Assembling 2-D Blocks into 3-D Chips

Johann Knechtel, *Student Member, IEEE,* Igor L. Markov, *Senior Member, IEEE,* and Jens Lienig, *Senior Member, IEEE*

***Abstract*—Despite numerous advantages of 3-D integrated circuits (ICs), their commercial success remains limited. In part, this is due to the wide availability of trustworthy intellectual property (IP) blocks developed for 2-D ICs and proven through repeated use. Block-based design reuse is imperative for heterogeneous 3-D ICs where memory, logic, analog, and microelectromechanical systems dies are manufactured at different technology nodes and circuit modules cannot be partitioned among several dies. In this paper, we show how to integrate 2-D IP blocks into 3-D chips without altering their layout. Experiments indicate that the overhead of proposed integration is small, which can help accelerate industry adoption of 3-D integration.**

***Index Terms*—3-D IC design styles, 3-D integrated circuits (ICs), floorplan optimization, intellectual property (IP) blocks, through-silicon via (TSV) planning, TSV islands.**

I. Introduction

3-D INTEGRATION is a promising design option to keep pace with steadily increasing demands on functionality and performance of electronic circuits. It is motivated by applications combining heterogeneous manufacturing processes, separate die testing for increased yield, shorter and lower-power interconnects, as well as a smaller form factor. Originating with vertical stacked dies in a system-in-package (SiP), wire bonding is used to interconnect separate dies, as illustrated by the Apple A4 package that places two DRAM dies on a ARM logic die. However, wire-bonding interconnect can become a bottleneck in such an SiP, and the next logical step is to provision for direct die-to-die interconnect without package-level detours, resulting in 3-D integrated circuits (ICs) (Fig. 1). Such interconnects are implemented using through-silicon vias (TSVs)—vertical plugs that connect two silicon dies. The use of TSVs enables *chip-level integration*, which promises shorter global interconnect while retaining the benefits of *package-level integration*. Recently, S. Borkar [1] presented an energy-efficient, high-performance 80-core system with stacked SRAM.

Manuscript received June 13, 2011; revised August 14, 2011; accepted October 21, 2011. Date of current version January 20, 2012. The work of J. Knechtel was supported by the German Research Foundation, under Project 1401/1. This paper was recommended by Associate Editor C.-K. Koh.

J. Knechtel and J. Lienig are with the Institute of Electromechanical and Electronic Design, Dresden University of Technology, Dresden 01062, Germany (e-mail: knechtel@ieee.org; jens@ieee.org).

I. L. Markov is with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109 USA (e-mail: imarkov@eecs.umich.edu).

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

Digital Object Identifier 10.1109/TCAD.2011.2174640

The 2009 edition of the International Technology Roadmap for Semiconductors prominently features 3-D IC integration in the section on interconnect and the section on assembly and packaging [2]. Industry analysts are forecasting that the global 3-D IC market will reach $5.2B by 2015 [3]. However, the progress in commercial applications of 3-D ICs is currently limited, despite these forecasts and apparent benefits. Academic research in this field is pursuing 3-D floorplanning [4]–[9], thermal management [5], [6], [8]–[11], TSV planning [12]–[17], and cost-effective design-space exploration [18]. Recent studies also address TSV-induced-stress analysis and accurate TSV alignment [19]–[21], the impact of intradie variation [22], and signal-integrity analysis for 3-D ICs [23].

Existing publications often neglect important obstacles to 3-D IC integration. One is given by design constraints and overhead associated with TSVs. At the 45 nm technology node, the area footprint of a $10\mu\text{m}\times10\mu\text{m}$ TSV is comparable to that of about 50 gates [24]. Furthermore, manufacturability demands landing pads and keep-out zones [21] which further increase TSV area footprint. Previous work in physical design often neglects this area overhead [5], [7]–[9]. Some studies explicitly consider thermal TSV insertion but not signal TSVs [11], [25]. Tsai *et al.* [14] observed that previous work also neglects the impact of TSV locations on wirelength estimates for floorplanning.

While the usage of TSVs is generally expected to reduce wirelength, Kim *et al.* [24] reported that wirelength reduction varies depending on the number of TSVs and their characteristics. Their case studies show that this tradeoff is controlled by the *granularity* of interdie partitioning. The wirelength typically decreases for moderate (blocks with 20–100 modules) and coarse (block-level partitioning) granularities, but increases for fine (gate-level partitioning) granularities.

Depending on the technology choices, TSVs block some subset of layout resources. Via-first TSVs are manufactured before metallization, thus occupy the device layer and result in placement obstacles. Via-last TSVs are manufactured after metallization and pass through the chip. Thus, they occupy both the device and metal layers, resulting in placement and routing obstacles [24].

A further impediment to 3-D IC integration is more subtle. It is related to feasible design styles and principles for 3-D integration. To achieve higher overall yield and reduce costs, separate testing of independent dies is essential [26], as also emphasized in a recent Intel study [1]. However, tight integration between adjacent active layers in 3-D ICs entails a significant amount of interconnect between different sections

0278-0070/$26.00 © 2011 IEEE

of the same circuit module that were partitioned to different dies. Aside from the massive overhead introduced by required TSVs, sections of such a module, e.g., a multiplier, demand for new testing approaches [26], [27]. Additionally, recent work [22] points out that intradie variation becomes a *first-order effect* for 3-D IC integration, while being only a *second-order effect* for 2-D chips.[1] The authors estimated that a 3-D layout will yield more poorly than the same circuit laid out in 2-D, contrary to the original promise of 3-D IC integration.

These wide-ranging considerations suggest that a successful and effective approach to 3-D IC integration must rely on proven and effective design methodologies. To this end, we make the following contributions.

1) We describe and compare several design styles, in particular the legacy 2-D (L2D) style which integrates existing 2-D intellectual property (IP) blocks not designed for 3-D IC integration (Section II).
2) Next, we extend the L2D style to L2D style with TSV islands (L2Di), where TSVs are clustered into TSV islands rather than placed in a spread-out manner (Section III). This technique can limit the overhead of TSVs, but does not necessarily exclude single TSVs.
3) To support the L2Di style, we propose a methodology and novel algorithms for net clustering, TSV-island insertion, and related tasks (Section V). The overall approach promises faster industry acceptance of 3-D IC integration.
4) We empirically validate our methodology, demonstrating 3-D IC integration of L2D IP blocks (Section VI).

The remainder of this paper is structured as follows. In Section II, we describe and compare several 3-D design styles. In Section III, we discuss options to connect blocks placed on separate dies. Additionally, we evaluate common wirelength-estimation techniques. The problem formulation for our L2Di style is given in Section IV. Our methodology is presented in Section V. In addition to proposing new techniques for 3-D integration, we also point out related results from graph theory. In Section VI, we present experimental results, validating our methodology. Our conclusions are given in Section VII.

II. 3-D IC Design Styles

3-D integration originated with package-level integration, which connects multiple 2-D chips through bonding pads, as illustrated by the quad-core variant of the Intel Core 2 processor. Finer granularity of 3-D integration is enabled by connecting dies with TSVs, which results in 3-D ICs [28]. In this paper, we consider signal TSVs. Simultaneously planning signal, power/ground and thermal TSVs requires further considerations [17]. Also, we consider face-to-back (F2B) stacking for 3-D IC integration. Next, we contrast gate-level and block-level integration styles for 3-D ICs (Fig. 2). Gate-level integration faces multiple challenges and currently appears less practical than block-level integration.

[1]When a die experiences process variation, all transistors become faster (or slower), perhaps at a different rate; the variations in transistor performance are therefore a second-order effect. However, several stacked dies may experience systematic variations in opposite directions—a first-order effect.

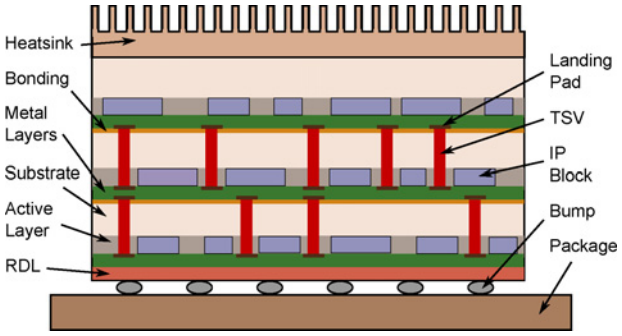

Fig. 1. 3-D IC containing three active layers, stacked using F2B technology. TSVs must not obstruct IP blocks and are therefore placed between them. Layer bonding can be realized with microbumps or direct bonding. Please note that routing interlayer nets through TSVs still requires vias in the metal layers. Also, connecting signals through bumps to the package requires a redistribution layer.

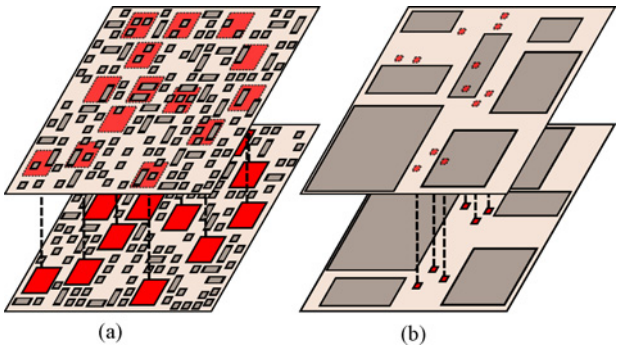

Fig. 2. Integration levels for 3-D ICs. TSVs are illustrated as solid, red boxes, and related landing pads as dashed, red boxes. (a) Gate-level integration, enlarged for illustration. This style is based on placing separate gates on multiple dies, likely resulting in a huge number of required TSVs. (b) Block-level integration relies on (2-D) blocks, which are partitioned between multiple dies and connected through global routes, thus limiting required TSVs.

A. Gate-Level Integration

One approach to 3-D integration is to partition standard cells between multiple dies in a 3-D assembly and use TSVs in routes that connect cells spread among active layers. This integration style promises significant wirelength reduction and great flexibility [6].

Its adverse effects include the massive number of necessary TSVs for random logic (Section I). The study by Kim *et al.* [24] revealed that partitioning gates between multiple dies may undermine wirelength reduction unless circuit modules of certain minimal size are preserved. A recent study [29] pointed out that layout effects can largely influence performance for highly regular blocks such as SRAM registers; a mismatch between TSV and cell dimensions may introduce wirelength disparities while routing regular structures to TSVs. Also, partitioning a design block across multiple dies requires new prebond testing approaches [26], [27].

Furthermore, gate-level 3-D integration requires to redesign all available IP, since existing IP blocks and electronic design automation (EDA) tools do not provision for 3-D integration. Even when 3-D place-and-route tools appear on the market, it will take many years for IP vendors to upgrade their extensive IP portfolios for 3-D integration.

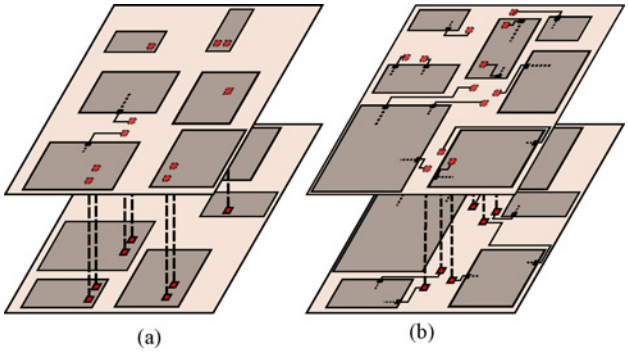

Fig. 3. Block-level integration for 3-D ICs. (a) Redesigned 2-D (R2D) style uses predefined TSV sites (small red boxes) within the block footprint. (b) L2D style places scattered TSVs preferably between blocks, thus limits stress for gates.

B. Block-Level Integration

Design blocks subsume most of the netlist connectivity and are linked by a smaller number of global interconnects [30]. Therefore, block-level integration promises to reduce TSV overhead. As pointed out in [29], TSVs may not scale at the same rate as transistors, thus the TSV-to-cell mismatch will likely remain for future nodes and may increase. The best option to limit the overhead of TSVs is to reduce their count by assigning only global interconnects to them.

Sophisticated 3-D systems combining heterogeneous dies are projected in a recent Cadence whitepaper [31]. Such systems require distinct manufacturing processes at different technology nodes for fast and low-power random logic, several memory types, analog, and radio frequency circuits, on-chip sensors, microelectromechanical systems (MEMS), and nanoelectromechanical systems. Thus, block-level integration appears crucial for future 3-D integration.

From an industrial perspective, 3-D integration will require 3-D-aware tools only for partitioning and thermal analysis [32]. Separate dies will be designed using (adapted) 2-D tools and 2-D blocks [32]. This is motivated by the broad availability of reliable IP blocks, as discussed later on. Furthermore, modern chip design often requires last-minute engineering changes. Restricting the impact of such changes to single dies is essential to limit additional cost.

When assigning entire blocks to separate dies and connecting them with TSVs, we distinguish two design styles (Fig. 3).

1) *R2D style*: 2-D blocks designed for 3-D integration (TSVs included within the footprints).
2) *L2D style*: 2-D blocks not designed for 3-D integration (TSVs preferably placed between blocks).

Depending on available blocks, mixing both styles may be practical. These styles promise a good tradeoff between necessary TSV usage and wirelength reduction, as discussed in Section I. However, the R2D style may be more constrained. For back-to-back (B2B) stacking, blocks may be required to align according to their predefined TSV locations, which would naturally increase placement complexity. This may further complicate design closure, e.g., due to congestion around densely packed obstacles. TSV-induced silicon stress increases the overhead of TSVs *inside* blocks, favoring TSVs *between* blocks.

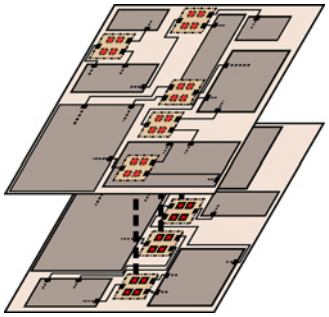

Fig. 4. L2Di style for 3-D integration. TSVs are grouped into TSV islands, which may include scan chains for test purposes and multiplex spare TSVs for redundancy. TSV island are illustrated as brown, dashed boxes containing TSVs (solid, red boxes). Related landing pads are illustrated as dashed, red boxes. Islands provide a beneficial option to connect blocks between several layers, but may also be difficult to insert into deadspace.

Several other important benefits of the R2D and L2D styles are described next. Design-for-testability (DFT) structures are a key component of existing (2-D) IP blocks and can therefore be used to realize prebond and postbond testing for the L2D style [26]. In general, test pins can be provisioned on each die and multiplexed/shared with other pins for prebond and postbond testing [33]. Block-level integration may be used to efficiently reduce critical paths, thus simultaneously allows limiting signal delay, increasing performance and reducing power consumption [34]. In [35], the authors propose optimal matching of *slow* and *fast* dies, based on accurate delay models with process variations considered. This approach assumes that dies can be delay-tested before 3-D stacking—a strong argument for block-level 3-D integration.

Another aspect of block-level integration styles deals with design effort (Section II-A). Modern chip design mostly relies on predesigned and optimized IP blocks; analysts at Gartner Dataquest point out that the IP market is still growing and will reach $2.3B by 2014. Existing IP blocks must be redesigned for use with the R2D style, despite their successful track record in applications and at the marketplace. Such a redesign would require new EDA tools for physical design and verification, increasing risks of design failures and being late to market. It is more convenient to use available L2D IP blocks and to place the mandatory TSVs in the deadspace between the blocks, as provisioned by the L2D style. An extreme form of design IP reuse possible with the L2D style is block-level mask reuse with changes only required for global routes at high metal layers—TSVs placed in deadspace do not modify silicon layers of the blocks.

III. Connecting 2-D Blocks in 3-D ICs

To connect 2-D blocks placed among multiple active layers, required TSVs can be inserted in several ways. First, one could use single, spread out TSVs [Fig. 3(b)]. The second option is to place TSVs on a gridded structure. The third option groups several TSVs into TSV islands, as required for the L2Di style. Depending on the manufacturing process, TSV insertion might be required to account for minimum TSV density, pitch, and spacing.

Fig. 4 illustrates TSV islands as blocks with densely placed TSVs. Optimizing the layout of TSV islands leads to several

benefits, as explained below. However, single TSVs can also be instantiated when necessary.

A study by Kim *et al.* [16] compared placing TSVs on a grid (regular placement) to placing scattered TSVs (irregular placement). The study revealed that irregular placement performs better in terms of wirelength reduction and design runtime. Since TSVs are placed near the blocks they are connected to, there is no need for a separate TSV-assignment process.

However, viewing TSVs as purely geometric objects would neglect several key technology issues. These include: 1) silicon stress in the neighborhood of TSVs, which alters transistor properties and motivates keep-out zones; 2) reliability and fault-tolerance issues in the TSVs themselves; and 3) complexities of connecting dies manufactured at different technology nodes, e.g., analog, digital logic, and memory dies. Regular TSV structures can be designed to address these concerns by optimizing spacing between TSVs, possibly sharing keep-out zones, performing electrothermal and mechanical simulations before layout, etc. In contrast, single TSVs would require greater care during layout. To this end, regular placement helps manufacturing reliable TSVs [13], [36], which favors assembling multiple TSVs into TSV islands.

A. TSV Islands

Grouping several TSVs into TSV islands of appropriate sizes is beneficial for several reasons, discussed below.

1) TSVs introduce stress in surrounding silicon which affects nearby transistors [21], but TSV islands do not need to include active gates. The layout of these islands can be optimized in advance; regular island structures help to limit stress below the yielding strength of copper [19]. Furthermore, using TSV islands limits stress to particular design regions [37]. Placing islands *between* blocks may thus limit stress on blocks' active gates.
2) TSV islands facilitate redundancy architectures [13], [20], where failed TSVs are shifted within a chain structure or dynamically rerouted to spare TSVs. Fig. 4 illustrates islands of four TSVs, including a spare.
3) Grouping TSVs can reduce area overhead. TSVs can be packed densely within TSV islands, possibly reducing keep-out zones without increasing stress-induced impact on active gates [37].
4) Regular, lithography-optimized layouts of predesigned TSV islands improve manufacturability by increasing exposure quality during optical lithography [13].
5) Each TSV experiences significant mechanical pressure (several hundred MPa [19]) which may affect even tungsten vias[2] over time, especially due to slight misalignments. The thinner the TSVs, the greater the pressure, and single TSVs are riskier than TSV islands that may balance out misalignments. For similar reasons, architectural pillars and columns usually appear in groups.

[2]Both copper and tungsten are used for TSV manufacturing. Currently, copper is more popular, but requires thicker TSVs due to its inferior mechanical properties (yield strength 600 MPa [19]).

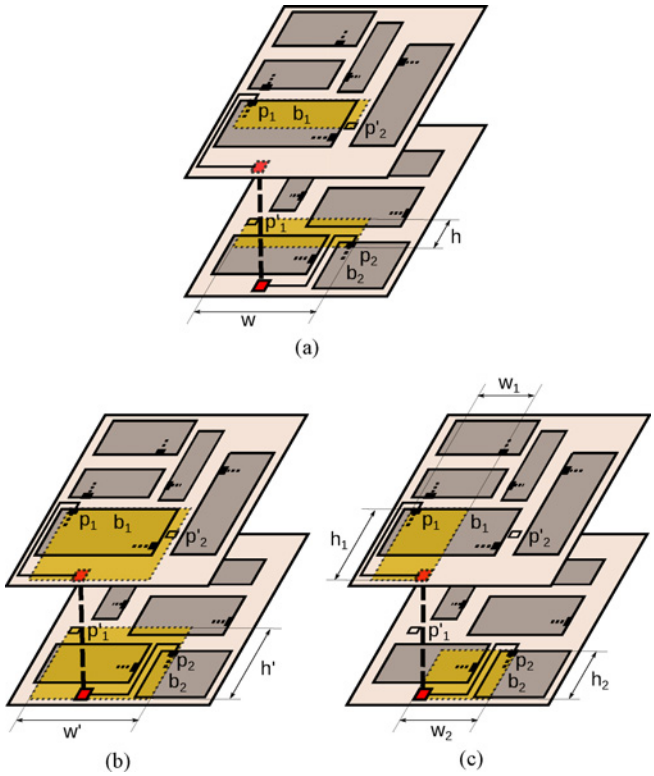

Fig. 5. Wirelength estimates for 3-D ICs based on bounding-box construction. Net pins are labeled p_n, projected pins as p'_n. (a) Considering only net pins provides lowest-accuracy estimation. Wirelength is calculated as NBB-3D-HPWL = $w + h$. (b) Using both net pins and the TSV location increases estimation accuracy. Wirelength is calculated as BB-3D-HPWL = $w' + h'$. (c) Most accurate estimation is achieved by separately considering net pins and TSVs on each die. Wirelength is calculated as BB-2D3D-HPWL = $\sum (w_l + h_l)$ for all related, active layers $l \in \mathcal{L}$.

6) Many designs suitable for 3-D integration, such as networks-on-chip, connect their modules by multibit buses. When such buses cross between adjacent dies, they will naturally form TSV islands.

Using TSV islands has some downsides. Connecting blocks through TSV islands can introduce wire detours, increase interconnect delays (also by coupling between TSVs), and increase the demand for routing resources. Large TSV islands may complicate floorplanning and placement. To address these challenges, we develop sophisticated algorithms for net assignment and TSV-island insertion.

The relevance of TSV islands may depend on technology details, which currently vary significantly among different manufacturers. We allow *trivial* TSV islands with only one TSV as well. This subsumes the straightforward handling of TSVs as a special case, thus our proposal is not restrictive.

B. Wirelength Estimation

Tsai *et al.* [14] showed that using TSVs not only affects the final wirelength but may also decrease the accuracy of wirelength estimation during floorplanning, if not appropriately addressed. As mentioned in Section I, previous work on 3-D integration mostly ignores TSV footprints and locations.

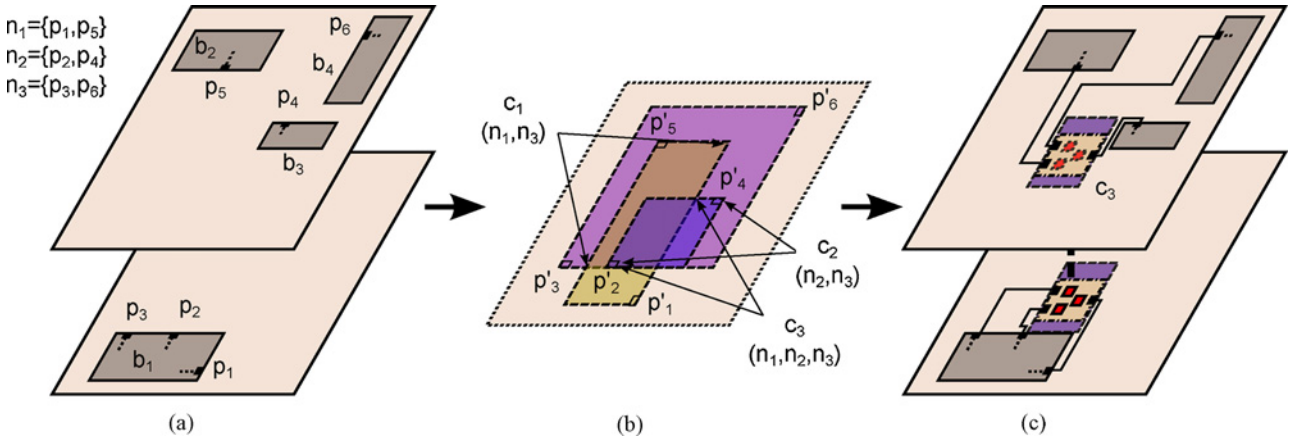

Fig. 6. Net clustering and TSV-island insertion. (a) Interlayer nets n_1, n_2, and n_3 need to be connected through TSVs. (b) Pins p_n are mapped to a virtual die as p'_n and corresponding net bounding boxes are constructed. Intersections of bounding boxes mark cluster regions c_1, c_2, and c_3 (intersection corners are pointed to). (c) Region c_3 is not obstructed by blocks and provides sufficient area, thus allows TSV-island insertion providing shortest routes for all nets.

Subsequently, TSV placement is likely to be hindered due to inappropriately distributed deadspace. The well-known metric half-perimeter wirelength (HPWL) of net bounding boxes is insufficient for 3-D ICs, due to possibly lacking deadspace for TSV insertion. However, this estimation technique using net pins provides a reference value for optimal TSV insertion. We refer to it as NBB-3D-HPWL [Fig. 5(a)]. Tsai *et al.* [14] proposed to extend it by considering TSV locations during bounding-box construction. We refer to this estimation technique as BB-3D-HPWL [Fig. 5(b)]. However, Tsai *et al.* neglected that using TSVs implies connecting blocks to TSV locations on all associated layers. To estimate resulting wirelength more precisely, Kim *et al.* [16] introduced net splitting. They construct bounding boxes on each layer separately and sum up resulting HPWLs. We refer to this technique as BB-2D3D-HPWL [Fig. 5(c)].

These estimation techniques assume only one TSV placed in each layer while connecting a net. However, for high-degree nets, e.g., those carrying enable signals, using multiple TSVs may be helpful for reducing wirelength and power consumption. Recently, Zhao *et al.* [38] proposed clock-tree-generation algorithms allowing multiple TSVs in each layer. Such techniques must simultaneously consider TSV capacitance and resistance, desired power savings, and wirelength tradeoff. Previously discussed wirelength-estimation techniques are unsuitable for such complex scenarios.

As mentioned in Section III-A, using TSV islands may require interconnect detours. To estimate them, one could compare BB-2D3D-HPWL to NBB-3D-HWPL wirelength. However, more reasonable is to compare BB-2D3D-HPWL wirelength for using TSV islands versus using single TSVs.

IV. Problem Formulation

For 3-D integration considering our L2Di style, the following input is assumed.

1) *Active layers*, denoted as set $\mathcal{L}$. Each layer $l \in \mathcal{L}$ has dimensions (h_l, w_l) such that every block assigned to l can fit in the outline without incurring overlap.
2) *Rectangular IP blocks*, denoted as set $\mathcal{B}$. Each block $b \in \mathcal{B}$ has dimensions (h_b, w_b) and pins, denoted as set $\mathcal{P}^b$. Each pin $p \in \mathcal{P}^b$ of block b is defined by its offset (δ_p^x, δ_p^y) with respect to the block's geometric center (origin).
3) *Netlist*, denoted as set $\mathcal{N}$. A net $n \in \mathcal{N}$ describes a connection between two or more pins.
4) *TSV-island types*, denoted as set $\mathcal{T}$. Each type $t \in \mathcal{T}$ has dimensions (h_t, w_t) and capacity κ_t. Since predesigned TSV-island types may incorporate spare TSVs, κ_t defines the number of nets that can be routed through t.
5) *3-D floorplan*, denoted as set $\mathcal{F}$. Each block b is assigned a location (x_b, y_b, l_b) such that no blocks overlap. The coordinate of the block's origin is denoted as (x_b, y_b) and l_b denotes the assigned layer.

As mentioned in Section I, previous work on 3-D floorplanning often neglects design constraints and overhead associated with TSVs. However, these studies promise to provide optimized floorplans in terms of, e.g., minimal wirelength and thermal distribution. Therefore, 3-D integration following the L2Di style addresses the omission of TSV planning. It seeks to cluster interlayer nets into TSV islands without incurring excessive overhead. Such TSV islands, as well as single TSVs, are then inserted into deadspace around floorplan blocks. If TSV-island insertion is impossible due to lack of deadspace, blocks can be shifted from their initial locations without disturbing their ordering. Additional deadspace can be inserted when necessary.

V. Methodology

To connect blocks on different dies following the L2Di style, we need to know the locations of TSV islands. However, placing TSV islands, i.e., fixing these locations, must account for routing demand and routability, so as to avoid unnecessary detours. In order to solve this chicken-and-egg problem, we develop the following techniques.

1) *Net clustering*: groups nets to localize and estimate global routing demand.
2) *TSV-island insertion*: uses these groups to appropriately insert TSV islands.

Net clustering uses net bounding boxes, i.e., minimal rectangles containing net pins, which contain all shortest-path con-

nections in the absence of obstacles. The intersection of several net boxes forms a *cluster region* for respective nets. Placing TSV islands within the cluster regions facilitates shortest-path connections for all considered nets. Assigning nets to clusters furthermore helps to select the type and capacity of each TSV island. To formalize the clustering process, we consider a *virtual die*—the minimum rectangle containing projections of active-layer outlines.

TSV-island insertion utilizes cluster regions to determine where to insert TSV islands. This depends on available island types, deadspace, and obstruction (by blocks or other islands) of cluster regions. Also, given that net clustering determines different groups of nets, our proposed TSV-island insertion selects the most suitable cluster for each net to facilitate routing of all nets. Furthermore, our techniques can be extended to perform subsequent deadspace-related tasks like buffer insertion. Fig. 6 illustrates net clustering and TSV-island insertion for two dies.

In the following discussion, we refer to interlayer nets as just nets. Details of our techniques are discussed in Sections V-B–V-D, the overall flow is illustrated in Fig. 7. Please note that our methodology is performed stepwise for multiple active layers, as illustrated in Fig. 7(a). Key parameters used in our algorithms are defined in Table I (Section VI) along with their values.

A. Background on Intersections of Bounding Boxes

We provide the following definitions, lemmata, and theorems to discuss intersections of bounding boxes in detail. The discussion is related to a study by Imai and Asano [39] on intersections of axis-aligned rectangles in the plane.

Definition 1: The intervals $A[x^A_{\min}, x^A_{\max}]$ and $B[x^B_{\min}, x^B_{\max}]$ overlap if $x^A_{\min} \leq x^B_{\min} \leq x^A_{\max}$ or $x^B_{\min} \leq x^A_{\min} \leq x^B_{\max}$.

Lemma 1: Given two overlapping intervals, their set-intersection is also an interval.

Note: For an axis-aligned rectangle $A[x^A_{\min}, x^A_{\max}] \times [y^A_{\min}, y^A_{\max}]$ its projections onto the x-axis and y-axis are intervals $[x^A_{\min}, x^A_{\max}]$ and $[y^A_{\min}, y^A_{\max}]$, respectively.

Definition 2: Two axis-aligned rectangles $A[x^A_{\min}, x^A_{\max}] \times [y^A_{\min}, y^A_{\max}]$ and $B[x^B_{\min}, x^B_{\max}] \times [y^B_{\min}, y^B_{\max}]$ overlap iff their x-projections overlap and their y-projections overlap.

Lemma 2: Given two overlapping axis-aligned rectangles, their set-intersection is also an axis-aligned rectangle. Given n axis-aligned rectangles, if their set-intersection is nonempty, then it is an axis-aligned rectangle.

In their study, Imai and Asano proved that n axis-aligned rectangles (e.g., bounding boxes) have a single nonempty n-way intersection iff each pair of these rectangles overlap.

Theorem 1: Consider n axis-aligned rectangles. They overlap pairwise iff their n-way set-intersection is nonempty [39]. We will say that such rectangles overlap n-way.

Thus, rather than check all subsets of overlapping bounding boxes, we may search for cliques in a suitably defined *intersection graph*. This graph represents bounding boxes by vertices and connects overlapping boxes by edges.

Imai and Asano also provided an $\mathcal{O}(n \log n)$-time algorithm for finding the maximum clique in intersection graphs with n vertices, in spite of the fact that this problem is NP-hard for general graphs [40].

Theorem 2: Consider n axis-aligned rectangles where at least two rectangles do not overlap. A largest k-element subset of rectangles that overlap k-way can be found in $\mathcal{O}(n \log n)$ time [39].

In our context, however, determining a single (maximum) clique is insufficient. In general, such large cliques may exceed the capacity of the largest available TSV island. Several TSV islands can be combined to implement such a clique, but this increases routing congestion and mechanical stress, and aggravates signal integrity problems [21], [22]. Another

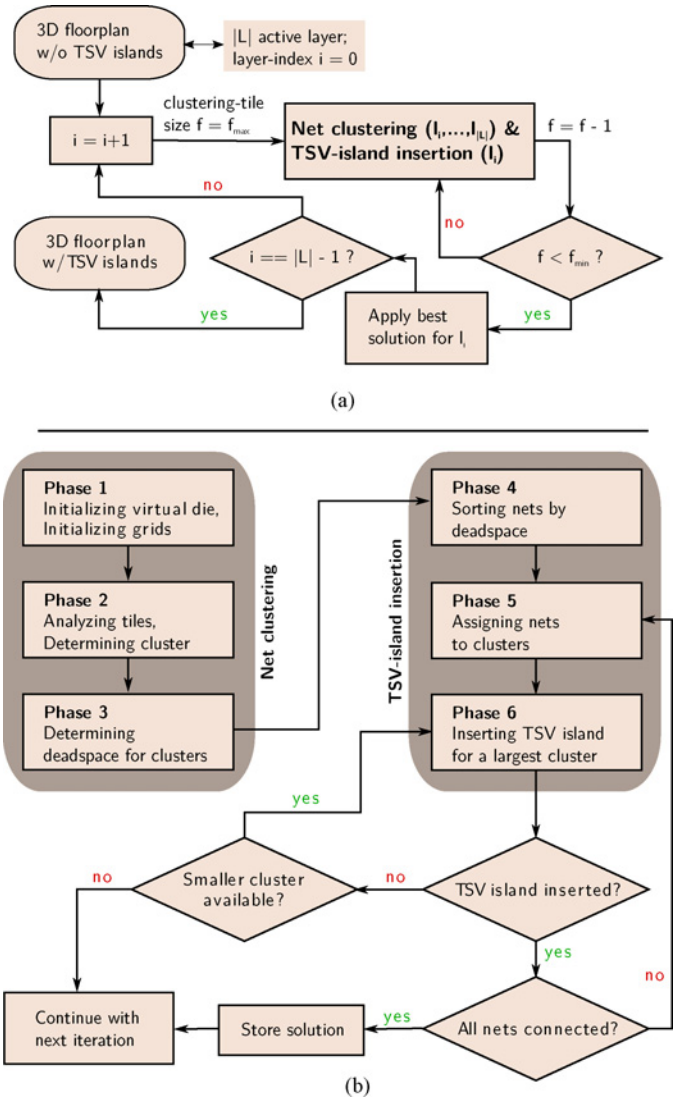

Fig. 7. Our methodology for 3-D IC integration. (a) Given a 3-D floorplan with $|\mathcal{L}|$ active layers, global iterations start with the lowest layer and perform net clustering and TSV-island insertion stepwise for all layers. Best solutions refer to solutions where TSV islands inserted in layer l_i result in smallest estimated wirelength. Clustering-grid tiles are resized in iterations, as explained in Section V-B. Assuming successful execution of TSV-island insertion on each layer, our techniques provide a 3-D floorplan with suitably placed TSV islands. (b) First, net clustering localizes global routing demand while determining cluster regions, described by intersections of net bounding boxes. Second, TSV-island insertion into cluster regions is iteratively attempted, based on dynamic scores, available TSV-island types, and deadspace.

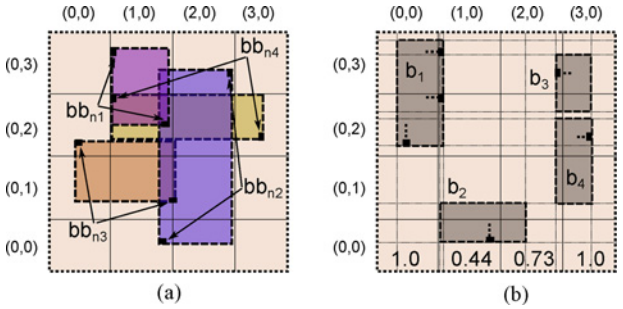

Fig. 8. Grid structures. (a) Uniform clustering grid $\mathcal{G}$ on the virtual die. According to projected bounding boxes, we link nets to covered tiles. The intersection of boxes must be explicitly checked during clustering. In tile (1, 2), e.g., net bounding boxes bb_{n_1}, bb_{n_3} and bb_{n_3}, bb_{n_4} do not overlap pairwise, but all four nets are linked to the tile. (b) In order to determine deadspace, a *nonuniform* grid $\mathcal{D}$ (and $\mathcal{G}$ only for illustration purpose) is constructed on the active layer. The per-tile ratio of deadspace is determined, as denoted in the last row. Deadspace is then annotated on each tile of $\mathcal{G}$.

problem with using large cliques is that corresponding (small) intersections of net bounding boxes may not include any deadspace, preventing the insertion of a TSV island. On the other hand, a smaller clique would imply fewer bounding boxes and a larger intersection that is more likely to admit TSV-island insertion. Thus, we seek to partition the edges in the intersection graph into a small set of cliques, which is the NP-complete clique cover problem [41]. Our algorithm, based on sophisticated analysis of bounding boxes, is presented next.

B. Net Clustering

The following algorithm is performed for subsets $\{l_i, \ldots, l_{|\mathcal{L}|}\}$ of active layers, where l_i denotes the lower layer. In order to identify clusters (cliques) of appropriate size, a *uniform* clustering grid $\mathcal{G}$ is constructed on the virtual die [Fig. 8(a)]. A clustering grid links each net n to each tile $\Xi \in \mathcal{G}$ covered by its net bounding box bb_n, and thus results in size-limited (appropriate) clusters. Nets connecting blocks on l_i to blocks on layers $l_{i+1}, \ldots, l_{|\mathcal{L}|}$ have to be considered. In this context, nets spanning three or more layers have to be adapted for following global iterations [Fig. 7(a)], as explained for TSV-island insertion (Section V-C). To calculate the amount of deadspace on l_i, a *nonuniform* grid $\mathcal{D}$ is constructed. Grid lines are drawn through the four edges of each block. Grid tiles not covered by blocks define deadspace. For m blocks overlapping with a particular tile Ξ, deadspace detection runs in $\mathcal{O}(m^2)$ time [11], which is not prohibitively expensive because typically $m < 50$. In the uniform grid $\mathcal{G}$, tiles with insufficient deadspace ($< \Xi^d_{\min}$) are marked as obstructed.

For the uniform grid $\mathcal{G}$, grid-tile size f influences per-tile net count. For example, quartering f in Fig. 8(a) would decrease the maximum per-tile net count from four to two. Having fewer nets per tile reduces the cluster size, increasing chances of TSV-island insertion. Therefore, we wrap our methodology into an outer loop [Fig. 7(a)], which iteratively decreases f from an upper bound $f_{\max}$ to a lower bound $f_{\min}$ (Table I). Afterward, the valid solution providing smallest estimated wirelength is chosen. The impact of grid-tile size on wirelength and success rate is discussed in Section VI-A.

Our clustering algorithm is illustrated in Fig. 9 (see also Fig. 7). *In Phase 1*, the virtual die and grid structures are

```
NET_CLUSTERING(i, L, N, B, F)
1  // Phase 1: initialize virtual die and grid structures
2  INITIALIZE_VIRTUAL_DIE(l_i, ..., l_|L|)
3  G = CONSTRUCT_CLUSTERING_GRID(l_i, ..., l_|L|, N, B, F)
4  D = CONSTRUCT_DEADSPACE_GRID(l_i, B, F)
5  // Phase 1: determine deadspace;
6  // mark tiles Ξ ∈ G where deadspace is < Ξ^d_min as obstructed
7  DETERMINE_DEADSPACE(D, G)
8  // Phase 1: link nets to grid tiles
9  foreach net n ∈ N where n connects l_i and l_{i+1}, ..., l_|L|
10     bb_n = DETERMINE_BOUNDING_BOX(n, B, F)
11     foreach grid tile Ξ ∈ G where Ξ is covered by bb_n
12        append n to Ξ.nets
13 // Phase 2: determine possible clusters
14 foreach grid tile Ξ ∈ G where Ξ.obstructed == FALSE
15     c = DETERMINE_CLUSTER(Ξ, Ω_min, O_nets, O_link)
16     if c ∉ C
17        insert c into C
18        foreach net n ∈ c.nets
19           n.clustered = TRUE
20     elseif |c.nets| > 0
21        UPDATE_CLUSTER_REGION(c, Ξ)
22 // Phase 2: handle yet unclustered nets
23 progress = TRUE
24 while progress == TRUE
25     RESET(unclustered_nets)
26     foreach net n ∈ N where n.clustered == FALSE
27        append n to unclustered_nets
28     c = DETERMINE_FURTHER_CLUSTER(unclustered_nets)
29     progress = (c ∉ C)
30     if progress == TRUE
31        insert c into C
32        foreach net n ∈ c.nets
33           n.clustered = TRUE
34 // Phase 3: determine available deadspace
35 foreach cluster c ∈ C
36     foreach grid tile Ξ ∈ G where Ξ is covered by c.bb
37        c.deadspace += INTERSECTION(c.bb, Ξ) × Ξ.deadspace
```

Fig. 9. Our net clustering algorithm. Input data are described in Section IV.

constructed. Then, each net is linked to each grid tile within the net's projected bounding box [Fig. 8(a)]. *In Phase 2*, for each unobstructed grid tile the largest cluster is determined in procedure DETERMINE_ CLUSTER—each linked net is considered as long as the resulting intersection of bounding boxes is nonempty.[3] Moreover, we impose a lower bound $\Omega_{\min}$ on the overlap area between the intersection and tiles, in order to assure the intersection is covering the unobstructed tile to some minimal degree and to maintain a minimal cluster size. An upper bound O_{nets} of nets assigned to each cluster c must not be exceeded. Each net n can be associated with at most O_{link} clusters. We note that intersections in general can overlap more than one tile, depending on the bounding boxes. Therefore, we allow cluster regions to be extended within procedure UPDATE_CLUSTER_REGION in cases where clusters are spread across several tiles. Next, we attempt to cluster yet-unclustered nets in procedure DETERMINE_FURTHER_CLUSTER. Nets are considered for clustering independent of related tiles, thus several combinations of nets are considered. Besides that, clustering is performed as described for procedure DETERMINE_ CLUSTER. Since this step allows one-net clusters, all nets

[3] For example, consider the second row from top of the clustering grid in Fig. 8(a). Note that tiles (0, 2) and (3, 2) are not used for cluster determination, since they are obstructed [Fig. 8(b)]. Clustering for tile (1, 2) results in c_1 with $c_1.nets = \{n_1, n_2, n_4\}$, and for tile (2, 2) in c_2 with $c_2.nets = \{n_2, n_4\}$.

```
TSV_ISLAND_INSERTION(i, L, C, N, T)
 1  // Phase 4: sort nets
 2  SORT_NETS_BY_AREA_SUPPLY(N, C)
 3  progress = TRUE
 4  while progress == TRUE
 5      // Phase 5: assign nets to clusters
 6      foreach net n ∈ N where n.inserted == FALSE
 7          c = FIND_HIGHEST_SCORED_CLUSTER(n, C, n^d_min)
 8          foreach net m ∈ c.nets
 9              append m to c.assigned_nets
10      // Phase 6: iteratively insert TSV island for a largest cluster
11      (c, t) = INSERT_TSV_ISLAND(l_i, C, T)
12      progress = (c != NULL)
13      // Phase 6: mark inserted nets
14      if progress == TRUE
15          foreach net n ∈ N where n ∈ c.nets
16              n.inserted = TRUE
17              n.TSV_island = t
18              // Phase 6: adapt nets spanning ≥ 3 layers
19              if n connects to layers l_{i+2}, .., l_{|L|}
20                  n.pin(l_{i+1}) = t.center
21          MARK_CLUSTER_HANDLED(c)
22          // Phase 6: remove all net assignments
23          foreach cluster c ∈ C
24              reset c.assigned_nets
```

Fig. 10. Our TSV-island insertion algorithm. Input data are described in Section IV.

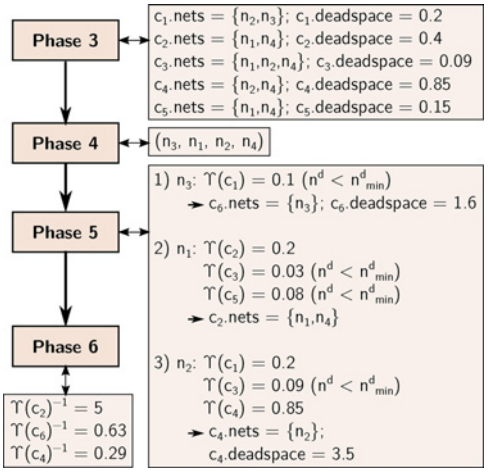

Fig. 11. Net assignment and cluster selection (refer to Fig. 8 for corresponding grid structures). In Phase 5, nets are assigned to clusters according to a score $\Upsilon(c)$. Not that there is no feasible cluster available for n_3, thus its bounding box defines a new cluster c_6. In Phase 6, TSV-island insertion is attempted using (sorted) clusters c_2, c_6, and c_4.

are guaranteed to be clustered afterward. *In Phase 3*, available deadspace is determined for each cluster region. It is summed up over available deadspace of related grid tiles while considering the particular intersection of the cluster region and tile.

C. TSV-Island Insertion

After running our net clustering algorithm, we now select cluster regions where TSV islands can be inserted in active layer l_i. Not all clusters need to have TSV islands inserted to allow routing all nets through TSVs—according to the bound O_{link}, each net may be *associated* with several clusters. Depending on the order of selecting clusters for TSV-island insertion, some clusters may become infeasible as island sites; deadspace accounted for a particular cluster may be shared with another cluster. Furthermore, clusters containing nets linked to obstructed tiles need to consider nearby deadspace. Both may result in TSV islands blocking each other.

Our TSV-island insertion algorithm (Fig. 10, see also Fig. 7) thus accounts for deadspace while iteratively assigning nets to clusters and inserting TSV islands. In the following discussion, we refer to nets assigned to a (inserted) TSV island as inserted nets, and to nets assigned to a (associated) cluster as assigned nets. *In Phase 4*, our algorithm sorts all nets by their total deadspace of associated clusters. Nets associated with clusters with little available deadspace are considered first, since corresponding TSV islands are difficult to insert. *In Phase 5*, associated clusters of each unassigned net are analyzed (Fig. 11). The highest-scored cluster with respect to a dynamic cluster score $\Upsilon(c) = c.deadspace \div |c.assigned_nets|$ (deadspace of cluster region divided by number of nets to be assigned) is chosen. Calculation of Υ for each cluster is performed dynamically within procedure FIND_HIGHEST_SCORED_CLUSTER. In order to facilitate TSV-island insertion, the cluster to be chosen must provide a minimal amount of deadspace $n^d_{\min}$ for each net to be assigned to it. Then, each (unassigned) net associated with the highest-scored cluster is assigned to this cluster, since it is most suitable for TSV-island insertion. Nets remaining unassigned after cluster analysis are assigned to one-net clusters, where the cluster region is defined by the net's bounding box. *In Phase 6*, TSV-island insertion for a largest cluster (in terms of $\Upsilon(c)^{-1} = |c.assigned_nets| \div c.deadspace$) is iteratively attempted—TSV-island insertion for clusters with many assigned nets and little available deadspace is difficult, thus these clusters are considered first. The procedure stops after inserting a TSV island for one largest cluster. Within the procedure, a local search over the cluster regions identifies contiguous regions with appropriate shapes. Therefore, the search aims to determine regions where a TSV island with sufficient capacity to connect all assigned nets can be inserted. Initially, deadspace is considered only within the cluster regions. If no contiguous regions of deadspace can be found, a second iteration expands the cluster regions by factors c^x_{ext}, c^y_{ext} (in terms of die dimensions) to widen the search. If no contiguous regions are found again for any cluster, iterative block shifting can be performed to increase deadspace (Section V-D). Therefore, the cluster providing maximal amount of deadspace is chosen first to minimize the total amount of shifting. After successful TSV-island insertion, inserted nets are marked as handled, and all noninserted nets are unassigned from remaining clusters—according to Υ, each noninserted net may be assigned to different clusters now. Furthermore, inserted nets connecting blocks on layer l_i to blocks on layers $l_{i+2}, \ldots, l_{|\mathcal{L}|}$ (spanning three or more layers) have to be adapted. The center of each related TSV island defines a virtual net pin, which is considered as respective net pin for following net-clustering iterations. Iterations continue with Phase 5 until all nets are inserted.

D. Deadspace Insertion and Redistribution

TSV-island insertion can fail because deadspace is unavailable where it is needed. To address these failures, we propose techniques to insert and redistribute deadspace.

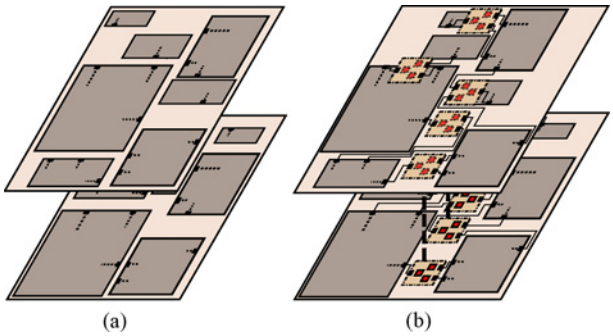

Fig. 12. Deadspace-channel insertion. TSV island are illustrated as brown, dashed boxes containing TSVs (solid, red boxes). Related landing pads are illustrated as dashed, red boxes. (a) Some floorplans exhibit only narrow channels between blocks. This obstructs insertion of buffers, glue logic, and TSV islands. (b) Inserting channels between blocks provides needed deadspace at the cost of larger chip area.

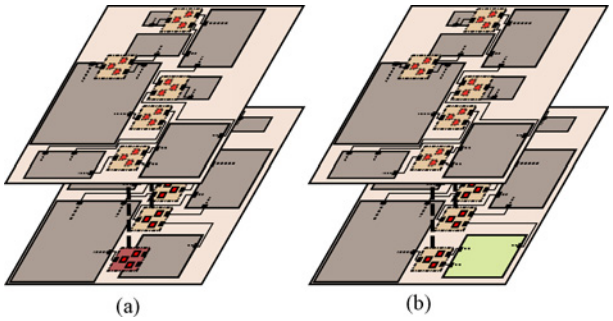

Fig. 13. Block shifting. (a) A given 3-D chip layout may provide sufficient, but inappropriately distributed deadspace. (b) Shifting blocks within the layout outline facilitates TSV-island insertion.

1) *Deadspace-channel insertion* provides regions to insert TSV islands (in case of too compact floorplans) and may facilitate routing (Fig. 12).
2) *Block shifting* allows to redistribute available deadspace to facilitate TSV-island insertion (Fig. 13).

Deadspace-channel insertion is often applied in industrial chip designs to facilitate routing, enable placement of buffers and glue logic, and increase flexibility of TSV-island insertion. However, this is less appropriate for compact floorplans.

Block shifting, on the other hand, facilitates compact floorplans (floorplan outlines are maintained) and TSV-island insertion. This approach is more complex, and success in gaining a sufficient amount of continuous deadspace is dependent on the actual floorplan. We develop two block-shifting techniques that rely on similar baseline algorithms: 1) *initial shifting*, and 2) *iterative shifting* (Fig. 14). Initial shifting performs block shifting once before our methodology is applied, as explained later on. Iterative shifting is performed during TSV-island insertion (INSERT_TSV_ISLAND, Fig. 10) when necessary.

Our algorithm for block shifting is based on the concept of spatial slack in floorplanning [42] and performs analysis of cluster regions. Slacks (for x-dimension and y-dimension) describe maximal possible displacement of a block within the floorplan outline. When blocks do not overlap, slacks are ≥ 0. We determine slacks for each layer separately and use standard linear-time traversals of floorplan constraint graphs [43], not unlike those in static timing analysis [44]. Floorplan modifications based on constraint graphs are discussed in detail in [45].

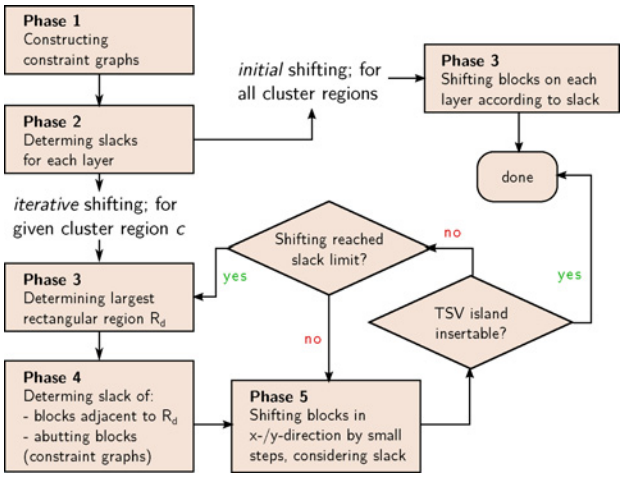

Fig. 14. Proposed flow for block shifting.

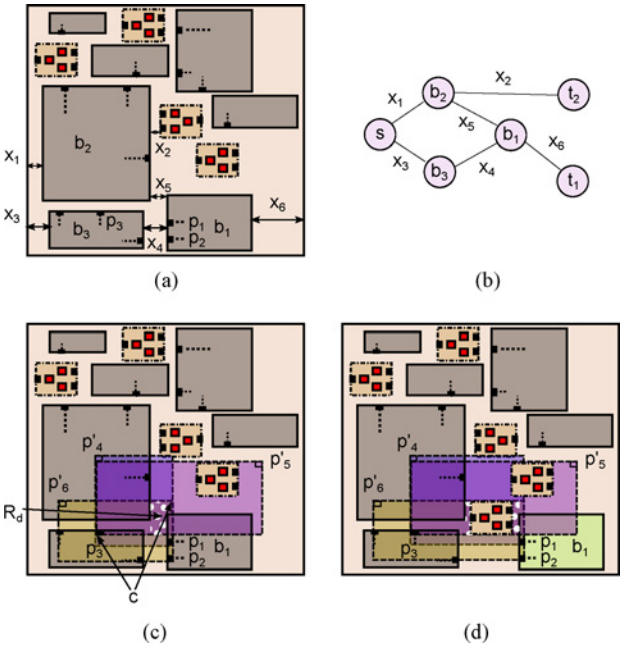

Fig. 15. Slack-based block shifting. (a) Connecting pins p_1, p_2, and p_3 to an adjacent layer (not illustrated) requires another TSV island. Related x-slacks are determined (labeled as x_n). (b) Slacks are then annotated on the constraint graphs (only the relevant part of the horizontal constraint graph is illustrated). (c) Cluster c (corners are pointed to) contains the deadspace region R_d (white dots); its area is too small for a TSV island. (d) Based on available slacks, block b_1 is shifted to resize R_d such that TSV-island insertion can succeed.

To calculate x-slacks, we: 1) pack blocks to the left boundary, and, independently, 2) pack blocks to the right boundary, both are invoked when dealing with constraint graphs. The x-slack for each block is computed as the difference of the block's x-coordinates in these two packings. The y-slack is calculated in the same way. Note that previously placed TSV islands are considered fixed obstacles. Allowing for TSV-island shifting might significantly increase wirelength, since our proposed TSV-island insertion aims for minimal wirelength.

An example for slack-based block shifting is given in Fig. 15. First, we determine slacks [Fig. 15(a)] and annotate them on the constraint graphs [Fig. 15(b)]. For iterative shifting, we determine the largest (rectangular) region R_d of deadspace for the cluster of interest [Fig. 15(c)]. If no

deadspace is found, we nominally consider the center of the cluster region as R_d. We then seek to consolidate additional deadspace around R_d by shifting away the blocks adjacent to R_d [Fig. 15(d)]. The distance by which each block is shifted cannot exceed its slack in the respective direction. Furthermore, the sum of such displacements in each direction cannot exceed the floorplan slack (the largest slack of any one block). Shifting a block may require shifting its abutting neighbors and other blocks. To this end, we maintain the floorplan configuration using constraint graphs. If R_d cannot be increased sufficiently, we choose another region of deadspace within the cluster region.

For initial shifting, we independently determine available slacks of blocks on all active layers and shift blocks such that they are centered according to slacks. This may facilitate TSV-island insertion around blocks since they are likely to be distributed toward the center of the die afterward, resulting in deadspace around them. Initial shifting is performed once before applying our methodology.

VI. Empirical Validation

We obtain 3-D floorplans by running state-of-the-art software [7][4] and configure the software to allow 10% deadspace on each active layer. We construct two sets of rectangular TSV islands, each containing via-first TSVs with footprints of $100\,\mu m^2$ and $50\,\mu m^2$, respectively. Each set contains islands with capacities for 1–30 nets while providing one redundant TSV, which is sufficient for practical TSV-failure rates [13]. Islands are designed by packing single TSVs in all possible configurations resulting in rectangular blocks. Packing accounts for practical spacing between adjacent TSVs of $10\,\mu m$ [19]. This facilitates manufacturing, the use of keep-out-zones and landing pads, and limits coupling between TSVs [46].

We implemented our algorithms using C++/STL, compiled them with g++ 4.4.3, and ran on a 32-bit Linux system with a 2.4 GHz AMD Opteron processor (using one processing unit) and 4 GB RAM. We configure parameters discussed in Section V according to Table I. We initially set $c^x_{\text{ext}} = c^y_{\text{ext}} = 10\%$. In cases where our algorithm terminates with no solution, we increase the value of both variables by 10%, and repeat the experiment until we obtain a valid solution or reach the maximum value of 50%. Table II reports results on representative GSRC benchmarks. As indicated in previous work [14], these benchmarks contain artificial, small blocks. To address this issue without modifying the floorplanner, every block was inflated by five times before floorplanning. After subsequently applying our methodology, active layers are contracted to the original size again to facilitate comparison with similar work. Thus, footprints of considered TSVs are implicitly shrunk to $4\,\mu m^2$ and $2\,\mu m^2$, respectively, in the contracted, final layouts. The benchmarks do not provide pin offsets, therefore we assume net bounding boxes to be defined by the bounding boxes of incident blocks. Since the used floorplanning software does not allow to account for I/O pins, nets connecting to such pins are not included in wirelength estimates. We consider intralayer nets by summing up the HPWL of their bounding box and include them in wirelength estimates. Since we do not perform net assignment to particular TSVs within islands, reported wirelength estimates consider the center of TSV islands to determine bounding boxes (resulting in pessimistic wirelength estimates). We also report runtime for our algorithms and methodology (summed up for global iterations), values for c^x_{ext} and c^y_{ext}, TSV count, and the area of the final layouts (defined as the product of the maximal height and maximal width over all active layers).

We consider two design configurations; one with guaranteed channels, one without channels. To insert channels between the blocks without modifying the floorplanner, every block was inflated (block dimensions by 5%) before floorplanning and contracted to the original size after floorplanning. However, this increases floorplan size (by 10.25%). An alternative is to pack blocks without channels, but carefully redistribute deadspace to facilitate TSV-island insertion. While more complex, this approach produces much more compact floorplans.

[4]We thank the authors of [7] and Y. Chen for sharing their infrastructure for 3-D floorplanning.

TABLE I
Parameters for Net Clustering and TSV-Island Insertion Algorithms, Along with Their Values

Metric	Meaning	Value
Ω_{min}	Min overlap area between cluster region and grid tile (tile size)	25%
Ξ^d_{min}	Min deadspace per clustering-grid tile (tile size)	70%
O_{nets}	Max nets per cluster	30
O_{link}	Max clusters per net	5
n^d_{min}	Min deadspace per net in a cluster (TSV footprint and keep-out zone)	110%
c^x_{ext}, c^y_{ext}	Extension of cluster region to search for nearby deadspace (die dimensions)	variable (10–50%)
f_{max}	Max clustering-tile size	15
f_{min}	Min clustering-tile size	5

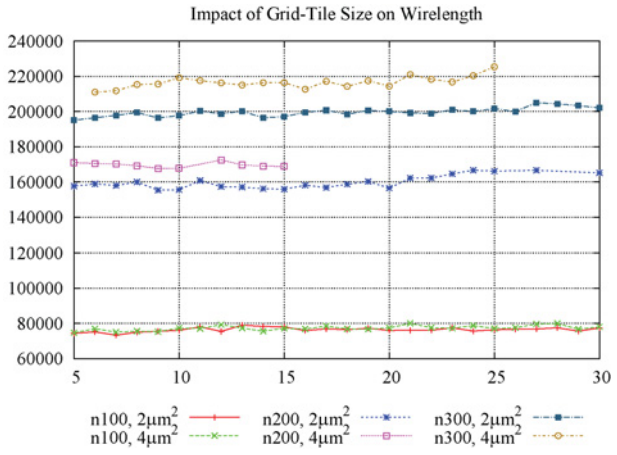

Fig. 16. Estimated wirelength (BB-2D3D-HPWL) over grid-tile size for L2Di integration of two dies. Results are obtained using initial shifting to redistribute deadspace. Note that intralayer nets are not considered.

A. Impact of Grid-Tile Size

As mentioned in Section V-B, our methodology is wrapped into a loop which iteratively decreases the grid-tile size f from an upper bound f_{max} to a lower bound f_{min} (Table I). Fig. 16 indicates that the density of valid solutions may increase with

TABLE II
L2DI INTEGRATION

Dies	Deadspace and TSV Footprint	Metrics	Redistributing Deadspace by Deadspace-Channel Insertion			Initial Block Shifting			Iterative Block Shifting		
			n100	n200	n300	n100	n200	n300	n100	n200	n300
2	10% 2 μm^2	BB-2D3D-HPWL	130 825	302 344	446 973	131 589	540 324‡	414 126	149 473†	482 071‡	530 096†
		TSVs	378	888	1162	420	941	1285	399	985	1294
		Runtime (s)	19.22	173.97	471.26	76.69	4075.93	1219.95	118.49	794.63	2689.91
	10% 4 μm^2	BB-2D3D-HPWL	133 085	345 474	455 082	143 039	543 962‡	442 638	151 272†	487 974‡	740 244‡
		TSVs	378	895	1154	422	943	1233	402	945	1209
		Runtime (s)	19.78	295.22	503.46	86.627	4698.35	1346.74	124.73	2030.43	1835.08
		Avg BB-2D3D-HPWL	131 955	323 909	451 028	137 314	542 143	428 382	150 373	485 023	635 170
		Normalized avg BB-2D3D-HPWL	0.961	0.597	1.053	1	1	1	1.095	0.895	1.483
		Area (mm^2)	0.1326	0.1301	0.2203	0.1203	0.1180	0.1998	0.1203	0.1180	0.1998
3	10% 2 μm^2	BB-2D3D-HPWL	125 263	254 756	349 766	107 556	260 285	329 024	134 568	297 761	354 223
		TSVs	534	1034	1480	541	1094	1467	582	1081	1519
		Runtime (s)	123.21	628.22	2124.44	222.39	2216.5	3268.41	146.05	701.43	1619.91
	10% 4 μm^2	BB-2D3D-HPWL	130 358	288 970	361 890	116 611	422 582‡	381 228	140 626	320 208	409 382
		TSVs	539	1038	1425	568	3809	1455	573	1055	1485
		Runtime (s)	126.37	1167.69	2166.21	172.13	3404.6	2977.85	144.91	518.07	1612.56
		Avg BB-2D3D-HPWL	127 811	271 863	355 828	112 084	341 434	355 126	137 597	308 985	381 803
		Normalized avg BB-2D3D-HPWL	1.140	0.796	1.002	1	1	1	1.228	0.905	1.075
		Area (mm^2)	0.1036	0.0979	0.1408	0.0939	0.0888	0.1277	0.0939	0.0888	0.1277
4	10% 2 μm^2	BB-2D3D-HPWL	113 884	235 542	312 764	112 720	245 816	313 033	136 135	270 877	329 745
		TSVs	654	1182	1597	696	1281	1700	698	1246	1758
		Runtime (s)	141.22	670.97	1590.08	269.25	1608.07	1295.67	181.35	593.72	1458.01
	10% 4 μm^2	BB-2D3D-HPWL	116 956	407 526‡	341 536	130 925	273 112	366 571	147 328	369 895‡	376 833
		TSVs	652	2257	1569	705	1252	1659	719	2018	1710
		Runtime (s)	147.2	1346.6	2316.36	383.66	2692.46	1576.98	154.36	1992.46	1415.2
		Avg BB-2D3D-HPWL	115 420	321 534	327 150	121 823	259 464	344 933	141 732	320 386	353 289
		Normalized avg BB-2D3D-HPWL	0.947	1.239	0.948	1	1	1	1.163	1.235	1.024
		Area (mm^2)	0.0653	0.0741	0.1177	0.0593	0.0673	0.1068	0.0593	0.0673	0.1068

Values for c_{ext}^x and c_{ext}^y are 10% unless otherwise noted († for 20% and ‡ for 50%).

decreasing grid-tile size. Small tiles limit the per-tile net count, thus also limiting the cluster size. In practice, smaller tiles lead to fewer nets being assigned per cluster, larger cluster regions, and easier TSV-island insertion. This also reduces wirelength, as expected. However, there is a lower bound for these relations. Very small tiles result in many clusters with few assigned nets, thus many small TSV islands might be inserted. Since placed TSV islands are fixed obstacles, this may complicate iterative block shifting. Also, our local search over cluster regions identifies contiguous deadspace for TSV-island insertion greedily. Therefore, determining appropriate regions for clusters considered late during TSV-island insertion is more likely to fail; there are already many TSV islands spread out within deadspace regions.

After confirming these trends in different experimental configurations, we set global iteration variables $f_{max} = 15$ and $f_{min} = 5$.

B. Results of TSV-Island Insertion

First, we evaluate our techniques for deadspace insertion and redistribution. Recall that deadspace-channel insertion increases floorplan's deadspace by inflating blocks (and contracting after floorplanning), which simultaneously increases floorplan area by 10.25%. In contrast, block shifting retains the floorplan's outline. Comparing wirelength estimates in Table II, we observe that deadspace-channel insertion on average is superior to *iterative* shifting, but inferior to *initial* shifting. On average, iterative shifting results in larger wirelength compared to initial shifting. During TSV-island insertion, previously placed islands represent fixed obstacles. Thus, the success of iterative shifting is undermined by decreased slacks compared to initial shifting. We therefore prefer initial block shifting for L2Di integration.

TABLE III
L2DI INTEGRATION FOR THREE DIES USING ONLY SQUARE TSV ISLANDS

Deadspace and TSV Footprint	Metric	n100	n200	n300
10% 2 μm^2	BB-2D3D-HPWL	114 795	Fail	413 472
	Normalized	1.067	–	1.257
10% 4 μm^2	BB-2D3D-HPWL	151 528	Fail	Fail
	Normalized	1.157	–	–

Initial block shifting is used to redistribute deadspace. Values are normalized to Table II.

Second, we analyze the impact of die count. We observe that wirelength estimates decrease on average for increasing die count. As expected, TSV counts increase on average.

Third, we analyze the impact of available TSV-island types, considering their capacity and dimensions. As expected, smaller TSVs simplify TSV-island insertion (Table II). Shape-flexible TSV islands increase chances for successful TSV-island insertion significantly; one particular setup using only square TSV islands is illustrated in Table III.

Fourth, we evaluate the overhead of TSV islands. Islands with more than a single TSV require larger continuous deadspace. On the other hand, the intersection of several

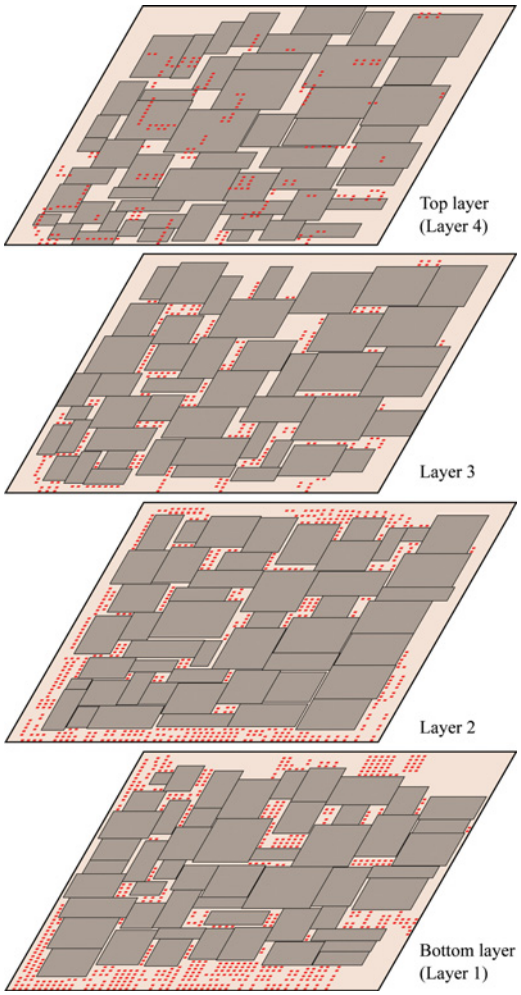

Fig. 17. L2Di integration of the GSRC benchmark n200. TSV footprints are 4 μm^2, and initial shifting is used to redistribute deadspace. TSV islands are shown as red dots. To enhance clarity, landing pads (red dots) are only illustrated on the uppermost layer.

TABLE IV
L2Di Integration for Four Dies Using ''Trivial'' TSV Islands (Single TSVs), Resulting in L2D Integration

Deadspace and TSV Footprint	Metric	n100	n200	n300
10% 2 μm^2	BB-2D3D-HPWL	110 047	223 880	277 914
	Normalized	0.976	0.911	0.888
10% 4 μm^2	BB-2D3D-HPWL	119 145	243 076	323 556
	Normalized	0.91	0.89	0.882

Initial block shifting is used to redistribute deadspace. Values are normalized to Table II.

net bounding boxes may be small in practice, depending upon net selection. However, our methodology accounts for sufficient deadspace while determining clusters and assigning nets to clusters in order to facilitate TSV-island insertion. Still, deadspace may be obstructed by iteratively placed TSV islands, which cannot be accounted for during net clustering. Therefore, using TSV islands may entail additional *overhead* in terms of increased wirelength. Table IV reports wirelength estimates for L2Di integration using *trivial* TSV islands (single TSVs) for a particular configuration. Here, we do not account for the possibly increased footprint of single TSVs (due to increased keep-out-zones in comparison to packed TSV arrays) and the loss of redundancy offered by TSV islands containing one spare TSV. These estimates are at least 91% (on average) of those in earlier experiments (Table II). Other configurations produced similar results. We conclude that the overhead of TSV islands is moderate and can be tolerated given their benefits (Section III-A).

Fifth, Fig. 17 illustrates an example of successful L2Di integration for the benchmark n200 using four active layers.

VII. Conclusion

Our work seeks to streamline the transition from existing practice in 2-D chip design to 3-D integration. Numerous technical challenges in this transition were pointed out in Sections I and II, as well as by Borkar [1] (Intel) and Topaloglu [47] (Global Foundries). TSVs tend to disrupt conventional layouts, each impacting several dies at once. Manufacturing of 3-D-enabled dies is complicated by considerations of yield for TSVs and thinned dies, as well as cost-effective testing. EDA tools need to support both 3-D path-finding efforts and comprehensive layout optimization, in particular TSV management. Power delivery, DFT, and reliability verification are further challenges for tool development.

The lack of commercial 3-D EDA tools hinders a cost-effective transition, and even when such tools become widely available, upgrading extensive 2-D IP portfolios for 3-D integration may take years. A key insight in our paper is that many of the benefits provided by 3-D ICs can be obtained while reusing existing 2-D IP blocks. In fact, such reuse is required for heterogeneous 3-D system-on-chips where circuit modules cannot be split between memory, digital, analog, and MEMS dies. Therefore, we analyze feasible design styles for 3-D integration of 2-D blocks. We introduce the design style L2Di, where TSVs can be clustered into TSV islands rather than always placed individually. This style appears most promising and least risky for 3-D IC design in the next 5–8 years.

To enable the L2Di style, we draw on graph-theoretical results to contribute novel techniques for net clustering and TSV-island insertion. We also develop techniques to insert and redistribute deadspace. Experiments validate the feasibility and efficiency of our methodology. Typically, initial block shifting is the most promising technique to redistribute deadspace.

Initial experiments conducted at the outset of our research indicated that naive algorithms for L2Di integration lead to very high interconnect overhead. The ISPD 2011 version of this paper reported smaller, but still significant overhead of roughly 13–17%. However, the highly optimized techniques developed in the course of our research reduce this overhead down to ≈9% for block-level interconnect, making it tolerable. Extensions of our core algorithms to 3-D ICs with more than two active layers appear in this paper for the first time. Compared to the entire interconnect stack, the wirelength overhead is negligible because the majority of wires are contained within individual blocks [30].

REFERENCES

[1] S. Borkar, "3D integration for energy efficient system design," in *Proc. Des. Automat. Conf.*, 2011, pp. 214–219.

[2] ITRS. (2009). *International Technology Roadmap for Semiconductors* [Online]. Available: http://www.itrs.net/Links/2009ITRS/Home2009.htm

[3] Global Industry Analysts, Inc. (2010). *3D Chips (3D IC): A Global Market Report* [Online]. Available: http://www.prweb.com/releases/3D_chips/3D_IC/prweb4400904.htm

[4] L. Cheng, L. Deng, and M. D. F. Wong, "Floorplanning for 3-D VLSI design," in *Proc. Asia South Pacific Des. Automat. Conf.*, 2005, pp. 405–411.

[5] J. Cong, J. Wei, and Y. Zhang, "A thermal-driven floorplanning algorithm for 3D ICs," in *Proc. Int. Conf. Comput.-Aided Des.*, 2004, pp. 306–313.

[6] J. Cong and Y. Ma, "Thermal-aware 3D floorplan," in *Integrated Circuits and Systems*. New York: Springer, 2010, ch. 4, pp. 63–102.

[7] P. Zhou, Y. Ma, Z. Li, R. P. Dick, L. Shang, H. Zhou, X. Hong, and Q. Zhou, "3D-STAF: Scalable temperature and leakage aware floorplanning for three-dimensional integrated circuits," in *Proc. Int. Conf. Comput.-Aided Des.*, Nov. 2007, pp. 590–597.

[8] Z. Li, X. Hong, Q. Zhao, S. Zeng, J. Bian, H. Yang, and C. K. Cheng, "Integrating dynamic thermal via planning with 3D floorplanning algorithm," in *Proc. Int. Symp. Phys. Des.*, 2006, pp. 178–185.

[9] X. Li, Y. Ma, and X. Hong, "A novel thermal optimization flow using incremental floorplanning for 3D ICs," in *Proc. Asia South Pacific Des. Automat. Conf.*, 2009, pp. 347–352.

[10] P. Jain, P. Zhou, C. H. Kim, and S. S. Sapatnekar, "Thermal and power delivery challenges in 3D ICs," in *Integrated Circuits and Systems*. New York: Springer, 2010, ch. 3, pp. 33–61.

[11] E. Wong and S. K. Lim, "Whitespace redistribution for thermal via insertion in 3D stacked ICs," in *Proc. Int. Conf. Comput.-Aided Des.*, 2007, pp. 267–272.

[12] M. Pathak, Y.-J. Lee, T. Moon, and S. K. Lim, "Through-silicon-via management during 3D physical design: When to add and how many?" in *Proc. Int. Conf. Comput.-Aided Des.*, 2010, pp. 387–394.

[13] A.-C. Hsieh, T.-T. Hwang, M.-T. Chang, M.-H. Tsai, C.-M. Tseng, and H.-C. Li, "TSV redundancy: Architecture and design issues in 3D IC," in *Proc. Des. Automat. Test Eur.*, 2010, pp. 166–171.

[14] M.-C. Tsai, T.-C. Wang, and T. T. Hwang, "Through-silicon via planning in 3-D floorplanning," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 19, no. 8, pp. 1448–1457, Aug. 2011.

[15] Y.-J. Lee, M. Healy, and S. K. Lim, "Co-design of reliable signal and power interconnects in 3D stacked ICs," in *Proc. Int. Interconn. Technol. Conf.*, 2009, pp. 56–58.

[16] D. H. Kim, K. Athikulwongse, and S. K. Lim, "A study of through-silicon-via impact on the 3D stacked IC layout," in *Proc. Int. Conf. Comput.-Aided Des.*, 2009, pp. 674–680.

[17] Y.-J. Lee, R. Goel, and S. K. Lim, "Multi-functional interconnect co-optimization for fast and reliable 3D stacked ICs," in *Proc. Int. Conf. Comput.-Aided Des.*, 2009, pp. 645–651.

[18] A. K. Coskun, A. B. Kahng, and T. S. Rosing, "Temperature- and cost-aware design of 3D multiprocessor architectures," in *Proc. Euromicro Conf. Digit. Syst. Des.*, 2009, pp. 183–190.

[19] M. Jung, J. Mitra, D. Z. Pan, and S. K. Lim, "TSV stress-aware full-chip mechanical reliability analysis andoptimization for 3D IC," in *Proc. Des. Automat. Conf.*, Jun. 2011, pp. 188–193.

[20] I. Loi, S. Mitra, T. H. Lee, S. Fujita, and L. Benini, "A low-overhead fault tolerance scheme for TSV-based 3D network on chip links," in *Proc. Int. Conf. Comput.-Aided Des.*, Nov. 2008, pp. 598–602.

[21] J.-S. Yang, K. Athikulwongse, Y.-J. Lee, S. K. Lim, and D. Z. Pan, "TSV stress aware timing analysis with applications to 3D-IC layout optimization," in *Proc. Des. Automat. Conf.*, Jun. 2010, pp. 803–806.

[22] S. Garg and D. Marculescu, "3D-GCP: An analytical model for the impact of process variations on the critical path delay distribution of 3D ICs," in *Proc. Int. Symp. Qual. Elec. Des.*, 2009, pp. 147–155.

[23] C. Liu, T. Song, and S. K. Lim, "Signal integrity analysis and optimization for 3D ICs," in *Proc. Int. Symp. Qual. Elec. Des.*, 2011, pp. 42–49.

[24] D. H. Kim, S. Mukhopadhyay, and S. K. Lim, "Through-silicon-via aware interconnect prediction and optimization for 3D stacked ICs," in *Proc. Int. Workshop Syst.-Level Interconn. Pred.*, 2009, pp. 85–92.

[25] Z. Li, X. Hong, Q. Zhou, J. Bian, H. H. Yang, and V. Pitchumani, "Efficient thermal-oriented 3D floorplanning and thermal via planning for two-stacked-die integration," *ACM Trans. Des. Automat. Elec. Syst.*, vol. 11, no. 2, pp. 325–345, Apr. 2006.

[26] H.-H. S. Lee and K. Chakrabarty, "Test challenges for 3D integrated circuits," *Des. Test Comput.*, vol. 26, no. 5, pp. 26–35, 2009.

[27] D. L. Lewis and H.-H. S. Lee, "Test strategies for 3D die stacked integrated circuits," in *Proc. Workshop 3D Integr. Technol. Architecture Des. Autom. Test in Conjunction with Des. Autom. Test Eur.*, Nice, France, Apr. 2009.

[28] R. Fischbach, J. Lienig, and T. Meister, "From 3D circuit technologies and data structures to interconnect prediction," in *Proc. Int. Workshop Syst.-Level Interconn. Pred.*, 2009, pp. 77–84.

[29] V. S. Nandakumar and M. Marek-Sadowska, "Layout effects in fine-grain 3-D integrated regular microprocessorblocks," in *Proc. Des. Automat. Conf.*, 2011, pp. 639–644.

[30] D. Sylvester and K. Keutzer, "A global wiring paradigm for deep submicron design," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Sys.*, vol. 19, no. 2, pp. 242–252, Feb. 2000.

[31] Cadence Design Systems, Inc. (2010). *3D ICs with TSVs: Design Challenges and Requirements* [Online]. Available: http://www.cadence.com/rl/Resources/white_papers/3DIC_wp.pdf

[32] L. K. Scheffer, "CAD implications of new interconnect technologies," in *Proc. Des. Automat. Conf.*, 2007, pp. 576–581.

[33] L. Jiang, Q. Xu, K. Chakrabarty, and T. M. Mak, "Layout-driven test-architecture design and optimization for 3D SoCs under pre-bond test-pin-count constraint," in *Proc. Int. Conf. Comput.-Aided Des.*, Nov. 2009, pp. 191–196.

[34] G. H. Loh, Y. Xie, and B. Black, "Processor design in 3D die-stacking technologies," *IEEE Micro*, vol. 27, no. 3, pp. 31–48, May–Jun. 2007.

[35] C. Ferri, S. Reda, and R. I. Bahar, "Strategies for improving the parametric yield and profits of 3D ICs," in *Proc. Int. Conf. Comput.-Aided Des.*, 2007, pp. 220–226.

[36] M. B. Healy, K. Athikulwongse, R. Goel, M. M. Hossain, D. H. Kim, Y.-J. Lee, D. L. Lewis, T.-W. Lin, C. Liu, M. Jung, B. Ouellette, M. Pathak, H. Sane, G. Shen, D. H. Woo, X. Zhao, G. H. Loh, H. S. Lee, and S. K. Lim, "Design and analysis of 3D-MAPS: A many-core 3D processor with stacked memory," in *Proc. Custom Integr. Circuits Conf.*, Sep. 2010, pp. 1–4.

[37] K. Lu, X. Zhang, S.-K. Ryu, J. Im, R. Huang, and P. S. Ho, "Thermo-mechanical reliability of 3-D ICs containing through silicon vias," in *Proc. Electron. Compon. Technol. Conf.*, May 2009, pp. 630–634.

[38] X. Zhao, J. Minz, and S. K. Lim, "Low-power and reliable clock network design for through-silicon via (TSV) based 3D ICs," *IEEE Trans. Compon., Packag., Manuf. Technol.*, vol. 1, no. 2, pp. 247–259, Feb. 2011.

[39] H. Imai and T. Asano, "Finding the connected components and a maximum clique of an intersection graph of rectangles in the plane," *J. Algorith.*, vol. 4, no. 4, pp. 310–323, 1983.

[40] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*. New York: Dover, 1998.

[41] R. Fowler, "Optimal packing and covering in the plane are NP-complete," *Inform. Process. Lett.*, vol. 12, no. 3, pp. 133–137, 1981.

[42] S. N. Adya and I. L. Markov, "Consistent placement of macro-blocks using floorplanning and standard-cell placement," in *Proc. Int. Symp. Phys. Des.*, 2002, pp. 12–17.

[43] A. B. Kahng, J. Lienig, I. L. Markov, and J. Hu, *VLSI Physical Design: From Graph Partitioning to Timing Closure*. New York: Springer, 2011.

[44] S. S. Sapatnekar, *Timing*. Norwell, MA: Kluwer, 2004.

[45] M. D. Moffitt, J. A. Roy, I. L. Markov, and M. E. Pollack, "Constraint-driven floorplan repair," *ACM Trans. Des. Automat. Electron. Syst.*, vol. 13, no. 4, pp. 1–13, 2008.

[46] D. H. Kim, S. Mukhopadhyay, and S. K. Lim, "TSV-aware interconnect length and power prediction for 3D stacked ICs," in *Proc. Int. Interconn. Technol. Conf.*, 2009, pp. 26–28.

[47] R. Topaloglu, "Applications driving 3-D integration and corresponding manufacturingchallenges," in *Proc. Des. Automat. Conf.*, 2011, pp. 214–219.

Johann Knechtel (S'11) received the M.S. (Diploma) degree in information systems engineering from the Dresden University of Technology, Dresden, Germany, in 2010. Currently, he is pursuing the Ph.D. degree from the Institute of Electromechanical and Electronic Design, Dresden University of Technology.

In 2010, he was a Visiting Research Student with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor. His current research interests include very large-scale integrated physical design automation with emphasis on 3-D integration.

Mr. Knechtel received a scholarship from the German Academic Exchange Service in 2010. He is a fellow and a Ph.D. Scholarship Holder of the research group Nano and Biotechnologies for Packaging of Electronic Systems, funded by the German Research Foundation.

Igor L. Markov (S'97–M'01–SM'05) received the Ph.D. degree in computer science from the University of California at Los Angeles, Los Angeles.

Currently, he is an Associate Professor of electrical engineering and computer science at the University of Michigan, Ann Arbor. He has co-authored three books and more than 180 refereed publications. His current research interests include computers that make computers.

Prof. Markov was the recipient of a DAC Fellowship, an ACM SIGDA Outstanding New Faculty Award, an NSF CAREER Award, an IBM Partnership Award, a Microsoft A. Richard Newton Breakthrough Research Award, and the inaugural IEEE CEDA Early Career Award. Some of his publications have received the Best Paper Awards at the Design Automation and Test in Europe (DATE), the International Symposium on Physical Design, and the IEEE Transactions on Computer-Aided Design conferences. In the 2011 redesign of the ACM Computing Classification System, he led the effort on the hardware tree. He is an Executive Board Member of ACM SIGDA. He is an Editorial Board Member of the *Communications of ACM* and IEEE DESIGN AND TEST, as well as several ACM and IEEE transactions. He has chaired tracks at DAC, ICCAD, ICCD, DATE, and GLSVLSI.

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

He is currently a Full Professor of electrical engineering at the Dresden University of Technology, where he is also the Director of the Institute of Electromechanical and Electronic Design. From 1999 to 2002, he was a Tool Manager with Robert Bosch GmbH, Reutlingen, Germany. From 1996 to 1999, he was with Tanner Research, Inc., Pasadena, CA. From 1994 to 1996, he was a Visiting Assistant Professor with the Department of Computer Science, University of Virginia, Charlottesville. From 1991 to 1994, he was a Post-Doctoral Fellow with Concordia University, Montréal, QC, Canada. His current research interests include physical design automation of very large-scale integrated circuits, multichip modules, and printed circuit boards, with a special emphasis on electromigration avoidance in physical design, 3-D design, and constraint-driven design methodologies.

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