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

***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 paper proposes a novel hierarchical physical design flow enabling the building of high-density and commercial-quality two-tier face-to-face-bonded hierarchical 3D ICs. Complemented with an automated floorplanning solution, the flow allows for system-level physical and architectural exploration of 3D designs. As a result, we significantly reduce the associated manufacturing cost compared to existing 3D implementation flows and, for the first time, achieve cost competitiveness against the 2D 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-$2.2\times$ compared with 2D, where all metrics are improved simultaneously, including up to $20\,\%$ power savings.**

***Index Terms*—Hier-3D; Physical Design Methodology; Wafer-level Bonding; Face-to-Face (F2F) Bonded 3D ICs**

I. Introduction

To deliver the perceived benefits of 3D ICs outside the purview of research and academia [1], a hierarchical 3D 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 2D designs. However, none of these flows are currently optimized for 3D 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 3D. 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, which include the floorplanning of blocks in 3D, are usually left to expert engineers and pre-constrained, like memory-on-logic. These can significantly impact the power, performance, area, and cost (PPAC) metrics, the latter recently becoming most important due to increasing wafer costs. In addition, when designing a 3D 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 3D 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 3D bottom-up hierarchical implementations to co-optimize power, performance, and area/cost combined design metrics, as modern-day 3D flows do not satisfactorily address the latter. The key contributions of this paper, an extension of [4], are as follows:

- We propose a first-of-its-kind hierarchical physical design flow for 3D designs, which significantly improves the placement and routing utilization across the 3D stack by reusing the unassigned silicon area of preceding hierarchy levels.
- Our Hier-3D flow exploits the inherent logical hierarchy to enable hierarchical multi-tier standard cell placement, greatly expanding the design space of 3D ICs. We demonstrate the flexibility of the proposed implementation flow by exploring the architecture configuration space for a selected design.
- We develop a high-level automated solution spanning RTL clustering, 3D logic-on-logic floorplanning, and holistic 3D physical implementations. This effort enables the automated architecture design exploration on a multi-level 3D physical hierarchy while still enabling predefined placements imposed by manual guidance from the human designer.
- We demonstrate 1.2-$2.2\times$ PPAC improvements and 1.2-$1.5\times$ runtime speedup on three highly diverse open-source and industrial low-power benchmarks and, for the first time, cost improvement compared to the 2D reference. These results are incommensurate with 2D and standard commercial and academic 3D flows.

II. Preliminaries

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

A. *Hierarchical Methodologies*

Designing modern SoCs requires a hierarchical methodology that spans multiple levels of physical hierarchy, due to the size and complexity of the designs. Nowadays, using up to five levels of physical hierarchy for designs with hundreds of millions of gates is common. However, handling more than two hierarchy levels significantly increases the implementation complexity. Without proper automation, this can hurt the quality of results (QoR). In particular, hierarchical EDA flows typically must provide the following capabilities:

- Partitioning the design logically and physically into the top-level design and the various partition blocks. In industrial-size designs, the highest-level modules typically exhibit independent functionalities (e.g., CPU core, graphics processor, memory controller, cache, interconnect PHY/controller). However, these modules are often too large to implement flat as a single block. Therefore, creating physical partitions at lower hierarchy levels where functionality-based partitioning is less evident may be necessary. This requires efficient block decomposition and floorplanning schemes.
- Automatic pin assignment for partitions, which guides the inter-partitions global routes.
- 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.
- 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.
- 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 2D designs and were not designed for high-level architectural exploration. They still involve massive manual effort and exhibit significant latency.

Moreover, 3D 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 3D exploration flow that combines human decision-making with automated assistance for less critical decisions is necessary to address this complexity. This approach will enable exploring a vast space of crucial decisions, such as complex 3D floorplanning with multi-tiered block partitioning.

B. *3D Floorplanning*

Floorplanning is a crucial step in the 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 2D or 3D 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.

Floorplanning is a revered task of historical importance in VLSI design automation that led to the development of many combinatorial and analytical algorithms. Even today, floorplanning is at the forefront of research, e.g., from the mixed-sized concurrent placement novelties from academia and commercial EDA tools [5], [6] and reinforcement learning (RL)-based macro placement [7].

However, research on 3D floorplanning is more limited. Due to the limited capabilities of EDA tools and manufacturing methods for 3D designs, current academic works and industrial applications are confined to memory-on-logic implementations with restricted architectural explorations and manual floorplanning [8]. One successful example of constrained memory-on-logic floorplanning is AMD's Ryzen V-Cache 3D IC [9], 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 2D algorithms to 3D. 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], [10], [11], are augmented to include the third layout axis. Then, transformations on the data structures are defined, including 3D changes to the floorplan, guided by optimization methods such as simulated annealing. The main sub-optimality reasons for these methods come from the difficulty of defining appropriate cost metrics to judge the quality of a 3D 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 3D hierarchical physical implementations is presented in Section III, and the proposed floorplanning methodology with new 3D cost components is shown in Section IV.

C. *State of 3D flows*

In the following, we present two existing state-of-the-art approaches to implementing face-to-face (F2F) wafer-to-wafer (W2W) bonded 3D ICs: a die-by-die-like Sequential-2D [12] and a Macro-3D flow [13]. We believe other available 3D flows unmatch the 3D awareness enabled by Macro-3D, making it most advanced towards a future, native 3D flow. The Sequential-2D is a realization of the well-known die-by-die integration scheme, which already has extensive tool support in the EDA industry. Both flows can be used for 3D hierarchical designs in a traditional black box, bottom-up approach and will serve as baselines for evaluating our proposed Hier-3D flow.

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

Key Idea. It enables 3D integration through the separate implementations of a die stack's top and bottom die with a standard 2D flow. 3D 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 2D 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 feed-throughs in the netlist of each die for each shared intra-die net, an approach highly inflexible and, therefore, 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 3D ICs in the memory-on-logic partitioning scheme.
Key Idea. The commercial 2D 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 3D integration requires matching die sizes, the Macro-3D flow increases the resulting manufacturing cost relative to the 2D 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

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 F2F-bonded 3D 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.

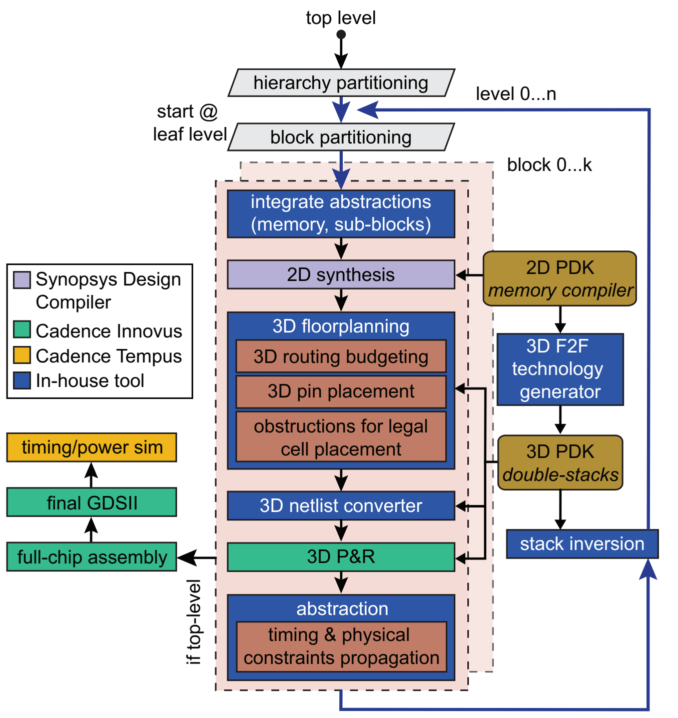

Fig. 1: The key steps in our Hier-3D flow. The hierarchy depth defines the number of outer cycles/iterations. Per iteration, synthesized block-level designs are prepared with 3D floorplanning, 3D placed-and-routed, and abstracted through our physical and timing constraints propagation. Finally, the stack and abstractions can be inverted to enable standard cell placement on the opposite die at the upper hierarchy level.

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 3D hierarchical flow is represented in Figure 1. The flow starts with an RTL whose hierarchy is predefined or 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 3D floorplanning step that places pins in the 3D stack and preserves routing resources for easier access and routing in the next step, respectively. Through whitespace modeling, placement space can also be strategically reserved for the upper levels. The netlist is subsequently modified to instantiate the cells of the 3D process design kit (PDK) corresponding to the tier assignments.

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 (=2D) or two dies (=3D), depending on the designer's choice.

TABLE I: Qualitative comparison among state-of-the-art physical design tools for 3D ICs and this work.

	Sequential-2D	Macro-3D [13]	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

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

B. *3D 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.

C. *Holistic 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`. However, more complex schemes can also be implemented to balance the routing resources more finely.

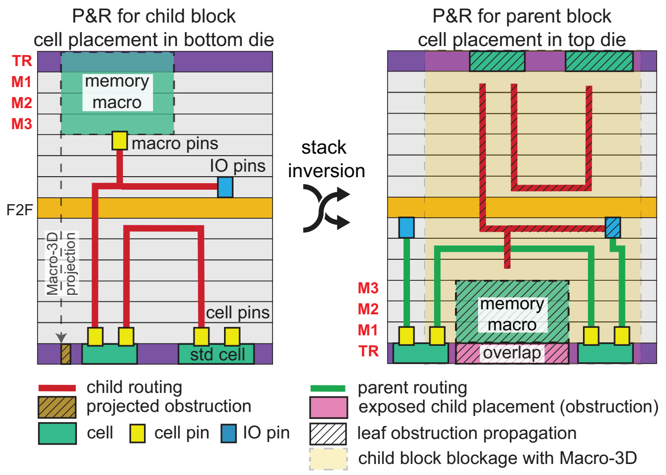

Fig. 2: Hier-3D's physical constraint propagation, stack inversion, and multi-tier cell placement. Top macros are projected as site-sized cells at the bottom to not obstruct standard cell placement, and IO pins can be placed anywhere inside the stack to facilitate upper-level connections (=left). The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

D. *Physical/Timing Constraint Propagation*

To enable the utilization of unused placement and routing resources by the P&R engines, we extend the Library Exchange Format (LEF) abstracts of implemented sub-blocks to enumerate all objects and structures in the placement and routing database 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 Figure 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 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 2D polygons and saved as rectilinear-shaped overlaps similar to the memory macros. We present this approach in the following subsection. This approach enables a full context view of the current level and implemented sub-

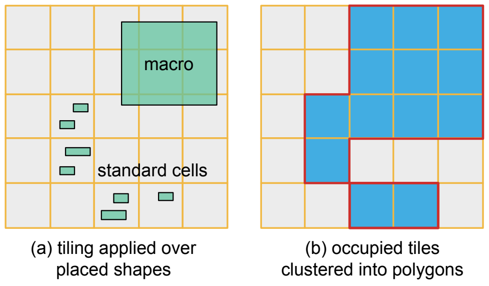
macro
standard cells
(a) tiling applied over placed shapes
(b) occupied tiles clustered into polygons

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 trade-off between runtime and area recovered. (b) Occupied tiles are merged into a set of rectilinear polygons to define `OVERLAP` obstructions for the detailed LEF.

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 Liberty (LIB) files.

E. 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 number of standard cells prohibits propagating their shapes individually, increasing LEF size and dramatically slowing down the EDA tool.

Thus, we propose a whitespace modeling method for efficient placement information propagation, depicted in Figure 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 corresponding to the occupied tiles are unioned to merge touching shapes, cropped to the floorplan box FP, yielding a set of connected components with polygon forms,

$$P = \cup_{\hexagon}\left(\left(\cup_{\square} O\right) \cap_{\square} FP\right), \tag{2}$$

where $\square$ and $\hexagon$ 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),

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

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.

F. Sequential Multi-Tier Cell Placement

Figure 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 resulting floorplan.

To better exploit the capabilities/assumptions of the P&R tool meant for standard 2D environments during clock tree synthesis (CTS) and routing, we invert the 3D 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 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 3D 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 LIB 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.

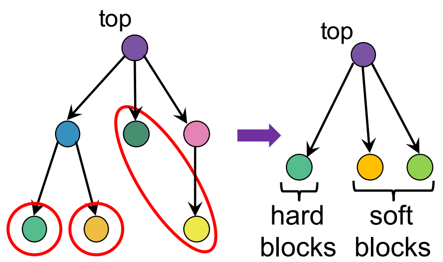

Fig. 4: An illustration of the RTL clustering algorithm formally described in Figure 5. Hard blocks form their own cluster in the resulting flattened netlist, while standard cells are clustered into soft blocks.

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.

IV. Automated Floorplanning

This section presents an automated approach to obtaining 3D 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 3D.

In some cases, the original RTL hierarchy may not be appropriate for deriving the physical implementation hierarchy. This is because while leaf modules tend to have cells placed closely in flat approaches, this is not the case when traversing the implementation hierarchy upwards, as modules tend to be connected to many others. This situation is prevalent in 3D designs.

A. Block Generation with RTL Clustering

Typically, the logical hierarchy determined by the RTL ultimately governs the physical hierarchy. However, adhering to this approach in certain situations may prove impractical, mainly when a logical hierarchy is not readily discernible. Furthermore, floorplanning engines that rely on combinatorial methods must operate on a simplified version of the netlist to handle the exponential complexity of the representation space in the number of blocks.

To overcome these limitations, we propose an automated clustering of the netlist, illustrated in Figure 5. 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 Figure 4. This method heeds that logical

```
input: netlist $N$, area threshold $t$
output: clusters from cluster($N \setminus macros(N)$, $t$) $\cup$ $macros(N)$ $\cup$
    remaining set of unclustered cells
    cluster($O$, $t$):
    $C = \varnothing$
    if $area(O) \geq t$ then
        $Q = sub\text{-}modules(O)$; sort $Q$ by decreasing area
        if $area(Q) \geq t$ then
            for $q$ in $Q$ do
                $R$ = sum of $area(i)$ for $i$ after $q$ in $Q$
                $C = C \cup$ 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. 5: Automated RTL clustering.

connectivity is a good predictor of physical position. As a matter of fact, flat placers typically follow logical information by placing cells from the same module together when optimizing wirelength, timing, power, and congestion.

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 3D 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 3D floorplan without real physical design feedback is difficult. Therefore, we propose

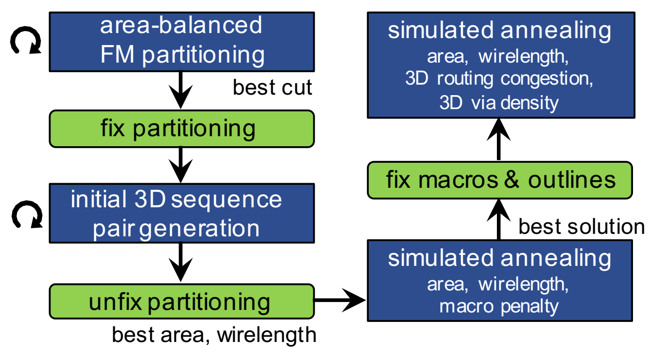
area-balanced FM partitioning
best cut
fix partitioning
initial 3D sequence pair generation
unfix partitioning
best area, wirelength
simulated annealing
area, wirelength, macro penalty
best solution
fix macros & outlines
simulated annealing
area, wirelength, 3D routing congestion, 3D via density

Fig. 6: Hier-3D's automated floorplanning methodology. A Fiduccia–Mattheyses (FM) partitioning [14] produces a min-cut tier assignment of the blocks. Then, an optimized 3D sequence pair is generated for the two-level simulated annealing (SA).

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 [6], [7] 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 3D 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),

$$c_{\text{area}} = a \cdot \left(1 + \frac{\max\{0, w - w_T\}}{w_T} + \frac{\max\{0, h - h_T\}}{h_T}\right)^2, \tag{5}$$

where the area cost is simply the area $a = (w, h)$ of the 3D 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 [15]. 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

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

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

$$w(e) = \left(1 - \frac{\mathrm{slack}(e)}{T}\right)^2, \tag{7}$$

where T is the clock period and the slack of each net e is extracted with static timing analysis (STA) on the synthesized netlist [16]. Finally, the wirelength cost is

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

3D Congestion Cost. We estimate congestion by using the simple and accurate estimation method called rectangular uniform wire density (RUDY) [17]. We extend the 2D formulation specifically for 3D designs by considering the 3D routing and F2F vias/bumps. The placement canvas is gridded, and for each bin b the horizontal/vertical (H/V) routing congestion is

$$\mathrm{RUDY}_{H/V}(b) = \sum_{e \in E_{\text{merged}}} \mathrm{RISA}(e) \cdot \frac{\mathrm{nbits}(e)}{\mathrm{capa.}_{H/V}(b)} \cdot \frac{\mathrm{OA}(e, b)}{h/w_e} \cdot Z(e), \tag{9}$$

where capa. is the number of metal resources, $Z(e) = 1/2$ if the net e is 3D, otherwise, $Z(e) = 1$. The smaller Z weight for the 3D nets models the availability of more routing resources from both tiers ($1/2$ assumes a 3D mirrored stack). $\mathrm{RISA}(e)$ serves as net weighting based on pin count to improve correlation with routed wirelength [18].

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 2D prefix-sum [19] 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,

$$\mathrm{SAT}(i, j) = \sum_{0 \le r < i} \sum_{0 \le c < j} \mathrm{RUDY}(r, c), \tag{10}$$

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, as SAT(LowerRight) + SAT(UpperRight) − SAT(LowerLeft) − SAT(UpperLeft).

F2F Via Density. The RUDY estimation does not model the pitch and spacing considerations of the F2F connections for the 3D nets. However, it is critical to model these, given that many DRCs occur when the required 3D interconnection density rises in our experiments. This is a typical problem when using traditional 2D global routers for 3D nets [20]. Therefore, we propose a RUDY-inspired method to model the F2F via density,

$$D_{\text{F2F}}(b) = \sum_{e \in E_{\text{merged}}} \mathbb{1}\left[e \text{ cut}\right] \cdot \frac{\mathrm{OA}(e, b)}{\mathrm{area}(b)} \cdot \frac{\mathrm{nbits}(e)}{\mathrm{nvias}(w_e, h_e)}, \tag{11}$$

where (w_e, h_e) is the size of the bounding box of net e, and $\mathrm{nvias}(w_e, h_e) = \dfrac{w_e \cdot h_e}{px \cdot py}$.

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

$$\begin{aligned} c_{\text{cong.}} = 2 \cdot \max_{b:\text{ bins}} \{D_{\text{F2F}}(b)\} + \max\{\overline{\mathrm{RUDY}_H}, \overline{\mathrm{RUDY}_V}\} \\ + \max\{\sigma(\mathrm{RUDY}_H), \sigma(\mathrm{RUDY}_V)\}, \end{aligned} \tag{12}$$

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

D. 3D Sequence Pair

Similarly to [2], we use a 3D sequence pair (SP) to represent a slicing floorplan solution. The 3D SP maintains one 2D SP per tier. In each 2D 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 3D SP to modify the floorplan. The main difference from a single 2D SP is that blocks can move from one pair to another in 3D to explore the 3D 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 2D 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. Note that predefined partitioning guidance is enabled by rejecting 3D moves to keep critical components in their initial 2D SP.

E. Floorplan Space Multi-Level Exploration

We use the simulated annealing (SA) algorithm [21] to explore the floorplan space stochastically. It is regarded as one of the most essential and successful metaheuristics for combinatorial optimization. It adopts a hill-climbing technique to reach global optima.

However, we observed that SA often cannot produce floorplans where the number of 3D interconnections is in the expected range. While many interconnections are only an issue when the F2F via density is high, the latter is also challenging to optimize. Therefore, we perform a preliminary area-balanced Fiduccia–Mattheyses (FM) min-cut partitioning [14] to propose a good starting point for SA. The FM cost is modified to handle timing, as in for the HPWL,

$$c_{\text{FM}} = \sum_{e \in E_{\text{merged}}} \mathbf{1}\left[e \text{ cut}\right] \cdot w(e) \cdot \text{nbits}(e). \tag{13}$$

Ulterior 3D moves changing the partitioning will be selected with very low probability during SA and rejected in case of a large F2F via count degradation.

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}. For example, for the designs of the MemPool 3D tile, we use 500 random starting seeds. Then, after inserting the blocks in their respective 2D SPs, we optimize the starting SPs by generating many random candidates (around 100K for the tile)—fixing the partitioning—and greedily picking the best configuration in terms of area and wirelength only.

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

Our multi-level optimization scheme is depicted in Figure 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 Eq. 5. Again, different user-predefined guidances are enabled here. The second stage of SA optimizes area, wirelength, and congestion using an adaptive temperature schedule, $T_i = (\Delta cost < 0) ? T_{i-1} : \alpha T_{i-1}$, 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 multi-objectives as follows,

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

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 Figure 6 takes less than a minute on the 3D MemPool tile (50 blocks/2K merged nets) and a few minutes on the 3D 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., [22]–[24]) to achieve better cost metrics.

F. From Floorplan to Implementation

The floorplanning solutions must be transformed for Hier-3D's sequential multi-tier 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 3D 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 3D 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 [20] 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.

Finally, we generate *Tcl* scripts for floorplanning inside the commercial EDA tool, defining the floorplan core outline, placing IO pins and hard macros, and generating soft placement regions for the standard cell clusters.

V. Industry Benchmarks

A. Architecture Description

To evaluate our proposed flow, we implement MemPool [25] as a representative example for tiled manycore architectures, the ARM Cortex-A53 representing ultra-high 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. Figure 7 details their very different hierarchical architectures. The diversity and complexity exhibited by these experiments illustrate the general applicability of our novel 3D integration flow for large complex modern multi-core SoCs.

First, MemPool integrates 256 RISC-V cores with 1 MiB of shared scratchpad L1 data memory (SPMs). The *group* connects

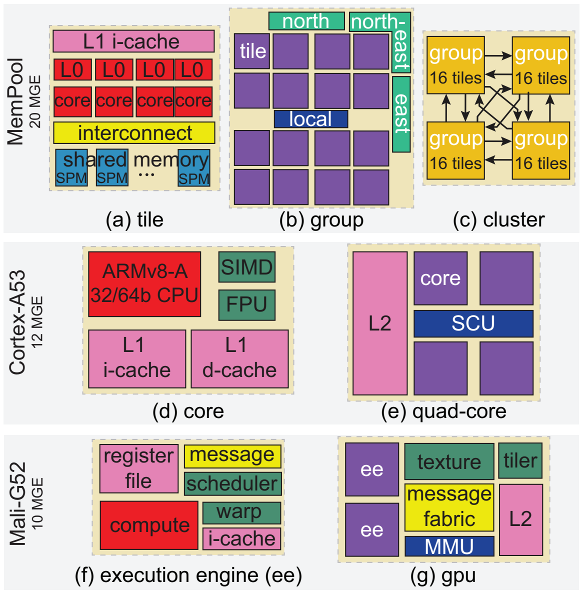

Fig. 7: Our three hierarchical benchmarks, MemPool [25], ARM Cortex-A53, and Mali-G52, implemented in TSMC's 28 and 16 nm processes.

sixteen *tiles* through a full crossbar interconnect locally and to tiles in the top-level *cluster* through a hierarchical crossbar. Second, the Cortex-A53 single-core includes a 32 KiB L1 d-cache, an advanced SIMD extension, and an FPU. The four cores share 1 MiB of L2 cache with a snoop control unit (SCU). Third, the Mali-G52 has one double-pixel shader core, with two execution engines (EE) with 48 warps each and 128 KiB of L2 memory.

The three designs have 20M, 12M, and 10M gate equivalent (GE) complexity, respectively, thus requiring a hierarchical implementation flow to keep tool execution time reasonable. The complexity per core of the Cortex-A53 is significantly higher than that of MemPool's tile and includes a multi-level cache hierarchy, while MemPool features a software-managed shared L1 SPM. In contrast, the Mali-G52 is dominated by a high standard cell logic, while MemPool's architecture is interconnect-centric with a large amount of memory.

We normalize our data to avoid revealing proprietary information for these commercial processors.

B. Default Hierarchy Management

We follow the flow shown in Figure 1 for hierarchy handling. The hierarchy is defined for MemPool following the original publication [25], while for the ARM designs, we use the recommended hierarchical cuts from the reference documentation, as depicted in Figure 7. Moreover, we apply the provided timing constraints for each level for the synthesis and P&Rs.

C. Default Macro Floorplanning

The benchmarks exhibit heavy memory/logic imbalance. The single-core Cortex-A53 is dominated by standard cells, while memory macros dominate its upper level. In contrast, standard cells dominate all hierarchy levels of the Mali-G52 and MemPool.

The 2D floorplans of the MemPool sub-designs are obtained from the original paper [25]. The 2D floorplans of the ARM Cortex-A53 and Mali-G52 are industrial-strength based on the official documentation. Instead, the Sequential-2D and Macro-3D floorplans are identical and obtained by projecting the 2D macro placement to the top tier and shrinking the floorplan to reach iso-logic density with the 2D counterpart. Note that the dimensions of the instantiated sub-blocks significantly constrain these highly area-optimized floorplans.

In Hier-3D, we implement the leaf level on the bottom die and invert the stack in each subsequent level. Note, however, that our flow further allows the study of diverse and complex floorplan constellations with different rules to decide which blocks are 3D or 2D and when and for which sub-blocks the stack inversion is applied. This enables exploring 3D logic-on-logic partitioning scenarios without compromising the theoretical partitioning design space.

VI. Experiments

A. Experimental Settings

Our proposed approach can be applied to face-to-back (F2B) or F2F die stacking. However, we focus on F2F W2W bonding for our case studies, where two prefabricated wafers are stacked and bonded together on their top metal layer. While integration techniques such as die-to-wafer bonding exist to process differently sized dies [26], W2W bonding offers a vertical connection pitch of less than 1 µm [27] but requires two dies with matching dimensions.

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 3D/2D automated floorplanning. In the 3D implementations, where we assume that the 3D technology is independent of the CMOS technology, the F2F via size, pitch, resistance, and capacitance are 0.5 µm×0.5 µm, 1.0 µm, 0.5 Ω, and 1 fF, respectively [27]. The 3D BEOL is defined in a custom 3D 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 [13] and implement the Sequential-2D flow as the construction of two sequential 2D implementations [12]. For a fair comparison, the 3D implementations include a (close to) balanced mirrored stack of the 2D 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

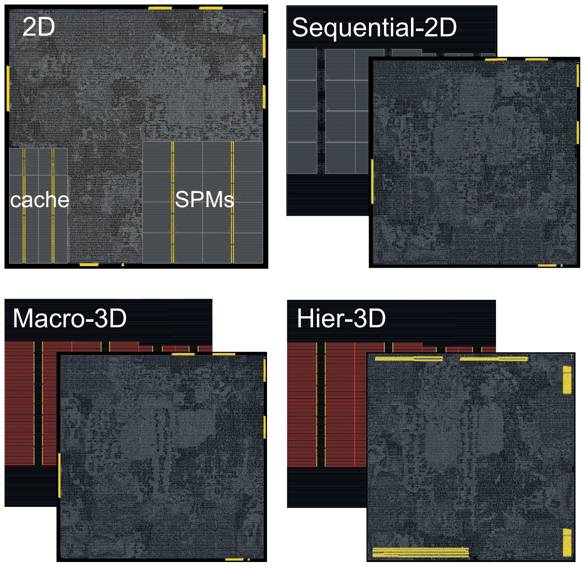

Fig. 8: Placement (to scale) of the MemPool single-tile designs, using TSMC 28 nm process: 2D vs. Sequential-2D vs. Macro-3D vs. Hier-3D.

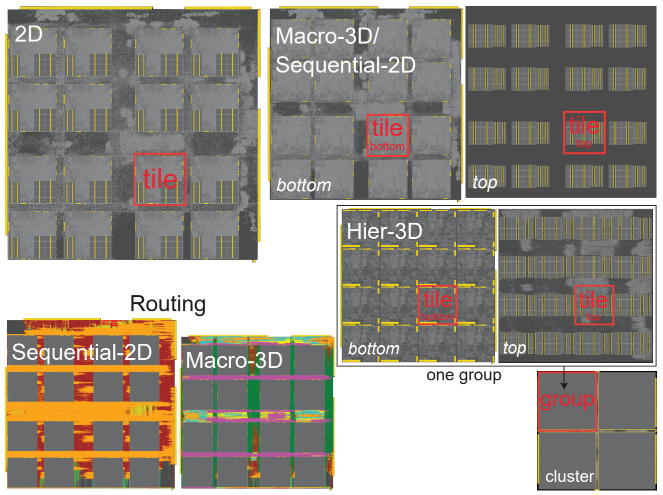

Fig. 9: Placement of the MemPool 16-tile group designs using TSMC 28 nm process: 2D vs. Macro-3D/Sequential-2D vs. Hier-3D and Sequential-2D, Macro-3D routing.

partitioning is automatized with *Tcl* inside *Cadence Genus*. The 3D 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 physical hierarchy and manual floorplan settings shown in Section V.

1) MemPool Design:

a) PPA Results: While the tile implementation PPA metrics are very similar across all integration flows (manual floorplans can be seen in Figure 8), 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

TABLE II: 64-tile MemPool Cluster 2D vs. Sequential-2D vs. Macro-3D vs. Hier-3D, using TSMC 28 nm process. 7.3M cells & 13.1M nets & 1536 memory macros. Values are normalized w.r.t. 2D 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

TABLE III: Clock tree metrics of the 16-tile MemPool Group 2D vs. Sequential-2D vs. Macro-3D vs. Hier-3D, using TSMC 28 nm process. Values are normalized w.r.t. 2D implementation.

MemPool Group	2D	Seq-2D	Macro-3D	Hier-3D
# clock buffers	1	1.13	0.88	0.60
max depth	1	0.90	0.72	0.46
avg. latency	1	0.95	0.87	0.71
max skew	1	1.33	0.87	0.51

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 Figure 9. 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. 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 [28] 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.

b) Clock Tree Comparison: Table III shows Hier-3D exhibits much better clock tree characteristics. The reduced floorplan considerably reduces the distances between the clock source pin and flip-flop sinks, simplifying clock optimization for the tool.

c) Wirelength Distribution: To study the performance gain offered by Hier-3D, we extract the wirelength of the nets in the group and plot the histogram in Figure 10. This shows an extreme trend towards a general net wirelength reduction in Hier-3D, indicative of the effectiveness of our flow.

d) PPA Evolution with Hierarchy: We summarize in Figure 11 the effects of the hierarchy on the implementation quality of the MemPool design in TSMC 28 nm. Our new flow exhibits excellent results for all metrics, which improve with

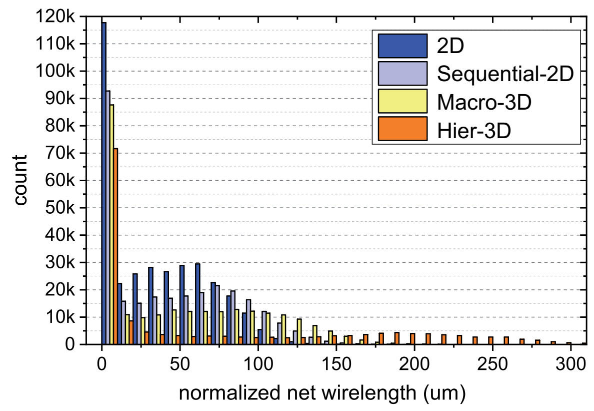

Fig. 10: Wirelength distribution of inter-tiles nets in the 16-tile MemPool Group designs using TSMC 28 nm process. We observe a more uniform distribution of net lengths for Hier-3D.

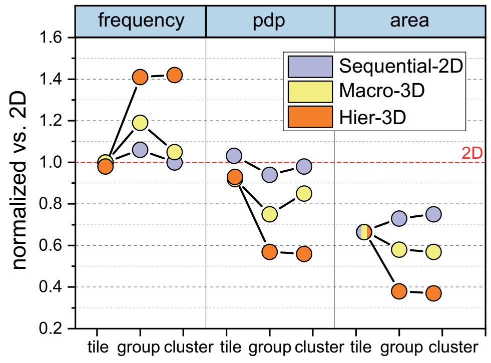

Fig. 11: Evolution of key PPA metrics depending on the hierarchy level for the MemPool design in TSMC 28 nm, normalized w.r.t. the 2D implementation. Hier-3D offers larger gains with the advance of the hierarchy complexity.

the complexity of the level: the PPA gap between Hier-3D and the other flows keeps increasing as we go up in the hierarchy tree, which makes Hier-3D more and more competitive for designs with many more levels of hierarchy, which represent the current industrial trend. Hier-3D embodies a suitable middle-ground between the benefits of a fully black-boxed hierarchical methodology (easy timing convergence, runtime scalability) and a flat approach (better optimization capability).

e) Technology Node Impact: 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 IV. 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 vs. memory

$$\frac{\text{area(die)} - \text{area(memory)}}{\text{area(site)}}, \tag{14}$$

increases by 1.2× from 28 nm to 16 nm for the MemPool Cluster, offering additional silicon area gains. This trend

TABLE IV: 64-tile MemPool Cluster 2D vs. Macro-3D vs. Hier-3D, using TSMC 16 nm process. Values are normalized w.r.t. 2D 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

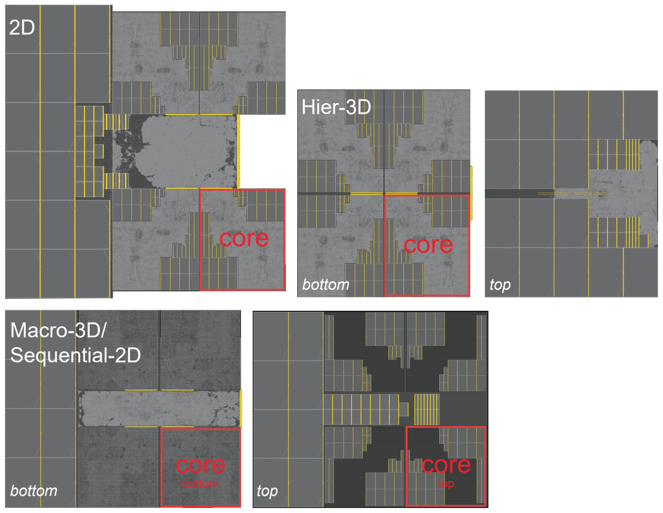

Fig. 12: Placement (to scale) of the Cortex-A53 4-core designs using TSMC 16 nm process: 2D vs. Macro-3D/Sequential-2D vs. Hier-3D.

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 Figure 12. The Hier-3D floorplan instantiates the 2D 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 V shows that the Hier-3D flow surpasses the Macro-3D flow in frequency and power, with

TABLE V: 4-Core Cortex-A53 2D vs. Sequential-2D vs. Macro-3D vs. Hier-3D, using TSMC 16 nm process. 2.4M cells & 2.5M nets & 165 memory macros. Values are normalized w.r.t. 2D implementation. 2D 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

TABLE VI: 2-Execution Engine Mali-G52 2D vs. Sequential-2D vs. Macro-3D vs. Hier-3D, using TSMC 16 nm process. 4.4 M cells & 4.8M nets & 141 memory macros. Values are normalized w.r.t. 2D 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

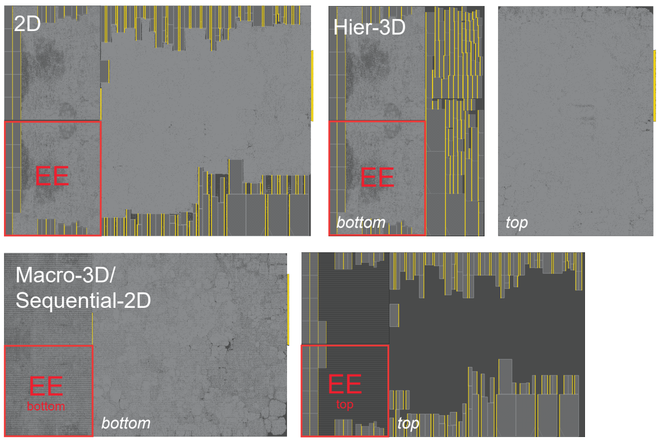

Fig. 13: Placement (to scale) of the Mali-G52 2-EE designs using TSMC 16 nm process: 2D vs. Macro-3D/Sequential-2D vs. Hier-3D.

drastic improvement in the silicon area. Again, all composing metrics are improved simultaneously, providing a total PPC bump of 51 % over the 2D reference.

3) Mali-G52 Design Results: We floorplan the Hier-3D GPU using a 2D bin-packing method, assigning the upper and lower edge macros of the 2D floorplan into two separate bins. The packed macros are placed next to the 2D EE abstractions on the bottom die, allowing the top-level standard cells to fully utilize the upper die, as shown in Figure 13.

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

4) Summary: Despite the excellent reference of the timing-optimized industry-recommended 2D IC floorplans, our implementations consistently achieve a smaller silicon area while delivering substantial PPC improvements for the three benchmarks, even though the three designs are pretty different. Typically, designs display a constant power-delay product, where a frequency gain increases power. However, compared to the competing flows, Hier-3D remarkably overcomes this

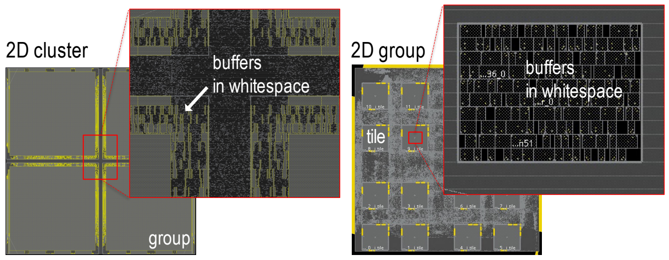

Fig. 14: Whitespace modeling: opening the group boundary for the cluster implementation using TSMC 28 nm process (= left), and reserving a small square in the middle of the tile for the group implementation using TSMC 16 nm process (= right).

power versus performance trade-off, improving both metrics simultaneously.

The experiments run on a Linux server with a 24-core Intel Xeon E5-2640 @ 2.5 GHz with 15 MiB L3 cache. The substantial runtime improvements of Hier-3D observed on all benchmarks—the Mali-G52 P&R runtime is about three to four days—make it scalable to sizeable modern multi-core SoCs.

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 2D examples. Note that this methodology can be applied similarly to 3D implementations, but the analysis in 2D 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 VII.

- First, we use whitespace modeling to open up placement space on the group boundary for the cluster level, as seen in Figure 14. 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.

TABLE VII: Whitespace modeling on 2D MemPool Cluster and Group designs. PPA metrics are for inter-groups and inter-tiles only, respectively.

MemPool	Cluster (28 nm) 2D	Cluster (28 nm) +whitespace	Group (16 nm) 2D	Group (16 nm) +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

TABLE VIII: 16-tile MemPool Group Hier-3D architecture exploration, using TSMC 28 nm process. 1.3M cells & 3.3M nets & 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

- 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 Figure 14. 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:

a) 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 3D physical 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 Figure 15, and the PPA results are tabulated in Table VIII. These show a reduced footprint area with an increased placement density while maintaining comparable performance.

b) 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

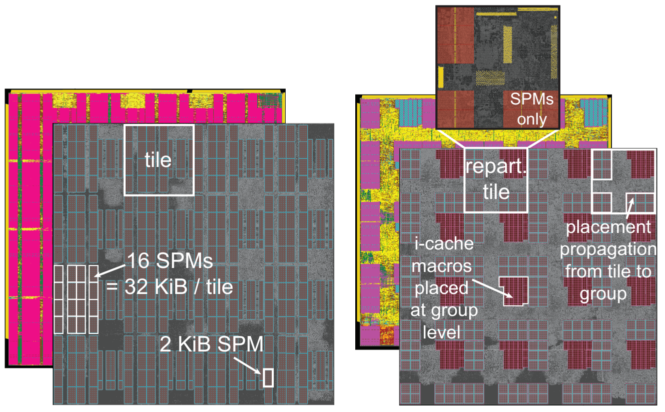

Fig. 15: MemPool physical and logical architecture exploration, using TSMC 28 nm process. Enlarging the SPMs from 1 KiB to 2 KiB (= left). Repartitioning of the tile instruction cache from the tile level to the group level (= right).

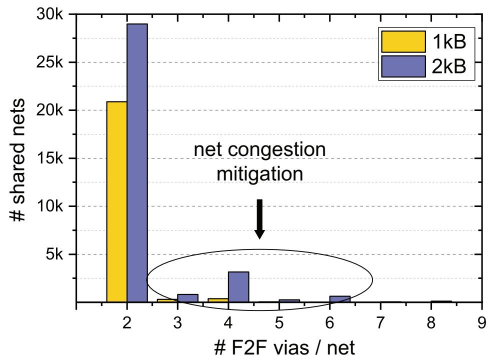

Fig. 16: Comparison of the number of vias used per shared 2D net in the Hier-3D 16-tile MemPool Group implementations, using TSMC 28 nm process. The group with 2 KiB shared memory macros (512 KiB total) benefits from a non-negligible number of nets alternating between two tiers, yielding a relief of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

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 [8] with promising results but lacking the flexibility of Hier-3D logic-on-logic capabilities. The corresponding placement and routing layouts are shown in Figure 15. Table VIII shows this change induces limited degradation in the PPA of the group, despite the much denser 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 Figure 16, 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 [8].

3) Automated Floorplanning: Table IX summarizes the floorplanning results. While the proposed solutions do not

TABLE IX: 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

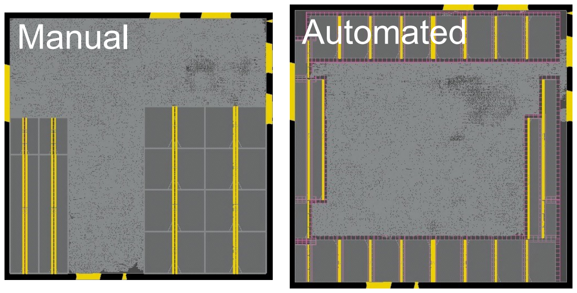

Fig. 17: Placement (to scale) of the MemPool single-tile designs using TSMC 16 nm process: 2D manual vs. 2D automated floorplan.

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.

a) 2D: Our automated floorplanning methodology designed for 3D designs is applied to 2D designs by skipping the partitioning and ignoring the components of the cost of 3D during SA. As shown in Table IX, the automated floorplan solution performs better than the manual one. Still, some area penalty is depicted in Figure 17. This could theoretically be recovered by manually refining the macro locations or increasing the target density of soft clusters during floorplanning, but potentially degrading other essential metrics.

b) 3D: Based on the assignments from the 3D 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. Table IX shows that the 3D 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 3D moves that worsen the cut during floorplanning. Ultimately, better floorplan solutions can be obtained from ensembles of optimization runs of candidate floorplans obtained with SA, a typical approach used in current industrial flows.

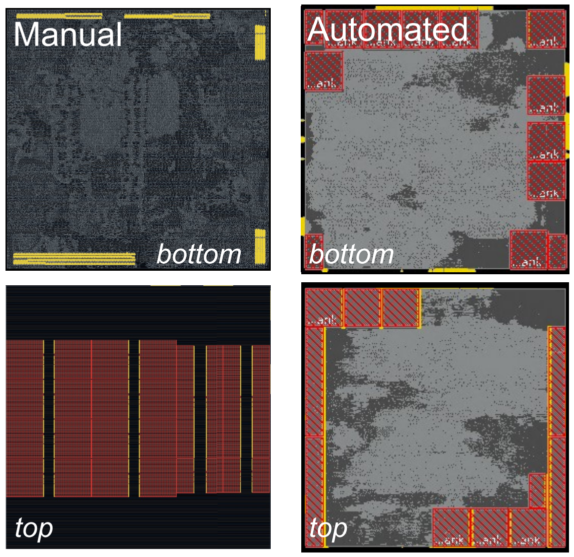

Fig. 18: Placement (to scale) of the MemPool single-tile designs using TSMC 16 nm process: 3D manual vs. 3D automated floorplan.

VII. Conclusions

We propose a full-chip RTL-to-GDSII physical design solution that offers a commercial-quality F2F-bonded 3D 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 and stack inversion, enabling multi-tier cell placement. This design flow steppingstone vastly expands the design space exploration options and can help explore physical hierarchy more efficiently on a multi-level for 3D ICs. Our proposed automated 3D 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 the 2D counterparts.

References

[1] C. Niessen, "Hierarchical design methodologies and tools for VLSI chips," Proceedings of the IEEE, vol. 71, no. 1, pp. 66–75, 1983.

[2] S. Panth, K. Samadi, Y. Du, and S. K. Lim, "High-density integration of functional modules using monolithic 3D-IC technology," in 2013 18th Asia and South Pacific Design Automation Conference (ASP-DAC). IEEE, 2013, pp. 681–686.

[3] J. Knechtel, I. L. Markov, and J. Lienig, "Assembling 2-D Blocks Into 3-D Chips," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 31, no. 2, pp. 228–241, 2012.

[4] A. Agnesina, M. Brunion, A. Garcia-Ortiz, F. Catthoor, D. Milojevic, M. Komalan, M. Cavalcante, S. Riedel, L. Benini, and S. K. Lim, "Hier-3D: A Hierarchical Physical Design Methodology for Face-to-Face-Bonded 3D ICs," in Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design, 2022, pp. 1–6.

[5] C.-K. Cheng, A. B. Kahng, S. Kundu, Y. Wang, and Z. Wang, "Assessment of Reinforcement Learning for Macro Placement," arXiv preprint arXiv:2302.11014, 2023.

[6] A. Agnesina, P. Rajvanshi, T. Yang, G. Pradipta, A. Jiao, B. Keller, B. Khailany, and H. Ren, "AutoDMP: Automated DREAMPlace-based Macro Placement," in Proceedings of the 2023 International Symposium on Physical Design, 2023.

[7] A. Mirhoseini, A. Goldie, M. Yazgan, J. W. Jiang, E. Songhori, S. Wang, Y.-J. Lee, E. Johnson, O. Pathak, A. Nazi et al., "A graph placement methodology for fast chip design," Nature, vol. 594, no. 7862, pp. 207–212, 2021.

[8] M. Cavalcante, A. Agnesina, S. Riedel, M. Brunion, A. García-Ortiz, D. Milojevic, F. Catthoor, S. K. Lim, and L. Benini, "MemPool-3D: boosting performance and efficiency of shared-l1 memory many-core clusters with 3D integration," in 2022 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, 2022, pp. 394–399.

[9] 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 2022 IEEE International Solid-State Circuits Conference (ISSCC), vol. 65. IEEE, 2022, pp. 428–429.

[10] D. Wong and C. Liu, "A new algorithm for floorplan design," in 23rd ACM/IEEE Design Automation Conference. IEEE, 1986, pp. 101–107.

[11] L. Cheng, L. Deng, and M. D. Wong, "Floorplanning for 3-D VLSI design," in Proceedings of the ASP-DAC 2005. Asia and South Pacific Design Automation Conference, 2005., vol. 1. IEEE, 2005, pp. 405–411.

[12] G. Sisto, P. Debacker, R. Chen, G. Van der Plas, R. Chou, E. Beyne, and D. Milojevic, "Design enablement of fine pitch face-to-face 3D system integration using die-by-die place & route," in 2019 International 3D Systems Integration Conference (3DIC). IEEE, 2019, pp. 1–4.

[13] L. Bamberg, A. García-Ortiz, L. Zhu, S. Pentapati, S. K. Lim et al., "Macro-3D: A physical design methodology for face-to-face-stacked heterogeneous 3D ICs," in 2020 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, 2020, pp. 37–42.

[14] C. M. Fiduccia and R. M. Mattheyses, "A linear-time heuristic for improving network partitions," in Papers on Twenty-five years of electronic design automation, 1988, pp. 241–247.

[15] A. B. Kahng, R. Varadarajan, and Z. Wang, "RTL-MP: Toward Practical, Human-Quality Chip Planning and Macro Placement," in Proceedings of the 2022 International Symposium on Physical Design, 2022, pp. 3–11.

[16] D. Z. Pan, B. Halpin, and H. Ren, "Timing-driven placement," Handbook of Algorithms for Physical Design Automation, pp. 423–446, 2008.

[17] P. Spindler and F. M. Johannes, "Fast and accurate routing demand estimation for efficient routability-driven placement," in 2007 Design, Automation & Test in Europe Conference & Exhibition. IEEE, 2007, pp. 1–6.

[18] C.-L. E. Cheng, "RISA: Accurate and efficient placement routability modeling," in Proceedings of the 1994 IEEE/ACM international conference on Computer-aided design, 1994, pp. 690–695.

[19] F. C. Crow, "Summed-area tables for texture mapping," in Proceedings of the 11th annual conference on Computer graphics and interactive techniques, 1984, pp. 207–212.

[20] S. Pentapati, Y.-H. Huang, A. Agnesina, M. Brunion, and S. K. Lim, "On Legalization of Die Bonding Bumps and Pads for 3D ICs," in Proceedings of the 2023 International Symposium on Physical Design, 2023, pp. 62–70.

[21] D. Bertsimas and J. Tsitsiklis, "Simulated annealing," Statistical science, vol. 8, no. 1, pp. 10–15, 1993.

[22] M. M. Ziegler, H.-Y. Liu, G. Gristede, B. Owens, R. Nigaglioni, and L. P. Carloni, "A synthesis-parameter tuning system for autonomous design-space exploration," in 2016 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, 2016, pp. 1148–1151.

[23] A. Agnesina, K. Chang, and S. K. Lim, "VLSI placement parameter optimization using deep reinforcement learning," in Proceedings of the 39th International Conference on Computer-Aided Design, 2020, pp. 1–9.

[24] A. Agnesina, S. Pentapati, and S. K. Lim, "A general framework for VLSI tool parameter optimization with deep reinforcement learning," in Proceedings of the NeurIPS 2020 Workshop on Machine Learning for Systems, Virtual, 2020, pp. 6–12.

[25] M. Cavalcante, S. Riedel, A. Pullini, and L. Benini, "MemPool: A shared-L1 memory many-core cluster with a low-latency interconnect," in 2021 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, 2021, pp. 701–706.

[26] A. Phommahaxay, S. Suhard, P. Bex, S. Iacovo, J. Slabbekoorn, F. Inoue, L. Peng, K. Kennes, E. Sleeckx, G. Beyer et al., "Enabling ultra-thin die to wafer hybrid bonding for future heterogeneous integrated systems," in 2019 IEEE 69th Electronic Components and Technology Conference (ECTC). IEEE, 2019, pp. 607–613.

[27] E. Beyne, S.-W. Kim, L. Peng, N. Heylen, J. De Messemaeker, O. O. Okudur, A. Phommahaxay, T.-G. Kim, M. Stucchi, D. Velenis et al., "Scalable, sub 2μm pitch, Cu/SiCN to Cu/SiCN hybrid wafer-to-wafer bonding technology," in 2017 IEEE International Electron Devices Meeting (IEDM). IEEE, 2017, pp. 32–4.

[28] A. Agnesina, M. Brunion, J. Kim, A. Garcia-Ortiz, D. Milojevic, F. Catthoor, M. Perumkunnil, and S. K. Lim, "Power, performance, area and cost analysis of memory-on-logic face-to-face bonded 3D processor designs," in 2021 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED). IEEE, 2021, pp. 1–6.