Shrunk-2-D: A Physical Design Methodology to Build Commercial-Quality Monolithic 3-D ICs

Shreepad Panth, *Member, IEEE*, Kambiz Samadi, *Member, IEEE*, Yang Du, *Member, IEEE*, and Sung Kyu Lim, *Senior Member, IEEE*

***Abstract*—Monolithic 3-D (M3D) integrated circuits (ICs) are an emerging technology that offer much higher integration densities than previous 3-D IC approaches. In this paper, we present a complete netlist-to-layout design flow to design an M3D block, as well as to integrate 2-D and 3-D blocks into an M3D SoC. This design flow is based on commercial tools built for 2-D ICs, and enhanced with our 3-D specific methodologies. We use the OpenSPARC T2 SoC as a case study, implement it in a 28-nm fully depleted silicon on insulator foundry process, and demonstrate that we can achieve up to 12% and 8% power savings for a single block and SoC, respectively, when compared with their 2-D counterparts implemented using commercial tools.**

***Index Terms*—3-D floorplanning, monolithic 3-D (M3D) IC, partitioning, placement.**

I. Introduction

As TECHNOLOGY scaling approaches its limits, 3-D integrated circuits (3-D ICs) have been proposed as one solution to the interconnect bottleneck. The conventional technique of fabricating 3-D ICs is using through-silicon-vias (TSVs), where two or more layers of devices are fabricated separately, aligned and bonded. However, the relatively large pitch and parasitics of TSVs limit them to memory-on-logic or large logic-on-logic designs with relatively small number of global interconnects [1].

An emerging alternative is monolithic 3-D (M3D) integration, where the tiers are fabricated sequentially, one on top of another, and connected together using monolithic intertier vias (MIVs). Since no die alignment is required, these MIVs are roughly the same size as local vias [2]. Overall, M3D offers extremely high integration densities, and the size of MIVs ensure that they have negligible parasitics.

Full-custom circuits such as SRAM [3] or FPGAs [4] can be designed in M3D, but they require little changes to CAD tools and mainly rely on manual effort. With respect to general logic, three design styles are possible—transistor-level, block-level, and gate-level. Transistor-level integration is the most fine-grained technique [5], [6], where the pMOS and nMOS within standard cells are placed on different tiers. However, this style requires redesign and recharacterization of the standard cells, and the standard cell footprint does not reduce by 50% in 3-D due to the mismatch in the pMOS and nMOS sizes. The next design style is block-level, where 2-D functional blocks are floorplanned onto different tiers [7]. This style has the benefit of IP reuse, but does not fully take advantage of the fine-grained nature of MIVs. The last design style is gate-level [8], where existing standard cells and memory can be placed on multiple tiers. The advantage of this style is that existing cells can be reused, has no total area overhead, and a high integration density to obtain power benefits.

Gate-level M3D is the most attractive option to design a single logic dominated block as it offers significant performance improvements without any area overhead. However, previous works have only presented M3D results based on academic engines without timing optimization or a real clock tree [8], or design flows that are incapable of handling several real world constraints such as memory [5]. Furthermore, today's chips are far more complicated than a single block. If an entire SoC is to be designed in M3D, certain blocks will benefit from folding them (e.g., processor), and certain blocks would best remain 2-D (e.g., cache). No design flow exists that is capable of handling an M3D SoC implementation with 3-D blocks, let alone one with a mix of 2-D and 3-D blocks. This paper provides a design flow that is capable of handling all such real world constraints, and this paper is the first work to do each of the following.

1) Provide a high-quality design flow to design a single block in M3D, including all design and optimization stages.
2) Demonstrate how preplaced hard macros in a 3-D space can be handled using commercial 2-D IC tools, and also provide a technique for 2-D commercial-tool assisted 3-D memory placement.
3) Demonstrate a consistent, proven power benefit for an M3D IC when compared to a signoff quality 2-D IC designed using state-of-the-art commercial tools.
4) Present a design flow that is capable of handling 3-D blocks during physical design.

In addition to these contributions, we study the impact of various M3D folding options using the OpenSPARC T2 processor, and present design guidelines that are general enough to apply to other SoCs as well.

Manuscript received May 21, 2016; revised October 27, 2016; accepted December 15, 2016. Date of publication January 5, 2017; date of current version September 14, 2017. This paper was recommended by Associate Editor Y. Chen.

S. Panth is with Intel Corporation, San Jose, CA 95134 USA (e-mail: shreepad.panth@intel.com).

K. Samadi and Y. Du are with Qualcomm Technologies Inc., San Diego, CA 92121 USA.

S. K. Lim is with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA.

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.2017.2648839

0278-0070 © 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

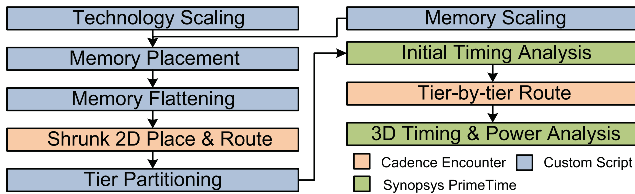

Fig. 1. Shrunk-2-D flow for designing a single M3D block.

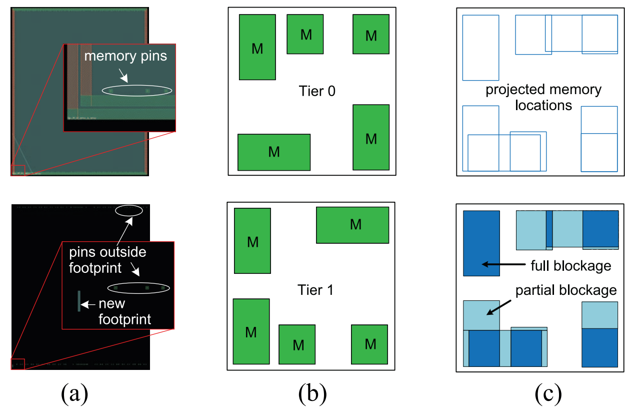

Fig. 2. Handling manually preplaced memory macros. (a) Isolating memory pins. (b) Sample preplacement of memory macros. (c) Handling memory blockages in a Shrunk-2-D footprint.

In the subsequent sections, this paper presents the first commercial-quality M3D design flow for a single block, followed by the first M3D SoC design flow, commercial-quality or otherwise. Next, experimental results that show the benefit of these flows, along with detailed case studies about different SoC floorplan options are presented. Finally, we discuss design learnings and guidelines.

II. Designing Single M3D Blocks

This section assumes that the footprint of the M3D block is known, as well as the tier and partition of all its IO pins. How these are generated in the context of the entire SoC is described in Section III. Note that this paper assumes only two tiers, but the flow can easily be extended to more tiers.

Assume that an optimal M3D placer exists that is capable of handling true 3-D timing optimization, routing, clock tree synthesis (CTS), etc. Now, take this optimal placement of all cells (and buffers), and remove the z dimension. Essentially, (x, y, z) of each cell transforms to (x, y), with cells overlapping. The idea behind this design flow is to perform this process in reverse.

1) Trick a commercial 2-D tool to achieve a placement such that it represents an M3D IC with all the tiers flattened, i.e., optimal (x, y) with cell overlap.
2) Post-partition this placement (assign z) with minimal change to the (x, y) location of cells.
3) Insert MIVs and reroute each tier with a commercial router while minimizing the routing change from 1).

The overall design flow is shown in Fig. 1. Several technology files are required to be scaled to handle point 1) above, which is discussed in Section II-A. Memory macros require additional steps such as scaling, placement, and flattening, which will be discussed in Section II-B. Once this is done, the commercial 2-D engine (Cadence Encounter) can be run on this "Shrunk-2-D" design. This result is then split into multiple tiers to obtain a DRC-clean sign-off quality design as described in Section II-D, and finally timing and power analysis is performed as described in Section II-E.

A. Technology Scaling

This section assumes a gate-only design, and handling memory will be introduced in Section II-B. The challenges that need to be overcome to enable commercial 2-D tools to design a gate-level M3D IC are as follows.

1) Enable an "overlapped" placement of all gates such that it represents both tiers of a superimposed M3D IC.
2) Ensure that any routing and extraction performed on this Shrunk-2-D design, even if it does not use the same metal dimensions as the final M3D design, accurately reflects the final M3D parasitics.

For a two tier design, superimposing the M3D tiers leads to a 2-D design with half the available placement area. Placing all the gates into this (using a 2-D tool) can be achieved by shrinking the area of each standard cell by 50%, or each side by $1/\sqrt{2}$ (0.707). The liberty (LIB) timing models of the cells are left unchanged to capture the actual timing behavior of these cells in the final M3D design.

To make the routing reflect M3D, we shrink the metal width and pitch by 0.707. We also need the extracted parasitics to reflect the full size wires in M3D. In this paper, we simply reuse the same capacitance tables as 2-D, and the correlation was found to be acceptable (discussed in Section IV-B). Tuning of the Shrunk-2-D capacitance tables may be required in future technology generations if this does not hold true.

B. Handling Memory Macros

This section first presents handling manually preplaced memory macros, and then discusses how the tool can be used to determine suitable locations for memory in a 3-D space.

1) Manual Preplaced Memory: Handling of preplaced memory within a commercial 2-D IC framework needs to overcome the following challenges: 1) memory macros are preplaced, cannot be moved, and thus cannot be shrunk down; 2) the commercial 2-D IC tool needs to be aware that the memory induces a placement blockage in its respective tier only; and 3) the timing model of the memory needs to be captured during Shrunk-2-D so that timing optimization and clock tree synthesis is performed accurately. These problems can be overcome by realizing that any given cell has a logical component, used for timing optimization, etc, and a physical component, which prevents overlap, etc. We can solve all three challenges by isolating these components.

First, we shrink down the footprint of the memory macros to the minimum size possible (that of a filler cell), while leaving it pins defined in the original locations. Therefore, the macro boundary will be smaller than the (x, y) of its pins [Fig. 2(a)]. This separates out the logical components, and we can now remove the z dimension of the preplaced memory to solve

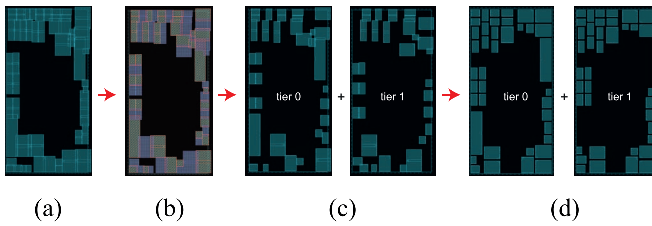

Fig. 3. Automatic memory partitioning. (a) Obtaining an initial memory placement with overlap. (b) Coloring overlapped memory. (c) Splitting the memory. (d) Legalizing the memory.

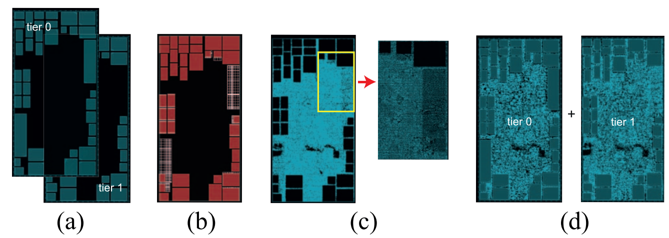

Fig. 4. Illustration of Shrunk-2-D flow. (a) Memory preplacement. (b) Input for Shrunk-2-D placement. Hashed areas indicate partial blockages while solid shaded areas indicated full blockages. (c) Shrunk-2-D result showing reduced placement density over partial blockages. (d) Post-partitioned result.

problem 1) above. We can also assign the original memory timing model to this filler-size cell. The tool sees all the pins, and a consistent LIB timing model, so this also takes care of problem 3) above.

Problem 2) is all that remains to be solved. As we have shrunk down the size of the memory, if we superimpose them, the tool thinks that either tier can be used for standard cell placement. To convey accurate whitespace information, we can use a combination of full and partial placement blockages. Consider the example of preplaced memory in Fig. 2(b). Regions with two memories overlapping can not have cells placed in either tier, so they can be represented as a full placement blockage. Regions with only one memory present can have cells in one tier, but not the other, and we use partial placement blockages to represent this. This allows us to specify a maximum placement density over a given region. If we specify a maximum density of half the target density of the chip, we can achieve our target [Fig. 2(c)].

2) Auto-Placing Memory: To achieve 2-D tool-assisted 3-D macro placement, the main challenge to overcome is that commercial 2-D tools have no concept of the third dimension in their in-built engines. An overview of how this 2-D engine can be leveraged to design M3D ICs is shown in Fig. 3.

We shrink the memory macros by 0.707 on each side, run through the automatic placement, and expand the memories back to their original area as shown in Fig. 3(a). This gives suitable (x, y) locations for each macro. The next step is to manually color the overlapping memories such that each color can be placed onto a separate tier to remove any overlap [Fig. 3(b)]. This is then split into different tiers and legalized manually as shown in Fig. 3(c) and (d). This gives preplaced memory locations, similar to Fig. 2(b) that can then be used to run through the rest of the flow.

C. Shrunk-2-D Place-and-Route Flow

The shrunk technology files and memory related pins and blockages can now be fed into Cadence encounter. We can then use this commercial tool to run through all design stages, all the way through CTS and post-route optimization. The benefit of doing optimization in such a fashion is that the tool can see the entire logic path, performing logic transformations as needed. This is not true for tier-by-tier optimization, where deriving timing constraints is a challenge, and even then, the tool cannot see the entire path.

D. Obtaining 3-D Design

As the MIV itself is of the size of a local via [2], has negligible cost, and can be ignored during timing optimization, we can assume that the shrunk 2-D solution represents the best true M3D placement solution that has been flattened down to a single tier by stripping the z dimension. However, to retain the quality of the Shrunk 2-D design, we need to split the logic into multiple tiers such that the change in (x, y) location is minimized. This can be done in a similar fashion to [8]. We define regular partitioning grids, and perform a global min-cut while ensuring area balance within each grid. A smaller gird size ensures less perturbation, but implies more MIVs (as the min cut is less effective). This could lead to longer routing detours to find suitable whitespace for MIVs. The sensitivity of wirelength to bin size is a few percent [8], and a reasonable tradeoff is obtained for bins with width of $10 - 20$ μm.

During this partitioning process, the cells in the clock tree can be handled separately to ensure minimal skew. One way of doing this is to first fix all the clock tree cells in one tier, and then later partition the regular logic. The advantage of this method is minimal changes from the Shrunk-2-D design, both in terms of clock buffer placement and clock tree routing. However, as will be seen in Section III, this is not always possible. In such cases, clock cells can be partitioned in a similar fashion to regular logic.

Once the locations of all cells are determined, MIVs need to be inserted into whitespace locations. This can be done by tricking a commercial router as in [8]. Essentially, the 2-D metal stack is duplicated, and the cell pins are moved to different metal layers depending on the tier in which it is placed. Since commercial routers are capable of routing to pins on multiple metal layers, this setup is just sent through routing, and the routing topologies are traced to get the MIV locations. With these locations, separate verilog and design exchange format (DEF) files are created for each tier, so that they can be opened in independent design windows for signoff routing. Note that for each tier, MIVs appear as ports on a given metal layer. This entire process is illustrated in Fig. 4.

E. Timing and Power Analysis

Once the MIV locations are determined, each tier is first trial routed and estimates of parasitics for each tier are dumped. The netlist for each tier, along with its parasitics is then fed into Synopsys PrimeTime. In addition, a top-level netlist and

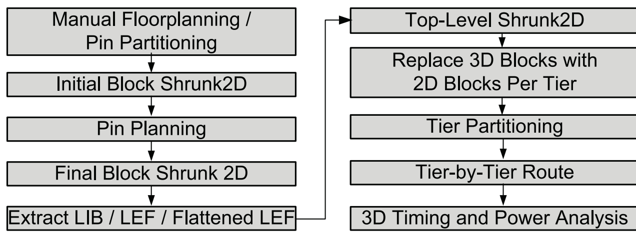

Fig. 5. Shrunk-2-D flow for designing a full-chip SoC.

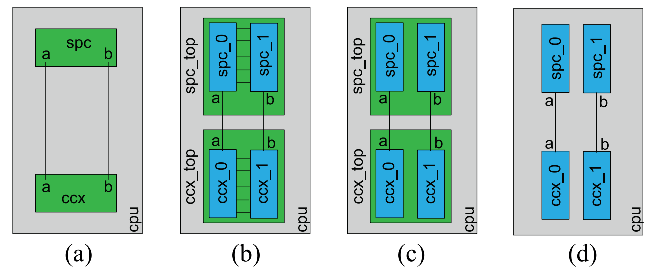

Fig. 6. Netlist generation for top-level pin planning. (a) Initial synthesis netlist. (b) Each block replaced with the netlist obtained after initial Shrunk-2-D. (c) Intrablock MIVs deleted. (d) One-level of hierarchy for each block removed.

parasitic file is created that contains the MIV connectivity and parasitics. We then perform an initial timing analysis to derive timing constraints for each tier. With these timing constraints, we go back to each tier, and run timing-driven routing. The real sign-off parasitics for each tier are then fed back into PrimeTime to get the final timing and statistical power simulation numbers.

III. Designing M3D SoCs

We have so far discussed how to design a single block in M3D. However, today's chips are complex systems, and involve several blocks, not all of which may benefit from 3-D. If every single block was 2-D, and the top-level implementation consisted of just floorplanning, each block could just be treated as a hard macro, and the methodology of Section II could directly be applied. However, designing a system with a mixture of 2-D and 3-D blocks introduces additional complexity that needs to be handled. More specifically, the new problems that arise while still needing to fit into the existing commercial 2-D IC framework are as follows.

1) Determine the pin locations of each 2-D and 3-D block so as to minimize the top-level routed wirelength.
2) Enable the shrunk 2-D flow to recognize 3-D hard macros and the corresponding intertier timing arcs during timing optimization and clock tree synthesis.
3) MIV planning and partitioning need to be modified to account for 3-D blocks.
4) Perform final tier-by-tier routing using a commercial tool with real 2-D foundry technology files where a tier can contain part of a 3-D block.
5) Perform extraction, parasitic back-annotation, and final timing and power analysis on a design where MIVs exist both within blocks as well as between them.

We make several assumptions relating to the design of the full-chip. First, we assume that the chip is floorplanned manually. Next, we assume that a bottom-up design style is followed, where timing budgets of each block are predetermined, each block closed separately, and then assembled at the top-level. Both these assumptions are valid for large SoCs, where only a handful of blocks are arranged manually at the top-level, and the timing paths between blocks are just direct register to register communication.

A. Overall Design Flow

The overall design flow is shown in Fig. 5. The first step is to decide which blocks are to be implemented in 3-D and decide on a manual floorplan. Guidelines on how to decide this are presented in Section V. In addition, the tier of all the block-pins needs to be decided. For a 3-D block, we can fix all the pins onto a single tier, or divide them onto all tiers. In this paper, we divide the pins such that the number of interblock nets is roughly equal on each tier. Note that this step does not decide the pin locations, only their tier.

Once the pin partitions for each block are known, we assign an initial random pin location, and go through a rough initial Shrunk-2-D for each block. This is to facilitate pin planning, and is explained in more detail in Section III-B. The pin planning step gives final pin locations, after which each 3-D block can go through a full Shrunk-2-D flow.

The next step is to extract LIB timing models for all blocks, which can be done from within Synopsys PrimeTime after timing the 2-D/3-D blocks. In addition, abstracts of the block in library exchange format (LEF) are extracted for each tier of the block separately from Cadence encounter. In addition, we create a "Flattened LEF" which is explained along with top-level Shrunk-2-D in Section III-C.

Section III-C also explains the next step in the flowchart, where the 3-D block is abstracted to look like a 2-D block in each tier. The output of this is a design that looks like what we see in Section II. The top-level now sees only a set of preplaced 2-D blocks and a Shrunk-2-D result corresponding to this. We can then simply reuse the flow developed in Section II-D to partition the design and route the top-level of each tier. Timing and power analysis of the entire SoC requires some special consideration, and is discussed in Section III-D.

B. Pushing Down Block Pin Locations

Given a set of pin partitions, the objective of this step is to find suitable locations for each pin along the block boundary such that the top-level wirelength and congestion are minimized. The first step is to generate a top-level netlist reflecting the pin partitions, and one technique is shown in Fig. 6.

We first start with a synthesis netlist, and a sample netlist with two blocks to be implemented in 3-D is shown in Fig. 6(a). The actual block and design names will be explained in Section IV. For simplicity, let each block have only two pins—"a" and "b." After manual pin-partitioning, assume that pin "a" of both blocks will be assigned to tier 0 and pin "b" of each block will be assigned to tier 1.

A random pin location for "a" and "b" are chosen, and each block is made to go through an initial Shrunk-2-D flow.

This need not include any CTS or timing optimization, as the only purpose is to obtain an initial 3-D netlist and partition for each block. The block netlists of Fig. 6(a) are then replaced with the initial 3-D netlists [Fig. 6(b)]. We then remove all intrablock 3-D connections [Fig. 6(c)], and then flatten one level of hierarchy as shown in Fig. 6(d).

From the top-level perspective, we now have a block-level netlist where each block is 2-D, and the partitions and locations of each block are known. The locations and connectivity of the intrablock MIVs are not relevant at the top-level. For such a block-level system, an iterative method to assign suitable block-pin locations was presented in [7]. Essentially, the block pins are initially assumed to be in the center of each block, and initial MIV locations for each interblock net is determined to be the center of its half-perimeter bounding box. With these MIV locations, separate verilog and DEF files are created for each tier so that they can be opened in Cadence Encounter. We can then use Encounter's internal pin planner to assign block pin locations based on connectivity to MIVs and other blocks. These block pin locations are used to drive the MIV planner of Section II-D to obtain new MIV locations, and the process is repeated again. The block pin locations converge fairly quickly, within a couple of iterations. These serve as the final block pin locations for full block Shrunk-2-D.

C. Handling 3-D Blocks During Shrunk-2-D Flow

After each block has gone through its own implementation, we need to perform top-level Shrunk-2-D to take care of interblock buffering and CTS. The main difference from Section II is that we now have a mix of 2-D and 3-D blocks.

After each block is implemented in 3-D (say spc), we extract LEF abstracts of each tier (spc_0 and spc_1) that contain the outline and pin locations. This also contains the block MIV locations (as from the blocks perspective, they are ports on each tier), and we delete them as they are not relevant to the top-level. The netlist of Fig. 6(d) contains all the interblock connections, along with only 2-D blocks. If we were to plug in this netlist along with the block tier LEFs, we could directly use the flow of Section II. However, there is one complication—the timing model of the block. Although spc_0 and spc_1 can be separated physically, they are still logically one block. There are timing arcs from pins on spc_0 to spc_1 and vice-versa that need to be captured. In addition, the clock input to the block is on one tier, so there will be a timing relationship between the pins on the other tier to the clock pin.

We can extract the timing model of the block in LIB format when we perform 3-D timing analysis using PrimeTime. We now need to reconcile a logical block that has all the block-pins, and the physical abstraction, where the pins are divided between two different LEF files. We do this by creating a Flattened LEF file for a 3-D block, where the block pins in both tiers are collapsed into a single tier. This is essentially stripping the z dimension, where we add all the pins from both tiers into a single LEF file.

Now, we use the netlist of Fig. 6(a), along with the flattened LEF file to go through the Shrunk-2-D portion of the flow of Section II. Note that for 3-D blocks, we do not really need to separate the pins and placement blockage component as we did for the memories, and we can directly use the flattened LEF file without changing its footprint. This is because no other block can overlap with a 3-D block in a different tier. In contrast, all 2-D blocks go through an identical process as the memories did in Section II-B.

Once Shrunk-2-D is completed, we still need to go through the process of partitioning top-level cells, MIV planning, etc. We can directly leverage all existing scripts developed by making a simple change to the netlist at this stage. For the partitioning stage, we swap the netlist of Fig. 6(a) to that of Fig. 6(d), while feeding it the (x, y) locations of all top-level cells obtained from Shrunk-2-D. This can be done because the partitioner does not need timing information. We can then proceed with MIV planning and tier-by tier routing as usual.

D. Timing and Power Analysis

Timing and power analysis is very similar to Section II-E, except that a few additional netlisting steps need to be performed. We obtain the parasitics for each block during block-design, and for the top-level nets during top-level design. We also can create an intertier SPEF for each block and the top-level. The actual netlist read into Synopsys PrimeTime is that of Fig. 6(b), and we just need to ensure that each net gets correctly annotated. Once this is done, we can proceed with 3-D timing and power analysis.

IV. Experimental Results

We implement all scripts and tools in python, tcl, and C/C++. Our target benchmark is the core/cache subsystem of the OpenSPARC T2 SoC [9] implemented in a foundry 28 nm fully depleted silicon on insulator (FDSOI) process. The MIV diameter is assumed to be 100 nm, with a resistance of 16 Ω, and a capacitance of 0.1 fF. Each tier in our design is built using six metal layers, and we assume copper on both tiers, which corresponds to a mature fabrication process. This assumption is reasonably valid as Panth *et al.* [7] have demonstrated that performance degradation can be overcome if one tier is degraded due to a poor fabrication process. According to [2], if the thickness of the intertier dielectric is greater than 100 nm, we can ignore intertier coupling, and hence we ignore such coupling in this paper. Therefore, the parasitics of a given 3-D net will be the sum of the extracted parasitics in each tier, plus the MIV parasitics. Since we lack a memory compiler for this technology, we scale down both the size and timing/power characteristics from a 130 nm foundry library. The frequency of the system is 870 MHz, which was the fastest we could close the entire SoC with the default commercial 2-D flow. All the analysis presented in this section is iso-performance, where we compare the power consumption of 2-D and 3-D SoCs.

In the rest of this section, we first present an overview of the structure of our target benchmark. Next, we correlate the wirelength, timing, and power results between Shrunk-2-D and M3D. We then study the benefits of a single block being implemented in 3-D, and finally we compare the benefits of different approaches to designing an M3D SoC.

TABLE I
CORRELATION BETWEEN SHRUNK-2-D AND M3D

	Routed WL (m)	Capacitance (pF) Wire	Pin	Total	Power (mW) Cell	Net	Leakage	Total
ccx								
Shrunk-2D	4.83 (1.00)	737.42 (1.00)	994.94 (1.00)	1732.35 (1.00)	213.30 (1.00)	252.80 (1.00)	0.72 (1.00)	466.90 (1.00)
M3D	4.87 (1.01)	603.35 (0.82)	994.94 (1.00)	1598.29 (0.92)	211.90 (0.99)	239.90 (0.95)	0.72 (1.00)	452.50 (0.97)
spc								
Shrunk-2D	5.52 (1.00)	752.66 (1.00)	589.98 (1.00)	1342.64 (1.00)	196.20 (1.00)	214.50 (1.00)	0.68 (1.00)	411.30 (1.00)
M3D	5.75 (1.04)	697.83 (0.93)	589.98 (1.00)	1287.81 (0.96)	195.50 (1.00)	207.10 (0.97)	0.68 (1.00)	403.20 (0.98)

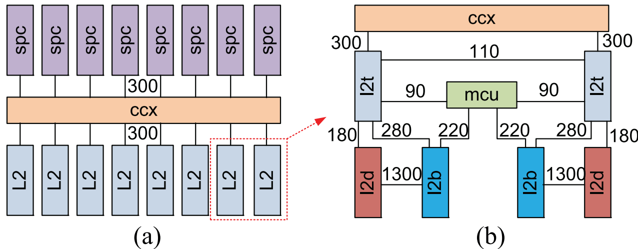

Fig. 7. Interblock connections in the OpenSPARC T2 subsystem. The width of buses are shown. (a) Core/cache subsystem of the OpenSPARC T2 processor. (b) Detailed view of the cache subsystem.

A. Overview of Target Benchmark

A rough overview of the various interblock connections in the core/cache subsystem of the OpenSPARC T2 SoC are shown in Fig. 7. It consists of six blocks—the SPARC core (spc), the cache crossbar (ccx), L2 tag (l2t), L2 bank (l2b), L2 data (l2d), and the memory controller unit (mcu). In this paper, we ignore any analog and PHY-layer related blocks. As can be seen from Fig. 7(a), this system consists of eight sets of spc and L2 related modules connected to the ccx. The L2 modules actually come in pairs, as can be seen in Fig. 7(b). The l2t modules are the only ones that connect to the ccx, and two l2t modules share a mcu. The l2b and l2d are heavily interconnected, and each communicate with their respective l2t. In addition, the mcu communicates with the pair of l2b modules. Pairs of L2 modules are independent units, and do not connect to other pairs. This interconnection structure motivates our floorplan choices in Section IV-D.

B. Correlation Between Shrunk-2-D and M3D

As was mentioned in Section II, we perform several technology scaling operations, then perform place and route, CTS, and buffer insertion using these scaled dimensions. However, we do not change the parasitics tech files, assuming that the parasitics per-unit length do not change from the unscaled dimensions. We then simply reuse the placement and buffering result while converting it to an M3D design. Final extraction, timing, and power analysis are performed using foundry provided technology files, and are signoff quality.

Final analysis always reflects the M3D design we have. However, this does not imply that the buffering and CTS are optimal. For this to happen, the wirelength, parasitics, timing and power of Shrunk-2-D, extracted using shrunk dimensions, should match that of M3D. This is the objective of reusing the 2-D capacitance tables, but the tool may behave differently. For example, if Shrunk-2-D predicts 50% lower parasitics than M3D, the design will be under-buffered, and the final M3D design will not meet timing. In contrast, if Shrunk-2-D predicts 50% higher parasitics than M3D, the final M3D design will still meet timing, but will be overbuffered, and will have higher power than necessary.

We pick the two logic dominated blocks (spc and ccx) and tabulate statistics for Shrunk-2-D and M3D in Table I. First, we note that the M3D wirelength is slightly higher than Shrunk-2-D, which is to be expected as zero disturbance is not guaranteed. We observe that Shrunk-2-D sees up to 18% lower wire capacitance, which translates to up to up to 8% lower total capacitance. In turn, this translates to up to 5% and 3% additional net power and total power reduction in M3D compared to Shrunk-2-D.

TABLE II
CLOCK TREE PARTITIONING OPTIONS

	Clock WL (m)	# Clock MIV	Clock Power (mW)	Clock Skew (ps)
ccx				
Shrunk-2D	0.42 (1.00)	-	79.60 (1.00)	324.73 (1.00)
Fixed Buf.	0.42 (1.01)	4968.00 (1.00)	81.90 (1.03)	231.77 (0.71)
Part. Buf.	0.43 (1.02)	5025.00 (1.01)	82.70 (1.04)	273.12 (0.84)
spc				
Shrunk-2D	0.20 (1.00)	-	41.30 (1.00)	198.92 (1.00)
Fixed Buf.	0.21 (1.04)	1431.00 (1.00)	43.10 (1.04)	171.20 (0.86)
Part. Buf.	0.21 (1.05)	1522.00 (1.06)	43.50 (1.05)	234.09 (1.18)

We also tabulate certain clock related metrics in Table II. This table includes both clock buffer partitioning options mentioned in Section II-D. The row corresponding to the Fixed Buf. case is where we fix all clock buffers onto one tier and only partition the flip-flops, and Part. Buf. is the case where both clock buffers and flip flops are partitioned across tiers. From this table, we observe that fixing the clock buffers on one tier gives both lower clock power and skew. Therefore, we only partition buffers when there is no other choice. For example, if we are fixing the buffers on tier 0, but there happens to be a memory cell placed in tier 0 in the same spot, we move that buffer to tier 1.

Overall, we observe that the correlation is not perfect, but acceptable pending further study. Since Shrunk-2-D sees higher parasitics than M3D, our M3D designs are likely slightly over-buffered, and power savings numbers are a little pessimistic. Tuning the tech files to obtain perfect correlation is beyond the scope of this paper.

C. Designing Single M3D Block

We now take each of the blocks in the T2 SoC, and implement them in M3D using the Shrunk-2-D design flow. The footprints of each the 2-D blocks come from the full-chip 2-D

TABLE III
ENCOUNTER 2-D VERSUS M3D POWER FOR EACH BLOCK

	Memory		Clk Network		FF Clk Pin		FF Out Pin		Comb.		Cell		Net		Leakage		Total	
	E2D	M3D	E2D	M3D	E2D	M3D	E2D	M3D	E2D	M3D	E2D	M3D	E2D	M3D	E2D	M3D	E2D	M3D
Logic Dominated Blocks																		
ccx	0.0	0.0	78.8	81.9	61.5	60.9	108.4	106.6	251.8	203.2	228.2	211.9	270.2	239.9	2.140	0.723	500.5	452.5
spc	30.7	27.3	51.1	43.1	51.6	51.5	105.6	103.4	230.0	177.9	212.7	195.5	255.3	207.1	0.942	0.679	468.9	403.2
Geo-Mean	30.7	27.3	63.5	59.4	56.3	56.0	107.0	105.0	240.7	190.1	220.3	203.5	262.6	222.9	1.419	0.700	484.4	427.1
Norm	1.000	0.889	1.000	0.936	1.000	0.994	1.000	0.981	1.000	0.790	1.000	0.924	1.000	0.849	1.000	0.493	1.000	0.882
Memory Dominated Blocks																		
l2b	11.6	11.5	7.7	9.3	12.1	12.1	9.9	8.8	41.3	39.3	36.1	36.7	46.3	44.1	0.202	0.140	82.6	80.9
l2t	15.6	15.5	26.2	23.7	44.8	44.5	31.4	30.4	64.2	59.6	97.0	93.3	85.0	80.2	0.218	0.148	182.2	173.6
mcu	0.7	0.7	5.3	6.0	5.5	5.5	9.3	9.2	6.3	5.9	14.2	15.0	12.9	12.3	0.032	0.033	27.1	27.3
l2d	16.2	14.3	8.1	7.7	7.5	7.6	9.9	14.0	34.7	26.5	34.8	32.9	41.5	37.2	0.205	0.094	76.4	70.2
Geo-Mean	6.8	6.4	9.7	10.1	12.2	12.2	13.0	13.6	27.6	24.6	36.3	36.1	38.1	35.7	0.131	0.090	74.7	72.0
Norm	1.000	0.953	1.000	1.041	1.000	1.001	1.000	1.047	1.000	0.891	1.000	0.994	1.000	0.936	1.000	0.688	1.000	0.964

floorplan that will be discussed in Section IV-D. The footprints for the 3-D blocks in this section are obtained by just scaling the width and height of each block by a factor of 0.707. We categorize the blocks into logic and memory dominated blocks, and tabulate the various components of 2-D and M3D power in Table III.

We first divide the power into memory output nets (memory), clock buffers (Clk Network), internal power at the clock pin of flops and memory (FF Clk Pin), power at the output of the flops (FF Out Pin), and the combinational power (Comb). The major source of power reduction is the combinational power, which benefits logic-dominated blocks more, where we see an average of 21% reduction, leading to an overall power reduction of 11.8%. However, memory dominated blocks benefit very little from this, and we see only an average of 3.6% power reduction.

We also observe that not all components of power reduce when going from 2-D to M3D. In memory dominated blocks, the placement of the memory macros and CTS settings plays a large role in determining the amount of power reduction. The l2d block contains the cache banks, and the M3D design has a very different placement of memory macros, which reduces the optimality of the flip-flop placement, leading to a marginally larger FF Clk pin and FF out pin power. Our CTS settings allow large clock buffers driving large fanout and loads, which implies that comparatively few buffers are present in the network. Even minor changes in the slew of clock nets has a large impact on the internal power of the clock buffers, and M3D does not always offer huge benefit under the CTS settings we have chosen. Tuning the CTS settings is possible to demonstrate greater M3D benefit, but we must be careful to choose the settings that offer the best 2-D IC result as well.

The second classification of power we make is into cell, net, and leakage power. The cell power reduction comes from reduced buffers, and the net power reduction comes from lower wirelength. Leakage reduction comes primarily from using more devices with a higher threshold voltage (V_t) to meet the same target frequency. As expected, we see some cell power reduction, while the net power reduction is larger (average of 15.1%) in logic dominated blocks. However, we see enormous amounts of leakage power reduction of 50.7% in logic dominated and 30.2% in memory dominated blocks. This is because leakage power has an exponential relationship to V_t. A low V_t cell has roughly 10× the leakage of a regular V_t cell, so converting even a few low V_t to regular or high V_t has a large impact on leakage. However, in this paper, we report power at full load, i.e., every region of every block is on. Real systems have a significant amount of dark silicon, which means that the total power reduction in M3D will go up significantly.

D. Monolithic 3-D SoC Design

We now design the T2 SoC in 2-D, and with three different M3D floorplan options, as shown in Fig. 8. The 2-D floorplan is quite similar to actual silicon. The M3D floorplans are motivated by the connection structure of Fig. 7. The first floorplan option is "logic on memory," where all blocks are 2-D, and floorplaned in 3-D. Here, we keep all logic dominated blocks on one tier, and all memory dominated blocks on another. The l2t and spc are kept as close to ccx as possible to minimize top-level interconnect. We place the mcu module in between the pairs of l2t and l2b modules to minimize wirelength. The second floorplan option ("logic folded") is where we fold the logic dominated blocks (spc and ccx), and floorplan these along with the memory dominated blocks implemented in 2-D. The floorplan flexibility is reduced, as a 3-D block uses up silicon in both tiers. Finally, the third floorplan option is "all folded," where we simply scale the location and dimension of all blocks in the 2-D floorplan by 0.707. In terms of ease of floorplanning, this is the easiest, as we can mimic existing 2-D floorplans, or even utilize existing 2-D floorplanning algorithms and then simply scale the result. It should also be noted that all floorplan options here have exactly the same total silicon area.

The overall power reduction for each of the floorplans is shown in Table IV. From this table, the general trend is that the more folded blocks we have, the more wirelength reduction and total power reduction we achieve. As in the block-level case, M3D power reduction comes primarily from net power reduction, with a little cell power reduction as well. In addition, we see huge leakage power savings. Therefore, we see a 8% reduction in the peak power of the chip, and this number will only go up when actual workloads are considered. We also use a significant number of MIVs, and the MIV maps for all floorplan options are shown in Fig. 9.

Although we say that more folded blocks lead to more power reduction, the reality is design dependent. Fig. 10(a) shows the percentage of power that each block

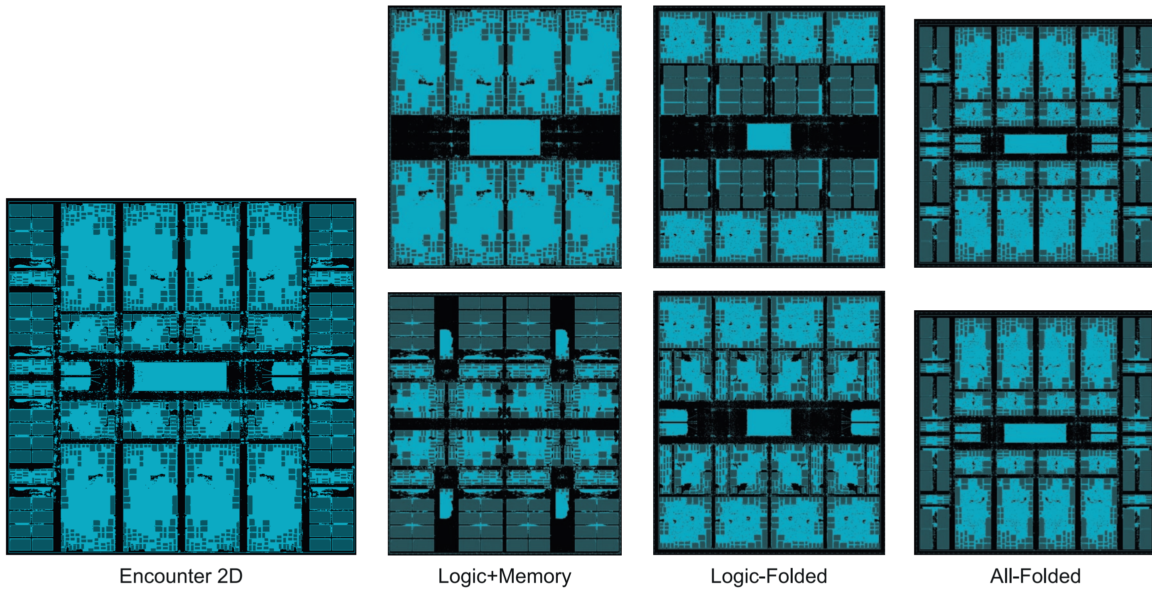

Fig. 8. Full-chip floorplans for 2-D and different M3D options of the OpenSPARC T2 SoC. The footprint for encounter 2-D, logic+memory, logic-folded, and all-folded layouts are $3.9 \times 4 = 15.6$ mm^2, $2.6 \times 3 = 7.8$ mm^2, $2.6 \times 3 = 7.8$ mm^2, and $2.76 \times 2.83 = 7.8$ mm^2, respectively.

TABLE IV
POWER RESULTS FOR THREE DIFFERENT M3D IMPLEMENTATIONS OF THE OPENSPARC T2 SOC

	Routed WL (m)	# MIV ($\times 10^3$)	Capacitance (nF)			Power (W)			
			Wire	Pin	Total	Cell	Net	Leakage	Total
Encounter 2D	99.54 (1.00)	-	11.76 (1.00)	6.71 (1.00)	18.46 (1.00)	2.94 (1.00)	2.74 (1.00)	0.016 (1.00)	5.69 (1.00)
3D - Logic + Mem	95.15 (0.96)	4.21 (1.00)	11.39 (0.97)	6.70 (1.00)	18.09 (0.98)	2.89 (0.98)	2.70 (0.99)	0.014 (0.89)	5.61 (0.98)
3D - Logic Folded	82.80 (0.83)	712.64 (169.47)	9.74 (0.83)	6.33 (0.94)	16.07 (0.87)	2.81 (0.96)	2.40 (0.88)	0.012 (0.76)	5.22 (0.92)
3D - All Folded	76.61 (0.77)	838.36 (199.37)	8.85 (0.75)	6.17 (0.92)	15.02 (0.81)	2.76 (0.94)	2.26 (0.83)	0.010 (0.66)	5.03 (0.88)

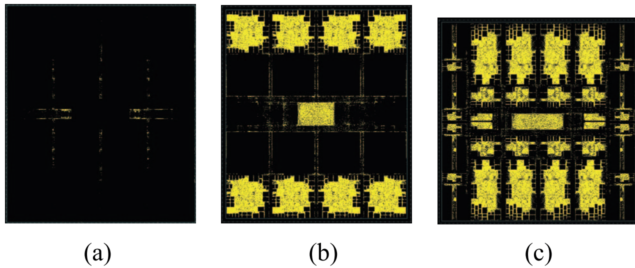

Fig. 9. MIV maps for different M3D floorplans of the T2 SoC. (a) Logic+Memory #MIV = 4205. (b) Logic-Folded #MIV = 712 640. (c) All-Folded #MIV = 838 360.

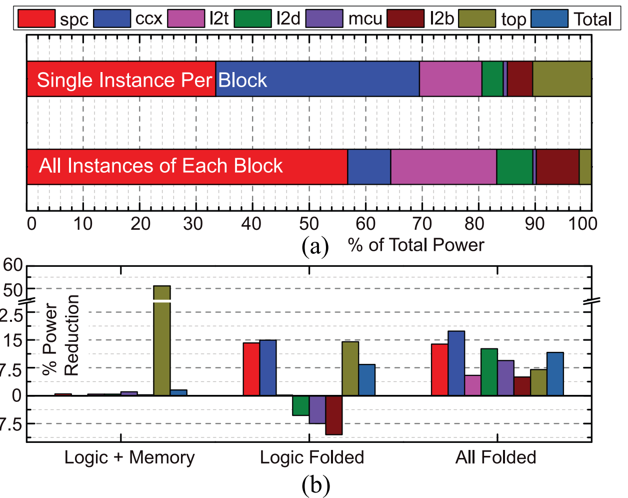

Fig. 10. (a) Percentage of full-chip power that each block consumes in the T2 SoC. (b) Power reduction % of each block for different floorplans.

consumes with respect to the total power of the 2-D design. The upper bar represents the hypothetical case where the SoC consists of only one instance of each block. Here, we observe that the spc and ccx contribute roughly equal portions of power, with the l2t and top-level close behind. However, when we consider that there are eight spc and l2t modules, and only one ccx and top-level, we see that the total power is now heavily influenced by what happens to spc and l2t.

Fig. 10(b) shows the power reduction of each block for each of the three M3D floorplan options. For the logic on memory floorplan, we see no significant change to the block power numbers, but huge reduction in the top-level power. However, as the top-level power is not a significant contributor to the total power, we do not see much total power savings. Next, for the logic folded case, we see large power savings for spc, ccx, and top-level, but this is offset by increased power in l2t. This is because of changes to the block shape and pin locations. Finally, the all folded case has the least top-level power reduction, but since it is skewed toward spc, l2t, and ccx power, and all those modules see improvements in power, we see the largest overall power savings.

V. DESIGN LEARNINGS AND GUIDELINES

This section presents several design guidelines based on the insight gained in previous sections. We first discuss how to predict which blocks are expected to show 3-D benefit, and

then we discuss how to decide on a good M3D SoC floorplan. First, when designing a single block in M3D, the wirelength savings over 2-D are fairly consistent irrespective of block type. However, how this wirelength savings translates to block power savings depends on several factors. First, blocks with lesser memory give more benefit. Next, blocks with fewer flops or heavily wire dominant ones are expected to give more benefit. Finally, if a block is expected to be in rest mode for a significant portion of the time, we can expect even larger power savings as leakage power becomes more dominant.

Next, when choosing an M3D SoC floorplan, there are several factors that need to be considered.

1) Fold blocks that have a higher chance of power reduction, i.e., logic dominated blocks.
2) Fold those blocks that are instantiated multiple times at the top level and hence contribute more to total power.
3) Generally, folding more blocks implies lesser top-level power saving. This tradeoff needs to be evaluated.
4) After floorplanning, it is better to keep the tier with more whitespace as the tier that will contain the MIV landing pads. This is because MIVs will land on the top-metal layer of this tier, and will be routed over blocks, where routing resources are very limited.

VI. Conclusion

In this paper, we have presented an netlist-to-layout design flow that produces M3D SoCs that show power benefits when compared to their 2-D counterparts designed with commercial tools. We have demonstrated that 2-D commercial tools can be used along with enhancements and scripts to produce M3D blocks. We have used the Oracle OpenSPARC T2 designed in a 28 nm FDSOI process as a case study, and used it to demonstrate the benefits of the proposed approach, and also to provide several design guidelines that make the results presented here general enough to be applicable to other designs.

References

[1] X. Dong, J. Zhao, and Y. Xie, "Fabrication cost analysis and cost-aware design space exploration for 3-D ICs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 29, no. 12, pp. 1959–1972, Dec. 2010.

[2] P. Batude *et al.*, "3-D sequential integration: A key enabling technology for heterogeneous co-integration of new function with CMOS," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 2, no. 4, pp. 714–722, Dec. 2012.

[3] S.-M. Jung, H. Lim, K. H. Kwak, and K. Kim, "A 500-MHz DDR high-performance 72-Mb 3-D SRAM fabricated with laser-induced epitaxial c-Si growth technology for a stand-alone and embedded memory application," *IEEE Trans. Electron Devices*, vol. 57, no. 2, pp. 474–481, Feb. 2010.

[4] T. Naito *et al.*, "World's first monolithic 3-D-FPGA with TFT SRAM over 90nm 9 layer Cu CMOS," in *Proc. IEEE Int. Symp. VLSI Technol.*, Honolulu, HI, USA, Jun. 2010, pp. 219–220.

[5] S. Bobba *et al.*, "CELONCEL: Effective design technique for 3-D monolithic integration targeting high performance integrated circuits," in *Proc. Asia South Pac. Design Autom. Conf.*, Yokohama, Japan, Jan. 2011, pp. 336–343.

[6] Y.-J. Lee, P. Morrow, and S. K. Lim, "Ultra high density logic designs using transistor-level monolithic 3-D integration," in *Proc. IEEE Int. Conf. Comput.-Aided Design*, San Jose, CA, USA, Nov. 2012, pp. 539–546.

[7] S. Panth, K. Samadi, Y. Du, and S. K. Lim, "Power-performance study of block-level monolithic 3-D-ICs considering inter-tier performance variations," in *Proc. ACM Design Autom. Conf.*, San Francisco, CA, USA, 2014, pp. 1–6.

[8] S. Panth, K. Samadi, Y. Du, and S. K. Lim, "Placement-driven partitioning for congestion mitigation in monolithic 3-D IC designs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 4, pp. 540–553, Apr. 2015.

[9] *Oracle OpenSPARC T2*. Accessed on May 2014. [Online]. Available: http://www.oracle.com

Shreepad Panth (S'11–M'15) received the B.S. degree from Anna University, Chennai, India, in 2009, and the M.S. and Ph.D. degrees from the Georgia Institute of Technology, Atlanta, GA, USA, in 2011 and 2015, respectively.

He is currently a Design Engineer and a Technical Staff Member with Altera Corporation, San Jose, CA, USA. He has authored over 20 publications. His current research interests include aspects of physical design for current and next generation 3-D ICs.

Dr. Panth was a recipient of the Best Paper Award at ATS'12 and IITC'14, and nominations for the Best Paper Award at ISPD'14 and DAC'14.

Kambiz Samadi (S'04–M'12) received the M.Sc. and Ph.D. degrees from the University of California at San Diego, San Diego, CA, USA, in 2007 and 2010, respectively.

He joined Qualcomm Research, San Diego, CA, USA, in 2011, where he is currently a Staff Research Engineer. He has over 25 publications in refereed journals and conferences. His current research interests include on-chip interconnection modeling and optimization for system-level design, 3-D IC modeling and optimization, very large scale integration design manufacturing interface, 3-D IC electronic design automation solutions, and 3-D IC architecture-level design space explorations.

Dr. Samadi was a recipient of the Best Paper Award and nominated for two best paper for his journals and conferences.

Yang Du (M'96) received the Ph.D. degree from Columbia University, New York, NY, USA, in 1994.

He held engineering positions with Analog Devices, Norwood, MA, USA, AMD, Sunnyvale, CA, USA, Motorola, Schaumburg, IL, USA, and Qualcomm, San Diego, CA, USA. He is currently a Director of Engineering with Qualcomm Research, where he leads a team in advanced nano-technology and semiconductor research. He has authored over 50 patents/patent publications and numerous conference/journal papers. His current research interests include span emerging semiconductor devices, predictive device and circuit modeling, novel very large scale integration (VLSI) circuits and architecture, next generation 3-D IC technology and design, 3-D VLSI circuit, architecture and system integration, design automation, thermal modeling, and thermal aware design methodologies.

Sung Kyu Lim (S'94–M'00–SM'05) received the B.S., M.S., and Ph.D. degrees from the University of California at Los Angeles, Los Angeles, CA, USA, in 1994, 1997, and 2000, respectively.

He joined the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA, in 2001, where he is currently the Dan Fielder Professor of Electrical and Computer Engineering. His current research interests include architecture, circuit design, and physical design automation for 3-D ICs. His research on 3-D IC reliability is featured as Research Highlight in the Communication of the ACM in 2014 and has authored *Practical Problems in VLSI Physical Design Automation* (Springer, 2008).

Dr. Lim was a recipient of the Best Paper Award from TECHCON'11, TECHCON'12, ATS'12, and IITC'14, and also nominated for the Best Paper Award at ISPD'06, ICCAD'09, CICC'10, DAC'11, DAC'12, ISLPED'12, and DAC'14. He is an Associate Editor of the IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems and the IEEE Design & Test of Computers. He was a member of the Design International Technology Working Group of the International Technology Roadmap for Semiconductors.