Hier-3D: A Methodology for Physical Hierarchy Exploration of 3-D ICs

Nesara Eranna Bethur, Anthony Agnesina, Moritz Brunion, Alberto Garcia-Ortiz, *Senior Member, IEEE*, Francky Catthoor, *Fellow, IEEE*, Dragomir Milojevic, Manu Komalan, *Member, IEEE*, Matheus Cavalcante, *Student Member, IEEE*, Samuel Riedel, *Member, IEEE*, Luca Benini, *Fellow, IEEE*, and Sung Kyu Lim, *Fellow, IEEE*

***Abstract*—Hierarchical very-large-scale integration (VLSI) flows are an understudied yet critical approach to achieving design closure at giga-scale complexity and gigahertz frequency targets. This article proposes a novel hierarchical physical design flow enabling the building of high-density and commercial-quality two-tier face-to-face-bonded hierarchical 3-D ICs. Complemented with an automated floorplanning solution, the flow allows for system-level physical and architectural exploration of 3-D designs. As a result, we significantly reduce the associated manufacturing cost compared to existing 3-D implementation flows and, for the first time, achieve cost competitiveness against the 2-D reference in large modern designs. Experimental results on complex industrial and open manycore processors demonstrate in two advanced nodes that the proposed flow provides major power, performance, and area/cost (PPAC) improvements of 1.2-2.2× compared with 2-D, where all metrics are improved simultaneously, including up to 20% power savings.**

***Index Terms*—Bonded 3-D ICs, face-to-face (F2F), hier-3D, physical design methodology, wafer-level bonding.**

I. Introduction

TO DELIVER the perceived benefits of 3-D ICs outside the purview of research and academia [1], a hierarchical 3-D design flow must subdivide complex, manycore, large-memory giga-scale designs into sub-blocks, which are independently synthesized and physically placed-and-routed (P&R) as separate design units. Then, the resultant mapped sub-blocks are recombined into subsequent runs of higher-level blocks—a process repeated as the hierarchy is traversed up to the top level.

This hierarchical approach offers the following benefits: 1) large designs can be implemented with acceptable runtime and memory usage, where typically significant reuse is made of (nearly) identical sub-blocks. It is infeasible with today's machines and tools to implement industrial-size SoC designs flat; 2) concurrent implementation of design tasks can be split across multiple development teams; and 3) third-party intellectual property (IP) blocks can be elegantly integrated as a natural part of the process flow. While a hierarchical flow mitigates many issues of a flat approach, numerous tedious and error-prone tasks are still required to close timing and optimize PPA at the top-level netlist. These include budgeting for block-level timing constraints and setting appropriate hierarchical physical constraints.

EDA vendors' tools partially manage these remaining issues in their proposed hierarchical flows that can simplify the implementation of large 2-D designs. However, none of these flows are currently optimized for 3-D hierarchical implementations. Instead, academic work [2], [3] focuses on sequential die-by-die approaches, where hierarchy levels are artificially created to use standard block-level flows where blocks are placed on different tiers and routed in 3-D. Furthermore, in these hierarchical flows, the blockage of macros makes placement and routing much harder on higher hierarchy levels than in flat implementations.

Moreover, the physical hierarchy decisions, including the floorplanning of blocks in 3-D, are usually left to expert engineers and preconstrained, like memory-on-logic. These can significantly impact the power, performance, area, and cost (PPAC) metrics, the latter becoming most important due to increasing wafer costs. In addition, when designing a 3-D floorplan, having a structured approach that facilitates manual design decisions can be advantageous. While a human designer can provide valuable guidance on placing critical components in a global 3-D floorplan, creating a detailed floorplan for all components can prove challenging. Therefore, there is a strong need to automate this task for a more streamlined and efficient design process.

In this work, we propose *Hier-3D*, a physical design methodology for 3-D bottom-up hierarchical implementations to co-optimize power, performance, and area/cost combined design metrics, as modern-day 3-D flows do not satisfactorily

Manuscript received 13 May 2023; revised 30 October 2023; accepted 4 December 2023. Date of publication 14 December 2023; date of current version 20 June 2024. This work was supported in part by the Ministry of Trade, Industry and Energy of South Korea under Grant 1415187652 and Grant RS-2023-00234159; in part by the Semiconductor Research Corporation (CHIMES) under Grant 3136.002; and in part by the DOE Office of Science Research Program for Microelectronics Co-Design (Abisko). This article was recommended by Associate Editor V. Pavlidis. *(Corresponding author: Nesara Eranna Bethur.)*

Nesara Eranna Bethur is with the Advanced Micro Devices Inc. Austin, Austin, TX 78735 USA (e-mail: NesaraEranna.Bethur@amd.com).

Anthony Agnesina is with the NVIDIA Corporation, Santa Clara, CA 95050 USA.

Moritz Brunion, Francky Catthoor, Dragomir Milojevic, and Manu Komalan are with IMEC, 3001 Leuven, Belgium.

Alberto Garcia-Ortiz is with the Electrical and Computer Engineering Department, University of Bremen, 28359 Bremen, Germany.

Matheus Cavalcante, Samuel Riedel, and Luca Benini is with the Electrical and Computer Engineering Department, ETH Zürich, 8092 Zürich, Switzerland.

Sung Kyu Lim is with the Electrical and Computer Engineering Department, Georgia Institute of Technology, Atlanta, GA 30332 USA.

Digital Object Identifier 10.1109/TCAD.2023.3342753

address the latter. The key contributions of this article, an extension of [4], are as follows:

1) We propose a first-of-its-kind hierarchical physical design flow for 3-D designs, significantly improving the placement and routing utilization across the 3-D stack by reusing the unassigned silicon area of preceding hierarchy levels.
2) Our Hier-3D flow exploits the inherent logical hierarchy to enable hierarchical multitier standard cell placement, greatly expanding the design space of 3-D ICs. We demonstrate the flexibility of the proposed implementation flow by exploring the architecture configuration space for a selected design.
3) We develop a high-level automated solution spanning RTL clustering, 3-D logic-on-logic floorplanning, and holistic 3-D physical implementations. This effort enables the automated architecture design exploration on a multilevel 3-D physical hierarchy while still enabling predefined placements imposed by manual guidance from the human designer.
4) We design a novel whitespace modeling flow that works across different physical design flows to propagate the physical shapes of the standard cells as an abstraction to the following hierarchy. This promotes further fine-grained area savings across the 3-D stack as the area can now be utilized in a pure logic die for the subsequent hierarchies' placement but also enables feedthrough buffer insertion for better timing across hierarchies.
5) We demonstrate 1.2–2.2× PPAC improvements and 1.2–1.5× runtime speedup on three highly diverse open-source and industrial low-power benchmarks and, for the first time, cost improvement compared to the 2-D reference. These results are incommensurate with 2-D and standard commercial and academic 3-D flows.

II. Preliminaries

This section presents motivations for the proposed approach, including preliminaries about hierarchical implementations and current state-of-the-art 3-D block-level methodologies.

A. Hierarchical Methodologies

In general, hierarchical EDA flows typically must provide the following capabilities:

1) Partitioning the design logically and physically into the top-level design and the various partition blocks.
2) Automatic pin assignment for partitions, which guides the interpartitions global routes.
3) Feedthrough insertion of nets and buffers into partitions to allow routing nets to cross over partition areas without creating significant detours, maintaining the signal's integrity and performance.
4) Timing budgeting that apportions budgets to blocks. This is a chicken or the egg problem, as proper budgets depend on the timing inside the partitions.
5) Assembling partitions for top-level sign-off closure.

Commercial EDA tools offer efficient databases (DBs), flows, and commands for engineers to solve these practical issues. However, the tools are currently restricted to 2-D designs and were not designed for high-level architectural exploration.

Moreover, 3-D integration introduces a new layout axis that provides opportunities for optimization but also increases design complexity. It requires making multiple complex choices for physical hierarchy and system architecture. Therefore, a flexible 3-D exploration flow that combines human decision-making with automated assistance for less critical decisions is necessary to address this complexity.

B. 3-D Floorplanning

Floorplanning is a crucial step in the very-large-scale integration (VLSI) physical design flow. Large sub-blocks representing IPs, already implemented partitions, clusters of unplaced standard cells, or memory macros must be placed on a 2-D or 3-D canvas at each hierarchy level. The decisions at that stage are essential as they heavily dictate the achievable P&R quality at the top level of the chip.

Today, 2-D floorplanning is at the forefront of research [5], [6]. However, research on 3-D floorplanning is more limited. Due to the limited capabilities of EDA tools and manufacturing methods for 3-D designs, current academic works, and industrial applications are confined to memory-on-logic implementations with restricted architectural explorations and manual floorplanning [7]. One successful example of constrained memory-on-logic floorplanning is AMD's Ryzen V-Cache 3-D IC [8], where L3 caches are stacked above the core units to increase the on-chip last level cache capacity by 200 % while maintaining the footprint. On the other hand, automated floorplanning research mainly concentrates on extending classical combinatorial 2-D algorithms to 3-D. First, the underlying data structures that efficiently encode the floorplanning space, e.g., normalized Polish expressions, sequence pairs, corner block lists, and O/B*-trees [2], [9], [10], are augmented to include the third layout axis. Then, transformations on the data structures are defined, including 3-D changes to the floorplan, guided by optimization methods such as simulated annealing (SA). The main suboptimality reasons for these methods come from the difficulty of defining appropriate cost metrics to judge the quality of a 3-D floorplan that correlates well with actual physical implementations and optimizing many competing metrics simultaneously. Moreover, these lack proper feedback from actual physical implementations. We will fill these gaps in our paper: the proposed flow for efficient 3-D hierarchical physical implementations is presented in Section III, and the proposed floorplanning methodology with new 3-D cost components is shown in Section IV.

C. State of 3-D Flows

In the following, we present two existing state-of-the-art approaches to implementing face-to-face (F2F) wafer-to-wafer (W2W) bonded 3-D ICs: 1) a die-by-die-like Sequential-2D [11] and 2) a Macro-3D flow [12].

1) Sequential-2D Flow: The Sequential-2D flow follows the EDA industry's approach to implementing 3-D designs.

Key Idea: It enables 3-D integration through the separate implementations of a die stack's top and bottom die with a

standard 2-D flow. 3-D physical awareness is established by defining pins inside the core area at valid copper pad locations.

Strengths: This approach does not restrict the partitioning scheme, as logic and memory cells can be placed in both dies. Moreover, it reuses the standard commercial 2-D tools capabilities, modeling the F2F bonding pads as IO pin shapes.

Limitations: Partitioning a design into a top and bottom die inherently introduces an additional level into the implementation hierarchy. If it does not follow the natural partitioning provided by the logical hierarchy, it requires a challenging additional constraint modeling step that can introduce PPA degradation. Moreover, sharing both dies' back-end-of-line (BEOL) resources would necessitate the insertion of feedthroughs in the netlist of each die for each shared intradie net, an approach highly inflexible and impractical without additional automation efforts.

Hierarchical Design: The Sequential-2D flow can be applied to a hierarchical design by introducing an additional hierarchy level into each block and forming a top and bottom sub-blocks. Implementations of subsequent hierarchy levels must respect the separated child block implementations in both dies.

2) Macro-3D Flow: Macro-3D provides state-of-the-art PPA optimization capabilities for 3-D ICs in the memory-on-logic partitioning scheme.

Key Idea: The commercial 2-D P&R tool is made aware of the complete die stack. The pins and routing obstructions of the memory macros are projected to the corresponding top layer to yield a holistic memory-on-logic flow. The copper pads are modeled as regular vias, allowing their automated insertion by a traditional global router and inherently incorporating the impact of their parasitics on timing and power.

Strengths: Standard P&R engines can optimize the complete design for timing closure because of the complete design view across both dies. Further, the holistic stack view enables a unified routing step of both dies, allowing metal layer sharing, i.e., nets with a start- and endpoint in the same die can borrow metal resources from the other die, resulting in a more uniform metal layer utilization.

Limitations: The silicon area of the memory and the logic die are usually very different. Therefore, as W2W-based 3-D integration requires matching die sizes, the Macro-3D flow increases the resulting manufacturing cost relative to the 2-D die if the lost space cannot be reclaimed.

Hierarchical Design: Macro-3D can implement designs hierarchically by abstracting sub-blocks as full-block obstructions. However, by obstructing all metal layers between the front-end-of-lines (FEOLs) of both dies, routing through the abstraction is prohibited, and routing over it is impossible.

III. Design Methodology

Focusing on die area savings is essential for semiconductor industries to save on die costs. Having this as our motivation, we solve the die area minimization problem via the 3-D stack interhierarchy area utilization. We propose our Hier-3D flow to mitigate the issues presented above. Targeted explicitly for silicon area minimization, it allows the building of high-density hierarchical commercial-quality

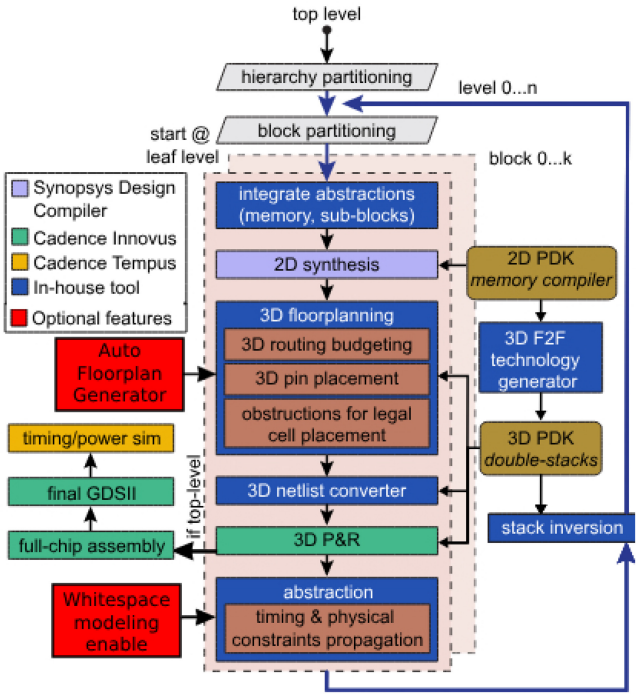


Fig. 1. Key steps in our Hier-3D flow. The hierarchy depth defines the number of outer cycles/iterations. Per iteration, synthesized block-level designs are prepared with 3-D floorplanning, 3-D P&R, and abstracted through our physical and timing constraints propagation. optional features of automatic floorplan generator and whitespace modeling are also provided. Finally, the stack and abstractions can be inverted to enable standard cell placement on the opposite die at the upper hierarchy level.

F2F-bonded 3-D ICs in a bottom-up fashion. It combines the previously presented advantages of the Sequential-2D and Macro-3D flows and introduces new key features to address their shortcomings, as summarized in Table I.

Further, we propose an automated floorplanning solution which allows predefined guidance by a human designer to produce physical hierarchy assignments that can be implemented using the Hier-3D methodology. For physical hierarchy assignments, we refer to the physical block creations enabled by our RTL clustering for optimum automated floorplans. This is further discussed in Section IV. Our flow also enables hierarchies defined manually from the existing logical hierarchies, which we discuss in the following section.

The high integration density targeted by Hier-3D has vast implications on the PPAC characteristics. Indeed, dense block packing can increase the number of dies per wafer for cost, eliminate long timing-critical wires and reduce interconnecting energy by reducing distances. In addition, maintaining a holistic view across the die stack avoids overconstraining the block interfaces and enables efficient power optimization capabilities.

A. Flow Overview

Our overall 3-D hierarchical flow is represented in Fig. 1. The flow starts with an RTL whose hierarchy is logically predefined or manually created, e.g., from high-level floorplanning. Then, we synthesize each block within a given hierarchy level using the timing abstractions of the lower-level sub-blocks. Next, each block undergoes a 3-D floorplanning step that places pins in the 3-D stack and preserves routing

TABLE I
QUALITATIVE COMPARISON AMONG STATE-OF-THE-ART PHYSICAL DESIGN TOOLS FOR 3-D ICS AND THIS WORK

	Sequential-2D	Macro-3D [12]	Hier-3D
key idea	implement two 2D dies separately	holistic memory-on-logic	hierarchy scheme for 3D
3D stack	two separate dies	double metal stack	double metal stack
main strengths	reuse of 2D environment, macros and standard cells in both tiers	full utilization of metal resources	full utilization of metal resources, maximize silicon area utilization
main weakness	dies are designed separately: limited optimization	limited to memory-on-logic: best suited for equal die area	inherently hierarchical: best suited for large designs
restricted partitioning	no	yes	no
holistic routing	no	yes	yes
utilize unused space	no	no	yes

resources for easier access and routing in the next step, respectively. If the automated floorplan is enabled, the 3-D floorplanning step also incorporates the automated floorplan generation described in Section IV.

This PDK includes shrunk cover memory macros and is updated at every level with the newly generated sub-blocks abstractions. The current block is then implemented using the timing and physical abstractions of the lower hierarchy levels. Please note that a given block implementation can span one die (=2-D) or two dies (=3-D), depending on the designer's choice. Next, the resulting implemented block is abstracted with our physical/timing constraints propagation method, including an optional whitespace modeling. Finally, the stack and the view of the abstracted block can be inverted for the next step to place standard cells on the opposite die. This loop continues until all hierarchy levels have been implemented.

The remainder of this section focuses on the detailed presentation of the critical steps of the Hier-3D flow.

B. Hierarchy Partitioning

The proposed flow offers the flexibility to use the logic hierarchy as is or manually define hierarchies from the existing logic hierarchies. We have developed a flow step around *Cadence Genus* to restructure the netlist, generate new SDCs, and manually place the IO pins. This provides the designer an additional knob for extracting the most from the 3-D stack interhierarchy area utilization, promoting further die area savings.

C. 3-D Floorplanning

Apart from the automated floorplanning that will be discussed in Section IV, the other main steps in this stage are as follows:

1) Holistic 3-D Routing Resource Budgeting: The routing resources may need to be budgeted individually by planning and reserving resources at the block level to ease the routing in the upper hierarchy level. For example, if the first level utilizes all metal layers in the doubled stack, very inefficient detours of critical nets through the die stack might occur in the following hierarchy level. We circumvent this by constraining the routing of the nets that do not require both BEOLs' traversal. In our experiments, our budgeting balances the routing resources between sequentially implemented sub-blocks by restricting nets to the die where standard cells are being placed, through the intermediary of the *Cadence Innovus* command `set_route_attributes`.

2) 3-D Pin Placement in Double Metal Stack: Our proposed methodology can exploit the tool's capability to freely assign a layer (z dimension) and all the block area (x, y dimensions) when placing the pins. By appropriately selecting the layers of the pins, the routing of the subsequent hierarchy level can be guided to utilize a particular die, offering additional options to plan the routing resource allocation.

Our in-house script automates the pin placement by creating a staggered pin grid with routing keep-out-zones (KoZ). These zones force the router to legalize in advance the F2F via placement on the pin in the following step. The pin grid also allows a denser signal routing for very wide IO busses, which would otherwise allocate many routing and placement resources for fan-out and fan-in only. The KoZ dimensions must be superior to the F2F pitch and are empirically set to $5\times$ the F2F pitch in our experiments to allow a design rule check (DRC)-clean routing solution in advanced nodes.

D. 3-D P&R: Sequential Multitier Cell Placement

Fig. 2 illustrates the standard cell placement on the second hierarchy level in Hier-3D. Standard cells can utilize the unused silicon area of the upper die, while in Macro-3D, an additional silicon area around the block is required, increasing the floorplan.

To better exploit the capabilities/assumptions of the P&R tool meant for standard 2-D environments during clock tree synthesis (CTS) and routing, we invert the 3-D stack, including the abstractions, while traversing the implementation hierarchy. The inversion of a block/stack consists in swapping the top and bottom layer names inside the Library Exchange Format (LEF)/technology LEF (TECHLEF) files, as *Mj_bot* $\leftrightarrow$ *Mj_top*. As a result, standard cells are always placed on the "bottom" die from the tool's perspective. During CTS, tree segment definitions assume that the top segment is defined to a higher metal layer than the trunk. However, with a holistic 3-D stack, the clock tree should ideally branch into two trunk and leaf definitions for the two FEOLs. This assumption remains valid with the inverted stack during the standard cell placement step. The clock balancing across tiers is simplified by the bottom-up hierarchical approach and by placing all standard cells on one tier per step. In addition, the opposite-tier macros—blocks or memories—have a balancing requirement automatically integrated inside their Liberty (LIB)

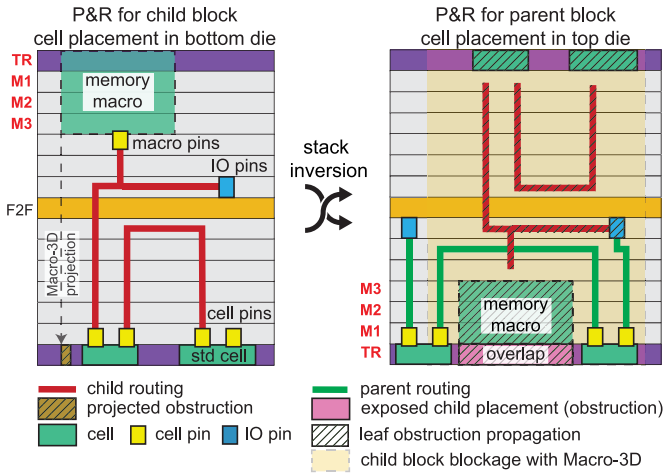


Fig. 2. Hier-3D's physical constraint propagation, stack inversion, multitier cell placement. Top macros are projected as site-sized cells at the bottom to not obstruct standard cell placement, and IO pins can be placed anywhere inside the stack to facilitate upper-level connections(=left). The physical routing/placement information is propagated to the next level to allow the P&R of the top die with the inverted stack(=right).

through the max_clock_tree_path attribute. Moreover, during routing, the assumption of reducing electrical resistance with higher metal layers holds for the FEOL of the standard cells currently being placed, leading to a more standard metal layer configuration and, therefore, more effective use of the router's heuristics.

By default, the blocks' LEF obstructions do not prevent the P&R engine from placing cells at illegal positions due to the presence of routed wires from the lower-level hierarchy. Therefore, we purposely replicate the LEF obstructions by creating special wire shapes on metal layers where standard cells have their pin shapes in our target TSMC technologies (M1 for signal pins and M2 for power and ground rails). Finally, we force the tool to check for pin DRC violations during cell placement which then enforces a valid cell placement. We also insert routing blockages on M1 and M2 over the obstructions of the OVERLAP layer to model the presence of the internal pin shapes of the cells in the sub-blocks, which are not propagated as OBS in the detailed LEF.

E. Abstraction

1) Physical/Timing Constraint Propagation: To enable the utilization of unused placement and routing resources by the P&R engines, we extend the LEF abstracts of implemented sub-blocks to enumerate all objects and structures in the placement and routing DB instead of wholly occupying all resources in the sub-block area. The physical abstractions are represented as "detailed" LEFs with BLOCK class type. In particular, the top memory macros projected as site-sized virtual cells during the current block implementation are exported as obstructions on the OVERLAP layer and as detailed routing obstructions (OBS statements) for the next hierarchical level, as shown in Fig. 2. The FULLDRC attribute is added to the OBS statements in the LEF so that the router considers them as real shapes with full DRC checking rules and cross-coupling considerations. This reduces signal integrity issues

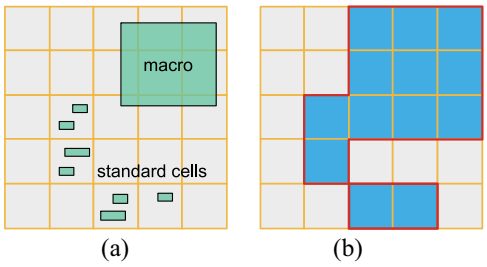


Fig. 3. Hier-3D's whitespace modeling. (a) Tiling applied over the placed shapes of cells (standard, macro, existing OBS on OVERLAP layer). The coarseness of the user-defined tiling provides a tradeoff between runtime and area recovered. (b) Occupied tiles are merged into a set of rectilinear polygons to define OVERLAP obstructions for the detailed LEF.

that can degrade the timing of the sub-blocks. Because single standard cells cannot be propagated individually due to the shape complexity, which would cause high memory usage and file size, their shapes can be first clustered into much larger 2-D polygons and saved as rectilinear-shaped overlaps similar to the memory macros. We present this approach in the following section. This approach enables a full context view of the current level and implemented sub-blocks with reduced memory requirements. Besides, because the router is now free to route through partitions, we improve the routing availability without the large runtime downsides of assembling sub-blocks as partitions. In addition, accurate timing representations are modeled by timing arcs from post-route extracted LIB files.

2) Whitespace Modeling: We must consider the placement of standard cells and macroblocks to enable fully-capable physical information propagation. This is useful to reserve placement space for the upper levels (e.g., for feedthrough buffers insertion) or when sub-blocks exhibit low cell density with large unused placement regions that should be reclaimed to optimize silicon area utilization. However, the standard cell count prohibits propagating their shapes individually, increasing LEF size and slowing down the EDA tool.

Thus, we propose a whitespace modeling method for efficient placement information propagation, depicted in Fig. 3. First, we reduce the complexity of handling the numerous geometrical shapes of the standard cells by tiling the floorplan. In practice, we use tiles of size $20\times$ the site size. This parameter is, however, tunable, defining the coarseness of the placement information propagation. From the tiling applied over the floorplan (that is the bottom die where standard cells were lastly placed), we build a binary matrix O which encodes the occupation of tiles

$$O(i,j) = \mathbf{1}\left[\exists \text{ cell} \in \text{tile}(i,j)\right] \tag{1}$$

where both standard and macro cells (memories and OBS on OVERLAP layer from sub-blocks) are considered. Finally, we rely on the efficient computational geometry primitives inside the EDA tool principally available for DRC checking to query objects and process shapes.

The rectangular shapes of the occupied tiles are unioned to merge touching shapes, cropped to the floorplan box FP, yielding a set of connected components with polygons

$$P = \cup_{\bigcirc}((\cup_{\square} O) \cap_{\square} FP) \tag{2}$$

```
input: netlist N, area threshold t
output: clusters from cluster(N\macros(N), t) ∪ macros(N) ∪ remaining set of unclustered cells
  cluster(O, t):
  C = ∅
  if area(O) ≥ t then
    Q = sub-modules(O); sort Q by decreasing area
    if area(Q) ≥ t then
      for q in Q do
        R = sum of area(i) for i after q in Q
        C = C ∪ cluster(q, t)
        if R < t then
          break
        end if
      end for
    else
      C = O
    end if
  else
    C = O
  end if
  return C
```

Fig. 4. Automated RTL clustering.

where $\square$ and $\bigcirc$ denote operations yielding rectangles and rectilinear and convex polygons, respectively. Reducing rectangles into polygons dramatically decreases the number of shapes to handle in the LEF for higher runtime scalability in the EDA tool. Moreover, one can apply a filter F to expose only a specific part of the placement (e.g., floorplan boundary only)

$$\widetilde{P} = F \cap_{\bigcirc} P. \tag{3}$$

During the implementation of the parent block using the extracted LEF with whitespace modeling, small holes in between polygons are filled with hard or soft placement blockages depending on their size, and computed as

$$\text{Holes} = \left(FP \setminus_{\square} \widetilde{P}\right) \text{INSIDE}_{\square}\, FP. \tag{4}$$

The whitespace modeling simplifies the buffer feedthrough insertion, even in high-density designs where very little space can be reclaimed organically. For example, a small area (e.g., a square of five standard cell rows) can be reserved at the lower level. This space will be available at the upper level to insert buffers for over-the-block routes without planning pins and introducing significant routing blockages for these routes only, given the block boundary and routing are entirely opened by the detailed LEF. However, note that routing blockages over M1-M2 must still be added to leave pin access points for the inserted buffers.

IV. Automated Floorplanning

This section presents an automated approach to obtaining 3-D floorplans. These floorplans are intended to guide designers and should be revised for high-quality Hier-3D implementations. Additionally, these floorplans can aid in exploring the system-level physical hierarchy and architecture of large designs more effectively in 3-D.

A. Block Generation With RTL Clustering

Our Automated Floorplanning solution enables automated sub-block creation by using an RTL clustering that promotes an optimal grouping of logic cells for better floorplans within a given hierarchy. Typically, the logical hierarchy determined by the RTL ultimately governs the physical hierarchy. However, adhering to this approach may prove impractical when a logical hierarchy is not readily discernible.

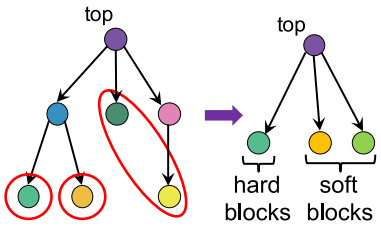


Fig. 5. Illustration of the RTL clustering algorithm formally described in Fig. 4. Hard blocks form their own cluster in the resulting flattened netlist, while standard cells are clustered into soft blocks.

To overcome the limitation, we propose an automated clustering of the netlist, illustrated in Fig. 4. Our algorithm clusters modules within the same hierarchy until the cluster's area exceeds a threshold. The user can easily specify exceptions to this process and add predefined guidance on clustering for some critical components. The netlist is then fully flattened as depicted in Fig. 5. This method heeds that logical connectivity is a good predictor of physical position.

B. Floorplan Settings

The clustered netlist is translated into *Bookshelf* format for the floorplanning engine and Verilog format for implementation in the EDA tool. Soft blocks of standard cells have an area based on a user-defined target density [$d_{\text{target}} = \sum_{c \in C_{\text{std}}} a(c)/a(\text{soft block})$], with a variable aspect ratio between 0.5 and 2. Their IO terms are assumed to be placed in the center. On the other hand, hard macros have fixed outlines and pins set at their exact locations. We increase the size of hard macros (padding) by a configurable constant to leave room for their legalization in the EDA tool. The unplaced IO pins move along with the floorplan area. They are snapped to the closest edge on the periphery based on the center of gravity of the module pins it connects to.

We observe that the wirelength and congestion estimation constitutes a runtime bottleneck during the combinatorial search. However, many nets connect the same pair of blocks in a bus-like structure. Therefore, we perform a net merging step, speeding up the 3-D floorplan exploration by $\times 5$. Nets are merged using a string hashing of the concatenated endpoint names sorted with lexicographical order. Merged nets are associated with a bit-width and weight from the original nets to estimate accumulated wirelength. The positions of associated merged pins on the hard blocks are computed as the barycenter of the original pin locations.

C. Cost Components

Precisely judging the quality of a 3-D floorplan without real physical design feedback is difficult. Therefore, we propose simple yet precise high-level components for our cost, including time-honored PPA representatives and novel features. This

cost will drive our combinatorial floorplan optimization. The latest macro placement works [5], [6] showed the effectiveness of relying on such high-level proxies.

Area Cost: The area is one of the essential PPA elements. The area of the 3-D floorplan is defined from the maximum of the tiers' sizes. However, there are cases where it is necessary to fix the outline of the floorplan, especially when decisions are made early in the design cycle in the physical hierarchical planning and layout. To that effect, we propose an area cost favoring outlines fitting inside the target outline (w_T, h_T)

$$c_{\text{area}} = a \cdot \left(1 + \frac{\text{Relu}(w - w_T)}{w_T} + \frac{Relu(h - h_T)}{h_T}\right)^2 \quad (5)$$

where the area cost is simply the area $a = (w, h)$ of the 3-D floorplan when there is no target outline.

Macro-Specific Cost: We include a penalty to mimic human-like floorplan rules for macro handling as in [14]. Because hard macros are preferably placed on the periphery of the floorplan to leave space for the standard cell placement and away from the IO pins to ease pin accessibility, our macro cost is

$$c_{\text{macro}} = \sum_{m\ :\ \text{macros}} \min_{s:\ \text{sides}} \{\text{d}(m, s)\} + \text{OA}(m, \text{koz}(\text{pins})) \quad (6)$$

where the first term attracts the macros to the sides of the floorplan. The second term computes the macro's overlapping area OA with the IO pins' keep-out zones in case IO pins are preplaced, pushing macros away from the IO pins.

Wirelength Cost: We use the half-perimeter wirelength (HPWL) as a proxy for routed wirelength. Because timing is also a crucial metric during floorplanning, we weigh nets in the wirelength cost with

$$w(e) = \left(1 - \frac{\text{slack}(e)}{T}\right)^2 \quad (7)$$

where T is the clock period and the slack of each net e is extracted with static timing analysis (STA) on the synthesized netlist [15]. Finally, the wirelength cost is

$$c_{\text{HPWL}} = \sum_{e \in E_{\text{merged}}} w(e) \cdot \text{nbits}(e) \cdot \text{HPWL}(e). \quad (8)$$

3-D Congestion Cost: We estimate congestion using the simple and accurate rectangular uniform wire density (RUDY) method [16]. We extend the 2-D formulation specifically for 3-D designs by considering the effects of 3-D routing and F2F bumps. For each bin b of the gridded placement canvas, and per horizontal or vertical (H/V) direction, the routing congestion is

$$\text{RUDY}(b) = \sum_{e \in E_{\text{merged}}} \frac{\text{RISA}(e) \cdot \text{nbits}(e)}{\text{cap}(b)} \cdot \frac{\text{OA}(e, b)}{h_e \| w_e} \cdot Z(e) \quad (9)$$

where cap is the number of metal resources in bin b, (w_e, h_e) is the size of the bounding box of net e, $Z(e) = 1/2$ if the net e is 3-D, otherwise, $Z(e) = 1$. The smaller Z weight for the 3-D nets models the availability of more routing resources from both tiers ($1/2$ assumes a 3-D mirrored stack). RISA(e) serves as net weighting based on pin count to improve correlation with routed wirelength [17].

The H/V congestion maps are then smoothed using a fast box filter to model that congestion can be alleviated by fanout routing outside the nets' bounding boxes. We use a summed-area table (SAT) or 2-D prefix-sum [18] to speed up the filtering. Each element of an SAT contains the sum of all elements above and to the left of the original maps, which can be computed efficiently in one-pass over the matrix. The tables are then used to compute the filtered maps by retrieving the sum of matrix values over any rectangular area in constant time.

F2F Via Density: The RUDY estimation does not model the pitch and spacing considerations of the F2F connections for the 3-D nets. However, it is critical to model these, given that many DRCs occur when the required 3-D interconnection density rises in our experiments. This is a typical problem when using traditional 2-D global routers for 3-D nets [19]. Therefore, we propose a RUDY-inspired method to model the F2F via density

$$D_{\text{F2F}}(b) = \sum_{e \in E_{\text{merged}}} \mathbf{1}[e \text{ cut}] \cdot \frac{\text{OA}(e, b)}{\text{area}(b)} \cdot \frac{\text{nbits}(e)}{\text{nvias}(w_e, h_e)} \quad (10)$$

where $\text{nvias}(w_e, h_e) = ([w_e \cdot h_e]/[p_x \cdot p_y])$ with $p_{x/y}$ the F2F bump pitch.

Finally, our total congestion cost, including routing and F2F via considerations, is empirically set as

$$\begin{aligned} c_{\text{cong.}} = {} & 2 \cdot \max\{D_{\text{F2F}}\} + \max\left\{\overline{\text{RUDY}}_H, \overline{\text{RUDY}}_V\right\} \\ & + \max\{\sigma(\text{RUDY}_H), \sigma(\text{RUDY}_V)\} \end{aligned} \quad (11)$$

encapsulating the average and standard deviation of the routing congestion, to balance via density and routing congestion.

D. 3-D Sequence Pair

Similarly to [2], we use a 3-D sequence pair (SP) to represent a slicing floorplan solution. The 3-D SP maintains one 2-D SP per tier. In each 2-D SP, the SP consists of an ordered pair of module name sequences that encodes geometric relationships between blocks. This representation covers many possible floorplans while offering efficient and flexible perturbations to the solution.

We introduce a few moves that act on the 3-D SP to modify the floorplan. The main difference from a single 2-D SP is that blocks can move from one pair to another in 3-D to explore the 3-D partitioning space. We allow the following moves: 1) Change the aspect ratio of a soft block (standard cells cluster) picked randomly. Hard macros' shapes and orientations are fixed; 2) Swap two blocks picked randomly in a 2-D pair in the positive sequence, negative sequence, or both, 3) Move a block picked randomly from one tier to another, or swap two blocks picked randomly between tiers.

E. Floorplan Space Multilevel Exploration

We use the SA algorithm [20] to explore the floorplan space. However, we observed that SA often cannot produce floorplans where the number of 3-D interconnections is in the expected range. Therefore, we perform a preliminary area-balanced Fiduccia–Mattheyses (FM) min-cut partitioning [13]

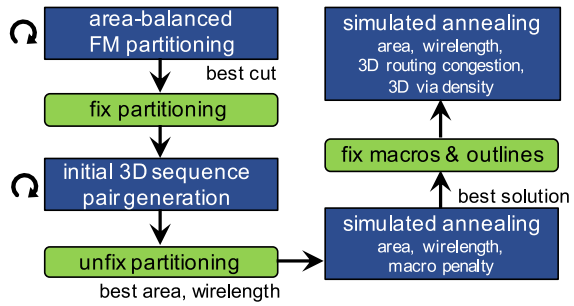


Fig. 6. Hier-3D's automated floorplanning methodology. A FM partitioning [13] produces a min-cut tier assignment of the blocks. Then, an optimized 3-D SP is generated for the two-level SA.

to propose a good starting point for SA. The FM cost is modified to handle timing, as in for the HPWL

$$c_{\text{FM}} = \sum_{e \in E_{\text{merged}}} \mathbf{1}[e \text{ cut}] \cdot w(e) \cdot \text{nbits}(e). \tag{12}$$

Recognizing the importance of the starting seed, we run the FM algorithm multiple times to improve the solution quality further, starting from numerous random permutations of the blocks and selecting the solution with the smallest c_{FM}.

Optimizing all goals simultaneously during SA is complicated and inefficient, mainly because moves affect the cost in a nonlinear and chaotic way. On the other hand, SA works better when the objective is "smooth." Therefore, we devise a multistage SA procedure where different goals are optimized at each stage, along with a multilevel temperature schedule.

Our multilevel optimization scheme is depicted in Fig. 6. The first stage optimizes the area, wirelength, and macro costs using a geometrical temperature schedule, $T_i = T_0^{\alpha \cdot i}$, with $T_0 = 100$ and $\alpha = 0.99$. The best solution from that step is used as the initial configuration for the next stage. Then, the hard macros and outline are fixed; in practice, by rejecting moves involving hard macros and by using the outline penalty in (5). Again, different user-predefined guidances are enabled here. The second stage of SA optimizes area, wirelength, and congestion using an adaptive temperature schedule, $T_i = (\Delta\text{cost} < 0)?\, T_{i-1} : \alpha T_{i-1}$, where the final temperature of the first stage is used as the initial temperature.

The costs of the two SA steps are scalarized to optimize our multiobjectives as follows:

1) $\text{cost} = \tilde{c}_{\text{area}} + \beta \cdot \tilde{c}_{\text{HPWL}} + \gamma \cdot \tilde{c}_{\text{macro}}$,
2) $\text{cost} = \tilde{c}_{\text{area}} + \mu \cdot \tilde{c}_{\text{HPWL}} + \rho \cdot \tilde{c}_{\text{cong.}}$,

where the terms are normalized based on the starting floorplan, and $(\beta, \gamma, \mu, \rho)$ are weights empirically set to $(1, 1, 2, 2)$.

Running the flow in Fig. 6 takes less than a minute on the 3-D MemPool tile (50 blocks/2K merged nets) and a few minutes on the 3-D 4-Core Cortex-A53 (369 blocks/5K merged nets). This scalability allows running this flow multiple times, tuning its parameters using some optimization method (e.g., [21]) to achieve better cost metrics.

F. From Floorplan to Implementation

The floorplanning solutions must be transformed for Hier-3D's sequential multitier hierarchical implementation. First, we leave the user to define the physical implementation sub-blocks and implementation dependency tree. Then, according to these decisions and in an automated fashion, the netlist is logically partitioned, and timing constraints are generated for the physical sub-blocks using the appropriate PDK assignments.

Moreover, we automate the placement of 3-D bumps for sub-blocks spanning multiple dies. First, a bump assignment grid is created based on required F2F via pitch assumptions. KoZ considerations are also considered when designing the grid. Then, for every 3-D net, we construct a point at the center of gravity of the pins connected to that net (pin locations are obtained from the floorplanning solution). A bipartite matching-based algorithm [19], then, assigns points to the bump grid, minimizing the timing-weighted total displacement. This results in the definition of the internal "IO" pins of the floorplanned block. This approach resembles the Sequential-2D one.

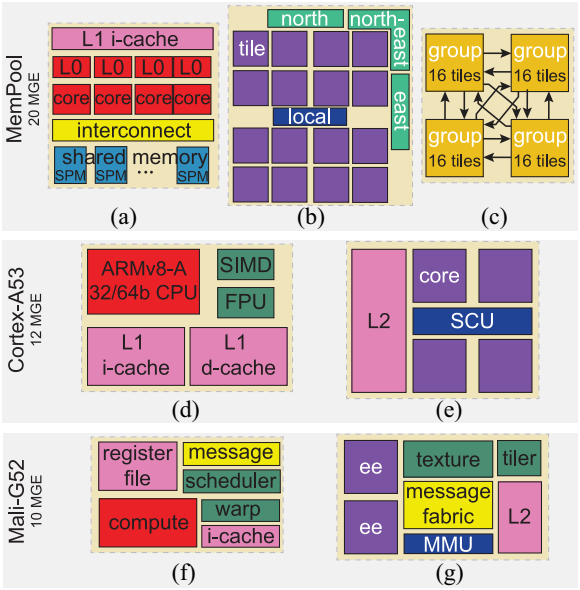


Fig. 7. Our three hierarchical benchmarks, MemPool [22], ARM Cortex-A53, and Mali-G52, implemented in TSMC's 28 and 16 nm processes. (a) Tile. (b) Group. (c) Cluster. (d) Core. (e) Quad-core. (f) Execution engine (ee). (g) Gpu.

V. Industry Benchmarks

A. Architecture Description

To evaluate our proposed flow, we implement MemPool [22] as a representative example for tiled manycore architectures, the ARM Cortex-A53 representing ultrahigh efficiency commercial multiprocessors, and the ARM Mali-G52 exemplifying mid-range graphics and multimedia processors. All three benchmarks are implemented using two advanced commercial process technologies. Fig. 7 details the different hierarchical architectures that we use for our benchmarks.

The 2-D floorplans of the MemPool subdesigns are obtained from the original paper [22]. The 2-D floorplans of the ARM Cortex-A53 and Mali-G52 are industrial-strength based on the official documentation. We normalize our data to avoid revealing proprietary information for these commercial processors.

VI. Experiments

A. Experimental Settings

We use a commercial TSMC 28 nm, high-κ metal gate, planar technology to first implement the MemPool design, and TSMC 16 nm, FinFET Compact technology for the Cortex-A53 and Mali-G52 implementations. We also further implement the MemPool design using the TSMC 16 nm process. Specifically, the MemPool design will serve as a vehicle to demonstrate the effectiveness of Hier-3D's advanced capabilities of whitespace modeling, architecture configuration exploration, and 3-D/2-D automated floorplanning. In the 3-D implementations, where we assume that the 3-D technology is independent of the CMOS technology, the F2F via size, pitch, resistance, and capacitance are 0.5μm$\times 0.5\mu$m, 1.0μm, 0.5 Ω, and 1 fF, respectively, [23]. The 3-D BEOL is defined in a custom 3-D TECHLEF file where metal layers are replicated and mirrored, separated by a F2F via layer of 0.175 µm thickness. In addition, the custom TECHLEF includes design rules for the double metal stack. Based on a custom interconnect technology (ICT) file, we simulate the metal layers' resistance/capacitance (RC) and copper pads.

We use the in-house tools of Macro-3D [12] and implement the Sequential-2D flow as the construction of two sequential 2-D implementations [11]. For a fair comparison, the 3-D implementations include a (close to) balanced mirrored stack of the 2-D configuration. Furthermore, we implement the designs with a *max-performance* target at the typical corner in all our experiments. Finally, our Hier-3D implementation flow is automatized with *Tcl* and *Bash* scripts inside the *Cadence Innovus* environment. The RTL clustering and logical partitioning are automatized with *Tcl* inside *Cadence Genus*. The 3-D floorplanning engine, including the bump planning and automated generation of *Tcl* scripts, is implemented using *Python* and *C++*.

B. Default Implementation Results

To highlight the PPA benefits of Hier-3D from the P&R side only, we first present the main implementation results using the default logical hierarchy and manual floorplan settings shown in Section V. Please note that neither whitespace modeling nor automated floorplanning is considered for these initial results. These optional features (whitespace modeling and automated floorplanning) and the hierarchy management are detailed in their dedicated subsections within this section.

1) MemPool Design: While the tile implementation PPA metrics are very similar across all integration flows, the group level is critical in the implementation of MemPool. The group is highly connected in the center, where the engine places most of the local interconnect logic. This creates heavy congestion, degrading timing, and increasing routing DRCs if the tiles are not spaced sufficiently. Thus, the floorplan size for Sequential-2D must be increased to obtain a DRC-clean design due to the reduced stack awareness compared to the Macro-3D implementation, as depicted in Fig. 8. With our flow, the block-to-block spacing can instead be reduced to only 5 µm thanks to the shared BEOL and the use of both FEOLs for standard cells, providing substantial area and cost reductions.

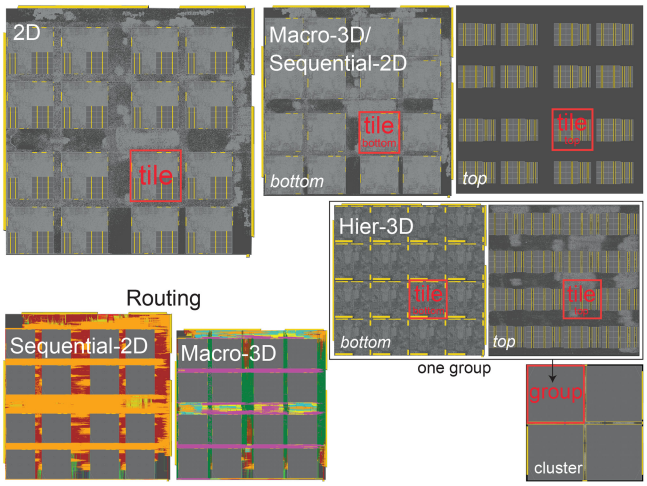


Fig. 8. Placement of the MemPool 16-tile group designs using TSMC 28 nm process: 2-D versus Macro-3D/Sequential-2D versus Hier-3D and Sequential-2D, Macro-3D routing.

TABLE II
64-Tile MemPool Cluster 2-D Versus Sequential-2D Versus Macro-3D Versus Hier-3D, Using TSMC 28 nm Process. 7.3M Cells, 13.1M Nets, and 1536 Memory Macros. Values are Normalized w.r.t. 2-D Implementation

MemPool Cluster	2D	Seq-2D	Macro-3D	Hier-3D
metals used	6	6 (bot) 6 (top)	6 (bot) 6 (top)	6 (bot) 6 (top)
silicon area	1	1.49	1.14	0.75
total WL	1	0.87	0.80	0.71
density (%) bot/top	56.2	50.8/22.6	62.4/29.6	84.4/53.4
buffer count	1	0.91	0.67	0.61
# F2F bumps	-	130K	813K	482K
effective freq	1	1.00	1.05	1.42
total power	1	0.98	0.89	0.80
power × delay	1	0.98	0.85	0.57
die cost	1	1.57	1.19	0.79
power perf cost	1	0.65	0.99	2.22
runtime	1	1.09	0.91	0.68

Moreover, this reduces the net lengths, resulting in significant power reduction and performance increase.

Table II highlights the huge PPA savings of Hier-3D and the resulting Power Performance Cost (PPC) metric computed using the methodology presented in [24] as PPC = Frequency / (Die Cost × Power). We see an impressive 2.2× PPC improvement, where all individual metrics are noticeably improved, which is quite unique. This result reflects the benefits of our flow in terms of higher die stack utilization.

We similarly implement the MemPool Cluster using TSMC 16 nm process to evaluate the impact of the technology node on Hier-3D's results, reported in Table III. Technology scaling further accentuates Hier-3D's PPC improvement, mainly from the higher reduction in silicon area, which we attribute to the difference between memory and logic scaling. Indeed, logic benefits more from innovative CMOS scaling features than memory, scaling more aggressively through fin depopulation. In contrast, the minimum size of a memory bit cell is relatively constant for FinFET-based standard cell architectures. The relative number of placement sites of logic versus memory

$$\frac{\text{area(die)} - \text{area(memory)}}{\text{area(site)}} \tag{13}$$

TABLE III
64-TILE MEMPOOL CLUSTER 2-D VERSUS MACRO-3D VERSUS HIER-3D, USING TSMC 16 nm PROCESS. NORMALIZED VALUES W.R.T. 2-D IMPLEMENTATION

MemPool Cluster	2D	Macro-3D	Hier-3D
metals used	6	6 (bot) 6 (top)	6 (bot) 6 (top)
silicon area	1	1.10	0.62
total WL	1	0.82	0.77
buffer count	1	0.81	0.84
# F2F bumps	-	1.06M	450K
effective freq	1	1.12	1.39
total power	1	0.93	0.86
power × delay	1	0.83	0.62
die cost	1	1.16	0.65
power perf cost	1	1.04	2.49
runtime	1	1.31	1.10

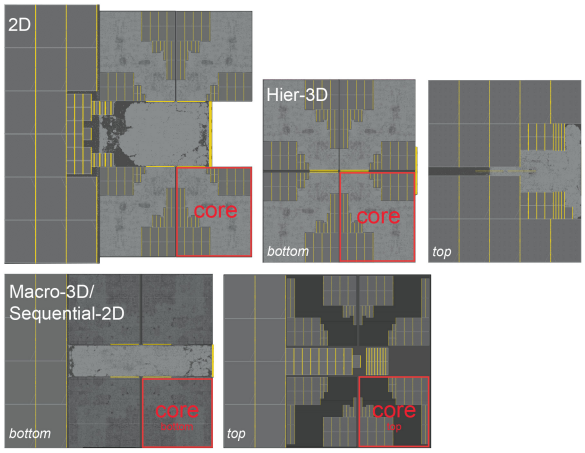


Fig. 9. Placement (to scale) of the Cortex-A53 4-core designs using TSMC 16 nm process: 2-D versus Macro-3D/Sequential-2D versus Hier-3D.

TABLE IV
4-CORE CORTEX-A53 2-D VERSUS SEQUENTIAL-2D VERSUS MACRO-3D VERSUS HIER-3D, USING TSMC 16 nm PROCESS. 2.4M CELLS, 2.5M NETS, AND 165 MEMORY MACROS. VALUES ARE NORMALIZED W.R.T. 2-D IMPLEMENTATION. 2-D SILICON AREA DOES NOT INCLUDE CUTOUTS

Cortex-A53	2D	Seq-2D	Macro-3D	Hier-3D
metals used	8	7 (bot) 6 (top)	7 (bot) 6 (top)	6 (bot) 7 (top)
silicon area	1	1.21	1.21	0.95
total WL	1	1.02	0.97	0.94
density (%) bot/top	77.5	73.7/62.6	72.1/62.5	81.0/90.7
buffer count	1	1.15	1.05	0.98
# F2F bumps	-	22K	81K	74K
effective freq	1	0.93	0.95	1.33
total power	1	1.14	0.97	0.97
power × delay	1	1.22	1.02	0.73
die cost	1	1.13	1.13	0.91
power perf cost	1	0.78	0.87	1.51
runtime	1	1.12	0.89	0.82

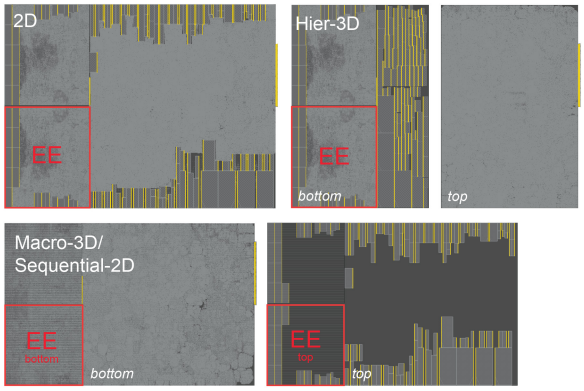


Fig. 10. Placement (to scale) of the Mali-G52 2-EE designs using TSMC 16 nm process: 2-D versus Macro-3D/Sequential-2D versus Hier-3D.

increases by 1.2× from 28 nm to 16 nm for the MemPool Cluster, offering additional silicon area gains. This trend will aggravate for 10 nm and below. The tile's silicon area requirement for the memory will become larger than the standard cell area required to implement the logic, which will cause a lower standard cell density in the logic die with more whitespace to be recovered in the memory-on-logic partitioning scheme.

2) Cortex-A53 Design Results: The PPA results of the single-core implementations are similar across all key metrics. The Macro-3D/Sequential-2D quad-core floorplan stacks two L2 data macros on top of each other, reducing the design footprint. However, a significant amount of silicon area in the upper die remains unused, as shown in Fig. 9. The Hier-3D floorplan instantiates the 2-D single-core abstraction, leaving the four single-core footprints unutilized in the upper die. Therefore, the top-level memory macros and SCU standard cells can be placed on top of the single cores, further reducing the silicon area. In the quad-core implementation, Hier-3D optimizes all the limiting competing paths between the SCU standard cells to the L2 data macros, the single-cores, and the IOs, thanks to the denser floorplan and increased routability, yielding a significant frequency increase. Table IV shows that the Hier-3D flow surpasses the Macro-3D flow in frequency and power, with drastic improvement in the silicon area.

3) Mali-G52 Design Results: We floorplan the Hier-3D GPU using a 2-D bin-packing method, assigning the upper and lower edge macros of the 2-D floorplan into two separate bins. The packed macros are placed next to the 2-D EE abstractions on the bottom die, allowing the top-level standard cells to fully utilize the upper die, as shown in Fig. 10.

Table V shows a 15 % increase in frequency while reducing the total silicon area compared to 2-D and a substantial die cost reduction compared with the two other 3-D flows. Modifications of the macro placement would yield further wire length reduction and PPC improvements at the expense of the silicon area gains.

C. Advanced Hier-3D Capabilities

Here, we present the advanced capabilities of Hier-3D using the MemPool design as our test case.

1) Whitespace Modeling: To verify the advantages of our proposed whitespace modeling, we focus on two 2-D examples. Note that this methodology can be applied similarly to 3-D implementations, but the analysis in 2-D helps to understand the general advantages of whitespace. The LEF modifications virtually come at no cost, and the additional placement area opened by the whitespace modeling offers incremental PPA improvements, as presented in Table VI.

First, we use whitespace modeling to open up placement space on the group boundary for the cluster level, as seen

TABLE V
2-EXECUTION ENGINE MALI-G52 2-D VERSUS SEQUENTIAL-2D VERSUS MACRO-3D VERSUS HIER-3D, USING TSMC 16 nm PROCESS. 4.4 M CELLS, 4.8M NETS, AND 141 MEMORY MACROS. VALUES ARE NORMALIZED W.R.T. 2-D IMPLEMENTATION

Mali-G52	2D	Seq-2D	Macro-3D	Hier-3D
metals used	8	7 (bot) 6 (top)	7 (bot) 6 (top)	6 (bot) 7 (top)
silicon area	1	1.45	1.45	0.99
total WL	1	0.88	0.86	0.98
density (%) bot/top	77.7	74.5/31.3	74.3/31.3	78.1/70.2
buffer count	1	0.93	0.93	0.97
# F2F bumps	-	56K	325K	149K
effective freq	1	0.98	0.95	1.15
total power	1	1.14	0.96	0.98
power × delay	1	1.16	1.01	0.85
die cost	1	1.35	1.35	0.95
power perf cost	1	0.64	0.73	1.24
runtime	1	0.99	1.08	0.81

TABLE VI
WHITESPACE MODELING ON 2-D MEMPOOL CLUSTER AND GROUP DESIGNS. PPA METRICS ARE FOR INTERGROUPS AND INTERTILES ONLY, RESPECTIVELY

MemPool	Cluster (28 nm)		Group (16 nm)	
	2D	+whitespace	2D	+whitespace
WL	1	0.97	1	0.99
buffer count	1	0.96	1	0.78
buffer area	1	0.94	1	0.77
effective freq	1	1.02	1	1.09
power	1	0.96	1	0.95
runtime	1	1	1	1

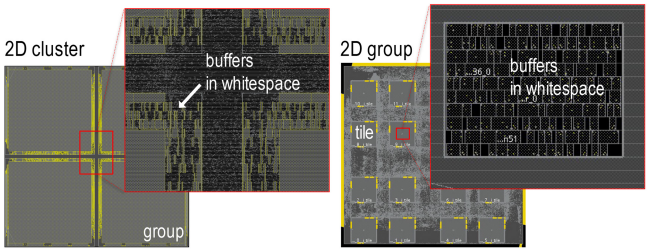


Fig. 11. Whitespace modeling in 2-D: opening the group boundary for the cluster implementation using TSMC 28 nm process (= left), and reserving a small square in the middle of the tile for the group implementation using TSMC 16 nm process (= right). The same whitespace modeling can also be performed on different flows including 3-D flows but was not explored here.

in Fig. 11. This allows cells to be placed by the tool in the whitespace, increasing the tool's optimization capabilities for placing these cells and buffers in the highly congested middle of the floorplan, resulting in a smaller buffer area and wirelength reduction without a DRC increase. Second, we purposedly reserve placement resources in the middle of the tile to allow the tool to directly insert feedthrough buffers in these locations at the group level, as seen in Fig. 11. This allows long routes overlapping the tiles to be buffered without detouring outside the tile blockages, reducing delays. This also provides significant buffer area reduction as, without whitespace, rerouting wires to completely avoid the large tile blockages often results in considerable wire detours, requiring expensive buffering to manage delay degradation.

2) Architecture Exploration (Hierarchy Restructuring): One of the main benefits of Hier-3D is that it offers a natural way to explore hierarchy and its effect on the 3-D physical

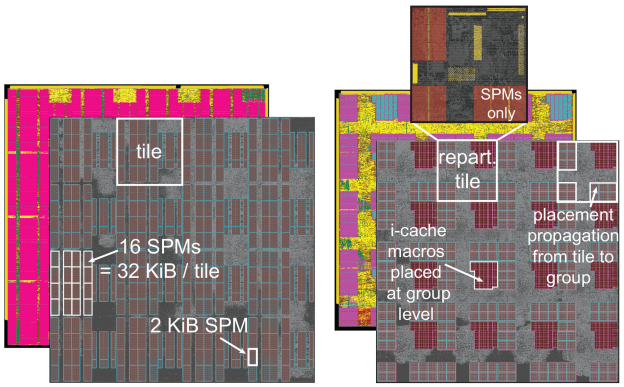


Fig. 12. MemPool physical and logical architecture exploration, using TSMC 28 nm process. Enlarging the SPMs from 1 KiB to 2 KiB (= left). Repartitioning of the tile instruction cache from the tile level to the group level (= right).

TABLE VII
16-TILE MEMPOOL GROUP HIER-3D ARCHITECTURE EXPLORATION, USING TSMC 28 nm PROCESS. 1.3M CELLS, 3.3M NETS, AND 384 MEMORY MACROS. VALUES ARE NORMALIZED W.R.T. BASE HIER-3D IMPLEMENTATION

MemPool Group	Hier-3D base	Hier-3D repartitioned	Hier-3D 2 KiB SPMs
metals used	6 (bot) 6 (top)	6 (bot) 6 (top)	6 (bot) 6 (top)
silicon area	1	0.95	1
total WL	1	1.03	1.06
density (%) bot/top	87.1/55.4	84.3/70.6	87.6/71.8
buffer count	1	1.05	1.09
# F2F bumps	106K	161K	149K
effective freq	1	0.96	0.92
total power	1	1.04	1.09
power × delay	1	1.08	1.18
die cost	1	0.95	1
power perf cost	1	0.97	0.84
runtime	1	1.03	1.05

implementation. To that effect, we restructured the MemPool netlist to move the tile's instruction cache logic and 1 KiB to the group level—we used *Cadence Genus* to restructure the netlist and generate new SDCs, and manually placed the IO pins. This allows a smaller tile footprint and better utilization of both tiers, as the tile partitioning no longer creates a silicon area requirement imbalance. The layouts of the group implementation are shown in Fig. 12, and the PPA results are tabulated in Table VII. These show a reduced footprint area with an increased placement density while maintaining comparable performance.

Memory Size Increase: To study the generality of the strengths of Hier-3D with more difficult implementation constraints, we look at the effects of increasing the capacity of MemPool tile's shared L1 data memory macros from 1 KiB to 2 KiB. This increases the utilization of the top tier and reduces the imbalance at the tile level. However, this does not require changing the tile footprint and allows to keep the footprint in our Hier-3D group constant. Similar experiments with Macro-3D were carried out in [7] with promising results but lacking the flexibility of Hier-3D logic-on-logic capabilities. The corresponding placement and routing layouts are shown in Fig. 12. Table VII shows this change induces limited degradation in the PPA of the group, despite the much denser

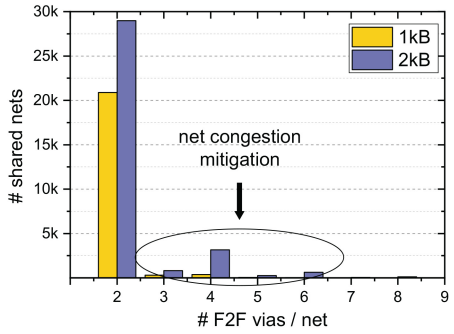


Fig. 13. Comparison of the number of vias used per shared 2-D net in the Hier-3D 16-tile MemPool Group implementations, using TSMC 28 nm process. The group with 2 KiB shared memory macros (512 KiB total) benefits from a nonnegligible number of nets alternating between two tiers, yielding a relief of congestion in the top tier.

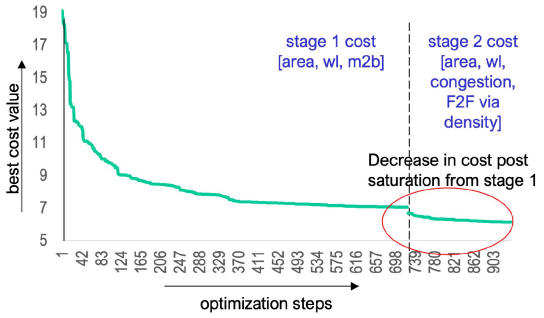


Fig. 14. Cost optimization in our multistage SA flow. The cost function is normalized across the stages to visualize the improvement. Due to our multistaged approach, we see a decrease in the overall cost post-cost saturation from stage 1 of SA.

TABLE VIII
VERIFICATION OF OUR AUTOMATED FLOORPLANNING ON THE MEMPOOL SINGLE-TILE, USING TSMC 16 nm PROCESS

MemPool Tile	2D		Hier-3D	
	manual	automated	manual	automated
metals used	6	6	6 (bot) 6 (top)	6 (bot) 6 (top)
silicon area	1	1.06	1	1
total WL	1	1.09	1	1.4
density (%) bot/top	81.0	73.0	88.0/48.1	68.1/79.0
buffer count	1	1.01	1	0.73
# F2F bumps	-	-	3K	39K
effective freq	1	1.05	1	1
total power	1	1	1	1.12
power × delay	1	0.95	1	1.12
floorplanning effort	hours + expertise	< 1 min push-button	hours + expertise	< 1 min push-button

floorplan. The noticeable effect is the abrupt increase in the number of F2F bumps. The tool can mitigate congestion using metal layer sharing, i.e., using routing resources of the bottom tier. This is corroborated by Fig. 13, which reports the number of F2F bumps used per shared net, i.e., a net that connects two top cells but is routed in the bottom tier. This positive effect leads to more efficient routes in heavily congested areas and is enabled by our detailed LEF abstraction. Overall, the Hier-3D PPA of the group with 2 KiB memory macros is comparatively much superior to the one reported in [7].

3) Automated Floorplanning: Before presenting the automated floorplanning implementation results, we illustrate how

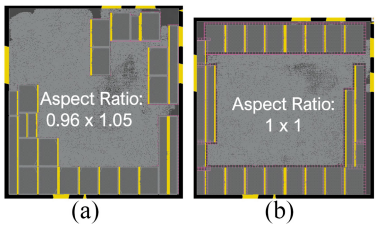


Fig. 15. 2-D floorplans of the MemPool tile using different macro-specific cost weights in TSMC 16nm process. (a) Is the floorplan where the macros are not pushed to the boundary due to a low weight assigned to Macro-Specific cost in the overall cost function. (b) Is the floorplan where the assigned weight is higher putting the macros on the corner.

the cost functions evolve throughout the multistage SA flow in Fig. 14. As shown by the stage 2 cost decrease, the multistage SA escapes the cost saturation of stage 1.

In our study, we compare the manual floorplans with the automated floorplans, using the reference floorplans from [7] and [22] for the manual floorplans. In general, extensive human expertise, effort, and numerous iterations of P&R runs feedback are required in manual floorplanning to achieve high-quality macro placement solutions. This method is also not scalable runtime-wise, given the latency of P&R from hours to days. On the other hand, the automated floorplanning solution delivers floorplans with competitive PPA in a minute for small benchmarks, which may rise to only a few minutes for more considerable benchmarks.

Table VIII summarizes the floorplanning results. While the proposed solutions do not consistently beat the manual solutions, our proposed auto-floorplanning solutions have minimal manual intervention and faster turnaround time than the human method. Still, our goal is to quickly generate satisfactory floorplans serving as starting points that can be refined later. In this regard, the automated solutions can enable faster system-level simulations for architectural experiments, which can tolerate less accurate PPA feedback.

2-D: To understand the influence of cost functions on the Automated Floorplan solutions, we select the Macro-Specific cost from Section IV and evaluate its effect on PPA. Fig. 15 shows the floorplans obtained with small and high Macro-Specific costs. The floorplan a) has macros not strictly on the boundary due to a lesser weight to the Macro-Specific cost, and floorplan b) is obtained with the weight set much higher. We observe that the total power and wirelength reduced significantly when macros were on the corner by 28% and 12% respectively. This can be attributed to better usage of routing resource and better standard cell placement when macros are on the corner. Similarly, the other cost functions in Section IV influence the overall PPA.

Our automated floorplanning methodology designed for 3-D designs is applied to 2-D designs by skipping the partitioning and ignoring the components of the cost of 3-D during SA. As shown in Table VIII, the automated floorplan solution performs better than the manual one. Still, some area penalty is depicted in Fig. 16.

3-D: Based on the assignments from the 3-D automated floorplanning, we used Hier-3D to implement the leaf level as the bottom die and the memory macros of the top die. Then we proceeded to implement the top die's standard cells.

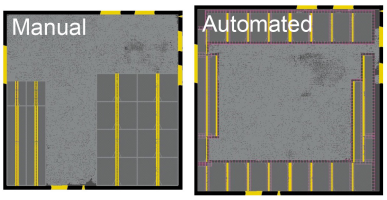


Fig. 16. Placement (to scale) of the MemPool single-tile designs using TSMC 16 nm process: 2-D manual versus 2-D automated floorplan.

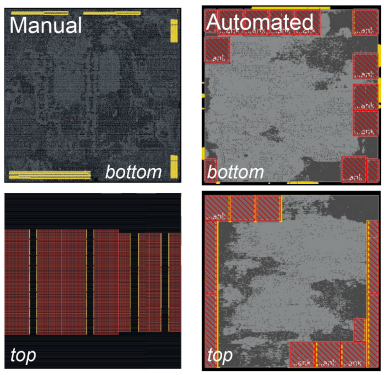


Fig. 17. Placement (to scale) of the MemPool single-tile designs using TSMC 16 nm process: 3-D manual versus 3-D automated floorplan.

Fig. 17 and Table VIII shows that the 3-D implementation is not as good as the manual version, especially regarding the wirelength and the number of F2F bumps. The latter can be improved by being more stringent when accepting 3-D moves that worsen the cut during floorplanning.

VII. Conclusion

We propose a full-chip RTL-to-GDSII physical design solution that offers a commercial-quality F2F-bonded 3-D IC physical layout for large hierarchical designs. Our flow includes new critical ideas, such as the routing and placement constraint propagation in the double metal stack view, stack inversion, and an optional whitespace modeling, enabling multitier cell placement. This design flow steppingstone vastly expands the design space exploration options and can help explore physical hierarchy more efficiently on a multilevel for 3-D ICs. Our proposed automated 3-D floorplanning methodology assists in executing this exploration and reduces its turnaround time. Our extensive experiments on large complex hierarchical designs of an open manycore processor and industrial ARM application and graphics processors show our flow offers 15 to 43 % power × delay reduction and more than 1.2× combined power, performance, and area/cost improvements compared with 2-D.

References

[1] C. Niessen, "Hierarchical design methodologies and tools for VLSI chips," *Proc. IEEE*, vol. 71, no. 1, pp. 66–75, Jan. 1983.
[2] S. Panth, K. Samadi, Y. Du, and S. K. Lim, "High-density integration of functional modules using monolithic 3D-IC technology," in *Proc. Asia South Pacific Design Autom. Conf. (ASP-DAC)*, 2013, pp. 681–686.
[3] J. Knechtel, I. L. Markov, and J. Lienig, "Assembling 2-D blocks into 3-D chips," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 31, no. 2, pp. 228–241, Feb. 2012.
[4] A. Agnesina et al., "Hier-3D: A hierarchical physical design methodology for face-to-face-bonded 3D ICs," in *Proc. ACM/IEEE Int. Symp. Low Power Electron. Design*, 2022, pp. 1–6.
[5] A. Agnesina et al., "AutoDMP: Automated dreamplace-based macro placement," in *Proc. Int. Symp. Phys. Design*, 2023, pp. 149–157
[6] A. Mirhoseini et al., "A graph placement methodology for fast chip design," *Nature*, vol. 594, no. 7862, pp. 207–212, 2021.
[7] M. Cavalcante et al., "MemPool-3D: Boosting performance and efficiency of shared-l1 memory many-core clusters with 3D integration," in *Proc. Design, Autom. Test Eur. Conf. Exhibit. (DATE)*, 2022, pp. 394–399.
[8] J. Wuu et al., "3D V-cache: The implementation of a hybrid-bonded 64MB stacked cache for a 7nm x86-64 CPU," in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*, 2022, pp. 428–429.
[9] D. Wong and C. Liu, "A new algorithm for floorplan design," in *Proc. ACM/IEEE Design Autom. Conf.*, 1986, pp. 101–107.
[10] L. Cheng, L. Deng, and M. D. F. Wong, "Floorplanning for 3-D VLSI design," in *Proc. ASP-DAC Asia South Pacific Design Autom. Conf.*, 2005, pp. 405–411.
[11] G. Sisto et al., "Design enablement of fine pitch face-to-face 3D system integration using die-by-die place & route," in *Proc. Int. 3D Syst. Integr. Conf. (3DIC)*, 2019, pp. 1–4.
[12] L. Bamberg et al., "Macro-3D: A physical design methodology for face-to-face-stacked heterogeneous 3D ICs," in *Proc. Design, Autom. Test Eur. Conf. Exhibit. (DATE)*, 2020, pp. 37–42.
[13] C. M. Fiduccia and R. M. Mattheyses, "A linear-time heuristic for improving network partitions," in *Proc. Papers Twenty-Five Years Electron. Design Autom.*, 1988, pp. 241–247.
[14] A. B. Kahng, R. Varadarajan, and Z. Wang, "RTL-MP: Toward practical, human-quality chip planning and macro placement," in *Proc. Int. Symp. Phys. Design*, 2022, pp. 3–11.
[15] D. Z. Pan, B. Halpin, and H. Ren, "Timing-driven placement," *Handbook Algorithms Phys. Design Autom.*, pp. 423–446, Nov. 2008.
[16] P. Spindler and F. M. Johannes, "Fast and accurate routing demand estimation for efficient routability-driven placement," in *Proc. Design, Autom. Test Eur. Conf. Exhibit.*, 2007, pp. 1–6.
[17] C.-L. E. Cheng, "RISA: Accurate and efficient placement routability modeling," in *Proc. Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, 1994, pp. 690–695.
[18] F. C. Crow, "Summed-area tables for texture mapping," in *Proc. Annu. Conf. Comput. Graph. Interact. Techn.*, 1984, pp. 207–212.
[19] S. Pentapati, A. Agnesina, M. Brunion, Y.-H. Huang, and S. K. Lim. "On Legalization of die bonding bumps and pads for 3D ICs," in *Proc. Int. Symp. Phys. Design*, 2023, pp. 62–70.
[20] D. Bertsimas and J. Tsitsiklis, "Simulated annealing," *Statist. Sci.*, vol. 8, no. 1, pp. 5–10, 1993.
[21] A. Agnesina, K. Chang, and S. K. Lim, "VLSI placement parameter optimization using deep reinforcement learning," in *Proc. Int. Conf. Comput.-Aided Design*, 2020, pp. 1–9.
[22] M. Cavalcante, S. Riedel, A. Pullini, and L. Benini "MemPool: A shared-L1 memory many-core cluster with a low-latency interconnect," in *Proc. Design, Autom. Test Eur. Conf. Exhibit. (DATE)*, 2021, pp. 701–706.
[23] E. Beyne et al., "Scalable, sub 2μm pitch, cu/SiCN to cu/SiCN hybrid wafer-to-wafer bonding technology," in *Proc. IEEE Int. Electron Devices Meeting (IEDM)*, 2017, pp. 32.4.1–32.4.4.
[24] A. Agnesina et al., "Power, performance, area and cost analysis of memory-on-logic face-to-face bonded 3D processor designs," in *Proc. IEEE/ACM Int. Symp. Low Power Electron. Design (ISLPED)*, 2021, pp. 1–6.

Nesara Eranna Bethur received the M.S. degree in electrical and computer engineering from the Georgia Institute of Technology, Atlanta, GA, USA, in 2023.

He joined Advanced Micro Devices Inc. Austin, Austin, TX, USA, as a Senior Silicon Design Engineer in 2023. Previously, he worked as a Design Engineer II with Cadence Design Systems Inc., Bengaluru, India, from 2019 to 2021. He was also a Technical Intern with Synopsys Inc., Bengaluru, India, in 2019. His research interests include design methodology for 3DIC, as well as the application of machine learning and algorithms in EDA.

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

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

Dr. Agnesina received the 2022 ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED) Best Paper Award.

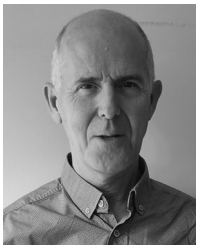
Moritz Brunion received the bachelor's and master's degrees in electrical and computer engineering from the University of Bremen, Bremen, Germany, in 2019 and 2022, respectively.

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

Alberto Garcia-Ortiz (Senior Member, IEEE) received the diploma degree in telecommunication systems from the Polytechnic University of Valencia, Valencia, Spain, in 1998, and the Ph.D. degree (summa cum laude) from the Department of Electrical Engineering and Information Technology, Institute of Microelectronic Systems, Darmstadt University of Technology, Darmstadt, Germany, in 2003.

He worked with Newlogic, Austria, Europe, for two years. From 2003 to 2005, he worked as a Senior Hardware Design Engineer with IBM Deutschland Development and Research, Böblingen, Germany. After that, he joined the start-up AnaFocus, Spain, where he was responsible for the design and integration of AnaFocus' Next Generation Vision Systems-on-Chip. He is currently a Full Professor for the chair of integrated digital systems with the University of Bremen, Bremen, Germany. His interests include low-power design and estimation, communication-centric design, 3-D integration, and hardware accelerators for AI.

Dr. Garcia-Ortiz received the "Outstanding Dissertation Award" in 2004 from the European Design and Automation Association. In 2005, he received from IBM an innovation Award for contributions to leakage estimation. Two patents are issued with that work. He serves as an Editor for Journal of Low Power Electronics and is a Reviewer of several conferences, journals, and European projects.



Francky Catthoor (Fellow, IEEE) received the Ph.D. degree in electrical engineering (EE) from the Katholieke Universiteit Leuven (KULeuven), Leuven, Belgium, in 1987.

Between 1987 and 2000, he has headed several research domains in the area of synthesis techniques and architectural methodologies. Since 2000 he is strongly involved in other activities with IMEC, Leuven, Belgium, including co-exploration of application, computer architecture and deep submicron technology aspects, biomedical systems and IoT sensor nodes, and photo-voltaic modules combined with renewable energy systems, where he is currently a Senior Fellow. He is also a part-time Full Professor with the EE department, KULeuven.

Dr. Catthoor has been an Associate Editor for several IEEE and ACM journals.

Dragomir Milojevic received the M.S. and Ph.D. degrees in electrical engineering from the Université Libre de Bruxelles (ULB), Brussels, Belgium, in 1994 and 2004, respectively.

He holds the position of a Professor with Digital Electronics and Systems Design, ULB. In 2004, he joined IMEC, Leuven, Belgium, working on multiprocessor and network-on-chip architectures for low-power multimedia systems. Since 2008, he has been working on enablement of 3-D stacked ICs, and system and design technology co-optimization of advanced technology nodes, and design methodologies for technology aware 3-D ICs.

Manu Komalan (Member, IEEE) received the integrated master's degree in nanotechnology from Amity University, Noida, India, in 2011, the first Ph.D. degree in electrical engineering from Katholieke Universiteit Leuven, Leuven, Belgium, in 2017, and the second Ph.D. degree in computer science from the Universidad Complutense de Madrid, Madrid, Spain, in 2017.

He then joined IMEC, Leuven, Belgium, as a Memory System Architecture Researcher, where he worked as a Program Manager with the Memory INSITE Program, from 2019 to 2021 on activities involving exploration, analysis, and optimization of NVMs across the different layers of abstraction, and is currently an Architect and a Manager with the Compute Systems Architecture Unit, wherein, his focus is systems design for the AI and HPC domain.

Matheus Cavalcante (Student Member, IEEE) received the M.Sc. degree in integrated electronic systems from the Grenoble Institute of Technology (Phelma), Grenoble, France, in 2018. He is currently pursuing the Ph.D. degree with the Digital Circuits and Systems Group of Prof. L. Benini, ETH Zürich, Zürich, Switzerland.

His current research interests include the design of very-large-scale circuits and high-performance systems, namely vector and manycore architectures, and their co-optimization with emerging VLSI technologies.

Samuel Riedel (Member, IEEE) received the B.Sc. and M.Sc. degrees in electrical engineering and information technology with ETH Zürich, Zürich, Switzerland, in 2017 and 2019, respectively, where he is currently pursuing the Ph.D. degree with the Digital Circuits and Systems Group of L. Benini.

His research interests include computer architecture, focusing on manycore systems and their programming model.

Luca Benini (Fellow, IEEE) received the Ph.D. degree from Stanford University, Stanford, CA, USA, in 1997.

He is a Chair of Digital Circuits and systems with ETH Zürich, Zürich, Switzerland, and is a Full Professor with the Universita di Bologna, Bologna, Italy. His research interests are in energy-efficient parallel computing systems, smart sensing microsystems, and machine learning hardware.

Dr. Benini is a Fellow of the ACM and a member of the Academia Europaea.

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

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

Dr. Lim received the National Science Foundation Faculty Early Career Development (CAREER) Award in 2006, the ACM SIGDA Distinguished Service Award in 2008, the Best Paper Award from the IEEE TRANSACTIONS ON ELECTROMAGNETIC COMPATIBILITY and the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS in 2022, and the Best Paper Award from several conferences in EDA, including ACM Design Automation Conference in 2023. He joined the Defense Advanced Research Projects Agency in 2022 as a Program Manager with the Microsystems Technology Office.