CU3D: A Comprehensive Timing-driven Design Flow for Face-to-Face-Bonded 3D ICs (Invited)

Haoyang Xu, Leilei Jin*, Tsung-Yi Ho
The Chinese University of Hong Kong

Abstract

3D ICs with face-to-face (F2F) bonding technology offer significant advantages over traditional 2D designs, including improved power efficiency, enhanced timing performance, and reduced footprint. However, existing design flows fail to adequately address critical challenges in F2F 3D implementations, particularly tier partition quality and 3D clock tree quality across multiple tiers. We propose CU3D, a novel F2F 3D IC design flow featuring: (1) timing-driven tier partitioning that preserves critical path locality, (2) latency-aware 3D clock tree synthesis with per-die optimization, and (3) iterative cross-tier timing constraint propagation for accurate timing closure. Experimental results demonstrate that compared to state-of-the-art 3D design flows, our approach achieves 7.9% improvement in worst negative slack (WNS), 51.5% improvement in total negative slack (TNS), 17.9% reduction in power consumption, 3.9% reduction in footprint area, and meets hybrid bond (HB) constraints, demonstrating the effectiveness of our holistic F2F 3D design methodology.

1 Introduction

F2F stacking, enabled by sub-micron wafer-to-wafer hybrid bonding, has emerged as a transformative technology in advanced 3D ICs [1–4]. Its deployment in commercial products like AMD's 3D V-Cache [5] and META's AR SoC [6] demonstrates an industry-wide shift towards fine-grained integration. Unlike traditional through-silicon-via (TSV) approaches limited to block-level stacking with large interconnect pitch (typically 5–10μm), F2F HBs achieves sub-micron pitch (< 1μm) [7–9]. This bonding configuration creates a unique physical structure where logic cells and interconnects from both dies are in close proximity, significantly reducing wirelength for cross-die connections and fundamentally altering the design space for system-level optimization. Compared to conventional 2D designs, F2F 3D ICs offer substantial benefits, including improved power efficiency due to reduced interconnect capacitance, enhanced timing performance through shorter critical paths, and a smaller footprint by stacking multiple dies vertically.

Despite these advantages, designing F2F 3D ICs presents formidable challenges that are not adequately addressed by existing design methodologies. First, in advanced technology nodes, the number of available HBs is severely limited due to manufacturing constraints, placement blockages, and routing congestion, creating a critical bottleneck for clock and signal connections that must traverse between dies. This scarcity necessitates careful optimization of HB utilization, particularly for clock networks that require multiple cross-die connections to maintain signal integrity and skew control. Second, achieving timing closure in F2F 3D designs is extremely difficult due to the lack of cross-die timing awareness during physical design stages. Traditional tools and flows treat each die independently, leading to inaccurate timing estimation for paths spanning multiple tiers and resulting in suboptimal optimization decisions. These challenges collectively render the full-flow physical design of F2F 3D ICs highly complex, requiring new methodologies that can simultaneously address HB constraints, clock distribution quality, and true 3D timing optimization.

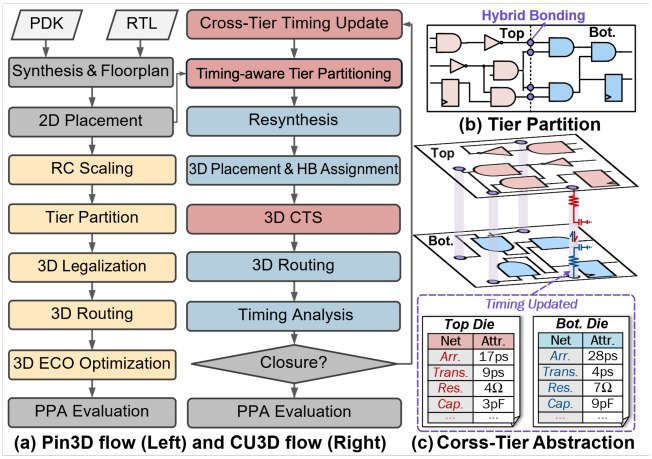


Figure 1: (a) CU3D flow for F2F 3D ICs compared to Pin-3D [10], comparison to existing 3D design flow, (b) tier partitioning, and (c) cross-tier timing update.

In this paper, we propose **CU3D**, an efficient timing-aware full-chip physical design flow for F2F 3D ICs. As shown in Figure 1(a), our flow begins with physical-aware tier partitioning directly from 2D placement results, ensuring placement-driven optimization from the outset. We then apply logic resynthesis techniques to unlock aggressive area compression opportunities in some regions, followed by die-by-die placement optimization that leverages the expanded solution space for superior placement quality. After strategic HB allocation as illustrated in Figure 1(b), we perform enhanced 3D-aware clock tree synthesis (CTS) that fully exploits vertical integration capabilities. Through iterative place-and-route with cross-die timing updates via external timing constraint exchange shown in Figure 1(c), we achieve true inter-die timing awareness throughout the optimization process. This comprehensive approach delivers substantial PPA improvements across all metrics, fully realizing the potential of F2F 3D integration.

Our contributions are listed below:

*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/10.1145/3831252.3845851

Table 1: Qualitative comparison between representative 3D flows and CU3D (Ours).

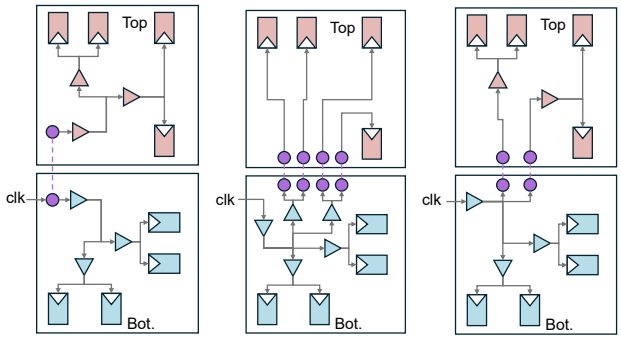
Method	Compact-2D [11]	AnchorGrid [12]	Pin-3D [10]	CU3D
Key Idea	Row halving	Array-style HB assignment	Pin projection	Cross-tier abstraction
3D Stack	Two separate dies	Two separate dies	Double metal stack	Two dies with awareness
Strength	Shrinking unnecessary	Optimizing with anchor constrains	3D ECO & Timing closure	**PPA-driven**
Weakness	DRVs due to row halving	Poor timing closure	Ignoring HB constraints	-
Placement	2D + Tier partitioning	Multi-die co-placement	2D + Tier partitioning	**Cross-tier aware placement**
Clock Tree Design	2D + Tier partitioning	Die-by-die	Optimized for 3D	**3D CTS under HB constarints**
Post Route Optimization	Both dies at once	Not supported	Die-by-die	**PPA abstraction**
HB Constraints	Not supported	Supported	Not supported	Supported
Techfile	Hack-necessary	Homogeneous-only	Hack-necessary	**Independent without hacking**

- We propose a full-chip PPA-driven F2F 3D IC physical design flow that handles array-style HBs in advanced technology nodes with commercial quality.
- We develop an efficient timing-aware tier partitioning method that leverages placement engine results to minimize HB usage while maintaining design quality.
- We introduce a unified HB assignment strategy for clock and signal nets with superior CTS performance.
- We establish accurate 3D timing modeling with cross-die constraint propagation, enabling early-stage inter-die timing awareness for reliable timing closure.
- Experimental results demonstrate that compared to the state-of-the-art design flow, our approach achieves significant reductions in HB usage alongside comprehensive PPA improvements.

2 Preliminarys

2.1 F2F 3D ICs in Advanced Technologies

Pseudo-3D flows [10–12] have been proposed to extend commercial 2D EDA capabilities to 3D ICs. In such flows, a 3D design is constructed by first creating an "intermediate 2D design" that closely emulates 3D parasitics, which is then partitioned into multiple tiers and routed to obtain the final 3D implementation. However, as technology nodes scale to 3nm and beyond, a fundamental disconnect has emerged between transistor scaling and HB density. While standard cell dimensions have reduced dramatically, the minimum manufacturable HB pitch has remained relatively stagnant, currently limited to approximately 0.2μm due to fabrication constraints [8]. This creates a severe scale disparity. A 3nm inverter cell occupies an area orders of magnitude smaller than the footprint required for a single HB [9]. Consequently, as chip dimensions shrink with technology scaling, the total area available for HB placement reduces proportionally, yet each HB still requires a comparably larger pitch footprint. To address this challenge, several 3DIC methods have been developed, leveraging information available at different stages, as summarized in Table 1. However, the achievable timing quality ultimately depends on the tier partitioning strategy in pseudo-3D flows. Without a timing-aware partitioning method that respects array-style HB constraints, reliable timing closure cannot be achieved. Moreover, hacking the technology file is no longer needed for CU3D, since hacking is difficult for commercial PDKs.



(a) Cascade-2D [13] **(b) Pin-3D [10]** **(c) FlexiCTS [14]**

Figure 2: Pseudo-3D CTS strategies: (a) Port partitioning strategy: independent clock trees built per die. (b) Centralized buffering strategy: all buffers constrained to one die. (c) Buffers are placed across dies based on 2D CTS solutions.

2.2 Challenges in 3D CTS

Prior research has not fully addressed this co-optimization challenge, with academic and industrial approaches taking fundamentally different paths. Academic work has largely focused on TSV-based architectures, with algorithms like 3D-MMM [15], DME [16], and inductance-aware methods [17] targeting wirelength minimization and TSV cost reduction. These methods assume high-cost vertical links and cannot be directly applied to HB, where the cost model for cross-die connections is fundamentally different due to their low parasitics. In the commercial context, the absence of native 3D CTS tools has led to pseudo-3D frameworks that adapt conventional 2D flows through rigid partitioning strategies. As illustrated in Figure 2, these approaches exhibit inherent limitations. The clock port partitioning strategy (Figure 2(a)), employed by Cascade-2D [13] and AnchorGrid [12], builds independent clock trees on each die. This minimizes HB usage but forces early path divergence, severely degrading timing performance. Conversely, the centralized buffering strategy (Figure 2(b)), adopted by Pin-3D [10], constrains all clock buffers to a single die to maintain 2D tool compatibility. While this maximizes common path length, it creates extreme imbalance in cross-die delays and consumes excessive HBs, leading to significant clock skew degradation. FlexiCTS [14] attempts adaptive buffer assignment based on local subtree characteristics (Figure 2(c)), yet remains constrained by two limitations: dependence on 2D CTS solutions as initial templates and lack of cross-tier timing awareness during buffer placement decisions. These constraints prevent true

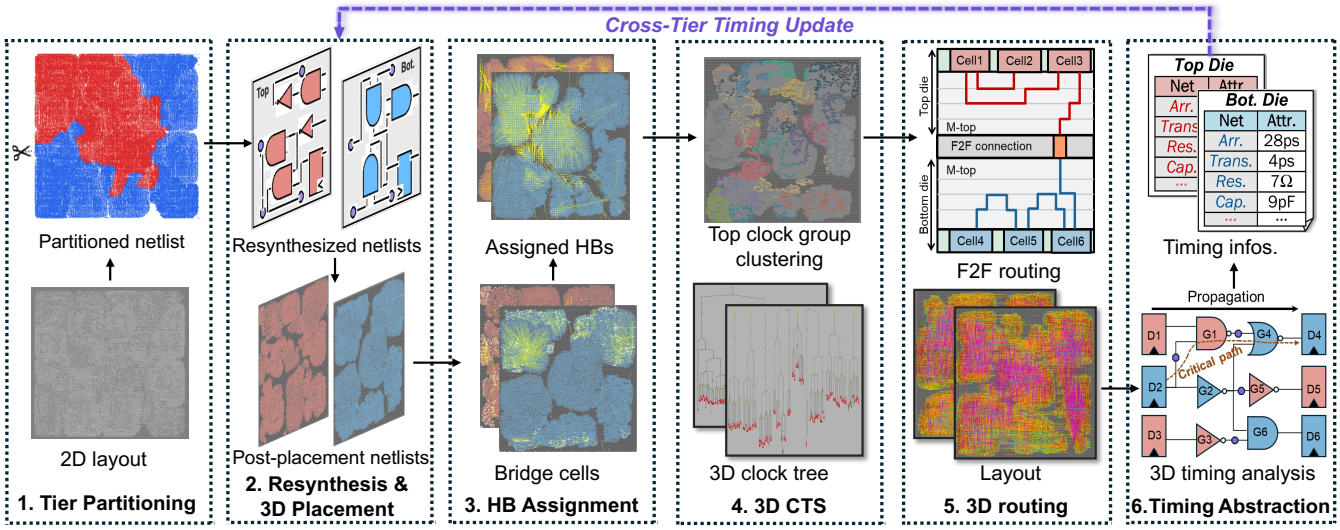


Figure 3: Overview of the proposed CU3D. The flow consists of six stages: (1) tier partitioning from 2D layout, (2) resynthesis and 3D placement of partitioned netlists, (3) HB assignment and bridge cell insertion, (4) 3D CTS with top clock group clustering, (5) 3D routing with vertical connections, and (6) timing abstraction with cross-tier timing propagation.

co-optimization of topology and 3D resource allocation, leaving substantial performance gains unexplored.

2.3 3D Timing Closure

Cross-die timing awareness is crucial for 3D IC design, yet achieving it with 2D commercial tools remains challenging due to mismatches between 2D timing models and 3D physical reality. Existing timing-driven tier partition methods [18–20] rely on 2D timing reports, which misrepresent 3D behavior because vertical interconnects alter delay propagation. Even incorporating HB RC parasitics, physical connectivity interference differs from logical timing interference, rendering optimized placements ineffective. Cross-die-aware placement algorithms cannot integrate with commercial tools, and their solutions are overridden during 2D reoptimization. Metal layer sharing methods [21] sidestep timing awareness, benefits stem from wirelength reduction rather than timing-driven optimization. Accurate cross-die timing analysis [22–24] requires parasitic files unavailable before detailed routing, creating circular dependency: timing decisions need accurate delays, but delays require completed design. A comprehensive timing-aware 3D flow integrating with commercial tools remains absent.

3 Overall Design Flow

In this paper, we propose a comprehensive F2F 3D design flow, CU3D. The complete CU3D workflow is depicted in Figure 3, which comprises six main stages: 1) Timing-aware tier partitioning for optimal trade-off between HBs limits and 3D benefits; 2) Resynthesis and 3D placement for early stage PPA optimization; 3) HB assignment for clocks and signals; 4) Latency-aware 3D clock tree generation; 5) 3D die-by-die routing and 6) Cross-tier timing abstraction for timing closure in the additional design round.

3.1 Timing-aware Tier Partitioning

In F2F 3D IC design, tier partitioning has a decisive effect on both HB usage and post-partition timing. We propose a timing-aware clustering scheme that operates on register clusters rather than individual cells, so that the partition respects the placement locality

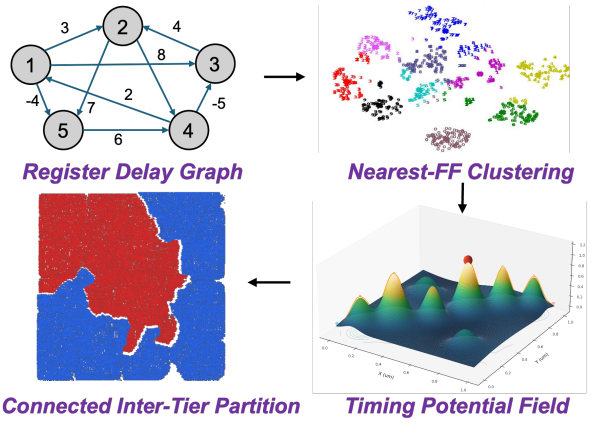


Figure 4: Timing-aware clustering for F2F tier partitioning, from a register delay graph to a connected inter-tier cut.

leveraged by geometric methods while explicitly accounting for the timing structure of the design. As illustrated in Figure 4, the core idea is to treat the register graph induced by timing paths as a set of interacting particles: each flip-flop (FF) is assigned a one-dimensional potential, and the criticality-weighted delay between two FFs acts as a spring that shapes this potential field. Connected combinational logic is folded onto its spatially nearest FF (nearest-FF clustering), so each cluster inherits the potential of its anchor register. The tier assignment is obtained by growing two connected cluster regions from the two extremes of the potential field, producing a single connected inter-tier boundary that co-locates timing-critical logic while keeping the interface short enough to satisfy the HB budget.

A critical insight underlying our approach is that the two objectives of tier partition pull in opposite directions. A delay-driven embedding alone, where registers are simply thresholded by their potential, cuts thousands of nets because the timing structure of a design rarely aligns with a spatially compact boundary; conversely, a purely spatial linear cut minimizes HB but may separate the two ends of a critical path. Our resolution is to decouple the two spaces:

Algorithm 1 Timing-Aware Tier Partitioning

Input: Placed netlist $\mathcal{N}$, timing paths $\mathcal{P}$.
Output: Tier assignment $T : \mathcal{N} \to \{0, 1\}$.

// Phase 1: Register delay graph and clustering
$G \leftarrow \{(x, y, F(x, y))\}$ from paths $\mathcal{P}$;
$\mathcal{C} \leftarrow$ ClusterComboToNearestFF($\mathcal{N}$);
// Phase 2: Potential equilibration on z (xy fixed)
for all FF particles **do**
  $F_z \leftarrow$ DelaySprings + SoftRepulsion + Confinement;
  $z \leftarrow z + \Delta t \cdot F_z/\text{mass}$ **until** $\max |F_z| < \epsilon$;
end for
// Phase 3: Field smoothing and connected two-seed growth
$z' \leftarrow$ SmoothOnKNNGraph(z);
$s_0 \leftarrow \arg\min z'$, $s_1 \leftarrow \arg\max z'$;
while area not balanced **do**
  grow lagging region from $\{s_0, s_1\}$ along k-NN edges
  in ascending/descending potential order;
end while
// Phase 4: Boundary refinement for HB
for each frontier cluster c **do**
  if majority of spatial neighbors in opposite tier **then**
    flip $T(c)$ if area balance permits;
  end if
end for
RebalanceArea(T);
return T;

timing determines a scalar potential field over registers, and spatial proximity determines where this field may be cut. By first smoothing the potential field (timing potential field in Figure 4) over the k-nearest-neighbor graph of register coordinates, the field becomes spatially coherent, so that a connected cut driven by the field inherits the locality of the 2D placement while still being guided by timing. This design philosophy preserves the hard-won optimization results of the 2D placer: congestion, timing, and power tradeoffs are retained because cells never move in the xy plane. The algorithm proceeds in four phases (Algorithm 1).

3.2 Die-by-Die 3D Resynthesis

After tier partitioning, we propose a die-by-die resynthesis methodology to exploit additional area optimization within each die while preserving inter-die communication. It exploits the potential of the library through the logic synthesis tool for early-stage improvement.

As illustrated in Figure 5, the die-by-die resynthesis process reveals opportunities for further cell merging and logic optimization that were previously constrained by the monolithic netlist structure. In the partitioned gate-level netlist (left panel), cells G2, G3, and G6 are initially mapped to inverters (INVD1) and a two-input NAND gate (NAND2D1) based on their logical functions. However, when these cells are analyzed within their respective die contexts during resynthesis, the optimization engine identifies that G6 can be re-implemented using a more compact two-input AND gate (AND2D1), as shown in the right panel. This transformation, enabled by local logic restructuring within the die boundary, achieves a 12% area reduction while maintaining identical functionality and timing characteristics. The layout views demonstrate the physical area benefit through reduced transistor count and more efficient cell geometry.

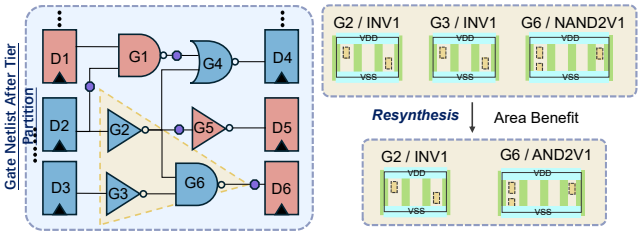


Figure 5: Die-by-die 3D resynthesis for PPA optimization, G2 and G3 can be optimized to G6 by the die-specific context.

Our resynthesis flow treats each partitioned netlist independently, applying constrained gate-level optimization with die-specific library restrictions. Critically, all `BRIDGE` cells on both dies are protected via `dont_touch` attributes to maintain HB topology. Within each die, the synthesis tool performs: (1) redundant logic elimination near partition boundaries, (2) buffer optimization along HB signal paths, (3) timing-driven cell downsizing on non-critical paths, and (4) fanout restructuring under die-specific constraints. The HB interface ports are treated as ideal network boundaries to prevent cross-die optimization violations.

3.3 3D Placement

The first placement phase focuses on the global placement of all standard cells, macros, and the newly inserted `BRIDGE` cells. A key feature is the deliberate decoupling of `BRIDGE` cell pairs: the `BRIDGE` cell on the top die, and its counterpart on the bottom die, move independently during global placement. This freedom enables the placement engine to optimize logic placement without being constrained by inter-die alignment. All large macros are placed and legalized during this phase, providing a stable physical context. `BRIDGE` cells can be positioned freely, even within memory macro regions, as they consume no silicon area, offering maximum flexibility for timing optimization.

After global placement is optimized, the second phase begins. All `BRIDGE` cells are snapped to the pre-defined, array-style HB grid, achieving precise alignment between the top die and bottom die bridge cells. Following this legalization, the `BRIDGE` cells are transformed into a `COVER` state: they become fixed and are treated as non-obstructing overlays. During subsequent detailed placement and optimization, the place-and-route engine can place and legalize standard cells directly underneath these fixed bridge cells. This two-phase flow successfully decouples early placement flexibility from the final requirement of legal HB alignment.

3.4 Hybird Bond Allocation Strategy

The assignment of HBs determines physical inter-die connection points and profoundly impacts wirelength, power, and timing. Figure 6 illustrates our allocation process.

Signal Net Assignment. Signal nets are timing-critical and have fixed endpoint locations after placement, making their HB assignment less flexible but more consequential. We therefore prioritize signal allocation. A naive approach would assign each cut net independently to the nearest HB candidate, but this ignores HB capacity constraints and may cause local congestion. Instead, we formulate the assignment as a bipartite matching problem to generate mapping between cut nets and HB locations, minimizing total wirelength while respecting one HB per net. To compute the cost matrix

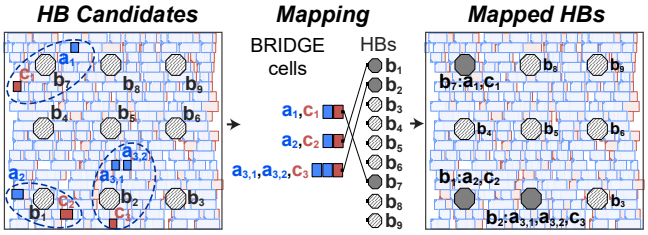


Figure 6: The illustration of HB assignment process. Top `BRIDGE` cells (blue) and bottom `BRIDGE` cells (red) will be mapped into the array-style HBs.

efficiently, we observe that each net's ideal HB location should minimize the sum of distances to its `BRIDGE` cells on both dies. The geometric center of these `BRIDGE` cells serves as a proxy for the optimal attachment point. We then query the nearest available HB using a KD-tree [25], a space-partitioning data structure that indexes HB locations for logarithmic-time nearest neighbor search. This reduces the matching complexity from $O(NM)$ to $O(N \log M)$, where N is the number of cut nets and M the number of HB candidates. Joint processing ensures that corresponding nets on both dies are assigned to the same HB, preserving signal integrity across the 3D interface.

Clock Distribution Assignment. After reserving HBs for signal routing ($\mathcal{H}_{signal}$), the remaining resources ($\mathcal{H}_{clock}$) serve clock delivery. Unlike signal nets, clock distribution does not have fixed endpoints, the placement of clock sinks is already determined, but the topology of the clock tree is flexible. This flexibility allows us to cluster sinks before HB assignment, reducing the number of clock HBs and minimizing insertion delay. However, naive clustering could increase clock skew if sinks within the same cluster have large spatial spread. We therefore apply K-means clustering to group spatially proximate flip-flops into K clusters, where $K = |\mathcal{H}_{clock}|$ is determined by the available clock HBs. This balances two competing objectives: minimizing intra-cluster distance to reduce local skew while maximizing HB utilization to avoid wasting resources. Each cluster is then assigned to its nearest clock HB. The bottom die outputs its local clock to the top die through these HBs, while the top die receives clock signals as HB inputs, forming a complete 3D clock tree spanning both dies (detailed in Algorithm 2). This two-phase priority-driven design ensures that timing-critical signal nets receive optimal HB assignment first, while clock distribution leverages remaining resources with minimal impact on overall skew.

3.5 Latency-Aware 3D CTS

Achieving a balanced and low-power clock network in a 3D stacked design is fundamentally more challenging than in 2D. The physical separation of tiers introduces additional latency and capacitance through HBs, while the two dies may operate under different PVT (process-voltage-temperature) corners, making traditional single-tree clock distribution across both dies impractical. Our method addresses these challenges through two key insights: (1) decoupling the clock network into per-die sub-trees reduces cross-tier coupling and enables localized optimization, and (2) propagating latency constraints from top to bottom allows commercial 2D CTS tools to be reused for 3D balancing.

Algorithm 2 Latency-Aware 3D Clock Tree Synthesis

Input: Placed netlist $\mathcal{N}$ with tiers, Set of HBs $\mathcal{H}$, Clock sinks $\mathcal{S}$.
Output: Synthesized 3D clock tree with balanced skew.

$\mathcal{S}_{top} \leftarrow \{s \in \mathcal{S} \mid \text{tier}(s) = \text{top}\}$;
$\mathcal{C}_{top} \leftarrow \text{KMeansCluster}(\mathcal{S}_{top}, K = |\mathcal{H}|)$;
$\mathcal{T}_{top} \leftarrow \emptyset$ //Set of top-die clock trees
for each cluster $c \in \mathcal{C}_{top}$ **do**
  $t_c \leftarrow \text{SynthesizeClockTree}(c)$;
  $\mathcal{T}_{top} \leftarrow \mathcal{T}_{top} \cup \{t_c\}$;
  $L[c] \leftarrow \text{ReportLatency}(t_c)$;
end for
$\mathcal{S}_{bot} \leftarrow \{s \in \mathcal{S} \mid \text{tier}(s) = \text{Bot}\}$;
$\mathcal{V}_{hb} \leftarrow \emptyset$; //Virtual sink nodes for HBs
for each HB $hb \in \mathcal{H}$ **do**
  $c_{hb} \leftarrow \text{GetAssociatedCluster}(hb, \mathcal{C}_{top})$;
  CreateVirtualSink(hb);
  SetLatencyConstraint($hb, L[c_{hb}]$);
  $\mathcal{V}_{hb} \leftarrow \mathcal{V}_{hb} \cup \{hb\}$;
end for
$\mathcal{S}_{bot}^{target} \leftarrow \mathcal{S}_{bot} \cup \mathcal{V}_{hb}$;
$t_{bot} \leftarrow \text{SynthesizeClockTree}(\mathcal{S}_{bot}^{target})$;
$\mathcal{T}_{3D} \leftarrow \text{IntegrateTrees}(t_{bot}, \mathcal{T}_{top}, \mathcal{H})$;
return $\mathcal{T}_{3D}$;

On the top die, all clock sinks (flip-flops) are grouped via K-means clustering. The number of clusters K is set to the number of available clock HBs, as each cluster requires a dedicated HB for clock delivery from the bottom die. Unlike grid-based partitioning, K-means minimizes intra-cluster distance, which directly translates to lower local clock skew and reduced buffer insertion within each cluster. This is critical because the top die has no direct clock input, all clock energy must come through HBs, making local power efficiency paramount. Each cluster is synthesized as an independent, balanced clock subtree. This multi-clock-domain design confines high-speed clock buffers and local skew management within each cluster, preventing clock glitches or skew from propagating across the entire top die.

After top-die CTS, we extract the latency from the HB input to each sink within the cluster. This latency is then annotated as a *virtual sink* constraint for the bottom-die clock tree. The key insight here is that bottom-die CTS does not need to know the internal topology of the top-die sub-trees; it only needs to balance the delay from its clock source to each HB output such that, when combined with the top-die latency, all sinks across both dies see the same total insertion delay. This *latency propagation* approach effectively decouples the two CTS problems, allowing standard 2D CTS tools to be used on each die independently.

On the bottom die, the primary clock signal enters through an I/O port. Each HB connecting to a top-die clock group is modeled as a virtual sink, annotated with the corresponding top-die latency. The bottom clock tree is then synthesized to balance insertion delay to two types of sinks: (1) its own local flip-flops on the bottom die, and (2) these virtual sinks representing top-die clusters. Balancing across these heterogeneous sinks is non-trivial because local sinks have no pre-annotated latency, while virtual sinks carry a fixed latency offset. Our solution treats the bottom-die CTS as a *zero-skew*

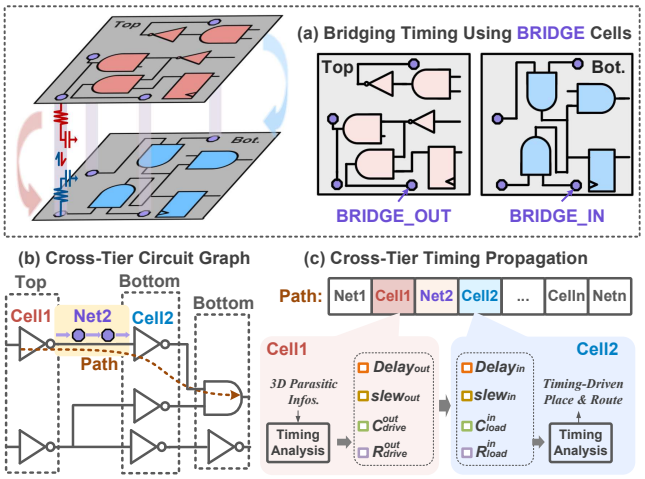


Figure 7: Cross-tier timing propagation methodology using `BRIDGE` cells for F2F 3D integration.

problem with offset targets, which commercial EDA tools can solve using their built-in latency budgeting capabilities. The resulting 3D clock tree achieves near-zero global skew while minimizing clock wirelength and power, as the bottom die's clock distribution can share buffers across multiple virtual sinks that are spatially aligned with its local sinks.

3.6 Cross-Tier Timing Constraint Propagation

Accurate timing closure in F2F 3D ICs requires jointly modeling cross-tier interconnect delay, which depends on source die routing, HB parasitics, and destination die load. We address this through unified 3D parasitic extraction and iterative SDC-based constraint exchange at tier boundaries. Cross-tier signals are represented by paired interface cells: `BRIDGE_OUT` on the driving die and `BRIDGE_IN` on the receiving die.

Figure 7(a) shows the physical `BRIDGE` cell placement at HB interfaces. Figure 7(b) depicts a representative cross-tier path: `Net1` connects `Cell1` on the top die, passes through the BRIDGE interface, and drives `Cell2` on the bottom die via `Net2`. A signal path traverses local routing on the source die, passes through `BRIDGE_OUT`, the HB structure, and `BRIDGE_IN` on the destination die, then continues to downstream logic. Figure 7(c) illustrates the timing propagation flow: merged parasitic extraction provides accurate load values; timing analysis computes output delay and slew at `Cell1`, which become input constraints for `Cell2`, enabling timing-driven optimization on the receiving die. The `BRIDGE_OUT` cell computes its output delay and slew based on the total load, which includes the net capacitance, HB parasitics, and the input capacitance of `BRIDGE_IN`:

$$C^{out}_{load} = C_{net} + C_{HB} + C_{input}, \tag{1}$$

$$R^{out}_{load} = R_{net} + R_{HB} + R^{next}_{wire}. \tag{2}$$

Accurate evaluation of Eqs. (1)–(2) requires extracting C_{net}, R_{net}, C_{HB}, and R_{HB} from the physical layout. We construct a merged parasitic database by combining post-route SPEF files from both dies. Cross-tier nets are unified with explicit HB RC models inserted at the interface. The `BRIDGE` interface cells are then abstracted: corresponding pins on paired cells are collapsed into a single virtual electrical node, enabling STA tools to compute end-to-end parasitic values without double-counting interface parasitics. The computed delay and slew at `BRIDGE_OUT` become the input boundary conditions for `BRIDGE_IN`, which propagates timing downstream using standard library characterization.

The physically propagated timing quantities are converted to standard SDC constraints at the `BRIDGE` interface pins. For each cross-tier net, timing analysis on the merged 3D design extracts the signal arrival time at the driver output and the required departure time. Note that "driving die" and "load die" refer to signal direction rather than physical tier position; either die can serve either role depending on the path. These map to SDC commands as follows:

$$\begin{aligned} \text{Load die: } & \texttt{set_input_delay}(Delay_{arrival}), \\ & \texttt{set_input_transition}(slew_{out}), \\ & \texttt{set_drive}(R^{in}_{load}), \\ & \texttt{set_load}(C^{in}_{load}) \qquad (3) \\ \text{Driving die: } & \texttt{set_output_delay}(Delay_{required}) \qquad (4) \end{aligned}$$

When multiple drivers fan into the same interface, worst-case timing is applied (maximum arrival for setup, minimum for hold).

The complete process operates iteratively: (1) perform independent STA on each die with current constraints, (2) extract timing at all `BRIDGE` interface pins, (3) run STA on the merged 3D design to compute accurate cross-tier delays per Eqs. (1)–(2), (4) generate updated per-die SDC files per Eqs. (3)–(4), and (5) apply these constraints for the next optimization iteration. When timing violations occur on the receiving die, the required arrival times propagate backward through the `BRIDGE` cells, informing placement and routing optimizations on the driving die.

Cross-tier delay values derive from the merged 3D SPEF while tier-internal parasitics remain from each die's individual extraction, preserving local RC accuracy. This approach enables timing-driven optimization within commercial place-and-route tools without requiring monolithic 3D implementation support. Cross-die OCV is naturally captured by applying different process corners to each die during merged timing analysis.

4 Experimental Results

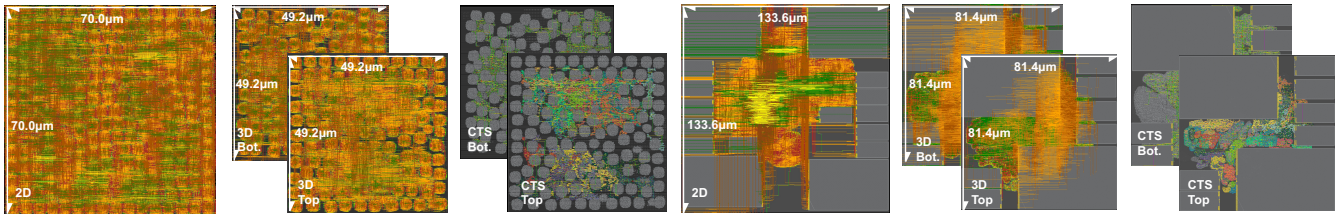
We evaluate CU3D using four logic-only benchmarks, AES, JPEG, wb_conmax [26], and b18 [27], and the open-source RISC-V processor NutShell [28]. Table 2 summarizes the benchmark characteristics. The designs cover a wide range of circuit scales and sequential densities, from the small bus interconnect matrix wb_conmax to the larger JPEG engine and NutShell CPU. All designs are implemented using an in-house predictive 3nm technology library at VDD = 0.70V and

Table 2: Experiment benchmarks.

Benchmarks	#Instances	#FFs	Description
AES	141k	10k	Cryptography hardware core
b18	47k	4.8k	Industrial control circuit
JPEG	259k	37k	Image compression engine
wb_conmax	18k	770	Bus interconnect matrix
NutShell	80k	20k	RISC-V processor core

Table 3: Comparison of PPA on four logic-only designs between 2D, Pin-3D [10], and CU3D.

Design Metrics	AES (3.3GHz)			wb_conmax (3.3GHz)			JPEG (2.5GHz)			b18 (3.3GHz)		
	2D	Pin-3D	CU3D	2D	Pin-3D	CU3D	2D	Pin-3D	CU3D	2D	Pin-3D	CU3D
Eff. Freq (GHz)	2.114	2.045	**2.146**	1.748	1.976	**2.217**	**1.522**	1.310	1.133	0.694	1.371	**1.605**
Cell Utilization (%)	53.2	54.6	53.3	50.0	61.9	62.5	52.2	48.9	56.8	52.5	54.9	58.7
Footprint (μm^2)	4900	2500	**2401**	652	**289**	324	13249	6854	**5776**	2025	1042	**961**
# Hybrid Bonds	-	8742	**2348**	-	1417	**1175**	-	14239	**2170**	-	2966	**1104**
WNS (ns)	-0.170	-0.186	**-0.163**	-0.269	-0.203	**-0.148**	**-0.257**	-0.363	-0.482	-1.139	-0.426	**-0.320**
TNS (ns)	-176.72	-190.66	**-106.57**	-68.48	-49.26	**-19.63**	-1763.79	-2203.42	**-1310.79**	-3331.29	-1168.38	**-452.16**
$P_{Internal}$ (mW)	73.7	99.9	**68.9**	**10.1**	11.2	11.4	**103.6**	140.8	105.7	47.0	45.9	**27.0**
$P_{Switching}$ (mW)	70.5	71.2	**61.0**	16.2	9.0	**8.1**	161.1	173.0	**155.2**	40.8	33.1	**30.0**
$P_{Leakage}$ (mW)	0.708	0.714	**0.664**	**0.081**	0.082	0.084	**2.110**	2.241	2.156	**0.305**	0.350	0.365
P_{Total} (mW)	144.9	171.8	**130.6**	26.4	20.3	**19.7**	266.9	316.0	**263.1**	88.2	79.4	**57.3**
Clock Skew (ns)	0.044	0.043	**0.034**	**0.017**	0.022	0.049	**0.068**	0.094	0.069	0.040	0.058	**0.035**
Clock Latency (ns)	0.198	**0.161**	0.177	0.082	**0.072**	0.119	**0.253**	0.276	0.309	0.160	0.157	**0.141**
Wirelength (m)	0.329	**0.242**	0.267	0.125	**0.046**	0.118	0.385	**0.359**	0.416	0.095	**0.075**	0.087



(a) AES (2D) (b) AES (3D) (c) AES (3D CTS) (d) NutShell (2D) (e) NutShell (3D) (f) NutShell (3D CTS)

Figure 8: Layout across 2D, 3D placement, and CU3D clock implementation on AES (a-c) and NutShell (d-f). Different colors indicate different clock domain in the top die.

Temp = 25°C. Cadence Genus [29] is used for synthesis and resynthesis, Cadence Innovus [30] is used for physical implementation, and Synopsys PrimeTime [31] is used for signoff timing analysis. For all 3D implementations, both dies use M6 as the top metal layer. To ensure a fair comparison, we reproduce Pin-3D [10] in the same technology, tool environment, and design constraints. We compare three implementations: the conventional 2D baseline, Pin-3D, and the proposed CU3D flow. The HB pitch in the implementation is set to 0.5μm in this work.

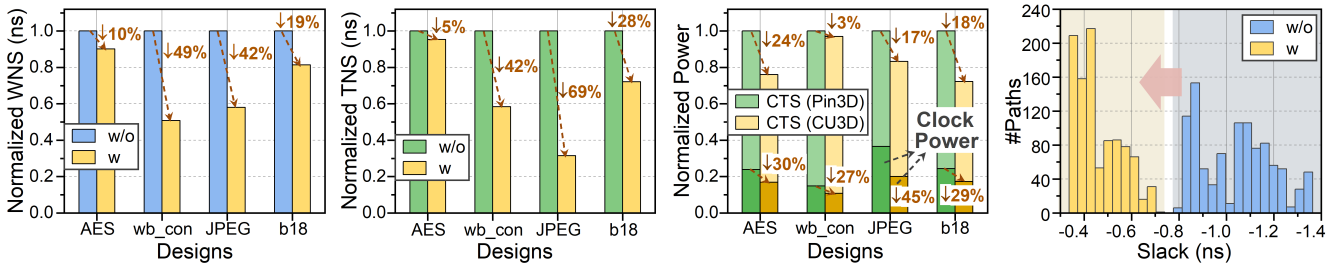
4.1 Overall PPA Comparisons

Table 3 shows the PPA comparison among 2D, Pin-3D, and CU3D for the four logic-only benchmarks. The layout comparison of AES and NutShell is shown in Figure 6. Overall, CU3D achieves better PPA than Pin-3D across most metrics while using significantly fewer HBs. Compared with Pin-3D, CU3D reduces the HB count, improves TNS, and lowers total power on all four logic-only benchmarks. Compared with the 2D baseline, CU3D provides the expected footprint reduction from vertical stacking while also improving effective frequency on AES, wb_conmax, and b18. The most significant improvement comes from HB utilization. Pin-3D relies on a pin-projection-based implementation and does not explicitly optimize the number of inter-tier cuts under array-style HB constraints. As a result, it requires a large number of HBs after tier partitioning. HB count control becomes increasingly critical in advanced technology nodes, where HB availability faces tighter constraints. In contrast, CU3D performs timing-aware tier partitioning from the 2D placement result and preserves placement locality.

CU3D also improves timing closure compared with Pin-3D on most benchmarks. For AES, WNS improves from -0.186ns to -0.163ns, and TNS improves from -190.66ns to -106.57ns. For b18, WNS improves from -0.426ns to -0.320ns, while TNS improves from -1168.38ns to -452.16ns. Although JPEG has worse WNS than Pin-3D, its TNS is reduced from -2203.42ns to -1310.79ns, indicating that CU3D still reduces the overall amount of timing violations. This trend suggests that CU3D improves the global timing distribution rather than only optimizing a small number of worst paths.

In terms of power, CU3D consistently achieves lower total power than Pin-3D. The power reduction mainly comes from shorter interconnects, reduced switching capacitance, and the proposed latency-aware 3D clock tree synthesis. For example, on AES, CU3D achieves 10% lower total power than the 2D baseline and 24% lower total power than Pin-3D. The footprint results further demonstrate the advantage of 3D integration. The layouts in Figure 6 also show this footprint reduction visually. For AES, the 2D layout occupies a 70.0$\mu m \times$ 70.0μm footprint, while the 3D layout uses a 49.2$\mu m \times$ 49.2μm footprint. For NutShell, the footprint is reduced to 81.4$\mu m \times$ 81.4μm.

Table 4 shows the detailed results on NutShell. NutShell is more challenging than the logic-only benchmarks because it contains a denser sequential structure and more complex timing paths. The 2D baseline achieves 0.51GHz effective frequency with 56.8mW total



(a) WNS improvement (b) TNS improvement (c) Power reduction (d) Slack histogram of NutShell

Figure 9: (a)-(b) WNS and TNS improvement after cross-tier timing optimization. (c) Normalized power comparison between Pin-3D and CU3D. (d) Slack histogram of NutShell after timing optimization.

Table 4: Comparison of metrics between 2D, CU3D without cross-die optimization, and full CU3D flow on NutShell CPU.

NutShell (1GHz)	2D	CU3D	
		Pre-Opt	Post-Opt
Eff. Freq (GHz)	0.511	0.426	**0.584**
Footprint (μm^2)	17689	6625	**6561**
# Hybrid Bonds	-	2808	2808
WNS (ns)	-0.955	-1.343	**-0.711**
TNS (ns)	-2522.2	-2596.0	**-639.2**
$P_{Internal}$ (mW)	20.6	13.9	**13.7**
$P_{Switching}$ (mW)	35.3	19.5	**17.8**
$P_{Leakage}$ (mW)	0.979	0.589	0.616
P_{Total} (mW)	56.8	34.0	**32.1**
Clock Wirelength (mm)	12.78	8.57	**8.18**
Clock Skew (ns)	0.05	0.09	**0.04**
Clock Latency (ns)	0.284	0.303	**0.235**
Wirelength (m)	0.299	0.241	**0.234**

power and a 17689μm^2 footprint. Before cross-tier timing optimization, CU3D already reduces the footprint to 6625μm^2 and total power to 34.0mW, but its WNS degrades to -1.343ns because each die is still optimized with incomplete cross-tier timing information. After cross-tier timing optimization, CU3D improves WNS from -1.343ns to -0.711ns and TNS from -2596.0ns to -639.2ns. The final effective frequency reaches 0.584GHz, which is 14.3% higher than the 2D baseline. At the same time, total power is reduced from 56.8mW to 32.1mW, and footprint is reduced from 17689μm^2 to 6561μm^2.

The NutShell clock statistics explain the power and timing improvements. CU3D reduces clock wirelength from 12.78μm in 2D to 8.18μm after optimization, directly lowering clock switching activity and reducing switching power from 35.3μW to 17.8μW. Clock skew improves from 0.05ns to 0.04ns, and latency from 0.284ns to 0.235ns. The proposed 3D CTS reduces power while maintaining a balanced clock network across tiers.

4.2 Ablation Study

To isolate the contribution of timing and clock optimizations, we conduct ablation studies in Figure 9. The "w/o" results denote implementation without cross-tier timing update, while "w" denotes the complete CU3D flow. Figure 9(a) and Figure 9(b) report WNS and TNS improvements, respectively. CU3D consistently improves both metrics across the four logic-only benchmarks. WNS improvement ranges from 10% to 49%, with JPEG showing the largest reduction of 49%. TNS improvement ranges from 5% to 69%, with JPEG again showing the largest reduction. By propagating delay, slew, and load across the tier boundary, CU3D enables optimization with timing information from the other die.

Figure 9(c) reports normalized power comparison between 3D CTS method of Pin-3D and CTS method of CU3D. CU3D reduces total power on all designs, with reductions ranging from 3% to 45%, mainly due to latency-aware 3D CTS. Instead of building a single long clock network, CU3D clusters top die clock sinks into local groups and performs bottom-up latency balancing from the bottom die. This reduces long lateral clock routes, lowers buffer insertion, and decreases switching capacitance. Figure 9(d) shows the slack histogram of NutShell after timing optimization. Compared with baseline, CU3D shifts the slack distribution right and reduces severely violating paths. Together with PPA results in Table 3 and Table 4, the ablation study demonstrates that CU3D's HB-aware partitioning, 3D CTS, and cross-tier timing update jointly improve F2F 3D IC implementation quality.

5 Conclusion

In this paper, we presented CU3D, a comprehensive F2F 3D IC design flow that systematically addresses tier partitioning, 3D CTS, and timing optimization challenges. Our timing-driven tier partitioning preserves critical path locality and satisfies array-style HB constraints while maintaining design quality. The latency-aware 3D CTS ensures balanced clock distribution across tiers with cross-tier timing propagation, eliminating path pessimism. Experimental validation on industrial benchmarks demonstrates substantial improvements over state-of-the-art methods in overall PPA, with significant reductions in HB count and power consumption, validating our holistic design methodology.

6 Acknowledgments

We extend our sincere gratitude to Professor Sung Kyu Lim and his student, MinGyu Park from the University of Southern California, for their technical assistance on the Pin-3D replication. This research was partially conducted by ACCESS – AI Chip Center for Emerging Smart Systems, supported by the InnoHK initiative of the Innovation and Technology Commission of the Hong Kong Special Administrative Region Government. This work is jointly supported by the Research Grants Council of Hong Kong SAR No. CUHK14211324 and No. T46-415/25-R.

References

[1] D. W. Fisher, S. Knickerbocker, D. Smith, R. Katz, J. Garant, J. Lubguban, V. Soler, and N. Robson, "Face to Face Hybrid Wafer Bonding for Fine Pitch Applications," in *IEEE Electronic Components and Technology Conference*, 2020.

[2] S. Pentapati and S.-K. Lim, "Heterogeneous Monolithic 3-D IC Designs: Challenges, EDA Solutions, and Power, Performance, Cost Tradeoffs," *IEEE Transactions on Very Large Scale Integration Systems (TVLSI)*, vol. 32, no. 3, pp. 413–421, 2024.

[3] V. Moroz, X. Xu, A. Svizhenko, X.-W. Lin, S. Popov, H. Sheng, and K. Larsen, "3DIC System-Technology Co-Optimization with a Focus on the Interplay of Thermal, Power, Timing, and Stress Effects," in *Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits)*, 2024.

[4] G. Yeap, S. Lin, H. Shang, and et al., "2nm Platform Technology Featuring Energy-Efficient Nanosheet Transistors and Interconnects Co-Optimized with 3DIC for AI, HPC and Mobile SoC Applications," in *IEEE International Electron Devices Meeting (IEDM)*, 2024.

[5] J. Wuu, R. Agarwal, M. Ciraula, C. Dietz, B. Johnson, D. Johnson, R. Schreiber, R. Swaminathan, W. Walker, and S. Naffziger, "3D V-Cache: the Implementation of a Hybrid-Bonded 64MB Stacked Cache for a 7nm x86-64 CPU," in *IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 65, 2022.

[6] T. F. Wu, H. Liu, H. E. Sumbul, L. Yang, D. Baheti, J. Coriell, W. Koven, A. Krishnan, M. Mittal, M. T. Moreira *et al.*, "11.2 A 3D integrated Prototype System-on-Chip for Augmented Reality Applications Using Face-to-Face Wafer Bonded 7nm Logic at< 2μm Pitch with up to 40% Energy Reduction at Iso-Area Footprint," in *IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 67, 2024.

[7] L. Witters, S. Van Huylenbroeck, S. Kang, and et al., "High-Density Wafer Level Connectivity Using Frontside Hybrid Bonding at 250nm Pitch and Backside Through-Dielectric Vias at 120nm Pitch After Extreme Wafer Thinning," in *Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits)*, 2025.

[8] J. De Vos, J. Myers, S. Van Huylenbroeck, L. Witters, E. Chery, M. Stucchi, P. Zhao, K. D'havé, M. Vanrusselt, S. Iacovo, H.-C. Tsai, S. Dewilde, L. Bogaerts, C. Heyvaert, S. A. Chew, B. Zhang, S. Kang, M. Brunion, D. Biswas, G. Beyer, G. Van der Plas, Z. Tokei, and E. Beyne, "Towards SoCs leveraging 200nm wafer-to-wafer interconnection pitch," in *IEEE International Electron Devices Meeting (IEDM)*, 2025.

[9] M. G. Park, P. Vanna-Iampikul, and S. K. Lim, "Closing the Gap: Advantages of Block-Level over Gate-Level in 3D IC Design for Advanced Nodes," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2025.

[10] 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 *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2020.

[11] 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," in *ACM International Symposium on Physical Design (ISPD)*, 2018.

[12] P. Vanna-iampikul, J. Yoon, C. Park, G. Yeap, and S. K. Lim, "Placement-Aware 3D Net-to-Pad Assignment for Array-Style Hybrid Bonding 3D ICs," in *ACM International Symposium on Physical Design (ISPD)*, 2025.

[13] K. Chang, S. Sinha, B. Cline, G. Yeric, and S. K. Lim, "Design-Aware Partitioning-Based 3-D IC Design Flow With 2-D Commercial Tools," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 41, no. 3, pp. 410–423, 2021.

[14] 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," in *ACM/IEEE Design Automation Conference (DAC)*, 2026.

[15] X. Zhao, J. Minz, and S. K. Lim, "Low-Power and Reliable Clock Network Design for Through-Silicon Via (TSV) Based 3D ICs," *IEEE Transactions on Components, Packaging and Manufacturing Technology (TCPMT)*, vol. 1, no. 2, pp. 247–259, 2010.

[16] T.-Y. Kim and T. Kim, "Clock Tree Synthesis for TSV-based 3D IC Designs," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 16, no. 4, pp. 1–21, 2011.

[17] J. Zhou, Z. Ge, C. Xi, and P. Zhou, "Inductance-aware Clock Network Synthesis Considering Hierarchical Interconnects in 3D ICs," in *ACM Great Lakes Symposium on VLSI (GLSVLSI)*, 2025.

[18] G. Kim and H. Park, "A Unified Design Flow for Homogeneous and Heterogeneous 3D Integration with Fine-Pitch Hybrid Bonding," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2025.

[19] Y.-H. Huang and S. K. Lim, "Snake-3D: Differentiable Learning for Cross-Tier Logic Path Snaking Optimization in 3D ICs," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2025.

[20] D. Kim, M. Kim, J. Hur, J. Lee, J. Cho, and S. Kang, "TA3D: Timing-Aware 3D IC Partitioning and Placement by Optimizing the Critical Path," in *ACM/IEEE Workshop on Machine Learning CAD (MLCAD)*, 2024.

[21] 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," in *ACM/IEEE Design Automation Conference (DAC)*, 2025.

[22] L. Zhou, S. Xiao, H. Wang, J. Wang, Z. Xu, B. Wang, and Z. Yu, "Toward Efficient Asynchronous Circuits Design Flow Using Backward Delay Propagation Constraint," *IEEE Transactions on Very Large Scale Integration Systems (TVLSI)*, vol. 32, no. 10, pp. 1852–1863, 2024.

[23] Y.-C. Lu, Z. Guo, K. Kunal, R. Liang, and H. Ren, "INSTA: An Ultra-Fast, Differentiable, Statistical Static Timing Analysis Engine for Industrial Physical Design Applications," in *ACM/IEEE Design Automation Conference (DAC)*, 2025.

[24] Y. Ji, Y. Lu, Z. Zhang, Z. Guo, Y. Lin, and B. Yu, "DiffCCD: Differentiable Concurrent Clock and Data Optimization," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2025.

[25] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[26] OpenCores: The reference community for Free and Open Source gateware IP cores, https://opencores.org/.

[27] F. Corno, M. Sonza Reorda, and G. Squillero, "RT-level ITC'99 benchmarks and first ATPG results. https://www.cerc.utexas.edu/itc99-enchmarks/bendoc1.html," *IEEE Design & Test*, pp. 44–53.

[28] H. Wang, Z. Zhang, Y. Jin, L. Zhang, and K. Wang, "A Linux-Compatible RISC-V Processor Designed by Undergraduates. https://github.com/OSCPU/NutShell," *RISC-V Global Forum*, 2020.

[29] Cadence, "Genus User Guide", https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/synthesis/genus-synthesis-solution.html, 2026.

[30] Cadence, "Innovus Reference Manual", https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/soc-implementation-and-floorplanning/innovus-implementation-system.html.

[31] Synopsys, "Primetime User Guide", https://www.synopsys.com/implementation-and-signoff/signoff/primetime.html, 2021.