Planning Massive Interconnects in 3D Chips

Johann Knechtel, *Member, IEEE,* Evangeline F. Y. Young, *Member, IEEE,* and Jens Lienig, *Senior Member, IEEE*

***Abstract*—Three-dimensional (3D) chips rely on massive interconnect structures, i.e., large groups of through-silicon vias (TSVs) coalesced with large multi-bit buses. We observe that wirelength optimization, a classical technique for floorplanning, is not effective while planning massive interconnects. This is due to the interconnects' strong impact on multiple design criteria like wirelength, routability and temperature. To facilitate early design progress of massively-interconnected 3D chips, we propose a novel 3D-floorplanning methodology which accounts for different types of interconnects in a unified manner. One key idea is to align cores/blocks simultaneously within and across dies, thus increasing the likelihood of successfully implementing complex and massive interconnects. While planning such interconnects, we also target fast, yet accurate, thermal management, routability and fixed-outline floorplanning. Experimental results on GSRC and IBM-HB+ circuits demonstrate our tool's capabilities for both planning massive 3D interconnects and for multiobjective 3D floorplanning in general.**

***Index Terms*—3D integration, floorplanning, thermal management, block alignment, massive interconnects, bus planning**

I. Introduction

THREE-DIMENSIONAL (3D) STACKING and die bonding, to obtain *3D chips*, is a promising and critical approach for meeting current and future design criteria such as, performance, functionality, delay and power consumption. *Through-silicon vias (TSVs)*, i.e., metal structures that pass vertically through whole dies, are short, low-power interconnects; they are key enablers for high-performance 3D chips, also known as *3D integrated circuits (3D ICs)*.

Massive interconnect structures for 3D chips have been proposed to increase communication capabilities for large-scale 3D logic integration [1]–[3] and 3D memory integration [4], [5]. More generally, we consider any *multi-bit bus that runs within a single die or passes between multiple dies*—in the latter case with additional deployment of large groups of TSVs—a massive interconnect structure in our study.

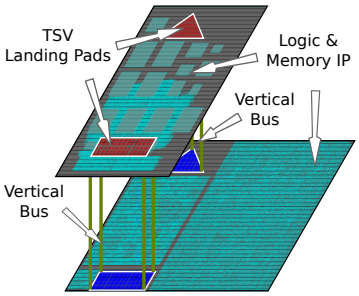


Fig. 1. Two macroblocks on adjacent dies, derived from [11]. Their massive vertical buses require the macros to be aligned so that routing paths are short and can be optimized for delay and power consumption.

In this context, the well-known *bus-planning approach*, i.e., grouping a large set of signals into adjacent wires, can be applied to 3D-chip design as well. The concept of *block alignment* has been successfully applied in two-dimensional (2D) layout representations for bus planning [6], [7], but it has been largely neglected in 3D representations. Some studies related to 3D design [8]–[10] deploy *fixed alignment*, i.e., blocks are aligned so that their relative positions exhibit specific distances. However, planning for vertical buses is only touched on in [9]. To the best of our knowledge, none of the studies heretofore consider *alignment ranges* where blocks are aligned so that their relative positions comply with upper and/or lower distance boundaries. Thus, *flexible block alignment* has not been deployed yet.

A key observation in our study is that flexible block alignment—along with "classical" fixed alignment—enables efficient and effective planning of massive 3D interconnects (Figs. 1 and 2). The resulting improvements, which we describe in this paper, stem from the simple fact that we can shorten and optimize wiring paths if the blocks to be interconnected are appropriately aligned and placed near each other in early design stages.

In this study, we propose a new methodology for planning massive interconnect structures early on in the design stages for 3D chips. We summarize our contributions as follows:

1) A 3D floorplanning methodology that emphasizes planning massive interconnect structures using block alignment is at the heart of our study. We have made this methodology available as an open-source 3D floorplanning tool [12]. Besides planning massive interconnects, the tool encompasses the following design objectives: optimized clustering of regular signal TSVs into massive interconnects; floorplanning within fixed outlines; fast thermal management; layout packing; wirelength and routability optimization.

Manuscript received October 22, 2014; revised February 23, 2015; accepted April 9, 2015. Date of publication TODO, 2015; date of current version TODO, 2015. This study was supported in part by grants from the German Research Foundation (Project 1401/1), from the Graduate Academy, TU Dresden, Germany, and from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. CUHK418611). This paper was recommended by Associate Editor S. K. Lim.

J. Knechtel was with the Institute of Electromechanical and Electronic Design, TU Dresden, Germany, when this study was conducted, and is currently with the Institute Center for Microsystems (iMicro), Masdar Institute of Science and Technology, Abu Dhabi, UAE (e-mail: jknechtel@masdar.ac.ae).

J. Lienig is with the Institute of Electromechanical and Electronic Design, TU Dresden, Germany (e-mail: jens@ieee.org).

E. F. Y. Young is with the Department of Computer Science and Engineering, Chinese University of Hong Kong, Shatin, Hong Kong (e-mail: fyyoung@cse.cuhk.edu.hk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier TODO

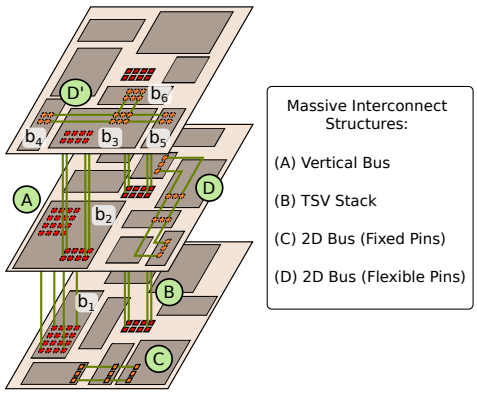


Fig. 2. Massive interconnect structures and associated block alignment in a 3D IC. Vertical buses (A) connect blocks placed onto adjacent dies. TSV stacks (B), comprising aligned bundles of TSVs, pass through two or more dies. Both (A) and (B) rely on inter-die alignment. Regular 2D buses with fixed (C) or flexible pins (D) connect blocks within dies.

2) We provide a new layout representation by extending the corner block list (CBL) [13] with a novel block-alignment concept. This concept allows us to simultaneously encode fixed alignments and alignment ranges, and to align blocks across dies (inter-die alignment) and within dies (intra-die alignment).
3) We introduce algorithms for efficient 3D layout generation, i.e., block placement and alignment as well as inherent layout packing.
4) We link the planning of massive interconnects with thermal management. Specifically, the thermal impact of TSVs is considered by our fast thermal-analysis approach which extends *power blurring* [14]. In addition, clustering regular signal TSVs into massive interconnects is thermally-aware, that is, it notably reduces maximum temperatures.

Our paper is organized as follows: First, we review a range of practical types of massive interconnects in 3D chips in Sec. II. We then provide the specifics for block-alignment encoding in Sec. III, and formulate our problem in Sec. IV. Next, we present our floorplanning methodology and its implementation in Sec. V. Experimental results are discussed in Sec. VI. Finally, we summarize our findings in Sec. VII.

II. Massive Interconnect Structures in 3D Chips and Related Block Alignment

The vertical components of massive interconnect structures in 3D chips have a strong impact on design quality. Therefore, such structures need to be considered early in the design process, to avoid re-design iterations at later stages.

The following aspects are particularly critical for TSV-based 3D ICs. First, TSVs introduce stress in the surrounding silicon which affects nearby transistors [15]. However, as *TSV islands* (i.e., grouped bundles of TSVs) need not include active gates, they mitigate this negative effect. Second, TSVs occupy a large design area. However, TSV islands can reduce area overhead: TSVs can be packed densely within islands, possibly reducing keep-out zones without increasing stress-induced impact on active gates [16]. Third, compared to spread-out single TSVs, TSV islands can more effectively reduce maximum temperatures by improving vertical heat conduction [17], [18].

Despite some drawbacks, such as the possibility of increased delays and signal-integrity issues due to coupling between TSVs [19], the benefits noted above confirm the superiority of TSV islands over single TSVs. This has also been borne out by previous studies, e.g., [2], [20], [21]. However, none of these studies proposed effective techniques for planning TSV islands in conjunction with classical highly-parallel buses. And to this day, no such *generalized* planning approach for massive interconnect structures in 3D chips has emerged.

A. Interconnect Types and Alignment Classification

We consider the (TSV-based) *block-level design* style. For 3D integration following this style, blocks / modules are typically handled as encapsulated entities and assigned to separate dies rather than split up among different dies. Block-level design is reputed to be reliable and efficient, especially for the first practical 3D-IC applications [2], [3], [20], [21].

Block alignment in 3D chips can be classified into (i) *inter-die alignment*, i.e., blocks placed among several dies are to be aligned, and (ii) *intra-die alignment*, i.e., blocks are to be aligned within one die. The alignment specifics arise from different scenarios for 3D integration and interconnects which are discussed next and illustrated in Fig. 2

Vertical buses (Fig. 2 (A)) connect blocks spread among multiple dies; in practice such interconnects are common in 3D chips. For example, consider the two macroblocks in Fig. 1. The tightly interconnected modules embed two vertical buses, which are implemented by large groups of TSVs[1]. Accounting for vertical buses during floorplanning requires capabilities for *inter-die alignment for block intersection*. That is, in order to include a large number of vertical interconnects, the related blocks must exhibit some minimal intersecting region when considering their projection onto a 2D plane.

It is important to note that blocks initially designed for 2D chips may need to be adapted for such embedded vertical buses. For example, hard design blocks with fixed layouts cannot include massive vertical interconnects. Such blocks will need to be either redesigned or encapsulated into macro blocks with sufficient design area for integrating the interconnects.

A special case of vertical buses are *aligned TSV stacks* (Fig. 2 (B)), where bundled TSVs are placed such that they pass straight through multiple dies. TSV stacks are, for example, applied for regular 3D NoCs, or to limit power-supply noise and to simultaneously improve thermal distribution [22]–[24]. Placement of aligned TSV stacks requires *inter-die alignment with fixed, zero offsets*. In other words, TSV stacks placed among separate dies must be arranged such that their outlines completely overlap in both x- and y-dimensions.

Fixed or flexible 2D buses (Fig. 2 (C, D)) are used to connect blocks within dies. Such buses are traditionally deployed to optimize datapath interconnects. Note that such structures rely on *intra-die alignment* of related blocks. Depending on whether blocks contain *fixed pins* (C) or *flexible pins* (D),

[1]Note that our approach is not restrictive in this context, and can be applied for different 3D manufacturing technologies.

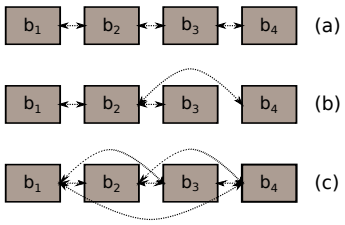


Fig. 3. Alignment topologies and related pairwise block relationships. The *chain* (a) relates blocks in a sequential manner; the *fixed reference* topology (b) defines one block as global reference (here b_2); and the *complete graph* (c) puts all blocks into a pairwise relationship with each other.

planning 2D buses requires support for *fixed alignment* or *alignment ranges*, i.e., alignment with fixed or flexible offsets.

B. Multi-Bend and Compound Interconnects

A flexible block arrangement, to enable *multi-bend interconnects*, is a crucial measure for massive interconnects comprising many blocks [7]. Furthermore, the interconnect structures introduced above are not necessarily disconnected.

Although our alignment approach (Sec. III) can encode any arbitrary block arrangement, we restrict it in the following to practical arrangements for massive interconnect structures. Arranging blocks in a pairwise fashion is at the core of our alignment approach. Specifically, we differentiate three topologies: *chains*, *fixed references*, and *complete graphs* (Fig. 3).

Chains and complete graphs are deployed for vertical buses. Chain-based alignment is more flexible; it enables lateral offsets of vertical-bus segments (Fig. 2 (A)). Complete graphs, in contrast, enforce TSV-stack-like interconnects.

Both chain and fixed reference topologies are applicable to TSV stacks. In addition, they produce comparable interconnect structures, assuming that the TSV stack blocks are described by fixed, zero offsets.

We implement flexible buses by selecting a fixed reference or by using a chain topology. Any straight-line and/or cross-like interconnect structure can be deployed using a fixed reference (Fig. 2 (D')). And any multi-bend interconnect structure can be created using chain topologies (Fig. 2 (D)). Thus, this latter topology is more flexible when planning massive buses.

Fixed buses exhibit similar structures like flexible buses, but account for fixed block pins using fixed offsets.

Compound interconnects, i.e., multiple interconnect structures linked together, can be defined explicitly or created implicitly. Take for example an explicit definition of a vertical bus linked to a flexible 2D bus in Fig. 2 (A & D')): in this case block alignment is needed for both interconnects ($b_{1..3}$ are aligned for the vertical bus and $b_{3..6}$ for the 2D bus) such that one block (here b_3) is covered in both alignments of the 2D bus and the vertical bus. We can also implicitly create compound interconnects by flexible alignment encoding—which is at the heart of our methodology—such that blocks are not restricted to particular dies. For example, think of one block in the 2D bus of Fig. 2 (C) being placed in another but the lowest die. This results in a compound structure of a 2D bus linked with a vertical bus. The notion of this feature is similar to allowing multi-bend structures; massive interconnects linking a large set of blocks may be easier to implement when they are flexible and can employ the full design space of the 3D chip.

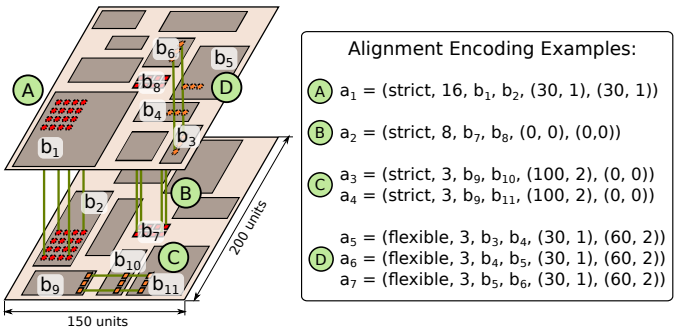


Fig. 4. Massive interconnect structures in a 3D IC with corresponding block-alignment encoding.

III. Block-Alignment Encoding

Each massive interconnect structure is described by a set of *block-alignment tuples* $\{a_1, \ldots, a_m\}$ and each tuple a_k is defined as

$$a_k = (AH, wires, b_i, b_j, (x, AT_x), (y, AT_y))$$

where AH denotes the alignment handling; $wires$ encodes the interconnect's signal count; and b_i and b_j denote the respective blocks of the massive interconnect. Further, (x, AT_x) and (y, AT_y) denote the partial alignment requests with respect to b_i's and b_j's x- and y-coordinates. This encoding allows us to *independently and simultaneously* align blocks in both the x- and y-directions. Thus, the different types of massive interconnect structures discussed in Sec. II can be implemented. Further encoding details are explained below.

A. Types of Block Alignment

The alignment handling AH can be either *strict* or *flexible*; the former requires that requests for the x- and y-directions are kept, while the latter allows x- and y-direction request to be swapped. As explained below in Sec. III-B, flexible handling is beneficial for planning of flexible buses.

The options for block alignment are *fixed offsets* ($AT = 0$), *minimal overlaps* ($AT = 1$), *maximum distances* ($AT = 2$) and *don't care* ($AT = -1$). The meaning and application of these types is explained next; examples are illustrated in Fig. 4.

Given a *fixed offset*, b_j is to be placed x/y units to the right/top ($x/y \geq 0$) or to the left/bottom ($x/y < 0$) of b_i, respectively. Placement offsets are defined with respect to (w.r.t.) the blocks' lower-left corners. Fixed-offset alignment is required for restricted interconnects, e.g., when connecting blocks with fixed pins (Fig. 4 (B, C)).

Considering a positive *minimal overlap*, the projected intersection of blocks b_i and b_j must be *at least* x units wide and/or y units high. The idea is to enable shortest-path interconnects within the blocks' projected intersection, e.g., as shown for a vertical bus in Fig. 4 (A). Note that this type of alignment—when defined for one dimension—is traditionally considered for 2D bus planning, e.g., in [25]. For convenience, it is also deployed for 2D buses in our methodology.

Defining a *maximum distance* requires that the center points of blocks b_i and b_j are *at most* x/y units apart. This way, massive interconnects can easily be limited in their length and/or width (Fig. 4 (C, D)).

B. Flexible Alignment Encoding

Note that the tuples introduced above can easily be used for both intra- or inter-die alignment. In other words, *our encoding does not restrict blocks to particular dies and does not require partitioning.* As discussed, this enables flexible implementation of compound interconnects. Such compound structures, in turn, naturally facilitate the optimization of massive interconnects along with other design criteria during floorplanning (Sec. V).

For *flexible 2D buses*, we prefer *flexible* alignment handling. This enables partial alignments in the x- and y-directions—typically minimal overlap and maximum distance—to be swapped dynamically during floorplanning optimization. This enables each block pair to form either a vertical or horizontal bus segment, and the overall bus structure is predominantly multi-bend in nature—thus increasing the probability of generating successful interconnects. For example, the tuple a_7 in Fig. 4 is handled dynamically so that blocks b_5 and b_6 have a minimal overlap in the y-dimension and a maximum distance in the x-dimension, contrary to the illustrated initial encoding.

In addition, we support *module preplacement*. Suppose block b_j needs to be placed at fixed coordinates: this request can be encoded as $(b_{LL}, b_j, (x, 0), (y, 0))$, where x and y denote the preplacement coordinates and b_{LL} is a dummy block representing the 3D IC's lower-left corner.

IV. Problem Formulation

We assume the following input for planning massive interconnects during floorplanning of 3D chips.

- (i) *Dies*, denoted as set $\mathcal{D}$. Each die $d \in \mathcal{D}$ has a fixed and common outline (h, w) so that every block assigned to d can fit without incurring overlap.
- (ii) *Rectangular blocks*, denoted as set $\mathcal{B}$. Each block $b \in \mathcal{B}$ has dimensions (h_b, w_b) and fixed or flexible pins, denoted as set $\mathcal{P}^b$. Furthermore, each block is assigned a power density b_{pd}.
- (iii) *Netlist*, denoted as set $\mathcal{N}$. A net $n \in \mathcal{N}$ describes a connection between two or more pins.
- (iv) *Massive interconnect structures*, denoted as set $\mathcal{MI}$. Each interconnect structure $mi \in \mathcal{MI}$ contains a set $\{a_1, \ldots, a_m\}$ of block-alignment tuples.[2]

The objective of our methodology is similar to classical 3D floorplanning: we seek to obtain a floorplan that optimizes design criteria (e.g., thermal management) while, at the same time, complies with constraints (e.g., fixed outlines). In a holistic approach, we incorporate the task of planning massive interconnects into our floorplanner's modular concept.

[2]Our formulation thus relies on specific alignment and/or offset ranges. We acknowledge that massive interconnects might be derived from a more abstract value, e.g., the specified number of wires for a bus. However, such considerations are highly technology-dependent, e.g., from minimum wire pitch and wire widths, and thus we prefer that the designer in charge manually defines appropriate alignment and/or offset ranges for massive interconnects.

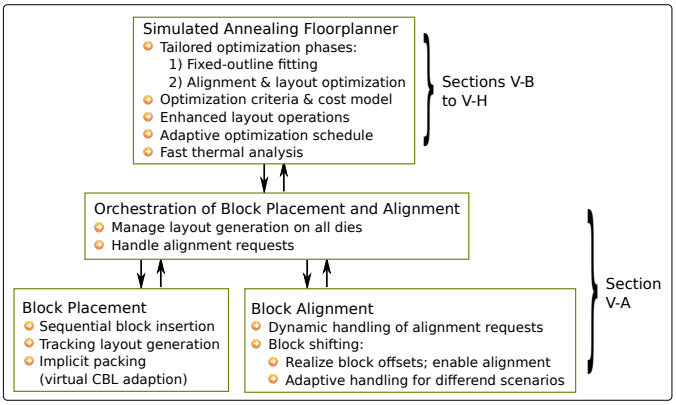


Fig. 5. Corblivar's core parts, embedded in an SA-based floorplanning tool. *Orchestration of Block Placement and Alignment* interacts with the SA heuristic for layout optimization, monitors the overall layout process, and delegates to *Block Placement* and *Block Alignment* in a synchronized manner.

V. 3D Floorplanning Methodology and Tool

For planning massive interconnects in 3D chips, we first propose the *corner block list for varied alignment requests (Corblivar)*, an extension of the classical CBL [13].[3] Corblivar represents a 3D IC containing n dies with an ordered sequence $\{CBL_1, \ldots, CBL_n\}$ of CBL tuples and the interconnects' set $\mathcal{MI}$. The key concepts of Corblivar's layout generation are discussed in Sec. V-A.

Like for any layout representation, we then embed Corblivar in a floorplanning methodology and tool; core parts and main features are outlined in Fig. 5. Our tool is a simulated annealing (SA)-based 3D floorplanner, implemented in C++ and made publicly available in [12]. Applying SA-based floorplanning during initial experiments, we observed the limitations of existing techniques w.r.t. solution-space exploration for block alignment as well as for "classical" 3D floorplanning. Thus, some effective upgrades were needed; notable features of our tool are (i) an SA framework that includes two global optimization stages and specific cost models (Secs. V-B and V-C); (ii) clustering of regular interconnects into massive interconnects (Sec. V-D); (iii) an adaptive SA optimization schedule (Sec. V-G); and (iv) fast yet sufficiently accurate thermal analysis (Sec. V-H).

A. Orchestrated Layout Generation

We extend the CBL technique [13] to simultaneously handle inter- and intra-die alignment, consider fixed offsets as well as alignment ranges and efficiently pack the layout. We implement this in an orchestrated fashion, such that a global process (i) generates the layout on each die; (ii) keeps track of all block-alignment requests; and (iii) delegates to separate processes for actual block placement and alignment. In the following, we describe key elements of these processes.

*1) **Orchestration of Block Placement and Alignment:*** The core concept behind CBL's sequential layout generation is maintained. As extension, a global orchestration process

[3]We leverage the CBL mainly for its efficiency (layout generation has a $\mathcal{O}(n)$ complexity) and expandability towards a 3D representation. For further details, e.g., the concept of T-junctions, please refer to [13].

Algorithm 1 Orchestration of Block Placement and Alignment. The *alignment stack* AS memorizes alignment requests in progress, *progress pointers* $p_i = b_j$ denote the currently processed block b_j for each die d_i, and a *die pointer* $p = d_i$ is used to keep track of the currently processed die.

```
1: p ← d_1                                          ▷ start without loss of generality on bottom die
2: p_i ← b_1
3: loop
4:     b_i ← p_i ← p
5:     if die d ← p is stalled then
6:         PLACE(b_i)
7:         mark b_i in any a ∈ mi ∈ MI as placed
8:         mark d as not stalled
9:         p_i ← p_{i+1}
10:    else                                                          ▷ d is not stalled
11:        if some a_k are defined for b_i then
12:            for all a_k do                          ▷ consider a_k w/ placed blocks first
13:                if a_k in AS then
14:                    ALIGN(a_k)
15:                    remove a_k from AS
16:                    mark b_i, b_j in any a ∈ mi ∈ MI as placed
17:                    mark d_j = die(b_j) as not stalled
18:                else
19:                    add a_k to AS
20:                    mark d as stalled
21:                    p ← die(b_j)
22:                end if
23:            end for                                          ▷ all a_k considered
24:            if d is not stalled then
25:                p_i ← p_{i+1}
26:            end if
27:        else                                          ▷ no a_k defined for b_i
28:            PLACE(b_i)
29:            p_i ← p_{i+1}
30:        end if
31:    end if                                                          ▷ b_i processed
32:    if p_i = end then
33:        if some p_j ≠ end then
34:            p ← p_j
35:        else
36:            return done
37:        end if
38:    end if
39: end loop
```

monitors the layout generation on each die d and any block alignment in progress. Details of this process are given in Algorithm 1; the main steps are also explained next.

Whenever a block b_i on d has to be aligned, we first check whether the *layout prerequisite* for this particular alignment is fulfilled. This means, we check whether the blocks preceding b_j are already placed on die d'—the die containing block b_j, which is being aligned with b_i. Only then, i.e., when b_j is the "next block to be placed" on die d', can we safely align b_i and b_j. Details of Algorithm 1 are discussed next.

Initially, we check whether the associated die d is currently marked as *stalled* (line 5), i.e., layout generation is halted due to another alignment request in progress—this occurs for *intersecting requests*, i.e., related blocks are arranged in the CBL sequences so that their placement is interfering. To resolve this, we need to unlock die d—we PLACE the current block b_i, mark related changes, and proceed with the next block (lines 6–9). Otherwise (for non-stalled dies), we check if some alignment requests a_k are applying to b_i (line 11). If no a_k are found, we directly PLACE b_i and proceed with the next block (lines 28–29). If some request(s) a_k are defined, we need to handle them appropriately (lines 12–23), as described next. For any given a_k, we search the stack AS for it (line 13) and continue accordingly. *Case a:* if a_k is found, it was previously handled while processing b_j, that is the block to be aligned with b_i. Thus, it is assured that preceding blocks

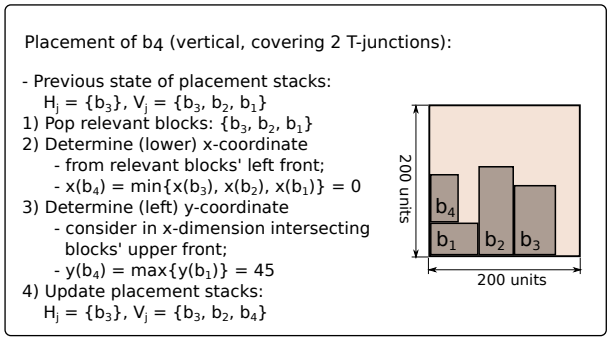


Fig. 6. Placement steps, to be performed for exemplary vertical insertion of block b_4 while covering 2 T-junctions.

on both related dies are placed at this point. We can now safely ALIGN both b_i and b_j, mark them as placed, and drop the request a_k (lines 14–17). Note that only in cases where *all* requests for b_i are handled, we proceed on the current die d (line 25). Otherwise, we continue layout generation without loss of generality (w.l.o.g.) on b_j's die d' (line 21). *Case b:* if a_k is not found in AS, b_j was not processed yet. We then memorize a_k as in progress, halt layout generation on d, and continue on d' (lines 19–21). Finally, if layout generation is done on d, we proceed on yet unfinished dies until the whole 3D layout is generated (lines 32–38).

Note that *deadlock situations*, i.e., layout generation on different dies waiting for each other to align particular blocks, *cannot* occur due to the process of *resolving paused dies*. With this technique, we simply place die d's current block b_i whenever we switch to a *paused* die d. However, this also implies that alignment of b_i must be deferred and implicitly resolved when its associated block b_j is processed.

*2) **Block Placement:*** To maintain a valid layout during placement, it is necessary to consider previously placed blocks. We implement a technique that allows us to efficiently keep track of relevant blocks and to perform implicit layout compaction (the latter is explained below). Besides this, our technique follows the principles of CBL's placement approach.

Placement is based on two stacks H_i / V_i for each die d_i. These stacks contain d_i's blocks currently covering the vertical right / horizontal upper front; these blocks define the *boundary fronts* for placement. Then during layout generation, we only need to check the respective boundary front to determine the coordinates for each block to be placed (Fig. 6).

For any block smaller than the "CBL room" it is supposed to cover, the next adjacent block will be packed into this room (Fig. 7). We refer to this technique as *virtual CBL adaptation* since it results in practice in different CBL encodings for the same compact layout. Hence, it supports efficient solution-space exploration towards compact layouts that produce—on average—shorter routing paths.

*3) **Block Alignment:*** Recall that our methodology supports different types of massive interconnects. We observe that all corresponding block-alignment requests depend on the blocks' *planar offsets*, i.e., the relative distances considering their projections onto a plane. This implies that we can rely on *block shifting* to handle alignment and thus massive interconnects.

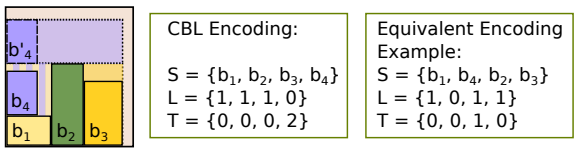


Fig. 7. Virtual CBL adaptation. Rooms and their assigned blocks are similarly colored. Block b_4 is "packed into the room" of b_1, providing a more compact layout. Without virtual CBL adaptation, b_4 would be placed as b'_4.

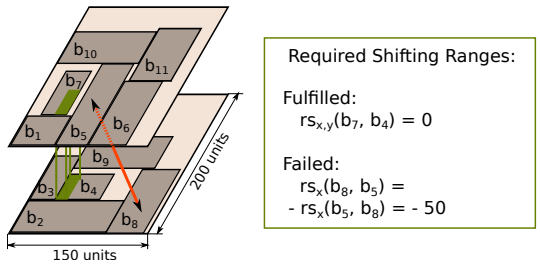


Fig. 8. Required shifting ranges. During alignment of block b_7 with previously placed b_4, shifting b_7 to the right was applicable so that $rs(b_7, b_4)$ was resolved. In contrast, shifting previously placed b_5 and the "inverse shift" of b_8 to the left are not applicable.

Our layout-generation process is tailored for block shifting: it is synchronized across the whole chip, i.e., blocks to be aligned "wait for each other's die to be ready", namely until preceding blocks are placed. However, depending on whether blocks to be aligned have been previously placed, shifting may not be applicable for all configurations. In particular, we omit "intrusive" block shifting, that is shifting when both blocks to be aligned have already been placed in order to resolve stalled dies. Based on experimental observations, such intrusive shifting requires adjacent (or likely even other nearby) blocks to be shifted in order to maintain a valid layout. This is impractical in the presence of different massive interconnects—shifting can undermine alignment for remaining interconnects, or even invalidate previously optimized interconnect structures. It is important to stress that such situations with both blocks previously placed only arise for particular CBL configurations, mainly due to an inappropriate placement order. This means that adapting the configurations, e.g., via dedicated layout operations (Sec. V-F), will resolve such issues.

If one block has been previously placed, we derive the *required shifting range* $rs_x(b_i, b_j)$ or $rs_y(b_i, b_j)$, i.e., the remaining offset of b_i and b_j to be compensated in order to fulfill the related block alignment (Fig. 8). In cases where $rs(b_i, b_j) < 0$, we would need to shift b_i left/down which is trivially prohibited for a valid (i.e., overlap-free) layout. Alternatively, we could shift previously placed b_j right/up; however, this is not practical in the presence of many blocks to be aligned, as mentioned above. Finally, if and only if $rs(b_i, b_j) \geq 0$, can we safely perform a forward shift of b_i in the x/y-direction—thus satisfying the block alignment criteria.

For scenarios where both blocks are not placed yet, we can trivially shift blocks so that the related massive interconnects are properly facilitated.

B. Optimization Metrics and Cost Models

Applied optimization metrics along with their cost models/functions are discussed next. Note that cost functions are formulated for SA's classical cost-minimization approach, i.e., lower cost is equivalent to more optimized layouts.

1) ***Outline:*** This metric unifies evaluation of the layout's bounding boxes (i.e., packing density) as well as fixed-outline fitting. This is achieved by extending Chen and Chang's aspect-ratio-based model [25]. Our model is defined as

$$c_{outline} = c_{PD} + c_{AR}$$
$$c_{PD} = \frac{1}{2}\alpha\left(1 + \frac{n_{feasible}}{n}\right) \times \max_{d_i}\left(\frac{A(b \in d_i)}{A(d_i)}\right)$$
$$c_{AR} = \frac{1}{2}\alpha\left(1 - \frac{n_{feasible}}{n}\right) \times \max_{d_i}\left(\Delta_{AR}(d_i)^2\right)$$

where c_{PD} and c_{AR} denote the respective cost terms for *packing density* and *aspect-ratio violation*, $A(b \in d_i)$ and $A(d_i)$ determine the blocks' and die's area, and $\alpha \leq 1$ denotes the cost weight. The aspect-ratio violation considers $\Delta_{AR}(d_i) = AR(b \in d_i) - AR(d_i)$ where AR encodes the aspect ratio for blocks or their assigned die, respectively.

The above model accounts for the SA process; calculations refer to previous n layout operations where $n_{feasible} \leq n$ operations resulted in a valid layout (i.e., blocks on all dies are fitting into the fixed outline). For best fitting solutions, we determine cost with fixed $n_{feasible} = n$, i.e., we locally ignore cost history for meaningful comparison of best solutions.

2) ***Wirelength and TSV Count:*** We assume that lowermost die d_1 is connected to the package board; nets connecting to terminal pins are thus routed through the whole stack down to d_1. For each net n, we determine the half-perimeter wirelength (HPWL) on each die d_i separately, denoted as $HPWL(n, d_i)$. To do so, we construct the bounding box by encircling the net's relevant pins: connected pins of blocks placed on die d_i and the connected terminal pins in case of $d_i = d_1$.

In order to model wires connecting blocks with TSVs, additional pins have to be considered. Depending on whether TSVs have been placed (for massive interconnects or clustered TSV islands, Sec. V-D) or not yet (i.e., for early estimations and for guidance during clustering TSV islands), these pins differ: for placed and connected TSVs, the TSVs' pins in d_i are to be considered; for yet unplaced TSVs, the pins of *blocks* to be connected in the next upper die d_j are to be considered. For the latter, assuming that TSVs can be placed into the resulting bounding boxes, this model provides the most accurate HPWL-based estimate prior to actual TSV placement [2]. Cost terms are defined as

$$c_{WL} = \sum_n \left(l_{TSV} \times TSVs(n) + \sum_{d_i \in n} HPWL(n, d_i)\right)$$
$$c_{TSVs} = \sum_n TSVs(n)$$

where $TSVs(n)$ denotes the required TSV count for net n. Note that we also account for "TSV wirelength" l_{TSV} in c_{WL}.

3) ***Routability:*** Cost is derived from the average and maximum of an *evenly distributed routing utilization* RU as

$$c_R = \text{avg}_{\forall n;\forall d}(RU(n, d_i)) \times \max_{\forall n;\forall d}(RU(n, d_i))$$
$$RU(n, d_i) = \omega_n \times ww_n \times \frac{x + y}{x \times y};\ x, y \in bb(n, d_i)$$

where $bb(n, d_i)$ refers to the outline of n's bounding box on d_i, ω_n to n's weight and ww_n to n's wire width. This model is an efficient and effective approach for estimating routability; it proved surprisingly accurate for practical benchmarks (i.e., the ISPD 2008 global-routing contest suite) and superior to other well-known estimations (e.g., Westra's net model) [26].

Assuming $ww_n = 1$, we model an "idealized" technology-independent utilization without consideration of minimum pitches. Then, $RU(d_i) = 1$ means that one metal layer of d_i is fully occupied.

This model is pessimistic / upper-bound for regions comprising vertical/TSV buses: net bounding boxes are considered to span across buses, thus nets are assumed to be routed over whole TSV-bus regions. However, assignments of nets to TSVs can be optimized later (e.g., during actual routing), most likely limiting routing utilization.

4) ***Thermal Management:*** The cost $c_T = \max_{\forall d}(T(d_{tc}))$ represents the estimated maximum temperature of the *thermally-critical* die d_{tc}.

Details on thermal-analysis-related material and chip properties are given in Sec. VI-A3; details of our fast thermal-analysis approach are explained in Sec. V-H.

5) ***Block Alignment:*** We denote the weighted *spatial alignment mismatches* between desired alignment and actual layout as cost term

$$c_{AMM} = \sum_{a_k} (|rs_x(b_i, b_j)| + |rs_y(b_i, b_j)|) \times wires$$

Recall that $rs(b_i, b_j)$ was introduced in Sec. V-A3 and Fig. 8, and $wires$ denotes the signal count of alignment a_k.

C. Optimization Stages

We consider two different stages for SA optimization; these stages support efficient solution-space exploration and layout optimization for 3D floorplanning with block alignment.

1) ***Stage I, "Fixed-Outline Fitting":*** The cost function is defined as

$$c_I = c_{outline}$$

There is no block alignment applied in this stage. The reason for initially focusing SA's search solely on the fixed-outline is simply that non-fitting layouts are a "knock-out", regardless of any block alignment and layout optimization that might be achieved. The transition to Stage II is made when the SA search triggers the first fixed-outline-fitting layout.

2) ***Stage II, "Alignment and Layout Optimization":*** We compose the cost function as

$$c_{II} = c_I + (1 - \alpha) \times c'_{II}$$
$$c'_{II} = \beta \frac{c_{WL}}{c_{WL_{in}}} + \gamma \frac{c_{TSVs}}{c_{TSVs_{in}}} + \delta \frac{c_R}{c_{R_{in}}} + \epsilon \frac{c_T}{c_{T_{in}}} + \zeta \frac{c_{AMM}}{c_{AMM_{in}}}$$

with cost weights $\beta + \gamma + \delta + \epsilon + \zeta \leq 1$.

Note that we memorize *inial cost terms* like $c_{WL_{in}}$ during transition to Stage II, i.e., we derive them from the first valid solution. Furthermore, we consider $c_I = c_{outline}$ as an essential term in this stage as well; based on our experiments, the multiobjective SA search still depends notably on outline fitting/optimization.

D. Clustering of Regular Interconnects

Regular signal TSVs, not explicitly set up as massive interconnect structures, are clustered into TSV islands during Stage II. Next, we motivate and outline our *thermal- and wirelength-driven clustering* technique.

As mentioned in Sec. V-B4 (and elaborated later on in Sec. V-H), there exists a thermally-critical die d_{tc}, exhibiting highest temperatures of the 3D-IC stack. We consider only d_{tc}'s thermal distribution during clustering, which has two benefits. First, the heat dissipation for hotspots on d_{tc} is increased, notably reducing overall maximum temperatures. Second, we mitigate adverse thermal coupling between dies which potentially introduces new hotspots in other dies when dissipating heat away from d_{tc}. The latter benefit is achieved by clustering into *nearly aligned stacks of TSV islands* which can serve as effective heatpipes. Previous studies have shown the benefits for thermal management of both stacking *single* TSVs [24] and TSV islands [17], [18].

For thermal- and wirelength-driven clustering, we leverage Knechtel's *net-cluster concept* [2] and Lindeberg's algorithm for *grey-level blob detection* [27]. We deploy the latter to determine *hotspot regions*; by considering these larger regions instead of simply relying on points of local temperature maxima, our algorithm mitigates thermal hotspots more effectively.

Initially, the bounding boxes $bb(n, d_i)$ for each net n are constructed separately on each related die d_i, as described in Sec. V-B2. For all dies d_i, the respective set of $bb(n, d_i)$ is then sorted by the boxes' covered area in descending order.

Next, the thermal distribution of d_{t_c} is determined (Sec. V-H). Based on the resulting 2D thermal map, grey-level blob detection is applied as described in [27, Sec. 9.1]. As indicated, hotspot regions are described by *blobs*—local temperature maxima along with their surrounding region (Fig. 12). The determined hotspot regions hs are sorted in descending order according to their *score*

$$s(hs) = (T_{max}(hs) - T_{min}(hs)) \times T_{max}(hs)^2 \times bins(hs)$$

where $bins(hs)$ denotes the (discretized) area covered by hs. Note that high-scored regions exhibit large thermal gradients, very high (due to second-power term) maximum temperatures and large covered areas. These regions have a strong impact on the thermal distribution and are thus considered first.

Based on net bounding boxes and hotspot regions, both sorted by their respective relevance, *net clusters* are determined stepwise on each die d_i. The idea is to gradually merge bounding boxes into net clusters with consideration of hotspot regions. More precisely, merging bounding box $bb(n_j, d_i)$ into net cluster c_k means redefining the cluster region as the intersection of c_k and $bb(n_j, d_i)$. Merging executes as long as (i) the intersection is trivially non-empty, and (ii) c_k still covers a hotspot region after the merging operation. In other words, whenever a merging step would relocate c_k away from a hotspot region, the respective box is ignored for cluster c_k. This step is repeated until all nets are merged into clusters.

Finally, TSV islands are placed into the clusters' center. These islands then yield massive interconnects which are optimized for both thermal and wirelength management.

E. TSV Placement

For massive vertical buses or clustered signal TSVs, the resulting TSV islands are sized and centred such that they fit into the bus site or cluster center. That is, islands are shaped according to their designated location and TSV dimensions and pitches. In case islands overlap, a greedy shifting is performed to legalize TSV placement.

For regular (non-clustered) TSVs, they are placed in the center of die-wise bounding boxes, as determined in Sec. V-B2. In case TSVs overlap, greedy shifting is applied as well.

Note that we seek to embed TSV islands into blocks—to limit local wiring between blocks and massive interconnects—which may induce additional cost. For example, hard blocks with fixed layouts cannot include TSVs. Such blocks will need to be either redesigned or encapsulated into macro blocks comprising TSVs and the actual block. In general, embedding TSVs requires sufficient unoccupied design area, i.e., deadspace. In our experiments (Sec. VI), deadspace utilization by TSVs was not excessive (well below 40%) for any setup; embedding TSVs into blocks is thus considered practical, without inducing notable additional cost.

F. Layout Operations

We deploy following operations to support the SA heuristic in its effective exploration of Corblivar's solution space:

- swapping blocks within or across dies / CBL sequences,
- swapping or moving whole CBL tuples within or across CBL sequences,
- switching a block's insertion direction,
- changing a block's T-junctions,
- rotating hard blocks, and
- guided shaping of soft blocks, as proposed in [25].

Operations and blocks / CBL tuples are randomly selected in Optimization Stage I. In Stage II, blocks related to failed alignment requests are preferably selected. These blocks are then swapped with adjacent blocks or moved across dies, depending on the particular alignment, so that $|rs(b_i, b_j)|$ is reduced. This way, the alignment is more likely to be successfully performed in the next iteration of layout generation.

G. Adaptive Optimization Schedule

As mentioned earlier, we need an adaptive optimization (SA cooling) schedule to improve the efficiency of solution-space exploration. Our schedule is capable of (i) effectively guiding the SA search within the optimization stages and (ii) escaping local minima. The schedule is composed of three different stages, explained below. Note that in the following, i labels the current step of i_{max} total temperature steps. An illustrative schedule example is given in Fig. 9.

1) ***"Adaptive Cooling" Stage:*** We deploy this cooling stage during Optimization Stage I. The cooling follows

$$T_{SA}(i+1) = \left(cf_1 + \frac{i-1}{i_{max}-1} \times (cf_2 - cf_1) \right) \times T_{SA}(i)$$

with cooling factors $cf_1 < cf_2 < 1$.

The cooling rate slows down; our intention here is to achieve initially fast cooling for the global scope, followed by slower cooling in a more confined, "local" solution space.

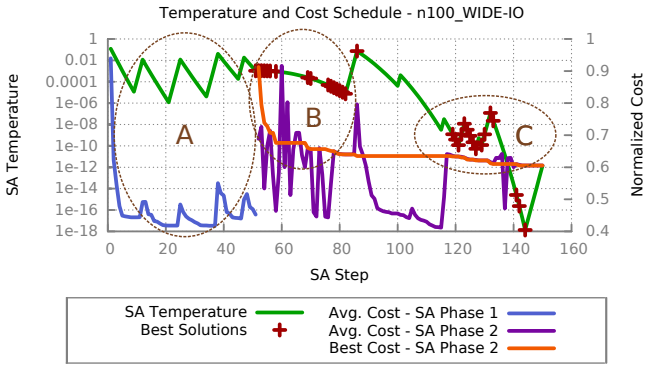


Fig. 9. Adaptive optimization schedule. "Brief Reheating" (e.g., A, C) enables to escape local minima while "Reheating and Freezing" (B) increases chances for triggering new best solutions with notably decreased cost.

2) ***"Reheating and Freezing" Stage:*** This type of cooling is invoked for Optimization Stage II, i.e., after a fitting layout has been found in the $i_{fitting}$ step. The function is defined as

$$T_{SA}(i+1) = \left(1 - \frac{i - i_{fitting}}{i_{max} - i_{fitting}} \right) \times cf_3 \times T_{SA}(i)$$

The cooling rate increases steadily, i.e., the temperature is reduced exponentially. However, setting factor $cf_3 > 1$ results in initial reheating. This way, the SA search increases chances of high-cost solutions being deployed in this "critical solution-space region" which covers the first fitting layout. According to our experiments, this allows to notably decrease cost during the subsequent search, i.e., facilitates high-quality solutions.

3) ***"Brief Reheating" Stage:*** This stage enables a robust cooling schedule. The function is defined as

$$T_{SA}(i+1) = cf_4 \times T_{SA}(i)$$

with cooling factor $cf_4 > 1$.

Brief reheating helps the SA search escape local minima. It is applied stepwise, in alternation with *individual* temperature steps during Stages I and II. Additionally, it is only applied when we observe $\sigma(c_{II}) \sim 0$ during previous k steps, that is when the search reached a local "cost plateau". This technique is inspired by Chen and Chang's study [25]; their approach, however, proposes reheating solely at one particular temperature step, which we believe is not as effective as our cost-controlled reheating.

H. Fast Thermal Analysis

For fast, yet accurate, steady-state temperature analysis, we extend Park et al.'s work on *power blurring* [14]. Instead of using computationally intensive finite differences or finite element analysis (FEA), power blurring is based on simple matrix convolution of thermal-impulse responses and power-density distributions. More precisely, to determine the thermal profile $T(d_i)$ on die d_i, multiple convolution results are superposed to model the effect of vertical heat transfer in the 3D IC (Fig. 10). The superposition is determined as

$$T(d_i) = \sum_{d_j} pdm(d_j, x, y) * tm(d_j, d_i, x, y)$$

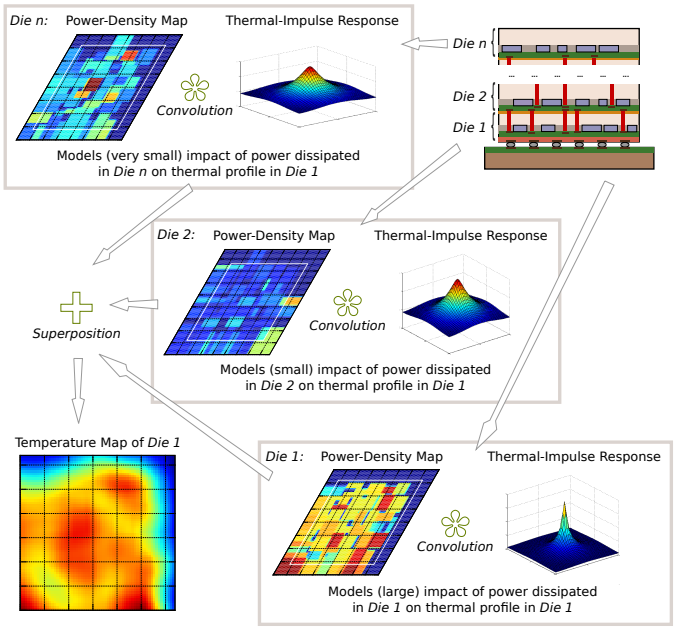


Fig. 10. Power-blurring-based thermal analysis (illustrated for *Die 1*). Given are a set of power-density maps and thermal-impulse responses. The latter describe how heat spreads from power sources placed in different dies; they model the overall heat conduction within a 3D IC. Each power-density map is to be convoluted with the related thermal-impulse response. The superposition of these convolution results provides the temperature map for a particular die.

Here, $pdm(d_j, x, y)$ denotes the gridded *2D power-density map* for die d_j's layout, with $0 \leq x \leq x_{max}$, $0 \leq y \leq y_{max}$, where $x_{max} = y_{max}$ represents the upper bins of the *uniformly gridded* die outlines. Furthermore, $tm(d_j, d_i, x, y)$ describes the discretized *thermal mask*—or thermal-impulse response—that models the impact of d_j's power dissipation on d_i's thermal profile.

During initial experiments, we found that d_1 (i.e., the die furthest away from the heatsink) generally experiences the highest temperatures; we refer to d_1 thus as the *thermally-critical die*. This is due to the fact that the bonding material, e.g., BCB adhesive polymer, has a significantly lower thermal conductance compared to the silicon in the chips—bonding layers are a *thermal barrier* in the 3D IC. It is important to note that this finding is based on heat conduction only towards the heatsink, not through bumps and thus ignoring the secondary heat path towards the package and PCB.[4] For scenarios with the secondary heat path considered, the thermally-critical die is more likely found in the middle of the 3D IC. Without loss of generality, we restrict $i = 1$ in the following.

1) ***Parameterization of Thermal Masks:*** For modelling the mask functions, we refrain from time-consuming FEA runs [14]. Instead, we model the masks using discretized *2D gauss functions*

$$\begin{aligned} tm(d_j, d_1, x, y) &= g(d_j, x, y) \\ &= w(d_j) \exp\left(-\frac{1}{s(d_j)} x^2\right) \exp\left(-\frac{1}{s(d_j)} y^2\right) \end{aligned}$$

[4]This is due to the lack of modelling secondary heat paths within the—for reference thermal simulations applied—3D extension of *HotSpot* [28].

where $w(d_j)$ is an amplitude-scaling factor and $s(d_j)$ a lateral-spreading factor specifically parametrized for each die d_j.

According to initial experiments, it is needless to spread the discretized gauss functions across the entire gridded power-density map. Instead, sweeping a relatively small 2D "mask window" ($x_m \times y_m = n \times n$, with $n \ll x_{max}, y_{max}$) stepwise across the map is practical [29]. This notably reduces computational efforts for the convolution calculations. However, for this method we need to parametrize the masks: their center points cp have to be aligned with the gauss functions' peaks and their boundary points need to feature a minimum value $g_{min} > 0$. To do this, we adapt the gauss functions—mainly $w(d_j)$ and $s(d_j)$—such that

$$\begin{aligned} g(d_j, x, y) &= g(d_j, x_m, y_m) \\ &= w(d_j) \exp\left(-\frac{m}{cp^2} x_m^2\right) \exp\left(-\frac{m}{cp^2} y_m^2\right) \end{aligned}$$

where

$$m = \frac{1}{2} \ln\left(\frac{w(d_j)}{g_{min}}\right); \quad w(d_j) = \frac{w}{j^{w_s}} = \frac{w(d_1)}{j^{w_s}}$$

The masks' center points are $cp = \lfloor n/2 \rfloor$. During convolution, the gauss-function parameters x_m, y_m sweep through the mask windows $-cp \leq x_m, y_m \leq cp$. Furthermore, $\max(j)$ represents the uppermost die and w_s is a scaling factor.

In the course of the parameterization of g_{min}, w, w_s, and pd_{PZ} and pd_{TSV} (the latter two will be introduced later), we determine an optimized parameter set for each different 3D-IC setup—different w.r.t. die count and dimensions. Therefore, we initially determine a *reference thermal distribution*. This is done by applying a 3D extension of *HotSpot* [28], a state-of-the-art academic thermal analyzer, on an exemplary 3D-IC layout. (Details on material properties for HotSpot simulations are given in Sec. VI-A.) For this reference thermal distribution, we then determine a best fit for the above parameters using a local search. Initial experiments revealed that the parameters have no consistent correlations. Thus, the local search treats each parameter independently, and derives candidate parameter values using normal distributions which are initially confined by a broad range. Whenever the local search finds a better parameter set, i.e., when the deviation of the power-blurring thermal distribution from the HotSpot distribution is reduced, the associated parameter values are considered as the new best set. Then, each parameter's normal distribution is adapted such that (i) the mean is set to the respective new best value, and (ii) the standard deviation is reduced by a user-defined factor. This loop is repeated several times, controlled by user parameters for accuracy and/or runtime.

2) ***Separated 1D Convolutions:*** The power-blurring approach can be sped up even further by separated 1D convolutions. Therefore, note that the 2D gauss function can be *separated* easily, i.e., it can be substituted by a 1D function's product $g_x \times g_y$. For separable functions defining a matrix, applying a 2D convolution is equivalent to applying two successive and orthogonal 1D convolutions [27, p. 43]. With the latter approach, however, we can reduce the number of operations needed from n^2 to $2n$ [27, p. 43]. For experiments

with $n = 9$, we observed an overall speedup of $\sim 4\times$ for such separated 1D convolutions.

*3) **Error Compensation – Padding Zones:*** We propose error compensation for thermal estimation near die boundaries, as declared necessary by Park et al. [14]. To this end, we introduce *power-density padding zones*. That is, we extend the power-density maps with a "ring" of $\lfloor n/2 \rfloor$ additional bins whose values are derived from blocks abutting the die boundaries, and weighted with a scaling factor pd_{PZ}. Note that this only affects the power-density maps, not the floorplan.

This approach is beneficial in two ways. First, the error compensation can be tuned flexibly via pd_{PZ} and is thus adaptable to different 3D-IC setups. Second, the convolution calculations themselves are freed from checking if bins are within well-defined matrix boundaries. These checks are computationally trivial, but required for innermost loops[5], and thus their omission reduces runtime notably; we observed an overall speedup of $\sim 3.5\times$ in our experiments.

*4) **Consideration of TSV Islands:*** In order to extend power blurring to account for TSV islands, we investigated three different approaches: (i) post-processing the thermal distribution; (ii) applying different thermal masks for design regions with and without TSV islands; and (iii) adaptation of power-density maps for design regions with TSV islands.

Options (i) and (ii) were ruled out after exploratory experimentation. A related key finding was that TSV islands have both a *local impact* on temperature gradients / values, and a *global impact* on the overall temperature ranges. While post-processing (option i) can easily account for the islands' global impact, the local impact has proven difficult to capture since it varies depending on island location and number, and on neighboring islands. Deploying different thermal masks (option ii) produced a more accurate estimation of the thermal distributions. However, this approach would considerably increase runtime: it requires "real" 2D convolution since different masks would introduce discontinuities in the result when the stepwise 1D convolutions are deployed.

Our approach (option iii) is predicated on scaling down the power density $pdm(d_j, x, y)$ for TSV regions. In other words, it simply models the improved vertical heat conduction in TSV regions with reduced power densities, subsequently resulting in (both globally and locally) lower temperatures. The scaling is defined as

$$pdm(d_j, x, y) \times = \left(1 + \frac{pd_{TSV} - 1}{100} \times d_{TSV}(d_j, x, y)\right)$$

where $0 \leq pd_{TSV} \leq 1$ is the scaling factor, and $0 \leq d_{TSV}(d_j, x, y) \leq 100$ is the TSV density on d_j at bin x, y. Note that pd_{TSV} is parameterized simultaneously with the other thermal-mask parameters (Sec. V-H1); the scaling factor pd_{TSV} allows us to model the local impact of TSV islands, while w and w_s now capture additionally the global impact of TSV islands.

[5]The 2D convolution of $pdm(d_j, x, y) * g(d_j, x_m, y_m)$ generally requires a fourfold nested loop [29], where the innermost loop is executed $x \times y \times x_m \times y_m$ times. In our case of separated 1D convolutions, the innermost loops are executed $(x \times x_m) + (y \times y_m)$ times. Within these loops, the convolution is stepwise performed, and the checks for well-defined samples, i.e., if data are within matrix bounds, are required for every step.

VI. Experimental Results

We conducted several experiments to validate Corblivar's capabilities; configurations and results are discussed next.

A. Configuration

*1) **Planning of Massive Interconnects:*** We consider two width- and length-restricted buses, each covering 5 blocks, along with two vertical buses, each connecting two blocks. We assume that each interconnect bundles 512 signals. Furthermore, we deploy a *Wide-IO interface* as outlined in [30], i.e., a vertical bus sized $600\mu m \times 160\mu m$ and connecting two dies through 960 TSVs. Note that Wide-IO interfaces are designed for memory integration; we apply it here mainly for its relevance to 3D integration.

As such a scenario has not been considered in previous studies, we cannot meaningfully compare it to other approaches.[6] However, we can evaluate whether these massive interconnects can be successfully considered within our methodology. To do so, we investigate two setups: (i) dedicated block alignment and (ii) classical optimization. For the former setup (i) we consider our proposed block alignment and for setup (ii) we deliberately ignore these measures. We consider the HPWL contribution of massive interconnects—weighted by the interconnects' signal count—for both setups, as an equitable comparison. Besides considering wirelength, we optimize for routing utilization, maximum temperatures, and also apply clustering of signal TSVs for both setups.

*2) **Regular and Large-Scale 3D Floorplanning:*** To evaluate Corblivar's efficiency w.r.t. key 3D floorplanning objectives, mainly for comparison with other relevant studies [31], [32], we look into layout packing, wirelength and thermal optimization in this setup. Note that we neglect massive interconnects and block alignment as well as clustering of signal TSVs here, since the other studies omitted such measures.

In addition, we demonstrate Corblivar's scalability by utilizing the *IBM-HB+* benchmark suite [33]. As far as we are aware, this is the first time that these large-scale circuits have been considered for 3D floorplanning.

*3) **3D-IC Configuration and Benchmarks:*** We assume face-to-back stacking of two or three dies; further technological properties are given in Table I. Terminal pins are available on the lowermost die d_1 which is assumed to be connected to the package board. Practical (i.e., stackable) fixed outlines up to $5 \times 5mm$ are considered. We apply *GSRC* [34] and *IBM-HB+* [33] circuits. To use die outlines in a more meaningful manner, we enlarged GSRC benchmarks by a factor of 5, and IBM-HB+ benchmarks by a factor of 2. In this context and for practical thermal management, power-density values of

[6]Previous studies on block alignment for 3D ICs considered different scenarios. Nain et al. [8] proposed to split up and align (sub-)modules among adjacent dies with fixed, zero offsets. However, as they didn't produce derived benchmarks containing split-up blocks, we cannot compare our approach with theirs. Law et al. [9] considered a more flexible problem with vertical bus planning: sets of blocks are defined separately for each die, and at least one block from each set is to be vertically aligned with one block from the other sets. This simplified problem is not compatible with our approach; we require all specified blocks to be aligned. Li et al. [10] refer to block-alignment capabilities but refrain from providing details and related experimental results. Finally, all studies consider only vertical alignment with fixed offsets.

TABLE I
CHIP AND MATERIAL PROPERTIES

Part (Material)	Height $[\mu m]$	Heat Capacity $\left[\frac{J}{K \times m^3} \times 10^6\right]$	Thermal Resistivity $\left[\frac{K \times m}{W}\right]$
Die (Si)	100	1.631	0.00851
Active Layer (Si)	2	—	—
BEOL Layer (Mainly Cu)	12	1.208	0.444
Bonding Layer (BCB)	20	2.299	5.0
TSV (Cu)	98	3.546	0.00253
TSV (Cu)	Diameter: $5\mu m$, Pitch: $10\mu m$		
Heat Spreader (Cu)	Dimensions: $3 \times 3mm$, Thickness: $100\mu m$		
Heat Sink (Cu)	Dimensions: $6 \times 6mm$, Thickness: $690\mu m$		

GSRC benchmarks are scaled down by a factor of 10, resulting in average power consumption of 2–3W. Note that layouts were packed where possible, enabling reduced die outlines. Deadspace utilization by TSVs was not excessive in any setup; we thus refrain from minimizing TSV counts.

*4) **General Setup:*** We conduct all experiments on an *Intel Core 2* system; reported runtimes are thus comparable. Corblivar and [32] are SA-based tools; best results are chosen from 5 to 25 runs. Applied Corblivar parameters are retrievable from [12]. Default settings [28] and material properties as detailed in Table I are deployed for *HotSpot*.

B. Results

*1) **Planning of Massive Interconnects:*** We observe that all massive interconnects can be successfully planned for both setups (dedicated block alignment and classical optimization, contrasted in Table II). However, the average success rate for planning all interconnects as well as respective TSV counts are higher for dedicated block alignment. Furthermore, final wirelength as well as maximum routing utilization are notably smaller for dedicated block alignment than for classical optimization. Recall that the estimated maximum routing utilization is pessimistic since nets are assumed to span across whole TSV-island sites (Sec. V-B3). Still, estimated utilizations larger than, e.g., 15, as experienced for classical optimization, are prohibitive: this translates to designs with some regions exceeding routing resources of 15 metal layers.

In short, our proposed block alignment is effective and greatly supports managing routability and wirelength during planning of massive interconnects.

Comparing die outlines and deadspace for both setups, we expect, and observe, a slight increase for dedicated block alignment—such measures limit the flexibility for layout packing and thus the increase in deadspace. Nevertheless, fixed die outlines were delivered in all cases. i.e., the proposed SA optimization stages are effective. An example of successfully planned interconnects and their corresponding block alignment is illustrated in Fig. 11.

*2) **Fast Thermal Analysis and Clustering of Signal TSVs:*** An initial observation is that our power-blurring approach shows some local deviations compared to *HotSpot*-verification runs. As indicated in [14], convolution-based thermal analysis particularly induces estimation errors at die boundaries; we reduced these errors after initial experiments (not illustrated) by introducing the power-density padding zones (Sec. V-H3).

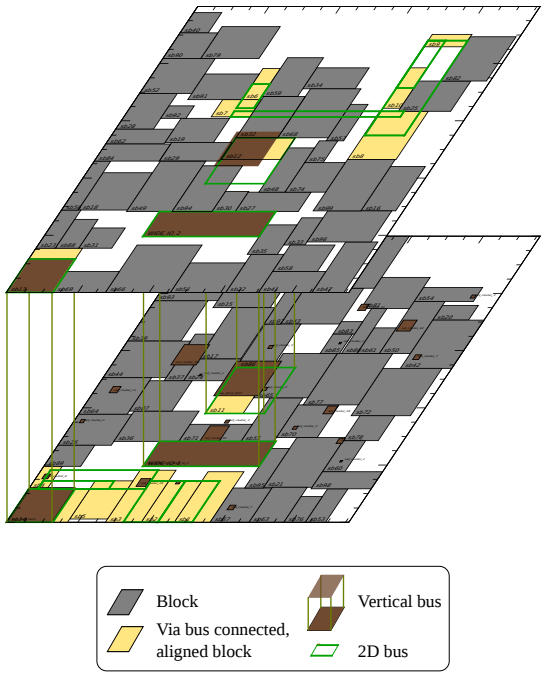
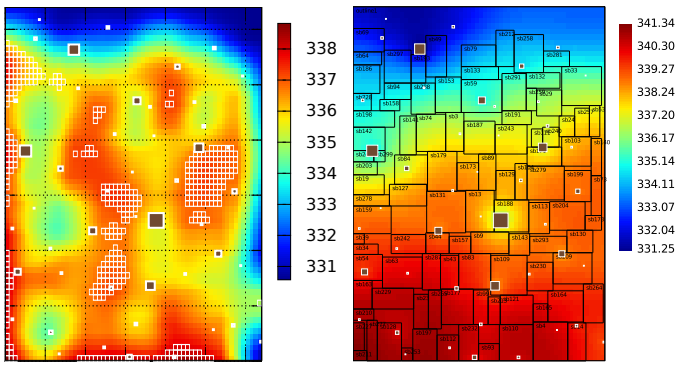


Fig. 11. Massive interconnect structures with corresponding block alignment, benchmark *n100*. The wide vertical bus contains the Wide-IO interface, the other two are regular embedded buses. For clarity, TSV landing pads in the upper die are only illustrated for vertical buses, not for signal-TSV clusters.



(a) (b)

Fig. 12. Thermal maps in Kelvin of critical die d_1 for benchmark *n300*; 3 dies in total, here without dedicated vertical buses. TSV islands are marked as white-framed, brown rectangles in (a) and (b). Hotspot regions (Sec. V-D) are marked as white-framed rectangles in (a). The temperature scale of our power-blurring analysis (a) largely matches the scale obtained by *HotSpot* (b); yet, local deviations are visible, especially for regions with TSV islands.

When TSV islands are considered, related errors increased again, as is clearly visible in Fig. 12. This is due mainly to inhomogeneous material properties (Cu TSVs and Si dies) and associated variations in heat conduction, especially near die boundaries. However, our approach is practical for the evaluation and localization of maximum-temperature regions—which we consider sufficient for thermal-aware layout optimization.

Refer to Tables II and III for comparison of estimated and by *HotSpot* verified maximum temperatures. In most cases, the error is below 1%, but generally increasing with the die count and for scenarios with few relatively large TSV islands (Table III). This is expected, since the thermal impact of (i) more dies stacked and (ii) few but large TSV

TABLE II
RESULTS FOR PLANNING OF MASSIVE INTERCONNECTS – UPPER PART REPRESENTS DEDICATED BLOCK ALIGNMENT, LOWER PART REPRESENTS CLASSICAL OPTIMIZATION

Metric	2 Dies				3 Dies			
	n100	*n200*	*n300*	Avg	*n100*	*n200*	*n300*	Avg
Wirelength $(\mu m \times 10^6)$	8.08	8.76	9.92	8.92	8.77	9.40	11.06	9.74
Die Outlines $(\mu m^2 \times 10^6)$	3.08	3.04	4.72	3.61	2.28	2.32	3.37	2.66
Total Deadspace (%)	26.29	26.88	27.11	26.76	33.77	36.20	31.89	33.95
Total TSVs (Count)	1992	2994	3156	2714	4954	6292	6578	5941
TSV Islands (Count)	32	42	49	41	62	74	82	73
Avg TSVs per Island (Count)	62	57	64	61	79	85	80	81
TSV-Deadspace Utilization (%)	12.31	18.04	12.40	14.25	21.42	24.97	20.42	22.27
Estimated Max Temp (K)	301.24	301.28	303.76	302.09	305.52	304.55	310.03	306.70
Max Temp [28] (K)	301.70	299.66	301.25	300.87	304.65	303.70	306.82	305.06
Max Routing Utilization	8.38	10.71	10.98	10.02	8.93	9.27	8.21	8.80
Runtime (s)	162	392	384	313	151	714	857	574
Wirelength $(\mu m \times 10^6)$	9.88	11.17	14.68	11.91	11.91	10.56	14.73	12.40
Die Outlines $(\mu m^2 \times 10^6)$	2.98	2.90	5.05	3.64	2.03	2.05	3.40	2.49
Total Deadspace (%)	24.01	23.34	31.89	26.41	25.55	27.78	32.49	28.61
Total TSVs (Count)	1992	2448	3167	2536	4954	6265	5621	5613
TSV Islands (Count)	29	41	46	39	56	71	85	71
Avg TSVs per Island (Count)	68	59	68	65	88	88	66	81
TSV-Deadspace Utilization (%)	13.90	18.11	9.84	13.95	31.83	35.67	16.98	28.16
Estimated Max Temp (K)	300.69	300.91	302.36	301.32	305.63	303.51	307.84	305.66
Max Temp [28] (K)	301.50	300.96	302.56	301.67	307.32	305.67	312.29	308.43
Max Routing Utilization	13.91	18.27	13.41	15.20	17.09	19.79	14.08	16.99
Runtime (s)	167	359	896	474	164	606	939	570

islands (which are not aligned, in contrast to clustered signal-TSV islands) is more complex and thus more difficult to model. Our thermal analysis is also very efficient due to fast computation—it can be conducted in ∼20ms—and is thus invoked in every optimization iteration. Compare this with *HotSpot* where one run can take on the order of tens of seconds up to a few minutes. *HotSpot* is thus not applicable during optimization iterations but only for final verification.

For clustering of signal TSVs (i.e., comparing results with clustering in Table II to results without clustering in Table III) we provide the following findings. First, wirelength is slightly increased for applied clustering. This can be expected [2]: some nets may necessitate minor detours to connect to TSV islands. Second, die outlines and deadspace ratios are comparable, i.e., clustering signal TSVs has little impact on chip area. Third, without clustering applied, routing utilization decreases for 2-die setups but increases for 3-die setups. This finding suggest that clustering is beneficial when stacking more dies; the floorplanner can then more effectively optimize routing utilization by (implicitly) rearranging few TSV islands rather than many single TSVs. Fourth, maximum temperatures are reduced when clustering is applied, especially for 3-die ICs. This indicates that our technique is helpful for "breaking the thermal barrier" posed by the bonding layers between stacked dies. Recall that the latter is achieved by stacking TSV islands during clustering.

TABLE III
RESULTS FOR PLANNING MASSIVE INTERCONNECTS BUT WITHOUT CLUSTERING OF SIGNAL TSVS

Metric	2 Dies			
	n100	*n200*	*n300*	Avg
Wirelength $(\mu m \times 10^6)$	8.13	7.16	11.02	8.77
Die Outlines $(\mu m^2 \times 10^6)$	3.19	3.03	4.69	3.64
Total Deadspace (%)	28.85	26.78	26.71	27.45
Total TSVs (Count)	2952	3389	3620	3320
TSV Islands (Count)	4	4	4	4
Avg TSVs per Island	624	624	624	624
Deadspace Util (%)	16.06	14.95	12.40	14.47
Est Max Temp (K)	305.82	307.01	308.61	307.15
Max Temp [28] (K)	301.66	303.84	309.18	304.89
Max Routing Util	7.74	9.60	7.97	8.44
Runtime (s)	81	228	337	215
Metric	3 Dies			
	n100	*n200*	*n300*	Avg
Wirelength $(\mu m \times 10^6)$	7.99	8.43	9.42	8.61
Die Outlines $(\mu m^2 \times 10^6)$	2.36	2.09	3.62	2.69
Total Deadspace (%)	35.88	28.97	36.76	33.87
Total TSVs (Count)	5915	7232	6593	6580
TSV Islands (Count)	9	10	8	9
Avg TSVs per Island	562	557	568	562
Deadspace Util (%)	19.53	34.64	14.09	22.75
Est Max Temp (K)	311.91	332.03	321.31	321.75
Max Temp [28] (K)	310.83	310.52	319.12	313.49
Max Routing Util	9.79	11.78	7.29	9.62
Runtime (s)	60	137	408	202

*3) **Regular 3D Floorplanning, Comparative Results:*** Next, we discuss comparative results for floorplanning with (equal) consideration of thermal and wirelength optimization (Table IV). Note that massive interconnects and dedicated block alignment as well as thermal- and wirelength-driven TSV clustering are not considered in this setup, since the other studies in comparison do not feature these functions. Also, benchmarks have not been enlarged in these experiments to ensure that the comparison is impartial.

We observe that Corblivar is superior to both a force-directed tool [31] and an SA-based tool [32]. In particular, we achieve notably reduced wirelength, smaller outlines along with lower deadspace ratios, and slightly decreased maximum temperatures when comparing to [31]. This indicates the

TABLE IV
COMPARATIVE RESULTS ON GSRC BENCHMARKS (NOT ENLARGED FOR UNBIASED COMPARISON)

Metric	Corblivar, 2 Dies				Corblivar, 2 Dies			Corblivar, 3 Dies				Corblivar, 3 Dies		
	n100	*n200*	*n300*	Avg	*ami33*	*xerox*	Avg	*n100*	*n200*	*n300*	Avg	*ami33*	*xerox*	Avg
Wirelength $(\mu m \times 10^5)$	3.31	6.04	8.35	5.90	1.29	12.46	6.88	3.56	6.36	8.35	6.09	1.34	12.26	6.8
Die Outlines $(\mu m^2 \times 10^5)$	1.06	1.04	1.67	1.26	7.74	132.44	70.09	0.93	0.75	1.13	0.94	5.60	96.45	51.03
Total Deadspace (%)	15.59	15.75	18.19	16.51	25.31	26.95	26.13	35.44	21.49	19.25	25.39	31.22	33.13	32.18
Max Temp [28] (K)	314.09	313.39	312.19	313.22	314.30	354.04	334.17	330.47	355.74	360.78	349.00	355.19	418.39	386.79
Runtime (s)	87	206	294	196	18	9	14	121	278	524	308	24	11	18
Metric	[31], 2 Dies				[32], 2 Dies			[31], 3 Dies				[32], 3 Dies		
	n100	*n200*	*n300*	Avg	*ami33*	*xerox*	Avg	*n100*	*n200*	*n300*	Avg	*ami33*	*xerox*	Avg
Wirelength $(\mu m \times 10^5)$	3.65	6.18	9.53	6.45	1.81	17.14	9.48	4.59	7.17	10.61	7.46	2.22	21.86	12.04
Die Outlines $(\mu m^2 \times 10^5)$	1.19	1.21	2.15	1.52	9.65	125.17	67.41	0.97	0.82	1.48	1.09	7.54	88.93	48.24
Total Deadspace (%)	25.02	27.87	36.69	29.86	40.09	22.70	31.40	38.89	28.64	38.65	35.39	48.87	27.47	38.17
Max Temp [28] (K)	313.31	313.74	314.63	313.89	336.36	366.48	351.42	348.55	360.60	361.35	356.83	384.39	482.16	433.28
Runtime (s)	439	446	526	470	193	47	120	266	497	574	446	193	48	121

TABLE V
RESULTS ON IBM-HB+ BENCHMARKS

Metric	2 Dies			3 Dies		
	ibm01	*ibm03*	*ibm07*	*ibm01*	*ibm03*	*ibm07*
Wirelength $(\mu m \times 10^6)$	15.51	39.14	74.00	13.53	36.95	68.64
Die Outlines $(\mu m^2 \times 10^6)$	11.16	23.71	29.92	7.12	17.16	19.38
Total Deadspace (%)	24.27	18.18	20.96	23.77	24.69	18.64
Runtime (s)	2219	4708	4937	2540	3792	5043

general effectiveness of our methodology. Compared to [32], however, we note that Corblivar's layouts are slightly larger. Nonetheless, we achieve notably reduced wirelength and lower maximum temperatures. Note that, in the interests of objectivity, we even neglect TSVs / TSV islands and related reductions in maximum temperatures caused by increased heat conduction. Thus, our methodology effectively addresses the trade-off between packing density and maximum temperature. Fixed outlines were obeyed in these experiments as well.

*4) **Large-Scale 3D Floorplanning:*** The IBM-HB+ suite does not include power information; we thus configured Corblivar only for wirelength optimization, layout packing and the (successful) consideration of fixed-outline constraints.

Results on arbitrarily selected circuits are provided in Table V. We observe that total wirelength (including TSVs' wirelength) for three-die chips is smaller than for two-die chips, which clearly indicates the benefit of 3D integration for these large-scale benchmarks. Recall that, in contrast, for small-scale GSRC benchmarks considered in previous setups, this was rarely the case. Our finding—only reasonably large circuits will generally benefit from 3D integration—is in accordance with other studies, e.g., [20].

VII. SUMMARY

In this work, we have extended 3D floorplanning towards effective planning of massive interconnects—an important, yet inadequately addressed, scenario for (future) 3D ICs.

To tackle deficits in previous studies, we promote block alignment. In the first place, we discussed how 3D (inter-die) and 2D (intra-die) alignment can be simultaneously applied for planning diverse interconnects like vertical buses. We then introduced *Corblivar*, a 3D layout representation with a novel alignment concept. In this regard, we developed effective techniques for layout generation, block alignment and layout evaluation, as well. We note that it is vital to synchronize alignment across the whole 3D IC. In particular, when aligning buses vertically, our algorithms need to consider each related die's layout in progress.

We have embedded Corblivar into an open-source, SA-based floorplanning tool. We have also developed necessary extensions including an adaptive SA schedule, clustering regular signal TSVs into vertical buses and convolution-based fast thermal analysis. For the latter, we proposed dedicated techniques (e.g., for efficient parameterization of thermal masks) which notably improve runtime and applicability.

Experimental results on GSRC and large-scale IBM-HB+ benchmarks demonstrate Corblivar's applicability for planning massive interconnects, as well as its competitive performance for "classical" 3D floorplanning, while considering fixed outlines, layout packing and thermal and wirelength optimization. A key observation is that classical floorplanning [35]—seeking to minimize the by signal count weighted wirelength—is also applicable for planning massive interconnects, but provides inferior results. Our methodology, in contrast, facilitates routability, reduction of wirelength as well as thermal management for massively-interconnected 3D chips.

REFERENCES

[1] F. Miller, T. Wild, and A. Herkersdorf, "Virtualized and fault-tolerant inter-layer-links for 3D-ICs," *Microprocessors and Microsystems*, vol. 37, no. 8, Part A, pp. 823–835, 2013.
[2] J. Knechtel, I. L. Markov, and J. Lienig, "Assembling 2-D blocks into 3-D chips," *Trans. Comput.-Aided Des. Integr. Circuits Sys.*, vol. 31, no. 2, pp. 228–241, 2012.
[3] M. Jung *et al.*, "How to reduce power in 3D IC designs: A case study with OpenSPARC T2 core," in *Proc. Cust. Integr. Circ. Conf.*, 2013.
[4] JEDEC Solid State Technology Association, "JEDEC Standard: JESD229 Wide I/O," http://www.jedec.org/standards-documents/results/jesd229, Dec 2011.
[5] T. Zhang *et al.*, "3D-SWIFT: A high-performance 3D-stacked Wide IO DRAM," in *Proc. Great Lakes Symp. VLSI*, 2014, pp. 51–56.
[6] H. Xiang, X. Tang, and M. D. F. Wong, "Bus-driven floorplanning," in *Proc. Int. Conf. Comput.-Aided Des.*, 2003, pp. 66–73.
[7] J. H. Y. Law and E. F. Y. Young, "Multi-bend bus driven floorplanning," *Integration*, vol. 41, no. 2, pp. 306–316, 2008.
[8] R. K. Nain and M. Chrzanowska-Jeske, "Fast placement-aware 3-D floorplanning using vertical constraints on sequence pairs," *Trans. VLSI Syst.*, vol. 19, no. 9, pp. 1667–1680, 2011.

[9] J. H. Law, E. F. Y. Young, and R. L. S. Ching, "Block alignment in 3D floorplan using layered TCG," in *Proc. Great Lakes Symp. VLSI*, 2006, pp. 376–380.
[10] X. Li, Y. Ma, and X. Hong, "A novel thermal optimization flow using incremental floorplanning for 3D ICs," in *Proc. Asia South Pacific Des. Autom. Conf.*, 2009, pp. 347–352.
[11] S. K. Lim, Personal communication, March 2013.
[12] J. Knechtel. (2015) Corblivar floorplanning suite. [Online]. Available: http://www.ifte.de/english/research/3d-design/index.html
[13] X. Hong *et al.*, "Corner block list: an effective and efficient topological representation of non-slicing floorplan," in *Proc. Int. Conf. Comput.-Aided Des.*, 2000, pp. 8–12.
[14] J.-H. Park, A. Shakouri, and S.-M. Kang, "Fast thermal analysis of vertically integrated circuits (3-D ICs) using power blurring method," in *Proc. ASME InterPACK*, 2009, pp. 701–707.
[15] H. Jao *et al.*, "The impact of through silicon via proximity on CMOS device," in *Proc. Microsys. Packag. Assemb. Circ. Technol. Conf.*, 2012, pp. 43–45.
[16] K. H. Lu *et al.*, "Thermo-mechanical reliability of 3-D ICs containing through silicon vias," in *Proc. Elec. Compon. Technol. Conf.*, 2009, pp. 630–634.
[17] Y. Chen *et al.*, "Through silicon via aware design planning for thermally efficient 3-D integrated circuits," *Trans. Comput.-Aided Des. Integr. Circuits Sys.*, vol. 32, no. 9, pp. 1335–1346, 2013.
[18] P. Budhathoki, A. Henschel, and I. A. M. Elfadel, "Thermal-driven 3D floorplanning using localized TSV placement," in *Proc. Int. Conf. IC Des. Techn.*, 2014, pp. 1–4.
[19] W. Yao *et al.*, "Modeling and application of multi-port TSV networks in 3-D IC," *Trans. Comput.-Aided Des. Integr. Circuits Sys.*, vol. 32, no. 4, pp. 487–496, 2013.
[20] D. H. Kim, R. O. Topaloglu, and S. K. Lim, "Block-level 3D IC design with through-silicon-via planning," in *Proc. Asia South Pacific Des. Autom. Conf.*, 2012, pp. 335–340.
[21] X. Zhao and S. K. Lim, "TSV array utilization in low-power 3D clock network design," in *Proc. Int. Symp. Low Power Elec. Design*, 2012, pp. 21–26.
[22] J. Knechtel *et al.*, "Multiobjective optimization of deadspace, a critical resource for 3D-IC integration," in *Proc. Int. Conf. Comput.-Aided Des.*, 2012, pp. 705–712.
[23] H.-T. Chen *et al.*, "A new architecture for power network in 3D IC," in *Proc. Des. Autom. Test Europe*, 2011, pp. 1–6.
[24] P. Hsu, H. Chen, and T. Hwang, "Stacking signal TSV for thermal dissipation in global routing for 3-D IC," *Trans. Comput.-Aided Des. Integr. Circuits Sys.*, vol. 33, no. 7, pp. 1031–1042, 2014.
[25] T.-C. Chen and Y.-W. Chang, "Modern floorplanning based on B*-Tree and fast simulated annealing," *Trans. Comput.-Aided Des. Integr. Circuits Sys.*, vol. 25, no. 4, pp. 637–650, 2006.
[26] T. Meister, J. Lienig, and G. Thomke, "Interface optimization for improved routability in chip-package-board co-design," in *Proc. Int. Workshop Sys.-Level Interconn. Pred.*, 2011, pp. 1–8.
[27] T. Lindeberg, *Scale-Space Theory in Computer Vision*. Kluwer Academic Publishers, 1994.
[28] A. Coskun, K. Kawakami, and D. Rossell. (2012) Hotspot 3D extension. [Online]. Available: http://lava.cs.virginia.edu/HotSpot/links.htm
[29] S. H. Ahn. (2005) Convolution. [Online]. Available: http://www.songho.ca/dsp/convolution/convolution.html
[30] M. Jung, D. Z. Pan, and S. K. Lim, "Chip/package mechanical stress impact on 3-D IC reliability and mobility variations," *Trans. Comput.-Aided Des. Integr. Circuits Sys.*, vol. 32, no. 11, pp. 1694–1707, 2013.
[31] P. Zhou *et al.*, "3D-STAF: scalable temperature and leakage aware floorplanning for three-dimensional integrated circuits," in *Proc. Int. Conf. Comput.-Aided Des.*, 2007, pp. 590–597.
[32] Y. Chen. (2010) 3DFP – thermal-aware floorplanner for three-dimensional ICs. For a related publication, refer to [36]. [Online]. Available: http://www.cse.psu.edu/~yxc236/3dfp/index.html
[33] A. N. Ng *et al.* (2006) IBM-HB+ benchmarks. [Online]. Available: http://vlsicad.eecs.umich.edu/BK/ISPD06bench/
[34] W. Dai, L. Wu, and S. Zhang. (2000) GSRC benchmarks. [Online]. Available: http://vlsicad.eecs.umich.edu/BK/GSRCbench/.
[35] A. B. Kahng *et al.*, *VLSI Physical Design: From Graph Partitioning to Timing Closure*. Springer, 2011.
[36] W.-L. Hung *et al.*, "Interconnect and thermal-aware floorplanning for 3D microprocessors," in *Proc. Int. Symp. Quality Elec. Des.*, 2006, pp. 98–104.

Johann Knechtel (M'11) received the M.Sc. in Information Systems Engineering (Dipl.-Ing.) in 2010 and the Ph.D. in Computer Engineering (Dr.-Ing.) in 2014, both from Dresden University of Technology, Germany.

Dr. Knechtel is currently a Postdoctoral Researcher at the Institute Center for Microsystems, Masdar Institute of Science and Technology, Abu Dhabi, UAE. From 2010 to 2014, he was a Research Associate and Scholar with the DFG Research Training Group Nano- and Biotechnologies for Packaging of Electronic Systems, and a Ph.D. Student at the Institute of Electromechanical and Electronic Design, Dresden University of Technology. In 2012, he was a Research Assistant with the Dept. of Computer Science and Engineering, Chinese University of Hong Kong, China. In 2010, he was a Visiting Research Student with the Dept. of Electrical Engineering and Computer Science, University of Michigan, USA.

His research interests cover VLSI Physical Design Automation with emphasis on 3D Integration.

Evangeline F. Y. Young (M'13) received her B.Sc. degree in Computer Science from The Chinese University of Hong Kong (CUHK). She received her Ph.D. degree from The University of Texas at Austin in 1999.

She is currently a professor in the Department of Computer Science and Engineering in CUHK. Her research interests include algorithms and VLSI CAD. She is now working actively on floorplanning, placement, routing, DFM and EDA on Physical Design in general.

Prof. Young has served on the organization committees of ISPD, ARC and FPT and on the program committees of conferences including DAC, ICCAD, ISPD, ASP-DAC, SLIP, DATE and GLSVLSI. She was the Area Chair in the TPC of ASP-DAC and DAC this year. She is also serving on the editorial boards of IEEE TCAD, ACM TODAES and Integration, the VLSI Journal. Besides, her research group has won several championships and prizes in many recent renown EDA contests.

Jens Lienig (M'97–SM'10) received the M.Sc. (diploma), Ph.D. (Dr.-Ing.) and Habilitation degrees in Electrical Engineering from Dresden University of Technology, Dresden, Germany, in 1988, 1991 and 1996, respectively.

He is currently a Full Professor of electrical engineering at Dresden University of Technology (TU Dresden) where he is also Director of the Institute of Electromechanical and Electronic Design (IFTE). From 1999 to 2002, he worked as Tool Manager at Robert Bosch GmbH in Reutlingen, Germany, and from 1996 to 1999, he was with Tanner Research Inc. in Pasadena, CA. From 1994 to 1996, he was a Visiting Assistant Professor with the Department of Computer Science, University of Virginia, Charlottesville, VA, and from 1991 to 1994, a Postdoctoral Fellow at Concordia University in Montréal, QC, Canada. His current research interests are in physical design automation, with a special emphasis on electromigration avoidance, 3D design, and constraint-driven design methodologies of analog circuits.

Prof. Lienig has served on the Technical Program Committees of the DATE, SLIP and ISPD conferences.