Placement and Routing in 3D Integrated Circuits

Cristinel Ababei, Yan Feng, Brent Goplen, Hushrav Mogal, Tianpei Zhang, Kia Bazargan, and Sachin Sapatnekar
University of Minnesota

Editor's Note:
Advanced manufacturing and packaging techniques are permitting a glimpse at the near-future, where wires can go in three dimensions, and ICs made in diverse processes can be assembled together—sandwich like—to achieve ever higher levels of integration. This article focuses on a specific CAD problem in connection with 3D ICs: how to partition, place, and wire a design subject to various constraints on power, timing, and manufacturability.
—Sani Nassif, IBM

In the everyday world, high-rise buildings are the solution to the problem of accommodating large populations in areas of prime real estate. In addition to permitting high population densities, such an arrangement also reduces the "interconnect bottleneck" that would come with the road network associated with an equivalent set of low-rise buildings, distributed over a larger area.

The silicon world is not much different; the need to densely pack circuits and locate critical blocks as close as possible to each other have led to the advent of 3D technologies with multiple tiers of devices stacked atop each other. The increased packing density improves the computation per unit volume and results in diminished on-chip interconnect problems because it reduces parasitic capacitances. A 3D design curtails parasitics by reducing the average interconnect lengths (in comparison with 2D implementations, for the same circuit size), as well as by denser integration, which results in the replacement of chip-to-chip interconnections by intra-chip connections.

Consequently, 3D integration can be an enabler for enhancements in system performance, power, reliability, and portability. Advances in industrial,[1] government,[2] and academic[3] research laboratories have demonstrated 3D designs with inter-tier separations on the order of a few microns. Recently, the Massachusetts Institute of Technology's Lincoln Laboratory has offered a MOSIS-like 3D integration program under the auspices of DARPA.

Fundamentally, the problem of 3D design relates to the topological arrangements of blocks, and physical design therefore plays a natural role in determining the success of 3D design strategies. Physical design in the 3D realm requires a fresh approach as new cost functions and new design structures become important, and ordinary extensions of 2D approaches are unequal to the task of solving these problems.

Here, we describe CAD techniques for placement and routing in 3D ICs, developed under our 3D Analysis and Design Optimization framework, which we call 3D-Adopt for short. These approaches address a dichotomy of design styles, both FPGA and ASIC.

The factors that are important in each style are different, so that a one-size-fits-all approach is impractical, and therefore, we present separate approaches for 3D physical design for each of these technologies. For example, thermal issues are much more important in ASIC designs than in FPGA architectures since the power densities in the former are higher. This is because both the operating clock frequencies and the density of the logic used are much higher in ASICs than in FPGAs.

Therefore, our ASIC tools tightly integrate thermal issues into the placement and routing algorithms. Another example that highlights the differences between ASICs and FPGA fabrics is the additional cost of the higher connectivities in FPGAs. The design of an FPGA must facilitate a larger number of possible connections (in the x, y, and z dimensions), and this entails an overhead of silicon real estate for implementing pass-transistor switches, buffers, and SRAMs. In ASICs, on the other hand, all we need to add is an inter-tier via that

0740-7475/05/$20.00 © 2005 IEEE Copublished by the IEEE CS and the IEEE CASS

connects one active device tier (layer) to another. (We should emphasize that inter-tier vias are valuable resources in the 3D ASIC context, too, but to a lesser degree than in 3D FPGAs.)

Hence, our FPGA placement method uses a two-step optimization process that minimizes inter-tier vias first, followed by further optimization within and across tiers. In contrast, the ASIC flow uses cost function weighting to discourage, but not minimize, inter-tier crossings.

FPGA-style designs

Although previous work has proposed 3D FPGA architectures, most of it falls short of proposing a complete 3D-specific system. Alexander et al. borrowed ideas from multichip module (MCM) techniques and proposed to build a 3D FPGA by stacking together several 2D FPGA bare dies.[4] Solder bumps or vias passing through the die would make the electrical contacts between different dies.

The number of solder bumps that can fit on a die determines the width and separation of vertical channels between FPGA tiers (layers). Wu, Shyu, and Chang analyzed universal switch boxes for 3D FPGA design.[5] It is important to point out that all previous FPGA work assumed that vertical wire segments connect each tier to only its adjacent tiers to provide inter-tier connectivity. With respect to developing CAD tools for 3D FPGA integration, Alexander et al. proposed 3D placement and routing algorithms for their architecture.[4]

FPGA architecture exploration

Architects must consider several factors while developing the architecture for a 3D FPGA. Designers must strike a balance between fabrication cost, area overhead, routability, and speed. Architectural evaluation should account for the context of the circuits that will run on the FPGA and the CAD tools that map such circuits to the FPGA device. An important factor affecting the performance and area efficiency of the 3D FPGA is the routing architecture. Switch boxes with too much connectivity will excessively waste area, and meager inter-tier via counts will hurt the design's performance.

Figure 1a shows an example 3D FPGA that incorporates multiple, 2D stacked FPGAs; a subset of the switches in the switch box provide connections between tiers. Figures 1b and 1c show 2D and 3D switch boxes. As Figure 1c shows, a switch that connects wire segments in all three dimensions—x, y, and z—will have connectivity $F_s = 5$. (The F_s of a switch box is the number of outgoing tracks that connect to an incoming track.) This translates into 15 pass transistors (and buffers) as opposed to a 2D connectivity of $F_s = 3$, which requires six pass transistors. As a result, we must minimize the number of high-connectivity switches without sacrificing routability. The routing architecture for this work uses multisegment routing with inter-tier wire segments of lengths one, two, and six logic block spans.

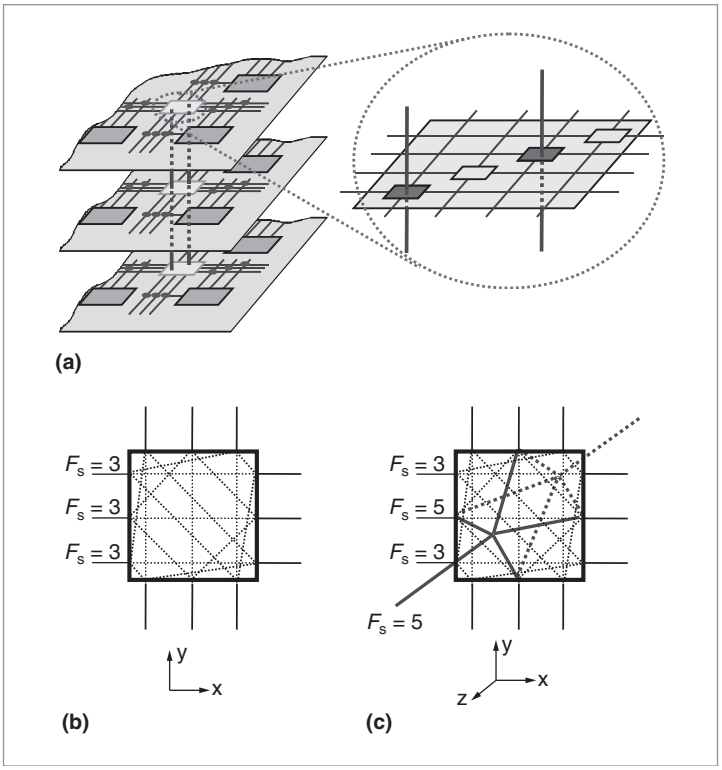

Figure 1. Example 3D FPGA (a), 2D switch (b), and 3D FPGA switch and its connectivity (c).

To fully evaluate the effect of architectural choices, designers need flexible physical-design tools that take architectural parameters as inputs and report wire length, channel width, area, and delay of benchmark circuits. We have developed a placement and routing tool called Three-Dimensional Place and Route (TPR) for this purpose.

TPR placement algorithms

The philosophy of our tool follows that of its 2D counterpart, the Versatile Place and Route (VPR) tool.[6] Figure 2 shows the flow of the TPR placement-and-routing CAD tool. The placement algorithm first employs a partitioning step using the hMetis algorithm[7] to divide the circuit into several balanced partitions, equal to the number of tiers for 3D integration. The goal of this first min-cut partitioning is to minimize the connections between tiers, which translates into reducing the number of vertical

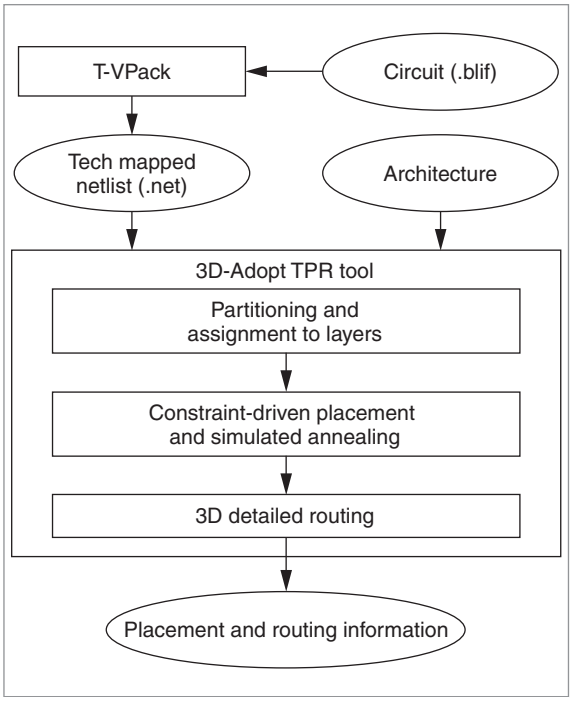

Figure 2. Flow of 3D-Adopt TPR tool.

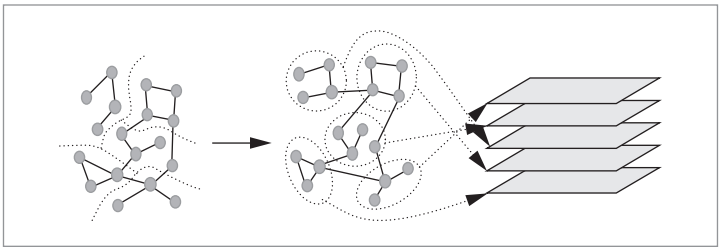

Figure 3. Partitioning of the netlist into tiers.

(inter-tier) wires and decreasing the area overhead associated with 3D switches as discussed in the previous section. After dividing the netlist into tiers, TPR continues with the placement of each tier using a hybrid approach that combines top-down partitioning and simulated annealing. The annealing step moves cells mostly within tiers.

Finally, the tool routes the cells to obtain a placed and routed solution. The following sections describe these steps in more detail.

Partitioning the circuit among tiers. Figure 3 shows a conceptual diagram of the TPR step that partitions the circuit and assigns tiers. After it partitions the netlist using hMetis, a novel linear placement approach arranges the tiers to minimize the wire length and the maximum cut size between adjacent tiers. (Cut size is the number of nets cut by a dummy plane parallel to the tiers.) TPR achieves this by mapping the problem to that of minimizing the bandwidth of a matrix, using an efficient matrix bandwidth minimization heuristic. (The bandwidth of a matrix is the maximum bandwidth of all its rows. The bandwidth of a row is the distance between the first and last nonzero entries.)

Figure 4b shows a graph in which each node corresponds to a cluster from the graph in Figure 3. For this graph, TPR forms an edges-vertices matrix in which each row corresponds to an edge, and the columns correspond to vertices. Entry a_{ij} in the matrix is nonzero if vertex j is incident to edge i and zero otherwise. Our algorithm seeks to minimize the bandwidth of this matrix by choosing an optimal ordering of the vertices.

Intuitively, we would like to minimize the bandwidth of every row, because the bandwidth of a row represents the number of tiers that the net corresponding to that row spans. Furthermore, it is desirable to distribute the bands of different rows among all columns. This is because the number of bands enclosing a particular column translates into the number of vertical vias that must pass through the tier corresponding to that column. Minimizing the matrix bandwidth achieves both goals: It minimizes the span of every row and distributes the bands across columns.

The bandwidth minimization problem is known to be NP-complete, and a solution for the tier assignment problem might not be optimal in terms of both objectives—minimizing wire length and the maximum cut size between adjacent tiers. Therefore, for this step, we use an efficient heuristic[8] that can find solutions with a very good trade-off between wire length and maximum cut. We briefly describe this technique in what follows using the graph example of Figure 4.

The procedure to solve the bandwidth minimization problem uses row (column) swaps to sort rows (columns) such that nonzero elements move toward the main diagonal.

For example, for the matrix of Figure 4a, to shift nonzero elements from the upper half toward the main diagonal (from right to left), you perform column swaps between columns 2 and 3, and then move column 6 between columns 2 and 4. Repeating this technique on rows and columns moves nonzero elements closer to the diagonal. When we ran this procedure on the graph in Figure 4b, it created the linear arrangement in Figure 4c. The goal of taking the matrix into a band form (which translates into the best linear ordering) serves two objectives:

- *Cut size minimization.* Having all 1s in the matrix

clustered along the main diagonal minimizes the cut size everywhere in the linear arrangement.

- *Wire length minimization*. Minimizing the bandwidth (maximum distance spanned by any of the nets) of the edges-vertices matrix minimizes the total wire length of all nets.

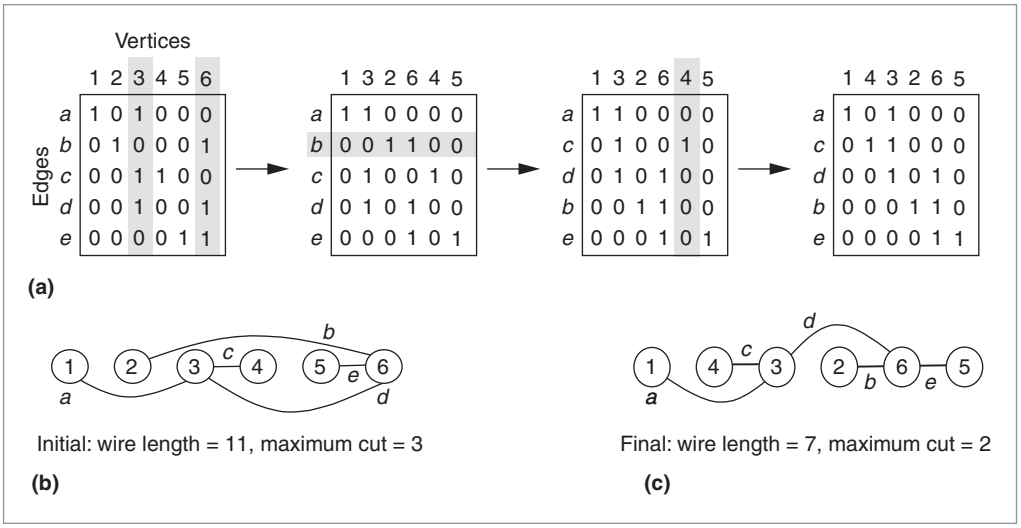

Figure 4. Edges-vertices matrix and steps to minimize both wire length and cut size (a). Performing these steps on an example graph (b) yields an optimized graph (c).

Figure 5 shows the pseudocode for minimizing edges-vertices matrix bandwidth. Referring to line 2 of the pseudocode, the left array corresponding to the leftmost matrix in Figure 4a would be 3, 6, 4, 6, 6, since the rightmost 1 elements in the rows are in columns 3, 6, 4, 6, and 6. Sorting this array requires swapping the second and third elements, which translates into swapping the second and third rows of the edges-vertices matrix.

1. Build EV-matrix
2. Array Left = indices of right-most non-zero elements in rows.
   Sort Left, swapping rows.
3. Array Top = indices of bottom-most non-zero elements in columns.
   Sort Top, swapping columns.
4. Array Right = indices of left-most non-zero elements in rows.
   Sort Right, swapping rows.
5. Array Bottom = indices of top-most non-zero elements in columns.
   Sort Bottom, swapping columns.

Figure 5. Algorithm for bandwidth minimization.

Partitioning-based placement within tiers. After the initial tier assignment, TPR performs placement on each tier, starting with the top tier and proceeding tier after tier. The placement of every tier is based on edge-weighted quad partitioning using the hMetis partitioning algorithm, and is similar to the approach used by Maidee, Ababei, and Bazargan,[9] which has the same quality as VPR but at three to four times shorter runtimes. Edge weights are usually computed as inversely proportional to the timing slack of the corresponding nets. However, we also selectively bias weights of the most critical nets. The set of critical nets consists of edges on the current k most-critical paths. To improve timing, TPR considers the bounding box of the terminals of a critical net that it has already placed. It projects this bounding box onto the lower tiers and uses it as a placement constraint for other terminals. Our earlier work presents the details of this partitioning-based placement phase.[10]

Simulated annealing placement phase. Following the partitioning-based placement step, we use a 3D-adapted version of VPR in the low-temperature annealing phase to further improve wire length and routability. We use the following cost function for each net:

$$Cost_{3D}(e) = q.Cost_{2D}(e) + \alpha.Span_z(e) + \beta.numTiers(e) \quad (1)$$

where $Cost_{2D}$ is the half-perimeter size of the *2D* projection of the bounding box of net e, $Span_z(e)$ is the total span of the net between tiers, and *numTiers* is the number of tiers on which we distribute the net's terminals. Parameters q, α, and β are tuning parameters (q has the same role as in VPR). Figure 6 shows an example to illustrate why we use the two components $Span_z$ and *numTiers*. In a 3D routing structure that employs multi-segment inter-tier connections, the left figure is more likely to use fewer vertical connections (of length 2) to connect the terminals on the first and the third tiers.

Routing algorithm. Our routing algorithm is an extension of VPR's routing engine. The 3D FPGA architecture (Figure 1a) described in the architecture file is represent-

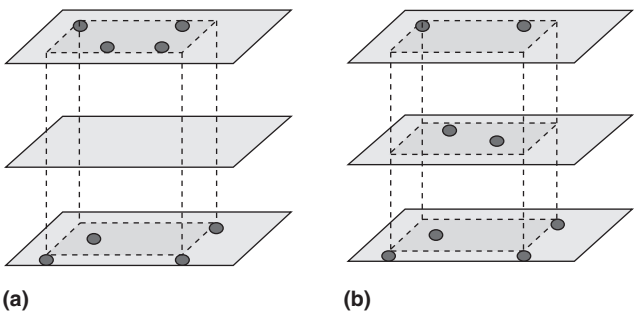

Figure 6. Example showing the difference between a net's span and number of tiers: for *Num_layers(e)* = 2, $Span_z$ = 2 (a); and for *Num_layers(e)* = 3, $Span_z$ = 2 (b).

ed as a routing resource graph. Each node of the routing resource graph represents a wire (horizontal tracks in the *x* and *y* channels of all tiers, and vertical vias in the *z* channels) or a logic block input or output pin. A directed edge represents a unidirectional switch (such as a tristate buffer). A pair of directed edges represents a bidirectional switch (such as a pass transistor). We add extra penalties to bends of a route created by a horizontal track and a vertical via, as well as to vertical vias themselves. This discourages the routing engine's preference for vertical vias, avoiding a net that could have been placed entirely in one tier from making a detour to other tiers and using vertical vias and routing resources in multiple tiers.

FPGA results. In our experiments, we used the 3D FPGA architecture of Figure 1a, where segment lengths of 1, 2, and *long* (spanning all tiers) form the inter-tier routing structure, and segment lengths of 1, 2, 6, and long are used within tiers. We assumed the delay of an inter-tier segment to equal that of an intra-tier segment of the same length. This is justified by the relatively short length of inter-tier vias in the emerging 3D technologies, and the fact that the dominant factors in the delay of an FPGA routing segment are the pass transistor and buffer delays. Our architecture definition file, however, is modifiable to reflect any parasitics on the vertical connections.

Figure 7 shows the results of our algorithm on the Microelectronics Center of North Carolina (MCNC) benchmarks. Figure 7a compares the quality of five-tier 3D circuits placed and routed using TPR to 2D circuits placed and routed using VPR. The bars show the ratio of the averages (over all the MCNC benchmark circuits) of metrics such as area and delay to those of their 2D counterparts. Total area is the footprint area multiplied by the number of tiers. We see that delay and wire length improve by about 20%, whereas total area increases because of the extra area used by 3D switches and the white space created on some tiers.

Figure 7b shows the improvement in delay for all MCNC benchmarks as we increase the number of tiers. We observe that for this circuit size, going up to five to six tiers has great benefits, but going beyond that results in diminishing returns.

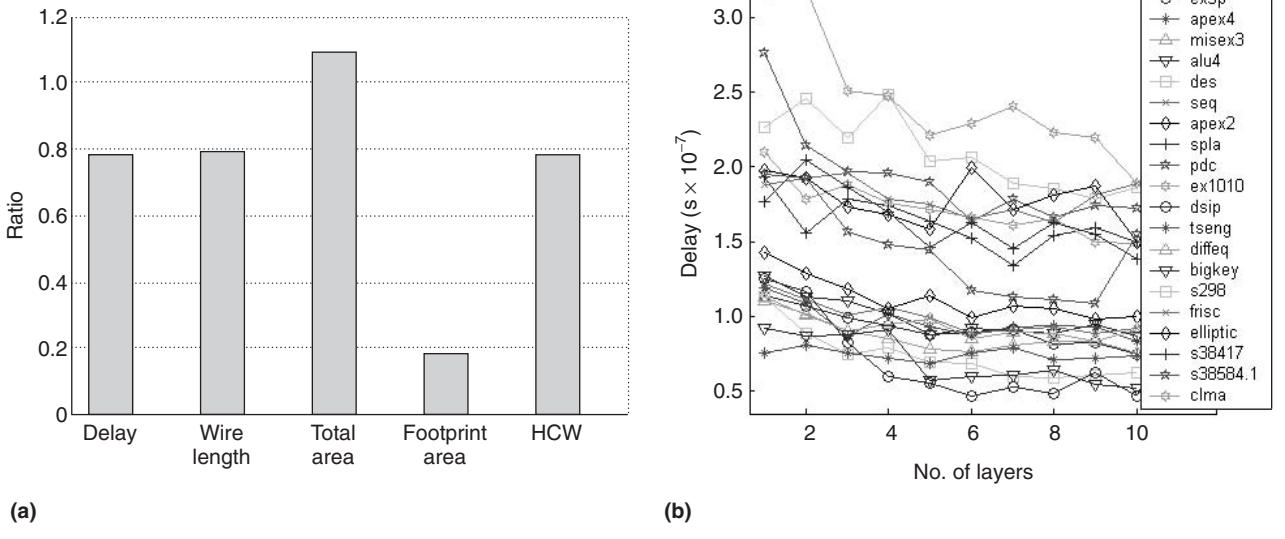

Figure 7. Ratio of wire length for TPR compared to 2D simulated-annealing design in VPR (a). For the TPR-placed-and-routed circuit, we plotted number of tiers versus delay (b) for MCNC benchmarks.

ASIC-style designs

For standard-cell-based 3D designs, we describe a flow, illustrated in Figure 8, for performing placement and routing with built-in techniques for thermal mitigation. The input to the system is a technology-mapped netlist and a description of the library (these could be, for example, LEF, library exchange format; DEF, data exchange format; or .lib descriptions), and the physical-design process consists of several steps. This flow treats temperature as a first-class citizen during the optimization, in addition to other conventional metrics, and it also considers inter-tier via reduction as a desirable goal. In the placement step, the standard cells are arranged in rows within the tiers of the 3D circuit.

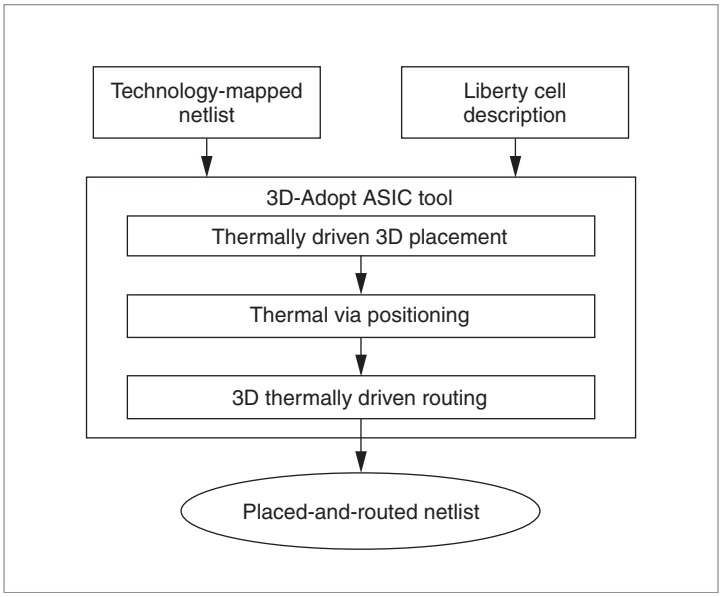

Figure 8. Physical-design flow for 3D ASIC-style implementations.

Because thermal considerations are particularly important in 3D ASIC-like circuits, this procedure must spread the cells to achieve a reasonable temperature distribution, while also capturing traditional placement requirements. In the second step, the flow makes the temperature distribution more uniform by the judicious positioning of thermal vias within the placement, which achieves improved heat removal. These vias correspond to inter-tier metal connections that have no electrical function, but instead, constitute a passive cooling technology that draws heat from the problem areas to the heat sink. Finally, the placement goes through a routing step to obtain a completed layout. Routing must take several objectives and constraints into consideration, including the avoidance of blockages because of areas occupied by thermal vias, incorporating the effect of temperature on the delays of the routed wires, and of course, traditional objectives such as wire length, timing, congestion, and routing completion. We will now describe each of these steps in detail.

3D thermally driven placement

Before describing the placer's internal workings, it is illustrative to view the result of a typical 3D placement obtained using the 3D-Adopt placer: a layout for benchmark circuit ibm01 in a four-tier 3D process. In the layout in Figure 9, the cells are in ordered rows on each tier, and the layout in each individual tier looks similar to a 2D standard-cell layout.

The heat sink is at the bottom of the 3D chip, and the lighter regions are hotter than the darker regions. The coolest cells are those in the bottom tier, next to the heat sink, and the temperature increases as we move to higher tiers. The thermal placement method consciously mitigates the temperature by making the upper tiers sparser, in terms of the percentage of area populated by the cells, than the lower tiers.

In the subsequent description, we will provide an overview of the algorithms that the placement engine uses, showing how they directly incorporate thermal objectives into placement.

Fast thermal analysis of 3D ICs. An essential ingredient of a thermally driven placement engine is a fast temperature analyzer. At the placement stage, it is adequate to consider the steady-state case, where the following differential equation describes heat conduction within the chip substrate:

$$K_x \frac{\partial^2 T^2}{\partial x} + K_y \frac{\partial^2 T^2}{\partial y} + K_z \frac{\partial^2 T^2}{\partial z} + Q(x,y,z) = 0 \qquad (2)$$

where T is temperature; and K_x, K_y, and K_z are the thermal conductivities along the three coordinate directions; and Q is the heat generated per unit volume. A unique solution exists for appropriately applied convective, isothermal, and/or insulated boundary conditions, which the nature of the packaging and the heat sink determine.

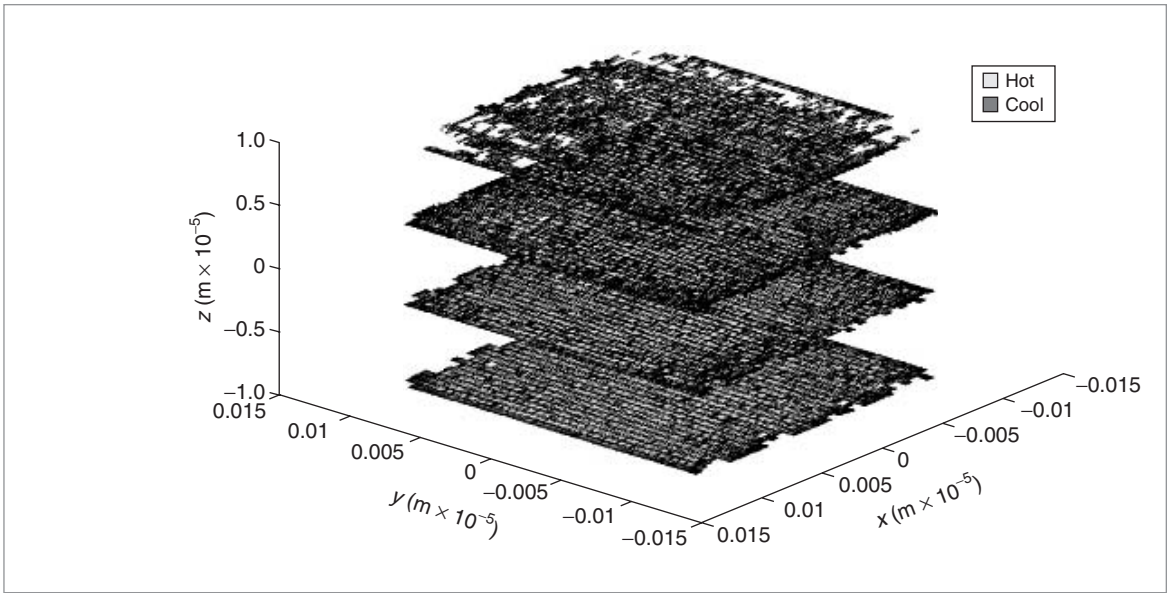

Figure 9. Placement for the benchmark ibm01 in a four-tier 3D technology.

We can solve this partial differential equation numerically using finite element analysis (FEA),[11] which discretizes the design space into regions known as elements. For rectangular structures of the type encountered in ICs, a rectangular cube-like element can simulate heat conduction in the lateral directions without aberrations in the prime directions.

FEA calculates the temperatures at discrete points (in this case, the nodes of the cube-like element), and it interpolates the temperatures within the elements using a weighted average of the temperatures at the nodes. In deriving the finite-element equations, FEA uses this interpolation within the elements to approximate Equation 2. For a specific element type, such as a rectangular prism, you can derive *element stamps* that are similar in character to the element stamps for electrical elements in the well-known, modified nodal analysis. The heat conduction stamp for the eight-vertex rectangular prism is an 8×8 matrix.

These stamps go to a global matrix to set up the global system of linear equations,

$$[K]\boldsymbol{T} = \boldsymbol{P}$$

where $\boldsymbol{T}$ is the vector of nodal temperatures and $\boldsymbol{P}$ the vector of node powers. In FEA parlance, matrix $[K]$ is the global stiffness matrix. We can similarly derive the stamps for boundary conditions.

Conductive boundary conditions simply correspond to fixed temperatures; since these parameters are no longer variables, they can be eliminated and the quantities moved to the right side of the equation so that $[K]$ is nonsingular. We can solve the FEA equations rapidly using an iterative linear solver, with clever adjustments to the convergence criteria to achieve greater or lesser accuracy, as required at different stages of the iterative placement process.

Force-directed paradigm. For 3D designs, the tools must carry out placement in not just the *xy* plane, but in the entire *xyz* space. Current technologies restrict the *z* dimension to just a few tiers.

To solve this problem, we base our placement engine on a force-directed approach, where we use an analogy to Hooke's law, representing nets as springs and finding the cell positions that correspond to the system's minimum energy state. In this analogy, we create attractive forces, illustrated in Figures10a and 10b, between interconnected cells; these forces are proportional to the quadratic function of the cell coordinates that represents the Euclidean distance between the blocks. We chose higher proportionality constants in the *z* direction to discourage inter-tier vias. Fixed locations, such as I/O pads or fixed blocks, are easy to incorporate into this formulation. We use other design criteria—such as cell overlap, timing, and congestion—to derive repulsive forces. In the 3D context, we use thermal criteria to generate repulsive forces, preventing hot spots. FEA uses the temperature gradient (which relates directly to the stiffness matrix and its derivative) to determine the magnitudes

and directions of these forces, as Figure 10c illustrates.

Once we generate the entire system of attractive and repulsive forces, we solve the system for the minimum energy state, that is, the equilibrium location. Ideally, this solution minimizes the wire lengths while at the same time satisfying the other design criteria, such as temperature distribution. The iterative force-directed approach takes the following steps in the main loop: Initially, this approach updates forces based on the previous placement. Using these new forces, the placer then recalculates the cell positions. The process repeats these two steps of calculating forces and finding cell positions until the layout satisfies the exit criteria. Goplen and Sapatnekar present the specifics of the force-directed approach to thermal placement, including the mathematical details.[12]

Once the iterations converge, a final postprocessing step legalizes the placement. Even though added forces discourage overlaps, the force-directed engine solves the problem in the continuous domain, and the task of legalization is to align cells to tiers, and to rows within each tier.

Applied to benchmark circuits with over 50,000 cells, this approach shows a runtime that is approximately linear as circuit size increases. The placement results in Figure 11 show average improvements of 17% in the average thermal gradient and 17% in the maximum temperature, for a nominal increase in wire length as compared to nonthermal placement.

Thermal via positioning. Although silicon is a good thermal conductor, with half or more of the conductivity of typical metals, many of the materials used in 3D technologies are strong insulators that severely restrict heat removal, even under the best placement solution. The materials include epoxy bonding materials used to attach 3D tiers, field oxide, or the insulator in an SOI technology. Therefore, the use of deliberate metal lines that serve as heat-removing channels, called *thermal vias*, are an important ingredient of the total thermal solution. The second step in the flow determines the optimal positions of thermal vias in the placement to provide an overall improvement in the temperature distribution. In realistic 3D technologies, the dimensions of these inter-tier vias are of the order of 5 μm × 5 μm.

In principle, we can view the problem of placing thermal vias as one of determining one of two conductivities (corresponding to the presence or absence of metal) at every candidate point where we can place a thermal via in the chip. However, in practice, it is easy to see that such an approach could lead to an extremely large search space with an exponential number of possible positions; the set of possible positions in itself is extremely large. Quite apart from the size of the search space,

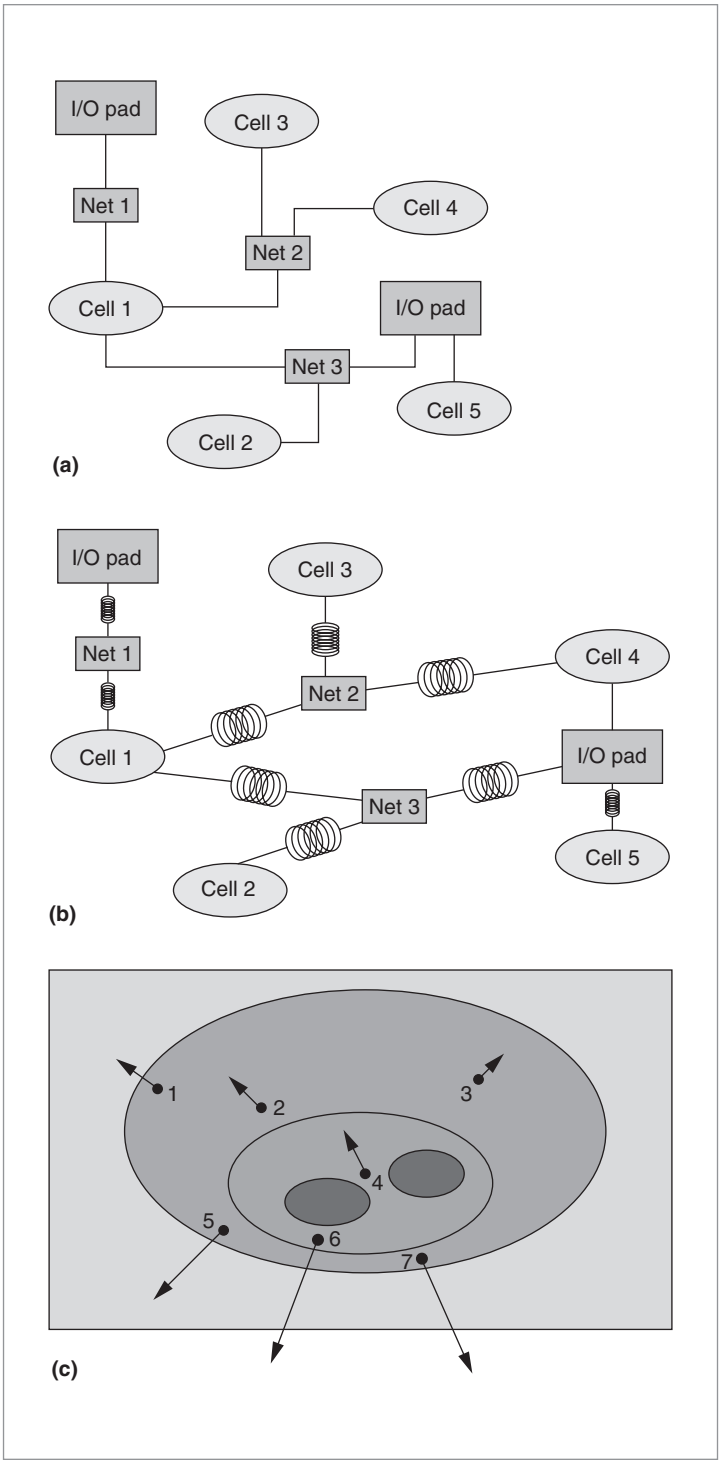

Figure 10. Sample netlist (a), attractive forces corresponding to wire connections (b), and repulsive thermal force vectors (c).

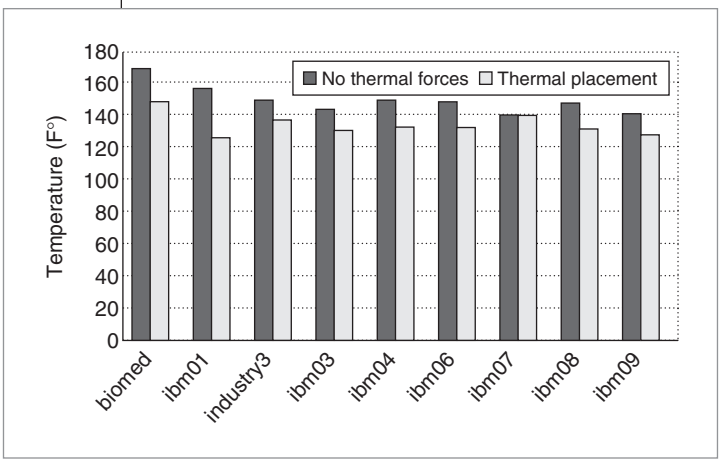

Figure 11. Temperature gradient improvements with thermally driven placement.

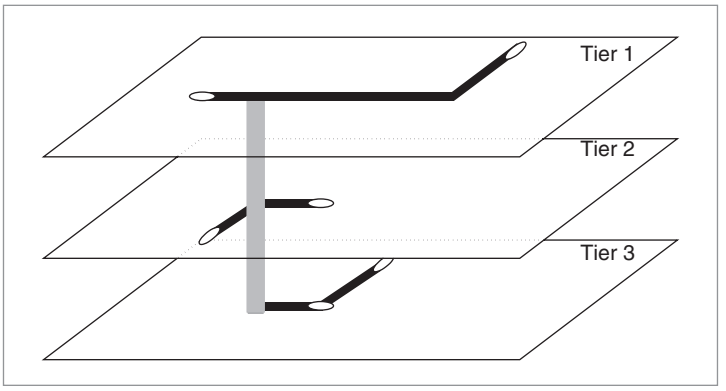

Figure 12. Example route for a net in a three-tier 3D technology.

such an approach is unrealistic for several other reasons.

First, the wanton addition of thermal vias in any arbitrary region of the layout would be nightmarish for a router, which would have to navigate around these blockages. Second, from a practical standpoint, it is unreasonable to perform full-chip thermal analysis, particularly in the inner loop of an optimizer, at the granularity of individual thermal vias. At this level of detail, individual elements would have to correspond to the size of a thermal via, and the size of the FEA stiffness matrix would become extremely large.

Fortunately, there are reasonable ways to overcome each of these issues. The blockage problem might be controllable by enforcing discipline within the design: designating, for example, a specific set of areas within the chip as potential thermal via sites. These could be specific inter-row regions in the cell-based layout, and the optimizer would determine the density with which thermal vias fill these regions. The advantage to the router is obvious, since only these regions are potential blockages, which is much easier to handle. To control the FEA stiffness matrix's size, we could work with a two-level scheme with relatively large elements, where the average thermal conductivity of each region is a design variable. Once we determine this average conductivity, we could translate it back into a precise distribution of thermal vias within the element that achieves that average conductivity.

Earlier work describes an algorithm to solve this problem.[13] We have applied this technique to a range of benchmark circuits, some with over 158,000 cells, and the insertion of thermal vias delivers an improvement in average temperature of about 30% with runtimes of a couple of minutes.[13] Therefore, thermal via addition has a more dramatic effect on temperature reduction than thermal placement.

Remarkably, the greatest concentration of thermal vias is not in the hottest regions, as you might first expect. The intuition behind this is as follows: If we consider the center of the uppermost tier, it is hot because the tier below it is at an elevated temperature. Adding thermal vias to remove heat from the second tier, therefore, also significantly reduces the temperature of the top tier. For this reason, the regions where the insertion of thermal vias is most effective are those that have high thermal gradients.

Routing algorithms. Once the process has placed the cells and determined the locations of the thermal vias, the routing stage finds the optimal interconnections between the wires. As in 2D routing, it is important to optimize wire length, delay, and congestion. In addition, several 3D-specific issues come into play. First, the delay of a wire increases with its temperature, so that critical wires should avoid the hottest regions as much as possible. Secondly, inter-tier vias are a valuable resource that the routing stage must optimally allocate among the nets.

Third, congestion management and blockage avoidance is more complex with the addition of a third dimension. For instance, signal or thermal vias that span two or more tiers constitute a blockage that wires must navigate around. Each of these issues are manageable through exploiting the flexibilities in determining the precise route within the bounding box of a net, or perhaps even considering slight detours outside the bounding box. Detours might be effective when an increase in the wire length improves the delay or congestion, or when they provide greater flexibility for inter-tier via assignment.

Consider the problem of routing in a three-tier technology, as illustrated in Figure 12. The layout has a grid of rectangular tiles, each with a horizontal and vertical capacity that determines the number of wires that can traverse the tile, and an inter-tier via capacity that determines the number of free vias available in that tile. These capacities account for the resources allocated for wires other than those for signals (those for power and clock, for example) as well as the resources used by thermal vias. For a single net, as Figure 12 shows, two degrees of freedom are available: the choice of locations for inter-tier vias, and the precise routes within each tier. The locations of inter-tier vias will depend on the resource contention for vias within each grid. Moreover, critical wires should avoid the high-temperature tiles as much as possible.

Figure 13 shows the overall flow of the solution technique. In the first step, we build a Steiner minimum tree for each net, with a cost that depends on the length of the net (with a penalty that discourages, but does not prohibit, the use of more than the minimum number of inter-tier vias).

This Steiner structure still affords considerable flexibility in the routing through the availability of soft edges,[14] and in each layer, assuming L and Z shaped routes, the second step determines the distribution of wire congestion. Next, a hierarchical procedure determines the precise assignment of inter-tier via locations: this corresponds to a sequence of assignment problems (assigning nets to vias) soluble using network-flow techniques. Once the inter-tier via locations are determined, the final step performs a minimum-cost maze routing in each layer, with a cost function based on wire length, temperature, and congestion, to yield the global routing solution. Finally, any standard 2D detailed router can perform the detailed routing of each tier.

Figure 14 shows the average delay improvements for the critical sinks for a set of benchmark circuits, as compared to a router that ignores thermal effects. The improvements range between 12% and 30%, and the total wire length remains nearly unchanged from that of the nonthermal case.

THREE-DIMENSION TECHNOLOGIES offer great promise in providing improvements in the overall circuit performance. Physical design plays a major role in the ability to exploit the flexibilities offered in the third dimension, and this article gives an overview of placement and routing methods for FPGA- and ASIC-style designs.

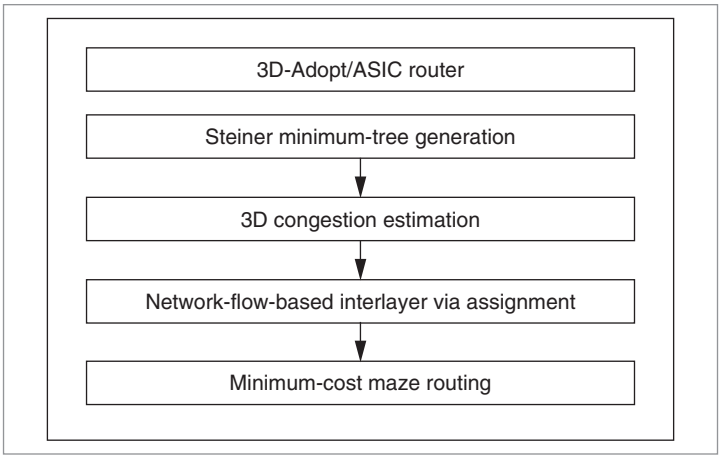

Figure 13. Steps in the overall router algorithm.

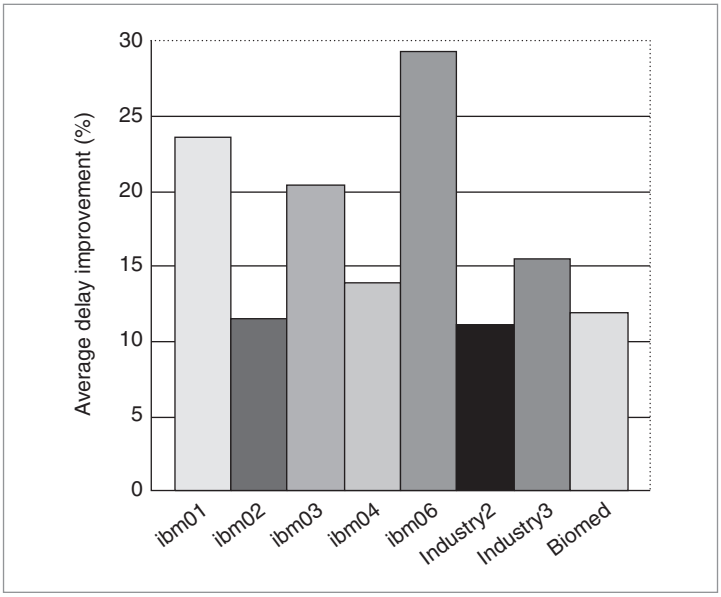

Figure 14. Average delay improvement for thermal versus nonthermal 3D routing.

Several promising directions remain to be explored, because 3D design enables other significant technologies. For example, 3D integration permits mixed-signal designs to isolate analog functionalities from digital blocks by placing them on different layers and/or using isolation ground planes between layers. It also permits heterogeneous integration using dissimilar technologies in each tier (for example, CMOS in one tier and gallium arsenide in another). Each of these ideas offer further challenges in the placement-and-routing arena, and these areas remain topics for further research. ■

Acknowledgment

This research was supported in part by DARPA

under grant N66001-04-1-8909.

References

1. K.W. Guarini et al., "Electrical Integrity of State-of-the-Art 0.13μm SOI CMOS Devices and Circuits Transferred for Three-Dimensional (3D) Integrated Circuit (IC) Fabrication," *Technical Digest of the IEEE Int'l Electron Devices Meeting*, IEEE Press, 2002, pp. 943-945.
2. J. Burns et al., "An SOI-Based Three Dimensional Integrated Circuit Technology," *Proc. IEEE Int'l SOI Conf.,* IEEE Press, 2000, pp. 20-21.
3. R. Reif et al., "Fabrication Technologies for Three-Dimensional Integrated Circuits," *Proc. Int'l Symp. Quality Electronic Design* (ISQED 02), IEEE Press, 2002, pp. 33-37.
4. A.J. Alexander et al., "Placement and Routing for Three-Dimensional FPGAs," *Proc. 4th Canadian Workshop on Field-Programmable Devices*, 1996, pp. 11-18; http://www.cs.virginia.edu/~robins/papers/fpd_camera.pdf.
5. G.-M. Wu, M. Shyu, and Y.-W. Chang, "Universal Switch Blocks for Three-Dimensional FPGA Design," *Proc. ACM/SIGDA Int'l Symp. Field Programmable Gate Arrays*, ACM Press, 1999, pp. 254-259.
6. V. Betz and J. Rose, "VPR: A New Packing Placement and Routing Tool for FPGA Research," *Proc. 7th Int'l Workshop Field Programmable Logic and Applications* (FPL 97), Lecture Notes in Computer Science, vol. 1304, Springer, 1997, pp. 213-222.
7. G. Karypis et al., "Multi-Level Hypergraph Partitioning: Applications in VLSI Design," *Proc. 34th ACM/IEEE Design Automation Conf.* (DAC 97), ACM Press, 1997, pp. 526-529.
8. C. Ababei and K. Bazargan, "Non-Contiguous Linear Placement for Reconfigurable Fabrics," Proc. Reconfigurable Architectures Workshop, 2004, p. 141b, http://www.ece.umn.edu/users/kia/Papers/2004/IJES04_espIssue_RAW04.pdf.
9. P. Maidee, C. Ababei, and K. Bazargan, "Fast Timing-Driven Partitioning-based Placement for Island Style FPGAs," *Proc. 40th ACM/IEEE Design Automation Conf.* (DAC 03), ACM Press, 2003, pp. 598-603.
10. C. Ababei, H. Mogal, and K. Bazargan, "Three-Dimensional Place and Route for FPGAs," *Proc. Asia-South Pacific Design Automation Conf.* (ASP-DAC 05), ACM Press, 2005, vol. 2, pp. 773-778.
11. D.L. Logan, *A First Course in the Finite Element Method*, 3rd ed., Brooks/Cole Publishing Co., 2002.
12. B. Goplen and S.S. Sapatnekar, "Efficient Thermal Placement of Standard Cells in 3D ICs Using a Force Directed Approach," *Proc. IEEE/ACM Int'l Conf. Computer-Aided Design* (ICCAD 03), IEEE Press, 2003, pp. 86-89.
13. B. Goplen and S.S. Sapatnekar, "Thermal Via Placement in 3D ICs," *Proc. ACM Int'l Symp. Physical Design* (ISPD 05), ACM Press, 2005, pp. 167-174.
14. J. Hu and S.S. Sapatnekar, "A Timing-Constrained Simultaneous Global Routing Algorithm," *IEEE Trans. Computer-Aided Design*, vol. 21, no. 9, Sept. 2002, pp. 1025-1036.

Cristinel Ababei is a member of the technical staff at Magma Design Automation. His research interests include CAD tools for layout and logic synthesis for robust VLSI circuits and FPGAs. Ababei has a BS in microelectronics from the Technical University of Iasi, Romania, and a PhD in electrical engineering from the University of Minnesota. He is a member of the IEEE and ACM.

Yan Feng is a post-doctoral researcher in the Electrical and Computer Engineering Department at the University of Minnesota. His research interests include algorithms and CAD for VLSI circuits. Feng has a BS in Engineering from the University of Science and Technology, Beijing; an MS and a PhD in computer science from the Colorado School of Mines. He is a member of the IEEE.

Brent Goplen is a PhD student in Electrical Engineering at the University of Minnesota. His research interests include CAD, the physical design of next-generation circuits, thermal issues, placement, and 3D ICs. Goplen has a BA in Chemistry and Biochemistry from Gustavus Adolphus College and an MS in computer engineering with a minor in mechanical engineering from the University of Minnesota.

Hushrav Mogal is a PhD student at the University of Minnesota. His research interests include computer-aided design tools and thermal issues for FPGAs and ASICs. Mogal has a BE in electronics engineering from the University of Mumbai, Bombay.

Tianpei Zhang is a PhD student in electrical engineering at the University of Minnesota. His research interest includes VLSI physical design and design for manufacturability. Zhang has a BS in applied physics from the University of Science and Technology of China and an MS in electrical engineering from Purdue University. He is a student member of ACM SIGDA.

Kia Bazargan is an assistant professor in the Department of Electrical and Computer Engineering at the University of Minnesota. His research interests include CAD, FPGAs, and reconfigurable computing. Bazargan has a BS in computer science from Sharif University in Tehran, Iran, and an MS and a PhD in electrical and computer engineering from Northwestern University. He is an associate editor for the *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* and an NSF Career award recipient. He is a member of IEEE and ACM.

The biography of **Sachin Sapatnekar** appears on p. 497 of this issue.

■ Direct questions and comments about this article to Kia Bazargan or Sachin Sapatnekar, Electrical and Computer Engineering Dept., Univ. of Minnesota, 200 Union St. SE, Minneapolis, MN 55455; {kia,sachin}@umn.edu.