359
	Force	1492(5.9%)	0/2*	2.6	≈0.0	<1	<1	1*	+
	Bipartite	1354(5.7%)	0/2*	28.2	≈0.0	<1	<1	1*	(Post Route Opt) 2729

logic-on-logic designs, as both the force and matching-based methods removed a significant amount of overlaps. In cases AP4 and AP5, all overlaps are removed after several additional legalization iterations, as stated in Table XII. For case NP1, due to the enormous number of vias and the relatively high via utilization rate (>40%), it remains unsolvable by our legalizer, even with additional legalization after post-route optimization. As stated in Section III, the root cause of this issue is that a high via utilization rate can affect the routability of vias and increase the likelihood of the router ignoring our suggested via locations. Also, when the utilization rate is this high, additional legalization after post-route optimization tends to have more disadvantages than benefits, as the rerouting process can introduce more via overlaps than it reduces. Similar to the results in Table VIII, the power and timing of the designs are not substantially affected, and the wire length even shows slight improvement. However, in case NP1, as the via utilization rate exceeds 40%, both legalizers begin to degrade design quality, particularly in terms of timing. For more advanced technology nodes, as the 3-D connectivity complexity increases rapidly, and considering the high timing criticality as well as the dense connectivity between dies, this partitioning style will not be compatible with the bump size and routing solution proposed here. Therefore, M3D ICs might be a better choice for such dense integration [4], as with the fine pitch of M3D vias, the overlaps issue of 3-D vias would be alleviated [5].

1) Comparison to [8]: Table X presents a comparison between different methods proposed in this work and those from previous work [8] in logic-on-logic cases. Similar to the findings in VII-B2, there is a significant reduction in the count of via overlaps when compared to methods from [8]. Additionally, as observed in the memory-on-logic cases, there is a noticeable reduction in displacement, especially for the force-based method. In terms of timing, unlike in memory-on-logic cases, for logic-on-logic cases, we have found that all the methods proposed in this article perform better than their counterparts in [8] in terms of WNS and TNS, except

TABLE XII
RUNTIME COMPARISON FOR LOGIC-ON-LOGIC 3-D IC WITH 1-μM PITCH HYBRID BONDING. FOR NP1, THE RESULTS PRESENTED HERE DO NOT INCLUDE LEGALIZATION AFTER POST-ROUTE OPTIMIZATION, AS IT IS NOT BENEFICIAL IN CASES WHERE THE VIA UTILIZATION > 40%

Implementation Details		Legalizer Stats							Reference
Circuit	Legalizer	#Vias(via$_{util}$)	#Viols	d_{max}	d_{avg}	Legalizer runtime	refinement overhead	#iter after Post Route Opt	Runtime of Default Pin-3D flow
Units				µm	µm	s	s		s
28 nm AP4 @1.50GHz	None	55710(21.9%)	19165	-	-	-	-	-	(Compact 2D) 7883
	Force [8]	52997(20.9%)	83	10.7	1.3	180	-	-	+
	Bipartite [8]	52824(20.8%)	25	6.1	0.6	30	-	-	(Pin-3D PnR) 12923
	Force	53106(21.0%)	0	5.2	0.1	8	<1	2	+
	Bipartite	52817(21.1%)	0	4.7	0.1	31	1	2	(Pin-3D Post Route Opt) 5210
28 nm AP5 @1.50GHz	None	212076(36.0%)	20735	-	-	-	-	-	(Compact 2D) 14404
	Force [8]	220105(37.3%)	1487	40.1	6.4	135540	-	-	+
	Bipartite [8]	219673(37.3%)	168	4.6	0.5	105	-	-	(Pin-3D PnR) 22755
	Force	219632(37.2%)	0	8.9	0.1	767	136	4	+
	Bipartite	219721(37.2%)	0	2.5	0.1	114	9	6	(Pin-3D Post Route Opt) 11666
16 nm NP1 @1.60GHz	None	59997(51.0%)	43547	-	-	-	-	-	(Compact 2D) 18369
	Force [8]	63899(54.3%)	72494	34.5	7.8	498	-	-	+
	Bipartite [8]	56407(47.9%)	100922	27.2	1	19	-	-	(Pin-3D PnR) 11116
	Force	51028(42.5%)	19573	31.0	1.2	54	24	-*	+
	Bipartite	54619(43.4%)	37411	29.2	0.8	23	4	-*	(Pin-3D Post Route Opt) 13680

for the bipartite-matching method in AP5. This suggests that, in cases where the via count and utilization rate are high enough, such as in logic-on-logic cases, the benefit of reducing displacement is more influential than the flaws of disabling the auto-snap function. Regarding power, similar to memory-on-memory cases, the difference is minimal.

2) Runtime Analysis: Table XII displays the runtime breakdown for methods proposed in this study, as well as those from [8]. The runtime of legalizers in logic-on-logic cases is significantly higher than in memory-on-logic cases, attributable to their much higher number of vias and via utilization rate. However, this increase in runtime is still negligible when compared to the PnR stage in Pin3D. The runtime overhead for the refinement stage during the legalization process is relatively small in comparison to the total runtime of the legalizer. The maximum overhead observed is 44% in the NP1 case, while in other cases, it remains below 20%, irrespective of the legalization methods used. Additionally, we observed that when the via utilization rate is low, the force-based method can resolve the problem in a relatively shorter time. For instance, in the AP4 case, the force-based method derived an overlap-free solution in just 8 s, compared to 31 s using the bipartite-matching method.

Regarding the legalization after post-route-optimization, we report only the number of iterations for AP4 and AP5, since in the NP1 case, the overlaps are never completely removed. In the AP4 case, due to its lower utilization rate, both methods require only two iterations to eliminate all via overlaps. For AP5, more iterations are needed to remove all overlaps. As stated in Table IV, in logic-on-logic cases, each iteration takes approximately as much time as the Pin-3D PnR section, with both being dominated by the routing stage.

E. Comparing Force-Based and Matching-Based Methods

In general, the matching-based method provides better result quality in terms of displacement, timing, and power. Additionally, the runtime of the matching-based method is significantly lower than that of the force-based method when via legalization is high (>25%). However, in the following scenarios, the force-based method still has advantages.

First, the force-based method is faster when via utilization is low because the runtime of the force-based method depends solely on via count and iterations, while the runtime of the matching-based method also heavily depends on the number of grid points. Case AP4 in Tables X and XII is a good example. It has relatively low via utilization (∼20%), and the force-based method is 3× faster than the matching-based method.

Second, the force-based method is also faster when most of the vias are legalized. In this scenario, the force-based method only needs a few iterations to solve the problem, while the matching-based method still needs to scan through all grid points and vias. The force-based method also moves fewer vias, resulting in less ECO routing, and fewer new vias and overlaps. This is the reason of force-based method needs fewer iterations after post-route opt in Tables XI and XII.

F. Takeaways

For realistic partitioning and 3-D bonding types, our two legalizers can remove all 3-D via spacing violations with only minor degradation in routing quality and PPA. The bipartite matching-based legalizer, along with the refinement, is versatile and compatible with various combinations of 3-D pitch values, bonding styles, and partitioning types. Furthermore, due to its optimality, it provides better solutions than other more straightforward and traditional legalizers, such as force-based legalization or greedy bump assignment, in terms of maximum and average via displacements and the overall PPA. The force-directed-based legalizer, on the other hand, serves as a handy option when only a small portion of vias needs to be legalized. Compared to the bipartite matching algorithm, the force-directed method does not snap vias without overlaps onto the grid, hence fewer vias are moved.

For designs in more advanced nodes, where finer pitch values are required and may not be practical in the near future, a fully integrated routing solution cooperating with the legalizers is still needed to achieve good PPA quality and clean DRCs.

VIII. CONCLUSION

When the 3-D via pitch size has the same or a larger magnitude than the gCell grid, the commercial router fails

to place 3-D via without cut short and spacing violations. To resolve this, we present a legalization flow, which fix all the violations during routing. With our techniques, the results shows better routing quality and fewer DRVs with negligible runtime impact, especially for hybrid-bonded 3-D ICs. For M3D ICs with high via count and utilization, while our proposed approach does offer assistance, having an improved router would provide greater benefits in such scenarios.

REFERENCES

[1] D. B. Ingerly et al., "Foveros: 3D integration and the use of face-to-face chip stacking for logic devices," in *Proc. IEEE Int. Electron Devices Meeting (IEDM)*, 2019, pp. 19.6.1–19.6.4.

[2] "AMD 3D V-cache Ryzen chiplets." 2018. [Online]. Available: https://www.anandtech.com/show/16725

[3] L. Bamberg, A. García-Ortiz, L. Zhu, S. Pentapati, D. E. Shim, and S. Kyu Lim, "Macro-3D: A physical design methodology for face-to-face-stacked heterogeneous 3D ICs," in *Proc. Design, Autom. Test Eur. Conf. Exhibit.*, 2020, pp. 37–42.

[4] S. S. K. Pentapati, K. Chang, V. Gerousis, R. Sengupta, and S. K. Lim, "Pin-3D: A physical synthesis and post-layout optimization flow for heterogeneous monolithic 3D ICs," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Design (ICCAD)*, 2020, pp. 1–9.

[5] E. Beyne, "The 3-D interconnect technology landscape," *IEEE Design Test*, vol. 33, no. 3, pp. 8–20, Jun. 2016.

[6] I. Jani et al., "Characterization of fine pitch hybrid bonding pads using electrical misalignment test vehicle," in *Proc. IEEE 69th Electron. Compon. Technol. Conf. (ECTC)*, 2019, pp. 1926–1932.

[7] D. W. Fisher et al., "Face to face hybrid wafer bonding for fine pitch applications," in *Proc. IEEE 70th Electron. Compon. Technol. Conf. (ECTC)*, 2020, pp. 595–600.

[8] S. Pentapati, A. Agnesina, M. Brunion, Y.-H. Huang, and S. K. Lim, "On legalization of die bonding bumps and pads for 3D ICS," in *Proc. Int. Symp. Phys. Design*, 2023, pp. 62–70.

[9] B. W. Ku, K. Chang, and S. K. Lim, "Compact-2D: A physical design methodology to build two-tier gate-level 3D ICs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 6, pp. 1151–1164, Jun. 2020.

[10] H. Eisenmann, "Force-directed placement of VLSI circuits," in *Proc. Symp. Int. Symp. Phys. Design*, 2015, pp. 131–132.

[11] J. Lu, H. Zhuang, I. Kang, P. Chen, and C. K. Cheng, "EPlace-3D: Electrostatics based placement for 3D-ICs," in *Proc. Int. Symp. Phys. Design*, 2016, pp. 11–18.

[12] B. Goplen and S. Sapatnekar, "Efficient thermal placement of standard cells in 3D ICs using a force directed approach," in *Proc. Int. Conf. Computer Aided Design*, 2003, pp. 86–89.

[13] Z.-W. Jiang, T.-C. Cheny, T.-C. Hsuy, H.-C. Chenz, and Y.-W. Changyz, "NTUplace2: A hybrid placer using partitioning and analytical techniques," in *Proc. Int. Symp. Phys. Design*, 2006, pp. 215–217.

[14] D. Z. Pan, B. Halpin, and H. Ren, "Timing-driven placement," in *Handbook of Algorithms for Physical Design Automation*. Boca Raton, FL, USA: CRC Press, 2008.

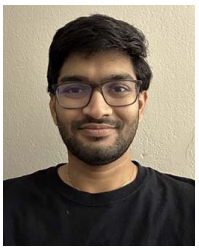
[15] R. Jonker and A. Volgenant, "A shortest augmenting path algorithm for dense and sparse linear assignment problems," *Computing*, vol. 38, no. 4, pp. 325–340, Mar. 1987.

[16] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian optimization of machine learning algorithms," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1–12.

[17] F. Nogueira. "Bayesian optimization: Open source constrained global optimization tool for Python." 2014. [Online]. Available: https://github.com/bayesian-optimization/BayesianOptimization

Yen-Hsiang Huang received the bachelor's degree in electrical engineering from National Taiwan University, Taipei, Taiwan, in 2022. He is currently pursuing the Ph.D. degree in electrical and computer engineering with the Georgia Institute of Technology, Atlanta, GA, USA.

He is presently a Member of the Georgia Tech Computer-Aided Design Laboratory, under the mentorship of Prof. S. K. Lim. His research interests focus on the physical design of 3D ICs.



Sai Pentapati received the B.Tech. degree from the Indian Institute of Technology at Kharagpur, Kharagpur, India, in 2017, and the M.Sc. and Ph.D. degrees from the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA, in 2022.

His main research interests include physical design of 3-D ICs and design technology co-optimization.

Anthony Agnesina received the M.S. and Ph.D. degrees in electrical and computer engineering from the Georgia Institute of Technology, Atlanta, GA, USA, in 2017 and 2022, respectively, and the Diplôme d'Ingénieur degree from CentraleSupélec, Rennes, France, in 2016.

He joined NVIDIA Research, Austin, TX, USA, in 2022. His research interests include 3-D integrated circuits, applied machine learning to EDA, and algorithms for computer-aided design of VLSI circuits.

Moritz Brunion received the bachelor's and master's degrees in electrical and computer engineering from the University of Bremen, Bremen, Germany, in 2019 and 2022, respectively.

He is currently a Researcher with IMEC, Leuven, Belgium. His current research focuses on technology-driven interconnect architecture design.

Sung Kyu Lim (Fellow, IEEE) received the B.S., M.S., and Ph.D. degrees from the Computer Science Department, University of California at Los Angeles, Los Angeles, CA, USA, in 1994, 1997, and 2000, respectively.

He joined the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA, in 2001, where he is a Professor. His research focus is on the architecture, design, test, and electronic design automation solutions for 2.5-D and 3-D ICs. His research is featured as Research Highlight in the Communication of the ACM in January 2014. He is the author of *Practical Problems in VLSI Physical Design Automation* (Springer, 2008) and *Design for High Performance, Low Power, and Reliable 3-D Integrated Circuits* (Springer, 2013). He has published more than 400 papers on 2.5-D and 3-D ICs. He joined the Defense Advanced Research Projects Agency in 2022 as a Program Manager with the Microsystems Technology Office.

Dr. Lim received the National Science Foundation Faculty Early Career Development (CAREER) Award in 2006. He received the ACM SIGDA Distinguished Service Award in 2008. He received the Best Paper Award from the IEEE TRANSACTIONS ON ELECTROMAGNETIC COMPATIBILITY and the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS in 2022. He received the Best Paper Award from several conferences in EDA, including ACM Design Automation Conference in 2023.