

Invited Paper: Toward Unified Open-source Implementation Platforms for 3D Integration Paradigms

Osama Hammad
Precision Innovations
osama@precisioninno.com

Junyeong Jang
UC San Diego
juj015@ucsd.edu

Arthur Koucher
Precision Innovations
arthurkoucher@precisioninno.com

Jingzheng Li
Fudan University
jingzhengli26@m.fudan.edu.cn

Matt Liberty
Precision Innovations
mliberty@precisioninno.com

Cho Moon
Precision Innovations
cmoon@precisioninno.com

Ioannis Savidis
Drexel University
is338@drexel.edu

Deepashree Sengupta
Precision Innovations
dsengupta@precisioninno.com

Yu Shao
ShanghaiTech University
shaoyu2025@shanghaitech.edu.cn

Zhiang Wang*
Fudan University
zhiangwang@fudan.edu.cn

Zehao Yang
Shanghai Jiao Tong University
yangzehao@sjtu.edu.cn

Zhiyu Zheng
Fudan University
zyzheng24@m.fudan.edu.cn

Abstract

Three-dimensional (3D) integration is evolving along different technology paths, from chiplet- and interposer-based systems to fine-grained logic-folding face-to-face (F2F) integrated circuits (ICs). These two paradigms introduce different physical design problems and therefore require different implementation flows. This paper describes open-source implementation efforts for both. For chiplet-based designs, we develop a 3Dblox-based flow that supports native multi-die representation, cross-die analysis, chiplet partitioning, bump escape routing and programmable-interposer routing. For logic-folding F2F designs, we describe ongoing research efforts toward an integrated native F2F 3D implementation flow spanning three-dimensional-aware synthesis, tier partitioning and global placement, hybrid bonding terminal (HBT) assignment, timing-driven detailed placement, clock tree synthesis, and native three-dimensional global routing. Together, these efforts illustrate how the OpenROAD infrastructure can support physical implementation across different forms of three-dimensional integration. We further discuss the remaining challenges toward practical design closure. Our goal is to move from fragmented solutions toward a coherent and extensible foundation for 3D IC and advanced packaging design.

CCS Concepts

• Hardware → Electronic design automation.

Keywords

3D integration, physical design, OpenROAD

ACM Reference Format:

Osama Hammad, Junyeong Jang, Arthur Koucher, Jingzheng Li, Matt Liberty, Cho Moon, Ioannis Savidis, Deepashree Sengupta, Yu Shao, Zhiang Wang, Zehao Yang, and Zhiyu Zheng. 2026. Invited Paper: Toward Unified Open-source Implementation Platforms for 3D Integration Paradigms. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD '26)*, November 08–12, 2026, San Jose, CA, USA. ACM, New York, NY, USA, 9 pages. <https://doi.org/10.1145/3831252.3845863>

*Authors are listed alphabetically by family name. Zhiang Wang is the corresponding author.



This work is licensed under a Creative Commons Attribution 4.0 International License. ICCAD '26, San Jose, CA, USA
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2873-0/2026/11
<https://doi.org/10.1145/3831252.3845863>

1 Introduction

As conventional transistor scaling becomes increasingly difficult and expensive, three-dimensional (3D) integration has emerged as an important path for continued system scaling. The transition is already visible in commercial processors and accelerators. AMD has widely adopted chiplet-based integration and advanced packaging in its CPUs and accelerators, e.g., the Instinct MI300 series [2] combines multiple compute chiplets, I/O dies, and HBM through advanced 2.5D and 3D integration. Intel has developed a broad packaging portfolio based on EMIB [24] and Foveros [25], ranging from side-by-side heterogeneous integration to vertically stacked dies with fine-pitch hybrid bonding. These technologies extend scaling beyond the boundary of a single monolithic die by integrating separately manufactured dies into a larger system.

A different direction is to move vertical integration deeper into the logic itself. Huawei recently introduced the τ -law [18, 22] and LogicFolding as an alternative path for continued semiconductor scaling. Rather than relying only on geometric transistor scaling, LogicFolding seeks to break the physical boundary of planar implementation and reduce signal propagation delay by folding logic into the vertical dimension. This represents a substantially different use of 3D integration. The design unit is no longer only a pre-designed chiplet. Logic cells and timing paths may be distributed across tightly coupled active dies, making vertical interconnections part of the physical implementation problem itself.

The distinction has direct implications for EDA. A chiplet-based system requires a design representation that can describe multiple dies, different process technologies, package structures, bonding interfaces, and die-to-die connectivity. The implementation flow must operate above the boundary of an individual die while preserving the physical information of each component die. In contrast, a face-to-face (F2F) 3D integrated circuit (IC) requires the physical design flow itself to become three-dimensional. Logic synthesis must understand the technology assigned to each tier. Partitioning and placement must account for vertical interconnections. Hybrid bonding terminal (HBT) [30] assignment directly affects timing and routing. Clock and signal networks must be constructed across both dies. A conventional 2D flow cannot address these two design regimes through the same set of abstractions and optimization steps.

This paper describes open-source implementation efforts for these two representative 3D integration paradigms. (i) For chiplet and advanced-packaging systems, we build a 3Dblox-based implementation flow on OpenROAD [1, 7, 23, 51]. We extend the OpenROAD database (ODB) to natively represent chips, chip instances,

heterogeneous technologies, bonding regions, bumps, physical connections and cross-die nets. The hierarchical database preserves the structure of the multi-die assembly, while an unfolded view provides global coordinates for full-system analysis. Bidirectional 3Dblox support provides a common interface between the database and external design tools. Based on this representation, the flow supports 3D visualization and structural checking, cross-die parasitic extraction and static timing analysis, and bonding-yield analysis. We further extend the flow to physical design applications, including cost- and thermal-aware chiplet partitioning, performance-driven bump escape routing, and routing on programmable silicon interposers. (ii) For fine-pitch F2F 3D ICs, we describe ongoing research efforts toward a native physical implementation flow in which vertical integration is considered throughout the design process. The envisioned flow starts from 3D-aware logic synthesis and continues through tier partitioning, global placement, HBT assignment, timing-driven detailed placement, clock tree synthesis, and native 3D global routing. Three-dimensional effects are introduced at the stage where they become relevant. Synthesis considers heterogeneous technology libraries and cross-tier interconnect parasitics. Partitioning considers placement geometry and HBT demand. Detailed placement and HBT assignment are refined using cross-tier timing feedback. Clock tree synthesis explicitly constructs clock delivery across the bonding interface. Global routing jointly uses routing resources from both dies and selects legal HBT locations as part of the routing solution. The objective is a physical design flow that treats the two tiers as one coupled implementation problem rather than two independently optimized 2D designs.

The two flows therefore address complementary levels of 3D integration. The 3Dblox flow targets *die-level integration*, where heterogeneous dies and package components are assembled and optimized as a system. The F2F flow targets *logic-level integration*, where the circuit itself is partitioned and implemented across vertically bonded active dies. This distinction is central to our view of future 3D EDA. Different 3D integration paradigms expose different design variables, physical constraints and optimization opportunities. They require implementation flows that match their integration granularity instead of forcing all 3D systems into a single extension of the conventional 2D flow. The main contributions of this paper are as follows.

- We present an open-source 3Dblox-based implementation flow for chiplet and advanced-packaging systems. The source code of 3Dblox-based ODB is available at [54]. The flow provides a native multi-die database, cross-die analysis, chiplet partitioning, bump escape routing and programmable-interposer routing.
- We describe ongoing research efforts toward an open-source F2F 3D IC implementation flow for fine-grained LogicFolding integration. The source code of the F2F 3D IC implementation flow will be available at [11]. The described techniques span 3D-aware synthesis, tier partitioning and placement, HBT optimization, clock tree synthesis and native 3D global routing.
- Based on these two flows, we identify key challenges toward practical 3D design closure, including native database support for logic folding, cross-die analysis and signoff, the impact of hybrid bonding yield and variation on physical design, and multi-physics effects.

The techniques described in Sections 3.1–3.6 represent ongoing research efforts toward native F2F 3D implementation and are intended for future availability through OpenROAD-Research [55].

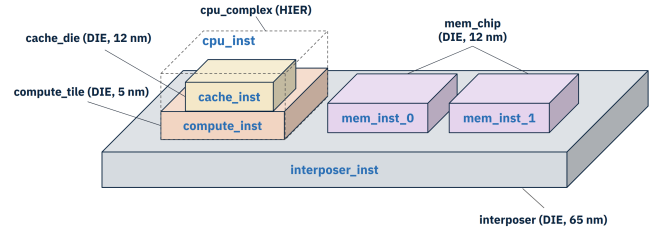


Figure 1: A five-die assembly spanning 3 PDKs.

Together, the efforts we describe exemplify the transition from isolated 3D design algorithms toward implementation flows tailored to different forms of 3D integration. As 3D integration continues to evolve from package-level chiplet composition toward increasingly fine-grained vertical logic integration, scalable and open implementation infrastructures will be essential for exploring the corresponding design and technology tradeoffs.

2 3Dblox-based Open-Source Implementation Infrastructure for Chiplet-based Designs

Chiplet-based integration moves physical design beyond the boundary of a single die. A system may contain dies from different technology nodes, multiple instances of the same die, several bonding interfaces, and an interposer or package substrate that connects them. The implementation flow must preserve this structure while allowing system-level analysis and optimization. Flattening all components into a conventional single-die database is not sufficient. It either mixes different technology stacks or loses the hierarchy and assembly information needed for package-level design.

To support this design paradigm, we build a 3Dblox-based implementation flow in OpenROAD. The flow has two parts. First, we extend ODB with a native multi-die data model that represents the complete assembly without modifying the underlying 2D representation of each die. Second, we build analysis and optimization capabilities on top of this representation. These capabilities range from cross-die timing and parasitic analysis to chiplet partitioning, bump escape routing, and programmable-interposer routing. The same 3Dblox design description is used throughout the flow.

2.1 A 3D Data Model in ODB

Chiplet-based integration requires a database that preserves heterogeneous dies while representing the complete assembly. We extend ODB with chip-level hierarchy, bonding interfaces, cross-die connectivity and an unfolded view for system-level analysis.

What the representation must express. A representation adequate for chiplet assemblies must simultaneously express dies fabricated in different process nodes, the same die placed more than once, hierarchical grouping of dies, the surfaces on which dies bond, and nets that leave one die and enter another.

Figure 1 illustrates every one of these required aspects. A compute tile in a 5 nm node and a cache die in a 12 nm node are grouped into `cpu_complex`, a chip with no layout of its own; a single 12 nm memory die is placed twice; and a 65 nm interposer carries everything. Note that a single-die database has two options, each of which loses information. Duplicating the twice-placed memory die copies a bump map that at hybrid-bonding pitch can hold hundreds of thousands of entries; aliasing the two placements loses the per-placement bump positions that bonding checks depend on.

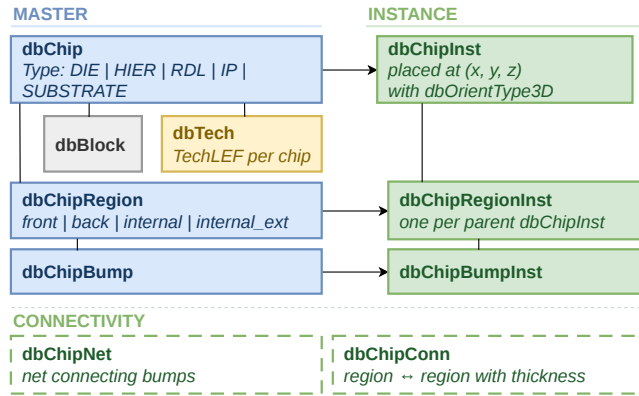


Figure 2: Overview of 3D Data Model.

Chip masters and chip instances. ODB already separates a definition from its placements: a dbMaster describes a cell and a dbInst places it. The 3D extension applies that same separation one level up, to the chip itself. Figure 2 shows the result: every master class (chip, region, and bump) gains an instance counterpart, while a chip’s technology and layout hang off the master. A dbChip is a chip master: a die, a redistribution layer, a hard IP, a package substrate, or a purely hierarchical grouping. A dbChipInst places a chip inside a hierarchical chip at a location and a 3D orientation. Because a master is stored once however often it is placed, the memory die’s layout exists once in the database and is placed twice.

Technology is attached to the chip, not to the design. Each die chip references a dbTech and owns a dbBlock, the ordinary 2D layout of that die; a dbTech may be referenced by any number of chips. The assembly’s four die masters reference only three distinct dbTechs, because cache_die and mem_chip are both 12 nm. The hierarchical chips, cpu_complex and the design top, own neither block nor technology; they exist only to place other chips.

Bonding interfaces. A dbChipRegion is an area on a chip surface (front, back, or internal) where that chip mates with another, and a dbChipBump is one bump within a region. A bump introduces no new geometric primitive: it wraps a dbInst already placed in the chip’s block, so the bump cell is a real instance with a master, pins, and a net terminating inside the die. This is where the chip-level and block-level models meet, and it is what lets cross-die extraction and timing continue into each die rather than stopping at its boundary.

Regions and bumps follow the same master/instance split as chips. Both are declared once on the master; creating a dbChipInst materialises a region instance for every region and a bump instance for every bump. Only the per-placement objects are duplicated, so placing a die with a large bump map a second time costs one small object per bump rather than a second copy of the map.

Physical bonding and logical connectivity are separate. Two objects tie the assembly together, and the model deliberately keeps them apart. A dbChipConn is the physical bond: it mates two whole region instances across a gap of a given thickness, and so covers every bump pair on that interface at once. A dbChipNet is the logical net: it spans a set of bump instances across chips, which may be a single pair or, for a supply, a large fraction of the interface. Neither object refers to the other. Figure 3 shows one such bond and a cross-die net inside cpu_complex, where cache_inst is placed face-down so that its front-side to_compute region mates with compute_inst’s front-side to_cache region.

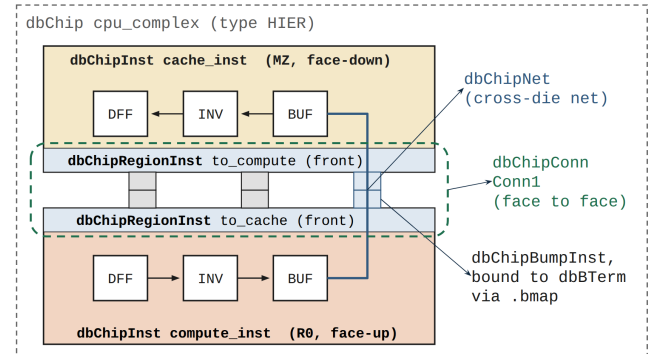


Figure 3: A face-to-face bond inside cpu_complex, and a net crossing it.

Separating them is what allows the two to be checked against each other: bump instances that coincide physically across a bond but belong to different chip nets are mis-wired, which is one of the checks of Section 2.2.1. Every object described so far is a native ODB object and persists in the standard .odb file, so an assembly survives a write/read cycle intact.

The unfolded view. The shared-master tree is the right structure to edit and to store, and the wrong one for analysis, which needs every physical die at an absolute position. The database therefore derives a second, flattened view: it walks the tree accumulating transforms and emits one object per physical occurrence of each chip, region, bump, connection, and net, in global coordinates. For the assembly of Figure 1, unfolding yields five leaf chips: cpu_complex disappears except as a path segment, and mem_inst_0 and mem_inst_1 appear as two objects at different absolute positions.

Unfolding also resolves what the folded model only implies. A region declared on a die’s front face reports an effective side of *bottom* once that die is placed flipped, as cache_inst is in Figure 3. The linter’s overlap and alignment checks, the 3D viewer, and cross-die extraction and timing all read this view. The view is strictly derived, a projection of the folded model rather than a second source of truth, so the two cannot drift apart.

Design interchange. The model is populated primarily from the 3Dblox format: OpenROAD reads chiplet definitions (.3dbv), the design and its stack (.3dbx), and bump maps (.bmap). Because a chiplet definition references technology and layout collateral, reading one also drives the ordinary LEF and DEF readers, so a single command populates the chip-level model and the per-die blocks it refers to; system-level connectivity arrives from the Verilog netlist named by the .3dbx. The path is bidirectional: OpenROAD also emits .3dbv, .3dbx, and a chip-level Verilog netlist, so it can act as a producer of chiplet collateral as well as a consumer. Class-level reference, format mapping tables, and the full command set are documented with the source [42].

2.2 3Dblox Application

The multi-die database is useful only if physical-design tools can operate directly on it. We therefore extend the 3Dblox flow with system-level analysis and optimization capabilities. The first group of applications validates and analyzes the assembled system. The second group uses the same representation to make physical-design decisions at the chiplet and package levels.

2.2.1 3D viewer and linter. The OpenROAD GUI has been extended to display a 3D assembly and to report violations found by a 3D linter. Figure 4 shows the viewer alongside the DRC viewer panel

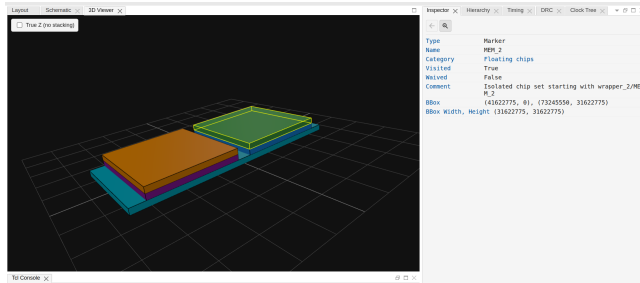


Figure 4: 3D Viewer in the OpenROAD GUI, with connectivity violations flagged by the 3Dblox linter (highlighted in yellow).

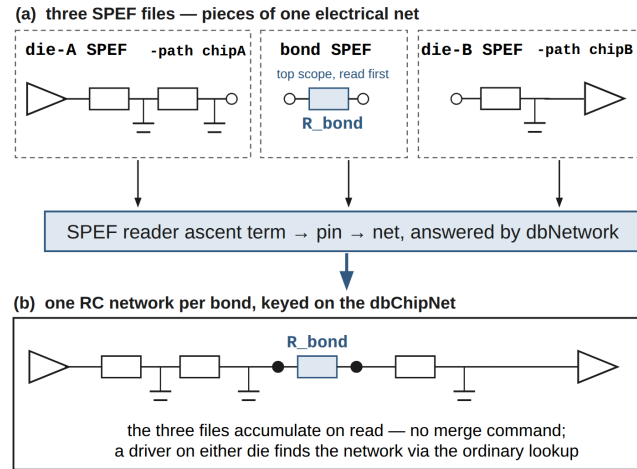


Figure 5: (a) Three disjoint SPEF files, from which (b) parasitic stitching produces one connected network for a cross-die net.

listing violations by type. A 3D linter runs over the unfolded view. Every check is a global-space geometric or connectivity query. It verifies that bonded regions actually face each other and overlap in all three axes, that bumps fall within their parent regions, that no two chips intersect, and that no chip is left floating (a union-find over valid connections, seeded from connections to the outside world). The most characteristic check is the cross-check that the physical/logical split enables: bumps that coincide physically across a bond must carry the same chip net, otherwise the stack is silently mis-wired. Violations are emitted as ordinary dbMarkers under a dedicated marker category, so 3D errors flow through the same reporting and GUI machinery as DRC results, at no additional cost. Complementing this, OpenROAD’s browser-based GUI provides a WebGL stack viewer that renders the chiplets as cuboids at their computed positions, with an optional de-stacked presentation for visual separation, and links a picked violation back to the corresponding marker in the DRC widget.

2.2.2 3D Parasitics. Parasitic extraction for 3D assemblies must account for interconnects that span multiple die technologies and the bonding interface. We extend OpenRCX [43] to extract each technology domain separately and stitch the resulting parasitics into a single electrical network for each cross-die net.

Extraction across a bond. OpenROAD’s parasitics extractor has been extended to an assembly of two face-to-face dies, the first configuration supported. The pass runs one extraction per die and one across the physical bond between them. The per-die extraction is the ordinary one: it walks the non-special signal nets of a die’s

block, its dbNets, and computes the resistance and capacitance of their metal shapes. The bond extraction walks the dbChipNets and computes the resistance between the two bumps of each pair. The two are separate passes because the bond belongs to neither die’s process: its dielectrics, conductors, and per-via resistances come from rules of their own. The pass is restricted to a single process corner, and the bond extraction assumes a negligible capacitive contribution from the bumps; multi-corner extraction, modelling of ground and coupling capacitance, and upstreaming of the pass itself are in progress.

Extraction rules. Extraction is driven by a rules file (for each process or assembly kit) that describes how to measure a wire’s parasitics from its physical surroundings. An assembly thus requires one rules file per distinct die technology, plus one for the bond.

Chip-level storage. Bond parasitics are stored at the chip level, mirroring the per-block model one level up: a dbChipCapNode is a capacitive node representing one side of a bump pair, and a dbChipRSeg is a resistive segment joining two such nodes, exactly as dbCapNode and dbRSeg do inside a die. Each capacitive node binds to a bump instance and its boundary terminal, which is where the chip-level network meets the die-internal one.

SPEF in and out. The writer emits one SPEF file per die and one for the bond. On the way back in, the files are read independently and accumulate into a single connected network per cross-die net, with no explicit merge step: a driver on either die reaches the whole network through the ordinary lookup (Figure 5). Each file therefore remains a valid SPEF in its own right, and a die’s parasitics can be reused unchanged in another assembly.

Toward physics-aware extraction. The above-described architecture establishes the important separation between die technology and assembly technology. The natural next step is to make the assembly pass *geometry-aware* rather than only rule-based. Geometry-aware closed-form models in [47, 48] enable evaluation within an iterative implementation loop; they compute resistance, capacitance, coupling capacitance, self inductance, and mutual inductance from physical parameters including via length, diameter, liner thickness, conductor and substrate properties, pitch to neighboring vertical interconnects, and spacing to a reference plane. Validations against 3D electromagnetic simulation have demonstrated < 8% maximum error for resistance, capacitance and inductance across the examined design space. Initial integration of [47, 48] uses resistance and grounded/coupling capacitances to construct the stitched RC network spanning die-internal and cross-die interconnects, consistent with the present RC-centric parasitic storage and timing flow. Adding native inductive objects will bring L_{self} and L_{mutual} models [47, 48] into future PDN, signal-integrity, clock, and high-speed interconnect analyses without changing the overall 3D data model.

2.2.3 3D Static Timing Analysis. Timing analysis needs one timing graph spanning the dies of an assembly, and we build it without flattening the database. OpenSTA [10], the timing engine, never touches the database directly: it reaches the design exclusively through an abstract network interface that answers structural questions such as “what are the children of this instance?” and “which pins does this net connect?” A small number of methods in OpenROAD’s implementation of that interface, dbNetwork, have been made aware of the 3D data model of Section 2.1, so that chip instances, cross-die nets, and bumps are presented to the timer as ordinary instances, nets, and pins, and a traversal that reaches a bump continues into the die on the other side of the bond. The timing engine is unmodified — levelization, arrival-time propagation,

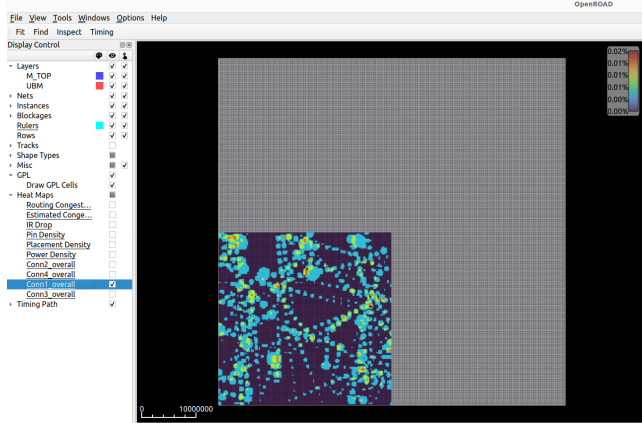


Figure 6: Bonding Yield Analysis based on UCLA's YAP engine.

and every reporting command work as in 2D because none of them know they are crossing a die boundary. This is the core claim of the approach: 3D timing is a network question, not a timer question.

Figure 3 shows a timing path crossing a die boundary. `compute_inst` and `cache_inst` are bonded face to face, with `cache_inst` flipped, and each carries a DFF-INV-BUF chain. `report_checks` reports the launch-to-capture path directly, starting at the flip-flop in `compute_inst`, crossing the bond through a bump pair, and ending at the flip-flop in `cache_inst`.

Without annotated parasitics the bump pads are zero-delay junctions — a zero-parasitic baseline in which the bond contributes no delay and appears as no timing node. Reading back the per-die and bond SPEF files of Section 2.2.2 closes that gap: the pieces accumulate into one RC network per cross-die net, so the bond resistance and the far die's wiring and loading enter delay calculation, and the same reports produce parasitic-annotated cross-die timing. This capability — cross-die `report_checks` with and without annotated parasitics, along with the accompanying regression tests — is available in OpenROAD.

2.2.4 Bonding Yield Analysis. Bonding is a yield-limiting step in 3D assembly, and its failure modes depend on the physical arrangement of bumps that the data model now captures. YAP, a yield analysis tool developed at UCLA, has been integrated into OpenROAD through a file-based interface. Bond failure can be analyzed for an assembly and the resulting risk reported per region. The results can be displayed in the GUI as a heat map alongside the other heat maps OpenROAD already supports. Figure 6 shows an overall-risk heat map loaded into the layout view.

2.2.5 Chiplet Partitioning. Chiplet-based partitioning determines the cost-effectiveness of 2.5D systems by defining how a large design is disaggregated into smaller chiplets. However, achieving cost-optimal partitions requires navigating complex tradeoffs among die yield, packaging and inter-chiplet communication overhead. ChipletPart [15] is a cost-driven 2.5D system partitioner that addresses these challenges through a unified optimization framework. The cost engine evaluates each candidate partition using the CATCH chiplet cost model [16]. A genetic algorithm drives the search over technology assignments and partitioning solutions, while a simulated annealing-based chiplet floorplanner ensures every candidate solution is floorplan-feasible.

Cost-driven partitioning alone may overlook thermal effects. We therefore extend ChipletPart with thermal-aware optimization.

Thermal-ChipletPart adds a thermal penalty to the cost objective and uses DeepOHeat [40] for fast temperature prediction. DeepOHeat is an operator-learning-based surrogate for steady-state thermal analysis. Its low inference cost allows temperature feedback to be evaluated directly during partitioning. Thermal-ChipletPart is integrated into the OpenROAD flow through the ODB database. The 3Dblox input files describe the chiplets, interposer, connectivity and related technology information. The partitioner reads these files and performs cost- and thermal-aware optimization. The final partition is written back to ODB database, allowing downstream stages such as bump assignment and chiplet routing to operate on the same design.

2.2.6 Bump Escape Routing. Bump escape routing assigns paths from dense bump arrays to the interposer or redistribution-layer routing fabric. At fine bump pitches, routing resources around the bump array are constrained, and neighboring connections compete for the same escape channels. The escape order therefore has a direct impact on routability, wirelength and timing, especially for performance-critical die-to-die connections.

We integrate the performance-driven escape routing method [8] into OpenROAD. Under monotonic single-layer routing, the escape order determines the legal routing topology through a set of ordering constraints. The routing problem can therefore be formulated as an optimization over escape sequences rather than independent net-by-net routing. The method first generates a high-quality initial ordering using dynamic programming, then explores alternative orderings with a pruned breadth-first search. Once the final order is selected, detailed routing realizes the corresponding paths and further reduces wirelength through local refinement.

The router operates directly on the ODB representation of the assembled design. Bump locations, bonding regions and net connectivity are obtained from the chip-level database objects, and the routed wires are written back into the same design state. This allows bump escape routing to remain consistent with assembly checking and timing analysis. On testcases with up to 1,024 nets per die, the router completes all connections without crossings in less than two seconds.

2.2.7 SAT-Based Programmable Interposer Routing. Programmable silicon interposers introduce a different routing problem from conventional package substrates. Routes are constrained to discrete tracks and programmable switches, and multiple chiplet connections compete for the same fabric resources [28, 33]. The resulting routing problem is therefore highly coupled: a locally feasible path for one connection can block several others and make the remaining design unroutable.

We integrate ExactSAT into OpenROAD to support routing on programmable interposers. The routing fabric is represented as a directed graph whose vertices and edges capture routing tracks, chiplet bumps, external ports and programmable switches. ExactSAT formulates the joint routing problem using Boolean variables and constraints [14, 41], allowing all connections to be considered simultaneously rather than routed sequentially. To improve scalability, the solver starts from restricted routing domains and expands them only when necessary. After a feasible solution is found, selected long routes are further refined using an ILP-based multi-commodity-flow formulation [50].

The router operates on the ODB representation of the assembled system and writes the final routing configuration back to the design flow. The selected routes are translated into programmable switch

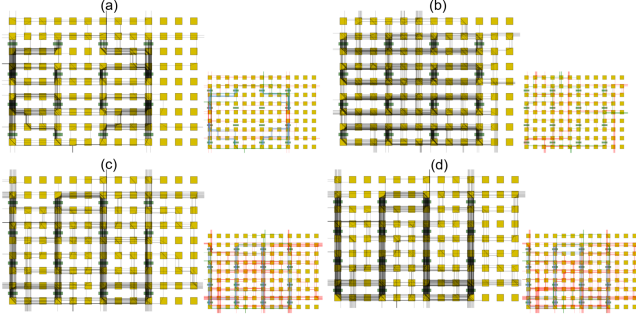


Figure 7: Routing results for the four cases unroutable by the original router. Yellow squares and green rectangles denote structures with programmable switches. Larger panels show ExactSAT solutions. Smaller panels show the nets left unrouted by the original router.

states and exported as a controlbits file for configuring the interposer fabric. Across 13 testcases with 68 to 1,010 source–sink connections, ExactSAT routes all cases, including four that cannot be completed by the original sequential maze router [28]. For cases solved by both methods, ExactSAT reduces wirelength by up to 8.3%, with an average reduction of 3.1% after excluding two cases that already achieve the shortest wirelength.

3 Integrated F2F 3D IC Implementation flow

Previous works, such as TaiWei [11], establish pseudo-3D methodologies for F2F 3D IC implementation by embedding the two tiers into a unified physical design abstraction. This enables mature 2D engines to be reused for end-to-end implementation. However, tier assignment, HBT assignment and cross-tier timing effects are still largely handled through the pseudo-3D abstraction rather than being modeled directly inside the optimization algorithms.

In this work, we describe ongoing work toward a *true 3D implementation flow* in which the third dimension is explicitly modeled in the design variables, objectives and physical constraints. Synthesis (Section 3.1) considers tier-dependent technologies and HBT parasitics before final technology mapping. Partitioning and global placement (Section 3.2) directly optimize tier assignment, 3D wirelength and HBT demand. Detailed placement (Section 3.4) jointly refines cell locations and HBT assignments (Section 3.3) using cross-tier timing feedback. Clock tree synthesis (Section 3.5) optimizes clock delivery across both tiers. Global routing (Section 3.6) directly explores routing resources on both dies and selects legal HBT locations during routing. Together, these stages describe a path for moving F2F implementation from pseudo-3D modeling toward native optimization of the underlying 3D physical design problem.

3.1 Tier-aware 3D Synthesis

This section describes preliminary work toward a 3D synthesis engine for the F2F 3D IC implementation flow. Existing 3D IC flows typically perform logic synthesis before tier partitioning [4, 5, 45, 49]. When different tiers use different technology nodes, synthesis is still performed using only a single technology library. Moreover, synthesis cannot identify cross-tier nets at this stage and therefore cannot consider the additional delay introduced by vertical interconnections. As a result, the synthesized gate-level netlist may not match the technology characteristics of its assigned tiers, while the criticality of cross-tier timing paths may be underestimated. To address these limitations, we describe 3D-Syn,

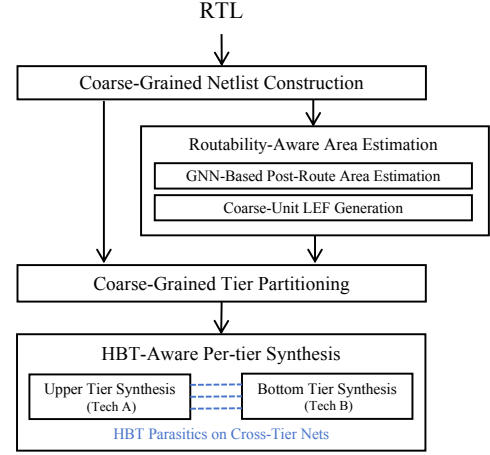


Figure 8: Overview of the 3D-Syn flow.

a 3D-aware synthesis flow that incorporates 3D implementation context into logic synthesis. Figure 8 shows the 3D-Syn flow.

3D-Syn first constructs a coarse-grained netlist to enable tier partitioning before final synthesis. It elaborates the RTL design into the Yosys [52] RTLIL representation and identifies arithmetic units and multi-bit registers as coarse-grained units. These tightly coupled logic structures are preserved, while the remaining logic is mapped into gates. This postpones the final technology mapping of the preserved units and prevents them from being split across tiers during partitioning. To provide the area of each preserved unit for coarse-grained tier partitioning, 3D-Syn uses a graph neural network (GNN)-based model to estimate its technology-dependent post-route area. The estimated post-route area accounts for the routing space required by each unit and is used for tier partitioning.

After tier partitioning (see Section 3.2 for details), 3D-Syn performs final synthesis using the obtained tier assignment and cross-tier connectivity. It constructs a hierarchical netlist with one submodule for each tier, while keeping cross-tier nets visible at the top level so that complete cross-tier timing paths can be analyzed during synthesis. 3D-Syn then assigns the resistance and capacitance of HBTs to all cross-tier nets [3, 32]. For tier-specific synthesis, the upper-tier submodule is synthesized using its corresponding technology library while the bottom-tier submodule is kept frozen. The bottom-tier submodule is subsequently synthesized using its corresponding technology library while the optimized upper-tier submodule is kept frozen. The resulting heterogeneous gate-level netlist is passed to the subsequent 3D physical design stages.

Early evaluations compare 3D-Syn against the conventional synthesis-first flow used in TaiWei using post-route metrics. In the heterogeneous ASAP7–Nangate45 stack, 3D-Syn reduces worst negative slack (WNS) and total negative slack (TNS) by 56.0% and 79.7% on average, respectively. Under the Nangate45–Nangate45 stack, corresponding reductions are 41.5% and 50.7%. These early results suggest that exposing tier-specific technologies and cross-tier interconnect parasitics during synthesis can produce netlists that are better aligned with downstream 3D physical implementation.

3.2 3D Partitioning and Global Placement

As described in the previous subsection, 3D-Syn invokes tier partitioning before final technology mapping. This makes tier assignment a key link between synthesis and physical design. Tier assignment determines which nets cross the bonding interface,

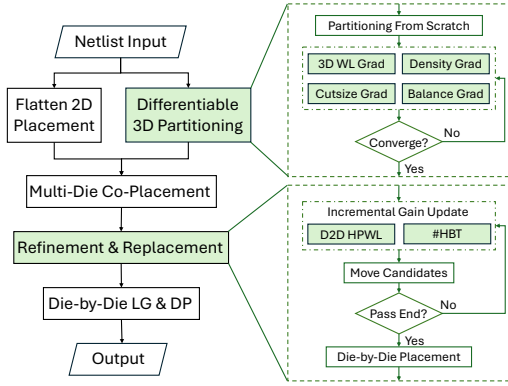


Figure 9: Overview of the 3D partitioning and global placement framework.

while the physical placement of cells determines the wirelength, density distribution and HBT demand created by those crossings. These two decisions are therefore tightly coupled. Conventional partitioning methods rely on min-cut heuristics such as hMETIS [29] and TritonPart [6], which minimize the number of cut nets without considering cell locations or the heterogeneous technology stacks of the different tiers. Moreover, partitioning and placement are typically performed in isolation: the partitioner cannot foresee how its tier assignment affects wirelength, density and HBT demand, while the placer cannot adjust the tier assignment to recover from poor partitioning decisions. As a result, the generated tier assignments may introduce long cross-die connections and congest HBT regions, degrading both wirelength and timing.

To address these limitations, we describe ongoing work on a differentiable 3D partitioning and global placement framework that directly optimizes placement-aware tier assignment through gradient-based optimization. The overview of the 3D partitioning and global placement framework is shown in Figure 9. The discrete tier assignment is relaxed into continuous probabilities, enabling placement objectives to be backpropagated end-to-end into the partitioner. A Dual-Max 3D wirelength model is introduced to capture the per-tier half-perimeter wirelength (HPWL) of each net, making the wirelength objective directly sensitive to the tier assignment. In addition, a terminal-aware cutsite penalty selectively suppresses cross-die nets in HBT-congested regions, steering vertical interconnect demand away from congested terminal areas, while a local balance constraint enforces grid-cell density equilibrium across the tiers to prevent placement hotspots from forming in either tier.

3.3 Hybrid Bonding Terminal Assignment

HBTs provide vertical interconnections for inter-tier nets in 3D ICs through high-density hybrid bonding [19, 26]. Unlike conventional vias, HBTs are discrete interconnect resources with fixed dimensions, spacing constraints and alignment requirements. Therefore, HBT locations must be selected from predefined candidates rather than freely generated during routing [39]. Since HBT locations directly affect inter-tier routing topology, RC distribution and interconnect delay, HBT assignment significantly impacts timing optimization. Existing HBT assignment methods mainly optimize routing-related costs [39], but cannot fully capture the timing impact of HBT location variations on critical inter-tier paths.

We describe a timing-driven HBT assignment framework consisting of routing-aware initial matching and STA-driven local reassignment. In the initial stage, following previous HBT assignment

formulations [39], the HBT assignment problem is formulated as a minimum-cost bipartite matching problem between inter-tier nets and legal HBT candidates. A quadtree-based spatial partitioning method is employed to decompose the global matching problem into multiple local subproblems. The routing cost of candidate assignments is estimated based on connection topology, and the Hungarian algorithm is applied to obtain the initial HBT assignment solution. Since HBT relocation may alter Steiner topology and RC distribution, the framework further includes a timing-driven local reassignment strategy based on STA feedback. For timing-critical inter-tier nets, local candidate regions are constructed around current HBT locations, and candidate assignments are evaluated according to their timing impact. A criticality-aware weighting model is included to prioritize nets with worse slack, and constrained bipartite matching is applied to obtain refined HBT assignments. Each reassignment is validated through Steiner topology reconstruction and STA analysis to avoid global timing degradation.

3.4 Timing-driven Detailed Placement

Existing 3D detailed placement methods have two main limitations. (i) HBT assignment is often performed after placement as a separate post-processing step [39]. Once the HBT locations are fixed, the placer cannot adjust them using timing feedback, which may introduce unnecessary detours on critical cross-tier nets. (ii) Many existing flows optimize the two dies independently [9, 37, 53]. This splits a cross-tier timing path into separate die-local segments and weakens the coupling between the two tiers. As a result, local placement improvements on one die may not improve the end-to-end cross-tier path and can lead to suboptimal timing closure.

To address these limitations, we describe a timing-driven 3D detailed placement framework that integrates HBT assignment directly into the placement loop. The framework preserves complete cross-tier timing paths and jointly refines standard-cell locations and HBT assignments. The inputs are a legally placed 3D netlist and a set of HBT candidates arranged on a two-dimensional grid. Static timing analysis is first performed to identify critical paths and timing-critical cells. A Lagrangian-multiplier-based timing-force model then converts timing slack into directional placement forces [17, 36]. These forces define the local search window and candidate locations for each critical cell. For inter-tier nets, HBT locations are updated using the timing-driven reassignment method described in Section 3.3. Cell movement and HBT reassignment are performed iteratively, with timing re-evaluated after each optimization round. The process terminates when no further timing improvement is observed. In preliminary experiments, the framework is integrated into the TaiWei flow and compared with conventional die-by-die optimization. Across the evaluated designs, the described method improves total negative slack by up to 35% over the die-by-die baseline.

3.5 Useful-skew-aware Clock Tree Synthesis

Clock tree synthesis (CTS) in F2F 3D ICs must distribute the clock across both active tiers while accounting for tier-specific sequential elements and HBTs used for inter-tier clock connections. Existing 3D CTS methods mainly focus on balanced clock delivery across the two tiers [27, 34, 45]. The flow described here further incorporates useful-skew optimization to improve timing closure.

As shown in Figure 10, starting from the placed 3D design, the described flow constructs a balanced 3D clock tree with buffers on both tiers and performs propagated-clock timing analysis. Useful-skew targets are then assigned to sequential elements [13, 46].

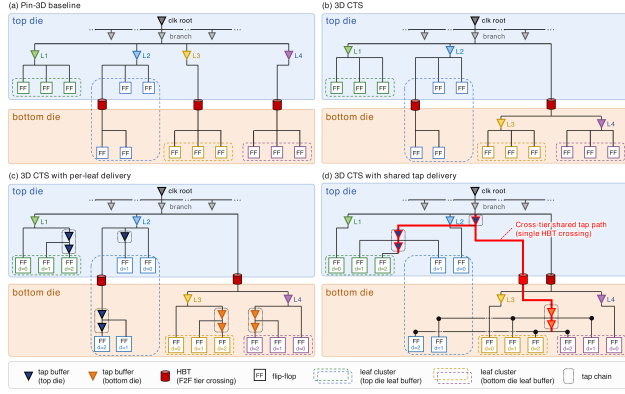


Figure 10: Clock structures for F2F 3D ICs. (a) Pin-3D baseline. (b) Balanced 3D CTS with clock buffers on both tiers. (c) Per leaf delay delivery. (d) Shared tap delivery.

These targets are realized using shared tap chains. Sinks driven by the same clock branch and located on the same tier share a cascaded tap chain while retaining different logical tap depths. This structure reduces redundant buffering and limits the number of HBTs required for inter-tier clock delivery. Timing analysis is then used to refine tap assignments and buffer sizes before routing. Across five designs and two F2F technology stacks, the described CTS flow improves post-route setup TNS by 6.4% to 97.3% over Pin-3D baseline [45], with a median cell-area overhead of 0.7%. In eight of the ten evaluated design–technology combinations, the implemented clock trees realize more than 89% of the requested skew delay.

3.6 Routing with Metal Sharing Awareness

Global routing for F2F 3D ICs must determine the routing topology, the die assignment of each route segment, and the locations of HBTs. Pseudo-3D flows typically merge the routing stacks of the two dies and model inter-tier connections as vias [4, 31]. The resulting crossings are then mapped to legal HBT sites after routing [21]. This simplifies routing, but HBT availability is not explicitly considered when the routing topology is constructed.

F2F integration also enables *metal sharing* [5, 20, 44]. A net whose terminals are located on the same die may use routing resources on the opposite die when local routing resources are congested. The route enters the other die through one HBT and returns through another, allowing the two metal stacks to share routing capacity. The F3DGR effort incorporates this additional routing freedom directly into global routing. It jointly considers routing topology, die assignment, legal HBT locations and routing capacity on both dies. To remain scalable, F3DGR first performs coarse 3D routing planning to determine die assignment, HBT demand, congestion distribution and candidate metal-sharing segments. The planned topology is then refined on the real routing graph, where routing segments, vias and legal HBT locations are determined. In preliminary experiments on ISPD 2025-derived benchmarks [35], F3DGR achieves valid routing with zero HBT-site overflow. Compared with TaiWei [11] and CUGR2.0 followed by HBT legalization [12, 21, 38], F3DGR reduces wirelength by 14.1% and 11.3% and improves runtime by 3.27× and 2.22×, respectively.¹

¹The largest testcase contains 12.05 million nets and the F3DGR is completed in 2.14 hours.

4 Conclusion and Future Work

The flows presented in this paper demonstrate that OpenROAD can support substantial portions of both chiplet-based and LogicFolding 3D IC implementation. By developing dedicated flows for these two integration paradigms, we show that their distinct physical abstractions and design constraints must be addressed explicitly throughout implementation. However, several challenges remain before either paradigm can achieve practical design closure. For chiplet- and interposer-based systems, the major gap lies between system-level planning, package implementation and signoff. Closing this gap requires better physical feedback.

- Early design decisions lack physical feedback. Cost and yield can be estimated early, but accurate routability, parasitic, thermal and power assessment requires detailed implementation. This mismatch makes it difficult to determine whether a chiplet partition is physically viable when the partition is created.
- System-level signoff remains incomplete. Signoff of individual dies does not verify the assembled system. Die-to-die links, interposer routing, package parasitics, power delivery and thermal coupling must eventually be closed together rather than verified as independent components.

For LogicFolding F2F 3D ICs, the challenge is more fundamental: physical design has become inherently three-dimensional, while much of the underlying EDA infrastructure is still built around a two-dimensional design model.

- There is no native physical design database for LogicFolding. Existing databases are organized around a two-dimensional block. LogicFolding instead requires cells, nets, timing paths, HBTs and routing resources to remain part of one design while being distributed across multiple active tiers. Today, tier assignment and cross-tier connectivity are still frequently maintained through external metadata or pseudo-3D abstractions.
- True cross-tier signoff is still missing. A timing path may traverse different cell libraries, BEOL stacks and HBTs. The same problem appears in parasitic extraction, power integrity and physical verification. Running conventional signoff separately on each tier cannot capture these interactions.
- HBT manufacturability is not yet fully captured in current physical design flows. Existing methods typically optimize HBT count and placement using nominal electrical models, while ignoring bonding defects, alignment errors, resistance variation and redundancy requirements. These non-idealities can directly affect timing, routability and the achievable benefits of LogicFolding. Their impact on PPA therefore needs to be considered together with HBT placement and utilization.
- Multi-physics effects should be considered during physical design. Fine-grained stacking couples temperature, power delivery, timing and mechanical stress at the cell level. A placement or tier assignment that is optimal under conventional PPA metrics may no longer be optimal, when vertical thermal coupling, IR drop and bonding-induced stress are included.

Addressing these challenges will be essential for moving both integration paradigms from research prototypes toward practical design closure. The flows that we describe provide a solid open-source foundation for continued advances in 3D IC design automation.

Acknowledgments. The authors consider Andrew B. Kahng of UC San Diego to satisfy IEEE and ACM criteria for authorship of this paper. However, with the assent of IEEE and ACM representatives, Prof. Kahng is not listed as an author due to applicable institutional or funding-related restrictions.

References

- [1] 3Dblox Consortium, *3Dblox specification*. <https://www.3dblox.org>.
- [2] AMD, *AMD Instinct MI300 Series Microarchitecture*. <https://instinct.docs.amd.com/develop/gpu-arch/mi300.html>.
- [3] B. Ayoub, S. Lhostis, S. Moreau, E. L. Perez, J. Jourdon, P. Lamontagne, E. Deloffre, S. Mermoz, C. de Buttet, V. Balan, C. Euvard, Y. Exbrayat and H. Frémont, "Impact of process variations on the capacitance and electrical resistance down to 1.44 μm hybrid bonding interconnects," *Proc. IEEE EPTC*, 2020, pp. 453–458.
- [4] L. Bamberg, A. Garcia-Ortiz, L. Zhu, S. Pentapati, D. E. Shim and S. K. Lim, "Macro-3D: A physical design methodology for face-to-face-stacked heterogeneous 3D ICs," *Proc. DATE*, 2020, pp. 37–42.
- [5] N. E. Bethur, A. Agnesina, M. Brunion, A. Garcia-Ortiz, F. Cathoor, D. Milojevic, M. Komalan, M. Cavalcante, S. Riedel, L. Benini and S. K. Lim, "Hier-3D: A methodology for physical hierarchy exploration of 3-D ICs," *IEEE TCAD* 43(7) (2024), pp. 1957–1970.
- [6] I. Bustany, G. Gasparyan, A. B. Kahng, I. Koutis, B. Pramanik and Z. Wang, "An open-source constraints-driven general partitioning multi-tool for VLSI physical design," *Proc. ACM/IEEE ICCAD*, 2023, pp. 1–9.
- [7] J. Chang, "3Dblox: Unleash the ultimate 3DIC design productivity," *Proc. ACM ISPD*, 2024, p. 215.
- [8] S.-H. Chen, Z.-W. Chen, P.-J. Lin, H.-J. Hsu, H.-Y. Lin, Y.-H. Chuang, H.-Y. Chen, J. Chang and Y.-W. Chang, "Performance-driven pre-assignment routing for high-speed package designs," *Proc. ACM/IEEE ICCAD*, 2025, pp. 1–9.
- [9] Y.-J. Chen, C.-H. Hsieh, P.-H. Su, S.-H. Chen and Y.-W. Chang, "Mixed-size 3D analytical placement with heterogeneous technology nodes," *Proc. ACM/IEEE DAC*, 2024, pp. 1–6.
- [10] J. Cherry, *OpenSTA—Parallax static timing analyzer*. <https://github.com/parallaxsw/OpenSTA>.
- [11] CODA-Team, *TaiWei: 3D system integration physical design platform*, GitHub repository, 2026. <https://github.com/CODA-Team/TaiWei>.
- [12] CUHK-EDA, *CUGR 2.0*, GitHub repository, 2023. <https://github.com/cuhk-eda/cugr-2>.
- [13] R. Ewetz and C.-K. Koh, "Scalable construction of clock trees with useful skew and high timing quality," *IEEE TCAD* 38(6) (2019), pp. 1161–1174.
- [14] H. Fraisse, A. Joshi, D. Gaitonde and A. Kaviani, "Boolean satisfiability-based routing and its application to Xilinx UltraScale clock network," *Proc. ACM FPGA*, 2016, pp. 74–79.
- [15] A. Graening, P. Gupta, A. B. Kahng, B. Pramanik and Z. Wang, "ChipletPart: Cost-aware partitioning for 2.5D systems," *ACM TODAES* 31(5) (2026), pp. 88:1–88:29.
- [16] A. Graening, J. Talukdar, S. Pal, K. Chakrabarty and P. Gupta, "CATCH: A cost analysis tool for co-optimization of chiplet-based heterogeneous systems," *IEEE TCAD* 45(3) (2026), pp. 1091–1104.
- [17] C. Guth, V. Livramento, R. Netto, R. Fonseca, J. L. Güntzel and L. Santos, "Timing-driven placement based on dynamic net-weighting for efficient slack histogram compression," *Proc. ACM ISPD*, 2015, pp. 141–148.
- [18] T. He, "A time scaling theory for multi-layer electronic systems," *Sci. China Inf. Sci.* 69 (2026), Article 183401.
- [19] H.-W. Hu and K.-N. Chen, "Development of low temperature Cu–Cu bonding and hybrid bonding for three-dimensional integrated circuits (3D IC)," *Microelectron. Reliab.* 127 (2021), Article 114412.
- [20] J. Hu, P. Vanna-iampikul, Z. Zhuang, T.-Y. Ho and S. K. Lim, "GNN-MLS: Signal routing in mixed-node 3D ICs through GNN-assisted metal layer sharing," *Proc. ACM/IEEE DAC*, 2025, pp. 1–7.
- [21] Y.-H. Huang, S. Pentapati, A. Agnesina, M. Brunion and S. K. Lim, "On legalization of die bonding bumps and pads for 3-D ICs," *IEEE TCAD* 43(9) (2024), pp. 2741–2754.
- [22] Huawei, "HUAWEI presents the Tau (τ) scaling law, enabling breakthroughs in transistor density and system performance," 2026. <https://www.huawei.com/en/news/2026/5/ieee-iscas-tau-scaling>.
- [23] IEEE Standards Association P3537 Working Group, "IEEE Draft Standard for 3Dblox–Chiplet Connectivity and Physical Properties Description Language," *IEEE P3537/D2*, Jul. 2026, pp. 1–134. <https://ieeexplore.ieee.org/document/11627229>.
- [24] Intel, *Advanced Packaging Innovations*. <https://www.intel.com/content/www/us/en/foundry/packaging.html>.
- [25] Intel, *Foveros Direct 3D Technology Brief*. <https://www.intel.com/content/dam/www/central-libraries/us/en/documents/2025-11/foveros-direct-3d-tech-brief.pdf>.
- [26] B. Jiang, Y. Wu, Z. Zhong, K. Fan, Q. Lin, J. Liang, H. Liu and X. Liu, "Cu–Cu hybrid bonding technology: From physical mechanisms to system integration for 3D ICs," *Moore and More* 2(1) (2025), Article 18.
- [27] X. Jiang, H. Lu, Y. Zhao, J. Wang, Z. Guo, H. Wu, B. Yu, S. K. Lim, R. Wang, R. Huang and Y. Lin, "A systematic approach for multi-objective double-side clock tree synthesis," *Proc. ACM/IEEE DAC*, 2025, pp. 1–7.
- [28] B. Jiao, L. Xu, X. Yu, H. Yang, H. Zhu, Y. Wang, J. Zhu, D. Wen, L. Wang, J. Tao, C. Chen, Y. Han, Q. Liu, N. Sun and M. Liu, "FPIA: Communication-aware multi-chiplet integration with field-programmable interconnect fabric on reusable silicon interposer," *IEEE TCAS-I* 71(9) (2024), pp. 4156–4168.
- [29] G. Karypis and V. Kumar, "hMETIS: A hypergraph partitioning package, version 1.5.3," Department of Computer Science, University of Minnesota, Technical Report, 1998.
- [30] S. E. Kim and S. Kim, "Wafer level Cu–Cu direct bonding for 3D integration," *Microelectron. Eng.*, vol. 137, pp. 158–163, 2015.
- [31] B. W. Ku, K. Chang and S. K. Lim, "Compact-2D: A physical design methodology to build commercial-quality face-to-face-bonded 3D ICs," *Proc. ACM ISPD*, 2018, pp. 90–97.
- [32] J. H. Lau, "Recent advances and trends in Cu–Cu hybrid bonding," *IEEE TCPMT* 13(3) (2023), pp. 399–425.
- [33] F. Li, Y. Wang, Y. Cheng, Y. Wang, Y. Han, H. Li and X. Li, "GIA: A reusable general interposer architecture for agile chiplet integration," *Proc. ACM/IEEE ICCAD*, 2022, pp. 1–9.
- [34] S. Li, L. Jin, S. Liang, Z. Zhuang, R. Chen, B. Yu and T.-Y. Ho, "FlexiCTS: CPPR-aware 3D clock tree synthesis for face-to-face bonded ICs," *Proc. ACM/IEEE DAC*, 2026.
- [35] R. Liang, A. Agnesina, W.-H. Liu, M. Liberty, H.-T. Chang and H. Ren, "Invited: ISPD 2025 performance-driven large scale global routing contest," *Proc. ACM ISPD*, 2025, pp. 252–256.
- [36] P. Liao, D. Guo, Z. Guo, S. Liu, Y. Lin and B. Yu, "DREAMPlace 4.0: Timing-driven placement with momentum-based net weighting and Lagrangian-based refinement," *IEEE TCAD* 42(10) (2023), pp. 3374–3387.
- [37] P. Liao, Y. Zhao, D. Guo, Y. Lin and B. Yu, "Analytical die-to-die 3-D placement with bistratal wirelength model and GPU acceleration," *IEEE TCAD* 43(6) (2024), pp. 1624–1637.
- [38] J. Liu and E. F. Y. Young, "EDGE: Efficient DAG-based global routing engine," *Proc. ACM/IEEE DAC*, 2023, pp. 1–6.
- [39] S. Liu, J. Jiang, Z. He, Z. Wang, Y. Lin, B. Yu and M. Wong, "Routing-aware legal hybrid bonding terminal assignment for 3D face-to-face stacked ICs," *Proc. ACM ISPD*, 2024, pp. 75–82.
- [40] Z. Liu, Y. Li, J. Hu, X. Yu, S. Shiau, X. Ai, Z. Zeng and Z. Zhang, "DeepOHeat: Operator learning-based ultra-fast thermal simulation in 3D-IC design," *Proc. ACM/IEEE DAC*, 2023, pp. 1–6.
- [41] G.-J. Nam, K. A. Sakallah and R. A. Rutenbar, "Satisfiability-based layout revisited: Detailed routing of complex FPGAs via search-based Boolean SAT," *Proc. ACM FPGA*, 1999, pp. 167–175.
- [42] The OpenROAD Project, *OpenDB 3D IC documentation*. <https://github.com/The-OpenROAD-Project/OpenROAD/blob/master/src/odb/doc/3dic.md>.
- [43] The OpenROAD Project, *OpenRCX—Parasitics extraction*. <https://openroad.readthedocs.io/en/latest/main/src/rcx/README.html>.
- [44] S. Pentapati and S. K. Lim, "Metal layer sharing: A routing optimization technique for monolithic 3D ICs," *IEEE TVLSI* 30(9) (2022), pp. 1355–1367.
- [45] 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," *Proc. ACM/IEEE ICCAD*, 2020, pp. 1–9.
- [46] S. Roy, P. M. Mattheakis, L. Masse-Navette and D. Z. Pan, "Clock tree resynthesis for multi-corner multi-mode timing closure," *IEEE TCAD* 34(4) (2015), pp. 589–602.
- [47] I. Savvidis, S. M. Alam, A. Jain, S. Pozder, R. E. Jones, R. Chatterjee, "Electrical modeling and characterization of through-silicon vias (TSVs) for 3-D integrated circuits," *Microelectronics J.* 41(1) (2010), pp. 9–16.
- [48] I. Savvidis and E. G. Friedman, "Closed-form expressions of 3-D via resistance, inductance, and capacitance," *IEEE TED* 56(9) (2009), pp. 1873–1881.
- [49] Y. Shi, C. Gao, W. Ren, P. Xie, S. Xu, K. Xue, M. Yuan, C. Qian and Z.-H. Zhou, "Open3DBench: Open-source benchmark for 3D-IC backend implementation and PPA evaluation," *arXiv preprint arXiv:2503.12946v2*, 2026.
- [50] E. Shragowitz and S. Keel, "A global router based on a multicommodity flow model," *Integr. VLSI J.* 5(1) (1987), pp. 3–16.
- [51] Siemens EDA, *3Dblox resources*. <https://www.siemens.com/en-us/company/electronic-design-automation/trending-technologies/3d-ic-design/3dblox-resources/>.
- [52] C. Wolf and J. Glaser, "Yosys—A free Verilog synthesis suite," *Proc. Austrian Workshop Microelectron.*, 2013, pp. 47–52.
- [53] Y. Zhao, P. Liao, S. Liu, J. Jiang, Y. Lin and B. Yu, "Analytical heterogeneous die-to-die 3-D placement with macros," *IEEE TCAD* 44(2) (2025), pp. 402–415.
- [54] 3Dblox-based OpenDB. <https://github.com/The-OpenROAD-Project/OpenROAD/tree/master/src/odb/src/3dblox>
- [55] OpenROAD-Research. <https://github.com/ieee-ceda-dac/OpenROAD-Research>