Investigating the Effects of Fine-Grain Three-Dimensional Integration on Microarchitecture Design

YUCHUN MA
Tsinghua University
YONGXIANG LIU, EREN KURSUN, and GLENN REINMAN
University of California, Los Angeles
and
JASON CONG
University of California, Los Angeles
California Nanosystems Institute

17

In this article we propose techniques that enable efficient exploration of the 3D design space, where each logical block can span more than one silicon layer. Fine-grain 3D integration provides reduced intrablock wire delay as well as improved power consumption. However, the corresponding power and performance advantage is usually underutilized, since various implementations of multilayer blocks require novel physical design and microarchitecture infrastructure to explore 3D microarchitecture design space. We develop a cubic packing engine which can simultaneously optimize physical and architectural design for efficient vertical integration. This technique selects the individual unit designs from a set of single-layer or multilayer implementations to get the best microarchitectural design in terms of performance, temperature, or both. Our experimental results using a design driver of a high-performance superscalar processor show a 36% performance improvement over traditional 2D for 2–4 layers and 14% over 3D with single-layer unit implementations. Since thermal characteristics of 3D integrated circuits are among the main challenges, thermal-aware floorplanning and thermal via insertion techniques are employed to keep the peak temperatures below threshold.

This research is supported by the National Science Foundation under Grant CCR-0096383, CCF-0430077, and CCF-0530261, the NSFC 60606007 and Tsinghua Basic Research Fund JC20070021.
Authors' addresses: Y. Ma, Department of Computer Science and Technology, Tsinghua University, Beijing, 100084, P. R. China; email: myc@mail.tsinghua.edu.cn; Y. Liu, E. Kursun, Computer Science Department, University of California at Los Angeles, 4732 Boelter Hall, Los Angeles, CA 90095; G. Reinman, J. Cong, Computer Science Department, University of California at Los Angeles, 4732 Boelter Hall, Los Angeles, CA 90095; email: {reinman,cong}@cs.ucla.edu.
 DOI 10.1145/1412587.1412590 http://doi.acm.org/10.1145/1412587.1412590

Categories and Subject Descriptors: B.7.0 [**Integrated Circuits**]: General; C.1.0 [**Processor Architectures**]: General

General Terms: Design, Performance

Additional Key Words and Phrases: 3D integration, thermal, microarchitecture, 3D packing

ACM Reference Format:

Ma, Y., Liu, Y., Kursun, E., Reinman, G., and Cong, J. 2008. Investigating the effects of fine-grain three-dimensional integration on microarchitecture design. ACM J. Emerg. Technol. Comput. Syst. 4, 4, Article 17 (October 2008), 30 pages. DOI = 10.1145/1412587.1412590 http://doi.acm.org/10.1145/1412587.1412590

1. INTRODUCTION

3D integration technology has attracted significant interest in recent years due to the improvement it provides for existing process technologies, such as reduced interconnect latency, improved packaging density, and heterogeneous integration of disparate technologies. An extensive amount of research has demonstrated the latency reduction through additional device layers [Das et al. 2003; Banerjee et al. 2001; Black et al. 2004; Puttaswamy and Loh 2006a; Loh 2008a, 2008b]; however, most prior arts on 3D [Cong et al. 2006, 2004; Das et al. 2004; Hung et al. 2006] have been limited to stacking traditional 2D dies or blocks. Such stacking offers reduction in interblock latency, yet does little to help *intrablock* wire latency. Although further improvement in both power and performance is possible from this type of fine-grain vertical integration, the lack of modeling and tool infrastructure limits the availability of such benefits.

Recent studies have provided block models for various architectural structures, including 3D caches [Tsai et al. 2005; Reinman et al. 2000; Kleiner et al. 1996; Ronnen et al. 2001], register files [Tremblay et al. 1995], 3D arithmetic units [Puttaswamy and Loh 2007, 2006b], and instruction schedulers [Puttaswamy and Loh 2006c], along with 2D-based placement tools such as Xie et al. [2006]. However, most of these models are limited to folding blocks by wordlines or bitlines. Other studies [Black et al. 2004; Xie et al. 2006] simply extended the 2D floorplan to multiple layers, which in turn caused heating problems due to vertical stacking of hotspots. At the same time, the overall performance impact of using a combination of 2D and 3D implementations of different components in the same design remains largely unexplored. That is to say, having a subset of the architectural units occupy a single device layer while others are implemented on multiple strata with potentially different heights in the z-dimension has not been explored at the architectural level. It is also important to note that using a locally best implementation of an individual unit may not necessarily lead to the best overall design at architecture level. Hence, the applicability of studies that focus on a single unit is limited, since such approaches do not effectively explore the interaction among blocks and global design constraints.

Another critical issue that has not been modeled in the prior art is the thermal effects of multilayer 3D stacking. Although multilayer blocks can reduce the area, delay, and power consumption for each individual block [Puttaswamy

and Loh 2006c; Xie et al. 2006], the best configuration for each single block may not lead to the best thermal design for the whole chip. In 3D design, the *co-optimization* of microarchitectural and physical design is essential, as most of the potential benefits are due to wirelength reduction. Recently, the MEVA-3D [Cong et al. 2006] framework was proposed to bridge the gap between physical planning and architectural design. This framework iteratively optimizes microarchitectural loop sensitivities in the floorplanning process to guide block placement and reduce the loop latencies. However, MEVA-3D is limited to single-layer blocks and does not enable multilayer block exploration, where each block can span more than one silicon layer.

To enable exploration with multilayer blocks, we model the multilayer 3D design as a cube packing problem. Although a number of 2D packing representations have been extended to handle the cube packing problem, such as ST [Yamazaki et al. 2000] and 3D sub-TCG [Yuh et al. 2004], they merely optimize the volume of cubic blocks based on simulated annealing using different representations. Additional design optimization needs to target the intrablock wire latency and the corresponding performance improvement beyond the improved packaging volume.

In this article we explore fine-granularity vertical integration and its impact on the micro-architecture design, where individual blocks are placed across multiple layers. By exploring multilayer implementations, our physical engine places blocks effectively within the given layer constraints, resulting in performance improvement and reduced power dissipation. We analyze the thermal characteristics in detail, while our thermal-aware floorplanning and via insertion stages keep the hotspot temperature below the silicon thermal thresholds. We make the following contributions.

—*Multilayer Architectural Blocks and Architecture-Driven 3D Packing*. We model components in multiple silicon layers and analyze the effects of different partitioning approaches on various components with respect to area, timing, and power, where individual blocks can occupy one or more strata. In this scheme a combination of single/multilayer block implementations are used in the same design. In addition to exploring the use of single-layer and multilayer blocks, we also analyze the impact of scaling the sizes of different architectural structures.

—*Architecture-Driven Cube Packing Engine*. During the packing optimization, we introduce the process of *alternative selection* to choose the best configurations among a wide range of implementations. Some heuristic methods are proposed to speed-up the convergence and reduce redundant searches on infeasible solutions.

—*Co-Optimization of Microarchitecture Design and Physical Design for 3D*. We extend the microarchitecture-physical codesign framework [Liu et al. 2007; Cong et al. 2006] to handle fine-grain 3D exploration. Given a frequency target, architectural netlist, and a pool of alternative block implementations, this framework finds the best solution in terms of performance (in BIPS), temperature, or both. The thermal-aware packing engine with our state-of-the-art temperature simulator tool is capable of optimizing the entire design

by choosing from various alternative implementations of each block. We also perform thermal via insertion to help mitigate the impact of increased temperatures in 3D integration.

—*Detailed Thermal Analysis of the Resulting 3D Design*. Since one of the main limitations of 3D integration is heat removal, we explore the thermal characteristics of experimental cases in detail to ensure feasibility of the resulting 3D design. We also utilize thermal via insertion to keep temperatures below the given thermal thresholds.

Our experimental analysis reveals that fine-grain 3D integration can provide 36% performance improvement over traditional 2D designs. Furthermore, peak temperatures can be kept within operation limits for moderate 3D stacks through the temperature-aware physical design-microarchitecture codesign framework and thermal via insertion. The remainder of this article is organized as follows. Section 2 briefly presents background on 3D integration technology; Section 3 discusses alternative 3D implementations of components; the physical exploration, along with details of the cube packing algorithms, is in Section 4; Section 5 provides the experimental methodology and experimental results. Finally, concluding remarks and future work are discussed in Section 6.

2. 3D TECHNOLOGY: PRELIMINARIES

3D IC fabrication refers to a wide range of technologies, including stacked multichip modules through wire bonding and similar MCM packaging [Tsui et al. 2003], chip-to-chip, chip-to-wafer, or wafer-wafer, bonding [Banerjee et al. 2001; Black et al. 2004; Topol et al. 2005; Das et al. 2003], and solid-phase recrystallization [Banerjee et al. 2001], each spanning a wide range of characteristics in terms of circuit performance, manufacturing cost, and thermal profile.

There has been an extensive amount of work on 3D integration; in this work we will only focus on closely related studies. Banerjee et al. [2001], Xie et al. 2007, 2006], Burns et al. [2006, 2001], and Topol et al. [2006, 2005] provide extensive discussion on current 3D research studies. We also use aggressive wafer-bonding 3D IC technology parameters. However, it is important to note that some 3D assumptions are more similar to sequential 3D technology with silicon growth in terms of aggressive fine-grain via pitch and interlayer bandwidth then to wafer-level integration; the analysis and results can be extended to similar technologies. 3D integration commonly involves face-to-face (F2F) and face-to-back (F2B) approaches, as illustrated in Figure 1. In Figure 1(a) the top device layer is flipped upside down so that the device layers are facing each other in F2F, and in Figure 1(b) the top device layer is oriented in the same direction as the bottom device layer in an F2B manner. We assume the use of F2B in this study, as shown in Figure 1(b).

3. DESIGN OF 3D ARCHITECTURAL BLOCKS

Traditional design space explored by architecture-level design includes different component sizes and characteristics, along with various ways of connecting

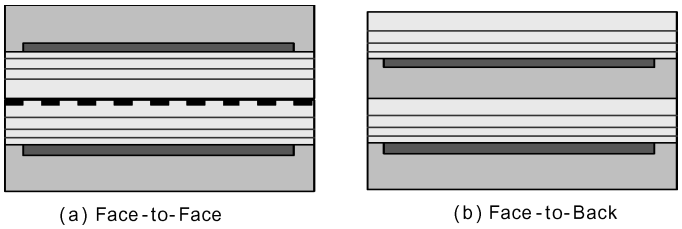

Fig. 1. 3D IC example with two device layers: (a) face-to-face (F2F); (b) face-to-back (F2B).

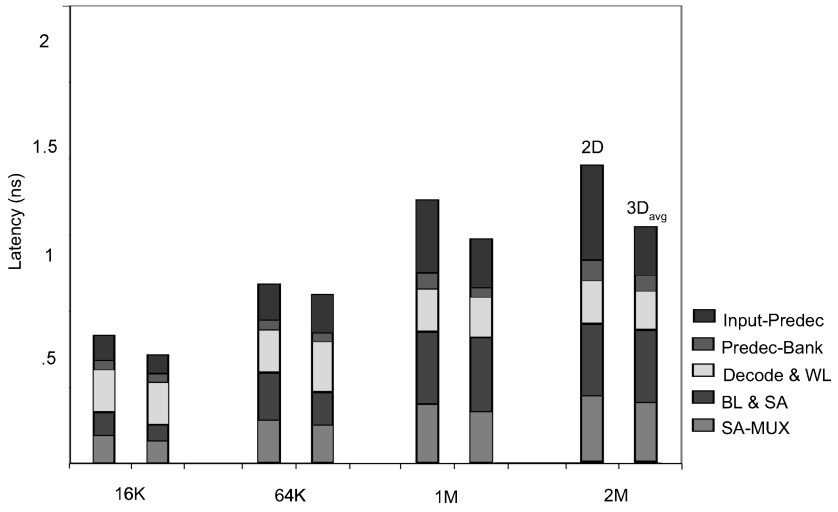

Fig. 2. Latency increase breakdown for increasing cache size.

the components. This design space is extended by vertical integration technology to include aspects such as multilayer implementations for each component and increased connectivity. For this purpose, we apply different partitioning techniques to model critical components in multiple layers and we analyze their area, delay, and power consumption. Even though the majority of performance improvement is due to the interblock global wiring in 3D integration, intrablock wiring latency is increasing as well. Figure 2 illustrates this increase in the wiring latency within the block and the corresponding breakdown analysis. The figure also illustrates the data-cache latency improvement possible through the use of 3D integration. However, it is important to note that this improvement is an average, and different folding techniques yield significantly different latency improvement depending on the size and characteristics of the architectural unit.

3.1 Multilayer Block Implementation Alternatives

To reduce intrablock interconnect latency and power consumption, we investigate two main strategies for designing blocks in multiple silicon layers: (i) *block folding* (*BF*) and (ii) *port partitioning* (*PP*). Block folding implies folding of the block in the x or y direction, potentially shortening the wirelength in

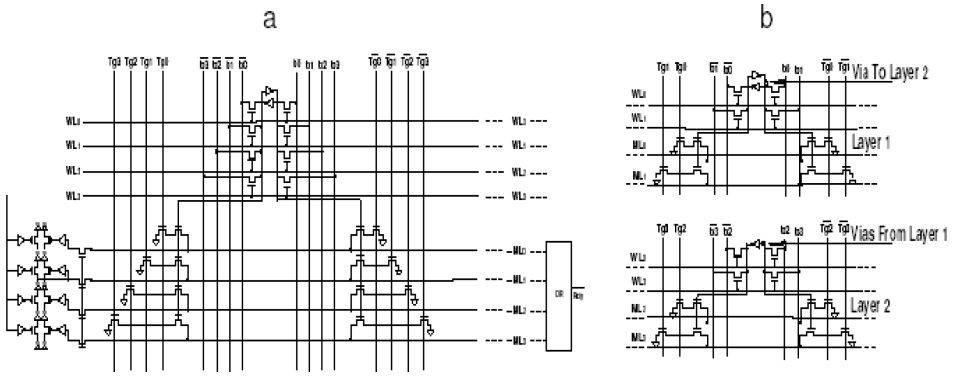

Fig. 3. (a) Single issue-queue cell with four tag lines and four access ports; most of the area is occupied by tags and access ports; (b) port partitioning: Tags and access ports are distributed into two layers. Width and height of each bit are reduced by half, and area by 75%.

one direction; port partitioning places the access ports of a cache-like structure in different layers. There are various architectural structures, including caches, register files, and wakeup and select logic. In this section we briefly describe the use of different partitioning strategies for issue queues and various cache-like blocks in our design driver architecture.

3.1.1 *Issue Queue.* Recent research [Cong et al. 2006] has shown that every additional pipeline stage of latency seen in the scheduling loop causes an average 5% performance degradation. Studies such as Folegnani et al. [2001] indicate that the average issue-queue power can comprise about 25% of a processor's total power consumption. For the purposes of this article, we study an issue queue based on Palacharla's implementations [Palacharla et al. 1997]. There are two main stages of issue-queue functionality: the wakeup stage where tags from completing register values are compared against input register tags stored in issue-queue entries, and a selection stage where ready instructions (as determined by the wakeup stage) are selected for execution.

Each issue-queue entry must track and compare the input register tags required by a given instruction in that entry. Figure 3 shows a single CAM cell used to store 1 bit of a register tag for an issue-queue entry. Assuming that at most four register values can be written back each cycle, and at most four new instructions can enter the issue queue each cycle, and an individual cell would have four 1-bit tags to compare against and have four write ports. In a processor with a 128-entry physical register file, register tags are composed of 7 bits. Therefore, each row would need 7 CAM cells for each operand, for a total of 14 CAM cells. In general, an N-entry issue queue has N such rows. In the wakeup stage, the match lines for each issue-queue entry are precharged high, and the tag lines are driven with the register tags of completed instructions. A match line only remains high if the register tag stored at the issue-queue entry is the same as a certain one of the register tags driven on the tag lines. If any match line for a given input register remains high, the ready bit for that operand is set in the issue queue. Once both ready bits are set, the operand is eligible

for issue (i.e., has woken up). In this stage, most of the delay comes from tag broadcasting and matching. In the selection stage, the select logic picks instructions to execute among all instructions that are eligible for issue [Palacharla et al. 1997]. For example, a selection tree for a 32-entry issue queue consists of three levels of arbiters. Each arbiter takes four input requests (i.e., four eligible instructions) and grants one request (i.e., selects one eligible instruction). In general, an N-entry issue queue needs a selection tree of level $L = \log_4 N$. In the issue queue, the delay due to wakeup logic contributes a large portion of the overall delay. Our simulations show that wakeup takes about 60% of the delay in a 32-entry issue queue with four incoming register tags to compare against, and four access ports. A significant contributor to delay is the wire latency of the tag bits and match lines. A 3D integrated issue queue can significantly reduce the length of these wires.

Issue-queue folding (BF). One way to reduce tag-line wire delay is to fold the issue-queue entries and place them on different layers. The issue queue is folded into two sets, and they are stacked in two layers. This approach effectively shortens the tag lines.

Issue-queue port partitioning (PP). For an issue queue with four tag comparison ports and four read/write ports, as shown in Figure 3(a), most of the silicon area is allocated to ports. The wire pitch is typically about five times the feature size [Palacharla et al. 1997; Reinman et al. 2000; Shivakumar et al. 2001]. For each extra port, the wirelength in both x and y directions is increased by twice the wire pitch [Reinman et al. 2000; Shivakumar et al. 2001]. On the other hand, the storage, which consists of four transistors, is twice the wire pitch in height, and has a width equal to the wire pitch. Hence, in the cell shown in Figure 3(a), the storage area is less than 1% of the total area, while tags and access ports occupy over 99% of the total area. One strategy for attacking the tag and port requirements is port partitioning, which places tag lines and ports on multiple layers, thus reducing both the height and width of the issue queue. The reduction in tag and match-line wirelength can help reduce both power and delay. The selection logic also benefits from this, as the distance from the farthest issue-queue entry to the arbiter is reduced. This will speed-up the comparison and also reduce power consumption.

3.1.2 *Caches.* Caches are regular structures composed of a number of tag and data arrays. Figure 4 shows a single memory cell for a 3-ported structure, which consists of four transistors that form two inverters: each inverts the output of the other. Each port contains bit, bitbar lines, a wordline, and two transistors per bit. For each extra port, the wirelength in both x and y directions is increased by twice the wire pitch. The wire pitch is five times the feature size [Folegnani et al. 2001; Farkas et al. 1996] in most designs. On the other hand, the storage, which consists of four transistors, is twice the wire pitch in height, and has a width equal to the wire pitch. Hence, in general, an increased number of ports in a cache-like structure corresponds to a larger port silicon area. Figure 5(a) demonstrates a high-level view of a number of cache tags and

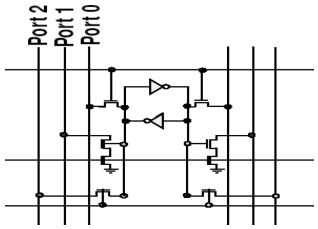

Fig. 4. A 3-ported SRAM cell.

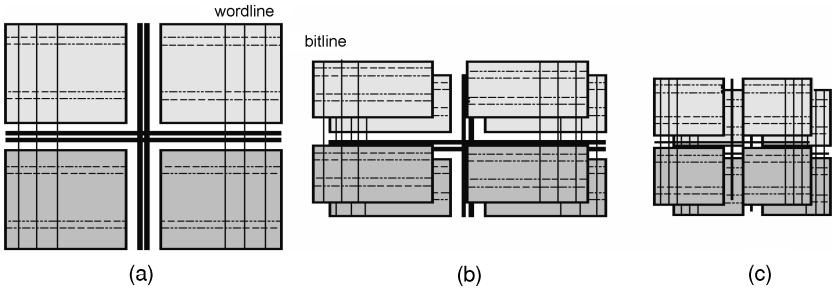

Fig. 5. 3D block alternatives for the cache: (a) a planar 2-ported cache, where the two lines denote the input/output wires of two ports; (b) wordline folding, where only the y direction is reduced, input/output ports duplicated; (c) port partitioning, where ports are placed on two layers, length in both x and y direction is reduced.

data arrays connected via address and data buses. Each vertical and horizontal line represents a 32-bit bus; we assume two ports on this cache, and therefore the lines are paired. Each box of the figure is a tag or data array, which is composed of a mesh of horizontal wordlines and vertical bitlines. Every port must have a wordline for each cache set and a pair of bitlines for each bit in a cache set. The regularity of caches means that their components can easily be subdivided. For example, the tag and data arrays can easily be broken down into subarrays. We make use of CACTI [Shivakumar et al. 2001] to explore the design space of different subdivisions and find an optimal point for performance, power, and area.

3D cache block folding (BF). We consider two options for block folding: wordline folding and bitline folding. In the former approach the wordlines in a cache subarray are divided and placed onto different silicon layers. The wordline driver is also duplicated. The gain from wordline folding comes from the shortened routing distance from predecoder to decoder and from output drivers to the edge of the cache. Similarly, bitline folding places bitlines into different layers, but needs to duplicate the pass transistors. Our analysis shows that wordline folding has a favorable access time and power dissipation in most cases compared to a realistic implementation of bitline folding. Hence, in the following sections we only present results using wordline folding.

3D cache port partitioning (PP). Partitioning the ports and placing them on different layers provides advantages, as shown in Figure 5(c). Port partitioning

allows reductions in both vertical and horizontal wirelengths. The width and height are simultaneously reduced by a factor of two, and the area by a factor of four. This reduces the total wirelength and capacitance, which translates into savings in access time and power consumption. Port partitioning requires vias to connect the memory cell to ports in other layers.

3.1.3 *Cache-Like Architectural Blocks.* We therefore adapt our CACTI to model register files because of their regular cache-like structure. Register files typically dissipate relatively large amounts of power due to their porting requirements, and the size of the physical register file can constrain the size of the instruction window in a dynamically scheduled superscalar processor. We will consider the same folding schemes for the register files as those used for caches. The register mapping units, load-store queue, and branch predictors can be approximated using only the data array portion of the cache.

3.2 Block Modeling

Each block has different implementation alternatives, according to the number of layers and different partitioning strategies. We can evaluate various alternatives using our block modeling approach. We assume a supply voltage of 1.0V and a 70nm process technology. Transistor and wire scaling parameters are derived from McFarland and Flynn [1995] and Tsai et al. [2005], and we assume copper interconnect in our simulation. Further transistor parameters are obtained from Cao et al. [2000]. 3D-CACTI [Tsai et al. 2005] is proposed as a tool to enable 3D exploration of caches and cache-like structures. However, it is restricted to wordline/bitline folding. We extended the 3D-CACTI framework by adding the capability of exploring port partitioning and block folding. Furthermore, we added an area estimation capability, including the area impact of 3D vias on the device layer. 3D-CACTI provides dynamic and leakage power models for caches and cache-like microarchitectural units. Via capacitance and resistance models are identical to Tsai et al. [2005]. We validated our modifications to 3D-CACTI with HSpice.

3.2.1 *Impact of 3D Technology.* In F2B designs, vias must pass through the device layer; this implies that space must be explicitly allocated for these vias to pass through. The insertion of vias will not only influence the area of the components but also the performance and power consumption for each block. In this study, the 3D via resistance is estimated to be 10^{-8}cm^2 [Tsai et al. 2005].

The height of the 3D vias is assumed to be 10μm per device layer. Current academic implementations of 3D via sizes vary from $1\mu\text{m} \times 1\mu\text{m}$ to $10\mu\text{m} \times 10\mu\text{m}$ [Tsai et al. 2005; Das et al. 2004]. As 3D bonding technology has advanced, 3D via size has rapidly scaled down. 3D via size has been reduced from 10μm to 1.75μm in MIT Lincoln Laboratory's 3D process technology [MIT 2006] at 180nm. Die-to-die via improvements have been announced in recent years [Li et al. 2006]. We expect 3D via size to continue to scale at smaller feature sizes. To demonstrate the impact of scaling via size, we plot the performance of the register file for via sizes ranging from 2.5μm to 0.5μm in 70nm technology. Via pitch is twice via size. The register file has four read ports and four write ports.

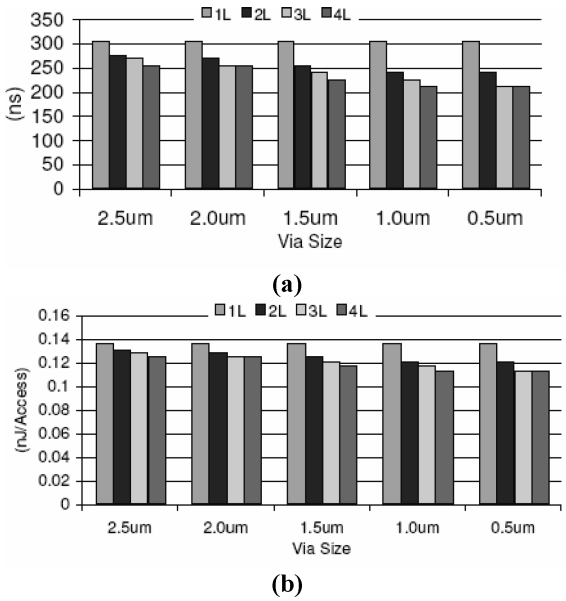

Fig. 6. Impact of via size: (a) impact of via size on timing using F2B port partitioning; (b) impact of via size on power using F2B port partitioning.

A single cell size is approximately $5.6\mu\text{m} \times 5.6\mu\text{m}$. In F2B bonding technology, 3D vias occupy silicon area in all layers except the bottom layer. Taking two-layer partitioning as an example, when via size is 2.5μm, the best solution is to place seven ports in the bottom layer and one port in the top layer, which only slightly reduces the wirelength. When via size is scaled to 0.5μm, the best solution places four ports in each layer. The wirelength is almost cut in half in both x and y directions. As shown in Figure 6, the reduction in wirelength reduces both delay and power as via size is scaled from 2.5μm to 0.5μm. As 3D technology advances, the 3D via size will decrease even further, yet it is important to note that even with the most aggressive TSV scaling trends such small pitches may not be achievable through wafer- or chip-level integration. Hence, sequential 3D integration based on silicon growth may be necessary for submicron TSV technologies to be possible [Xue et al. 2001]. In this study, we assume an aggressive via pitch of 1.4μm. An area of $0.7\mu\text{m} \times 0.7\mu\text{m}$ is reserved for each 3D via for the upper layers in F2B technology. We assume $7\mu\text{m} \times 7\mu\text{m}$ for power delivery vias and the thermal via infrastructure. We assume that the power vias can be arranged in clusters such that the microarchitectural unit layout and area are not impacted by the power delivery. Clock design for such fine-grain 3D integration is another important challenge; we assume that the signal vias are used for the clock distribution with minimum area overhead. Also, we assume periodic use of redundant vertical vias to synchronize the clock signal on multiple layers to minimize the skew.

3.2.2 *Scaling Architecture Sizes.* Increasing the microarchitectural unit sizes may cause extra cycles on critical loops in the microprocessor, which in

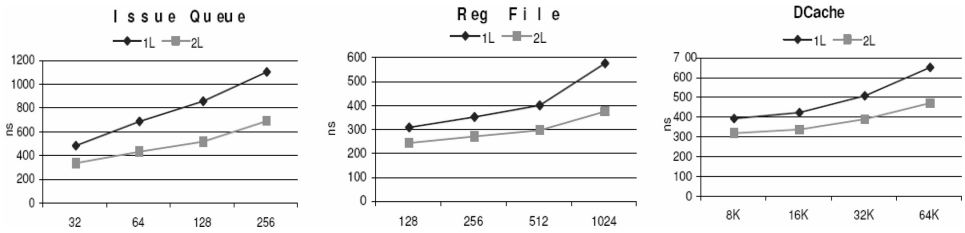

Fig. 7. Scaling the sizes of issue queue, register file, and data cache, and impact on latency for single-layer and two-layer cases (in nsec).

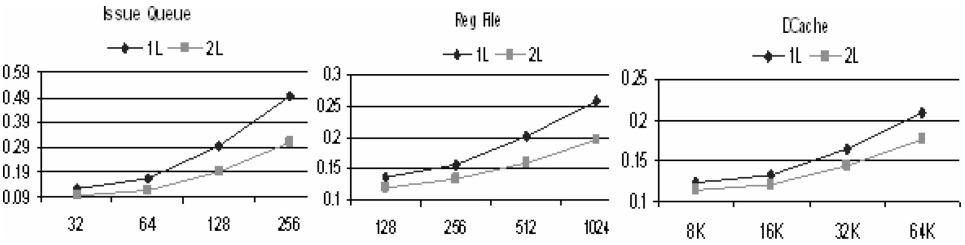

Fig. 8. Scaling the sizes of issue queue, register file, and data cache, and impact on power dissipation for single-layer and two-layer cases (in nW).

turn may even degrade the performance of the whole system. 3D implementation provides the possibility of reducing the delay for large-sized blocks. Figure 7 shows timing performance when three structures are scaled from default size to up to 16 times larger. With 3D integration technology, the access latency of increased-sized structures is still less than in that of 2D. The register file, and data cache can even quadruple their sizes while still outperforming the default blocks in 2D. Figure 8 shows the effect of issue queue, register file, and data cache size on power dissipation for single-layer and two-layer cases. 3D integration can reduce power dissipation when compared to the original 2D designs, while the larger 3D structures will dissipate more power than smaller-sized 3D blocks.

The size of architecture structure influences not only the area, delay, and power for individual blocks, but also the overall design of the microprocessors. By reducing the areas of blocks using 3D integration, the lengths of global wires that pass over these blocks may be reduced, thus providing useful timing slacks for circuit paths within microprocessors. Therefore, the increased latency in larger-sized structures may be tolerated within a given cycle of a critical loop, and the overall performance of the system can be improved. A packing-related analysis is necessary to investigate the effects of scaling architecture sizes for microarchitecture designs.

3.2.3 *Scaling to Multiple Silicon Layers.* We analyzed 3D implementations of various caches and other regular structures, including the instruction and data caches, register file, load-store unit, branch predictor, and issue queue. Further details of each unit are presented in Section 5, Table I. Among the experimental components, the issue queue has quite different characteristics

compared to the rest of the cache-like structures. To accurately model the issue-queue performance, we used HSpice simulations with a model similar to that in Palacharla et al. [1997]. The area is approximated by 3D-CACTI using a similarly configured cache. For all the simulations the supply voltage is 1.0V and the technology size is 70nm. Transistor and wire scaling parameters are derived from Tsai et al. [2005], McFarland and Flynn [1995], Cao et al. [2000], and we use copper interconnect in our simulations. Using these models, we quantified the gain of using 3D blocks in terms of area, timing, and power, as shown in Figure 9 for two, three, and four layers of silicon. All measurements are normalized to the performance of a single-layer block. In general, we observe that the reduction of area, power, and delay is further increased as the number of layers is increased. (Please note that the term "block area" refers to microarchitectural unit footprint in this context). For the issue queue (IQ) with PP, area reduction increases to 80% with three layers, and to 90% with four layers. Reduction in issue-queue delay increases to 43% with three layers, and to 50% with four layers. For the issue queue with block folding, there is less reduction in area and delay with additional layers. However, the impact on match-line wirelength from stacking more layers increases the power consumption for folding to 9% with four layers.

The following points were observed during our experimental analysis.

—*Area Reduction.* (Figure 9(a)) Port partitioning is consistently more effective for area reduction over all structures. This is because port partitioning reduces lengths in both X and Y directions.

—*Power and Timing Improvement.* The power or timing improvement in port partitioning does not increase with the number of layers when this number is larger than the number of ports. At the same time, the transistor layer must accommodate the size of vias for instruction cache, data cache, and RF. Four-layer designs do not outperform three-layer designs in terms of power or timing. On the other hand, with wordline folding, the trend continues, with consistent improvement for greater numbers of layers for most of the components. However, for IQs, the impact on match-line wirelength from stacking more layers increases the power consumption for folding to 9% with four layers.

—*Block Folding.* Block folding is more effective in reducing the block delay, especially for components with fewer ports. The data cache achieves up to 30% reduction in delay with BF, and a 23% reduction in delay with PP.

—*Multilayer Blocks.* Although multilayer blocks have reduced delay and power consumption compared to single-layer blocks, the worst-case power density may increase substantially by stacking circuits. Therefore, the power reduction in individual blocks alone cannot guarantee the elimination of hotspots. The thermal effect depends on not only the configuration of each block, but also on the physical information of the layout.

The diversity in benefit from these two approaches demonstrates the need for a tool to flexibly choose the appropriate implementation. The best 3D configuration of each component may not lead to the best 3D implementation for the

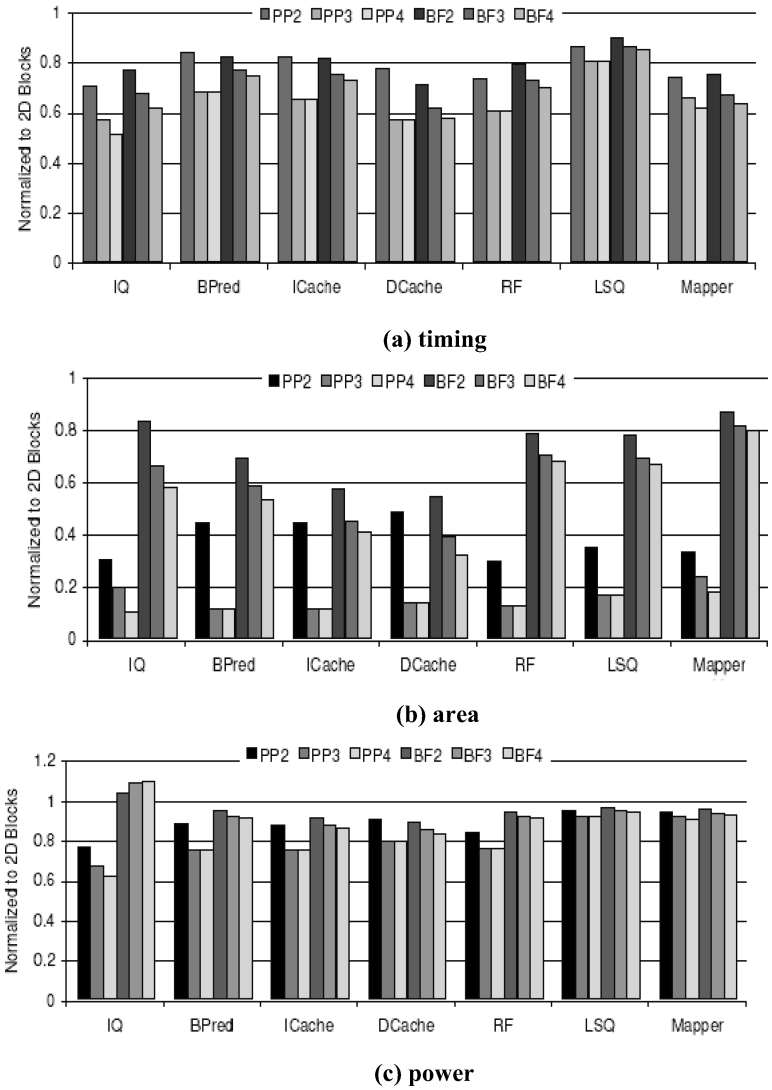

Fig. 9. Improvements for multilayer F2B design.

whole system. Furthermore, to favor the thermal effect, the reduction in delay of 3D blocks may provide the latency slack to allow a trade-off between timing and power. Therefore, fine-grain 3D integration needs to consist of more than purely architectural design optimization or physical design optimization. To enable the co-optimization between 3D microarchitectural and physical design, we need a 3D codesign engine that can choose the implementation while executing the packing optimization.

4. PHYSICAL EXPLORATION FOR 3D MICROARCHITECTURE DESIGN

Evaluating 3D architectures is a complex process that involves optimizing several parameters: frequency, die area, performance, white space for thermal via

insertion, and the temperature itself. With the various implementations for each critical component, the architecture design is partially defined. Without the physical information, it is impossible to obtain optimal implementations for components for the final chip. So the co-optimization of architecture design and physical design should also be able to choose the configurations for components, such as the number of layers, the partitioning approaches, etc.

As described in the previous section, each component is not restricted to a rectangle, but is likely to have cubic blocks, which have heights in the z-direction, in packing design. Therefore, a cube packing algorithm should be developed to arrange the given rectangular boxes in a rectangular box of minimum volume, without overlapping each other. And to evaluate a physical design for a processor, an efficient approach to estimate system performance should be developed based on the architecture structure and the model of components.

Therefore, our flow takes as input: (1) a microarchitecture description in terms of the parameters of the blocks, including the available sizes of the structures and number of functional units; (2) a target frequency for evaluating the microarchitecture; (3) a set of architecture-level critical paths whose latency will impact the performance of the microarchitecture, and the description of performance sensitivity for each critical path; and (4) estimated power density numbers for the blocks in the processor. Notice that items (3) and (4) are estimates of performance and power for those microarchitecture components that can drive a subsequent physical planning process.

We propose an automated floorplanner that can be configured to optimize the floorplan for die area, performance, and temperature while considering interconnect pipelining. We extended the CBL floorplanner based on a simulated annealing scheme [Hong et al. 2000; Ma et al. 2005] as the cube packing engine. This engine is flexible enough to dynamically choose the configurations of blocks while doing the packing, and allows one to configure the number of device layers for 3D integration. Thermal via insertion and global routing can be employed to obtain the optimized thermal and routing profile. Recent studies show that thermal via insertion can eliminate hotspots efficiently, which can reduce the peak temperature by 60% for four-layer designs [Li et al. 2006; Cong and Zhang 2005]. Once the exact block positions and wire latencies are known, this information is fed to our validation flow. A detailed cycle-accurate simulator that considers wire latency and power, and that is coupled with power and thermal models, is used to validate the results from our evaluation flow. In the sections that follow, we explain the components of our flow.

4.1 Performance Estimation

Our approach to calculating the performance of the processor during the floorplanning process is similar to the method used in Cong et al. [2006] and Jagannathan et al. [2005]. As the target frequency during floorplanning is fixed, we want to calculate the IPC degradation caused by the extra latency introduced by interconnect in the layout. Borch et al. [2002] showed that the IPC depends on a set of critical processor loops, and extra latency along these

loops can cause the IPC to degrade. For each critical path, we develop IPC performance sensitivity models, similar to Sprangle et al. [2002], which provides information about IPC degradation due to extra latency along the path. Recently, this kind of exploration has been used for performance improvement by Black et al. [2006]. However, that study was limited to single-layer blocks. Though the pipelining of the processors might need complex design, to obtain a high-level estimation of latency for the whole system , we assume inserting flip-flops on critical paths to maintain clock frequency. During floorplanning, we calculate the total latency of each critical path, including the blocks and wires, and determine the total number of cycles at the target frequency required to cover this path latency. Extra latency from the wires is used to compute the new IPC, and hence the performance of the processor for that floorplan.

4.2 3D Packing Algorithm

In previous 3D packing algorithms for microarchitecture design, each component can only occupy one layer, or the layer numbers for blocks are equal. Hence, the floorplanning algorithms used are merely extended 2D packing algorithms, not true 3D packing approaches. To enable the packing of 3D components which may occupy more than one layer, we constructed an architecture-driven packing engine that is a true 3D packing engine such that the 3D components in our design can be treated as cubic blocks to be packed in 3D space. The dimension in the z direction represents the layer information. The 3D packing algorithm is extended from the CBL floorplanner [Hong et al. 2000; Ma et al. 2005].

4.2.1 *Floorplanning for 3D Microarchitecture.* The floorplanning problem that we investigate here considers several components in its objective function that are important trade-offs in 3D architectures. Specifically, we consider the die area (footprint), the performance of the microarchitecture in *BIPS*, the maximum on-chip temperature, and the wirelength so that the interconnect power can be reduced. Formally, we define the problem as follows:

Given.

(1) target cycle time T_{cycle};
(2) clocking overhead $T_{overhead}$;
(3) target layer number of the chip Z_{con};
(4) list of blocks in the microarchitecture. Suppose for block i, there are k different implementations that are recorded in a candidate list as $\{c_1^i, c_2^i, \ldots c_k^i\}$. And each candidate c_2^i has the width(w_j^i), height(h_j^i), layer number (z_j^i), delay(d_j^i), and power(p_j^i); and
(5) set of critical microarchitectural paths with performance-sensitivity models for the paths.

Objective. Generate a floorplan that optimizes for the die area, performance, and maximum on-chip temperature.

Figure 10 shows the optimization flow based on a simulated annealing approach. Different than the previous simulated annealing-based floorplanning

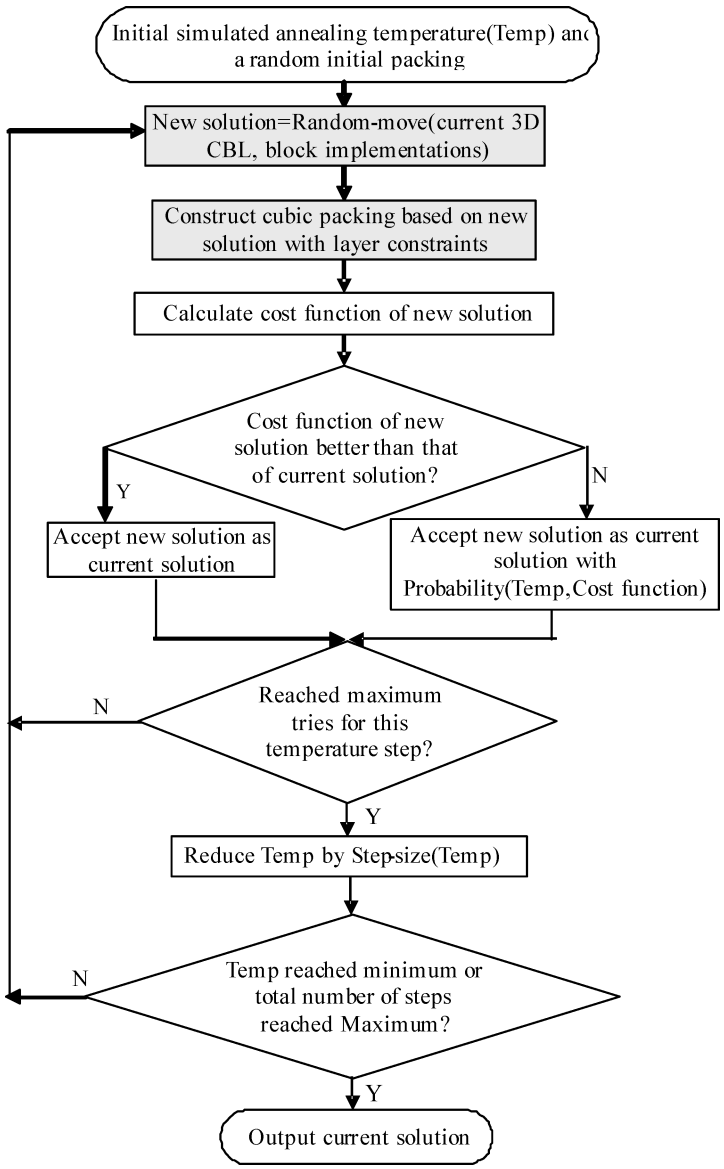

Fig. 10. Cube floorplanning flow based on simulated annealing approach.

approach, we not only extend the floorplanner to consider the performance of the design instead of a wirelength optimization objective, but also integrate the dynamically choosing of the blocks' configurations while doing the packing.

Our cost function uses a weighted combination of area, performance, and temperature and can be represented by

$$\cos t = w1 * \frac{1}{BIPS} + w2 * Area + w3 * Temp + w4 * \text{Wire},$$

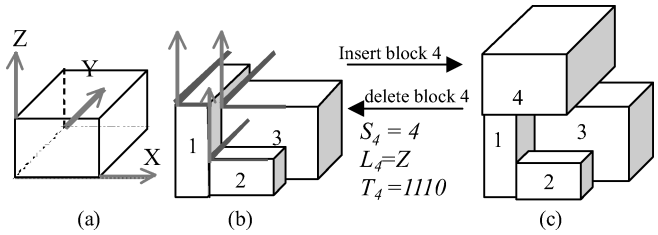

Fig. 11. The process of creating a corner cubic block: (a) x,y,z directions; (b) corner cubic block is 3 and uncovered block list in the z direction is {1,2,3}; (c) corner cubic block is 4, which covers 3 blocks in {1,2,3} since T4 = 1110; uncovered block list in the z direction becomes {4} and the corresponding 3D CBL: S = {1,2,3,4} L = {X, Y, Z} T = {10 10 1110}.

where BIPS corresponds to the performance of the microarchitecture. With this floorplan of the blocks, *Area* is the total area of the floorplan. The performance (BIPS) is calculated through the method presented in the previous section. *Temp* corresponds to the maximum on-chip temperature based on the CFD ACE+ temperature simulator [Wilkerson et al. 2004]. The coefficients of $w1$, $w2$, $w3$, and $w4$ are used to control the different weight for each component. In our test evaluation, the performance component is given a high weight and will be optimized when the simulated annealing engine tries to minimize the cost function.

4.2.2 *3D CBL Representation.* The 3D CBL representation records the topological covering relations between cubic blocks in the sequence from the lower-left front corner to the upper-right rear corner for a packing. The topology of cube packing is a system of relative relations between pairs of 3D blocks in such a way that block A is said to be left of block B when every point of A is left of every point of B. Relations of "right of," "above," "below," "front of," and "rear of" are analogously defined. Since we are representing the topological relations between blocks, the representation is independent of the sizes of blocks. The 3D floorplan divides the total packing region into cubic rooms with sides. Each cubic room is assigned to no more than one block. Therefore, to represent the topological relations in the packing, each cubic block is represented by a cubic room and rooms cover each other in the x, y, or z direction. During the packing process from the lower-left front corner to upper-right rear corner, if the room of block A covers the room of block B, then the room of block A is totally covered by a side and the side extension of the room of block B. As shown in Figure 11, we define the direction for each room with the direction in which it covers other rooms. Therefore, if we want to insert a new block 4 to the packing in Figure 11(b), the room of block 4 can cover those packed blocks {1,2,3} in x, y, or z direction. The new inserted block will be located at the upper-right rear corner so that it is defined as the *corner cubic block*. For each direction, not all the rooms of packed blocks are available to be covered, since some of them may have already been covered by the previous packed rooms. As shown in Figure 11(b), block 1 has already been covered by block 2 in the x direction; the new room of block 4 can only cover the rooms for block 2, block 3, or both

in the x direction. Therefore, we define the uncovered block list in a packing sequence for each direction, which records the current available blocks to be covered. In Figure 11(b), before block 4 is inserted, the uncovered block list in the z direction is {1,2,3}, the uncovered block list in the y direction is {1,3}, and the uncovered block list in the x direction is {2,3}.

With the covering direction and the uncovered block list in the corresponding direction, we still need information concerning which block or blocks should be covered to determine the position of the inserted block. Suppose that the uncovered block list in a direction in the packing sequence is $\{B_1, B_2, \ldots B_k\}$. Since the rooms in the CBL representation have flexible dimensions, if the room of B is placed to cover B_i, then B covers every room after B_i in the uncovered blocks list, and the room for block B does not cover the rooms before B_i. As shown in Figure 11(c), the uncovered block list in the z direction is {1,2,3}. If the room of block 4 covers the room of block 1, the room of block 4 will cover the rooms for blocks 2 and 3 at the same time. Therefore, to determine the position of inserted blocks, we can record the number of blocks covered by the room of this block in the uncovered block list. If m blocks will be covered by inserted block B,then the last m blocks in the uncovered block list should be covered by the room of B, so that the position of block B in the covering direction should be larger than any block from $\{B_{k-m+1}, \ldots B_k\}$ in this direction. With the new inserted block, the uncovered block list should be updated dynamically. The last m blocks $\{B_{k-m+1}, \ldots B_k\}$ are no longer available to be covered in this direction; hence, the updated uncovered block list after block B is inserted should be $\{B_1,\ldots B_{k-m}, B\}$.

Therefore, the information related to the packing process of the inserted corner block B should include the following: the block's name, the covering direction, and the number of blocks covered by B in the uncovered block list. To favor the generation of new solutions during the optimization process, we use a binary sequence T_i to record the number of blocks covered in the uncovered block list, in which the number of "1"s corresponds to the number of covered blocks. Each string of "1"s is ended with a 0 to separate it from the record of the next block. Given a 3D packing, we can have a sequence S of block names, a list L of orientations, and a list $\{T_2,T_3, \ldots, T_n\}$ of covering information. The three-element triple (S,L,T) composes a 3D CBL (as shown in Figure 11(c)).

To construct a packing based on a given CBL, the blocks are packed from the left-bottom front corner to the upper-right rear corner. For each insertion, the new corner block is packed at the upper-right rear corner of the current packing according to the corresponding direction in the 3D CBL, and all the blocks packed before are at the left of, or below, or in front of the current corner cubic block. Note that in a 3D CBL there are special cases where the number of successive "1"s in list T_i is greater than the number of available uncovered blocks in the corresponding direction. To amend this, we automatically insert a "0" when the number of successive "1"s in list $T$is greater than the uncovered blocks, so that the block will cover all the available uncovered blocks. Moreover, we can construct the floorplan accordingly, based on an arbitrary 3D CBL list. The transformation from a 3D CBL list to a cubic packing with n blocks can be described in the following algorithm.

Algorithm 3D CBL_Packing

Initialize the packing with cubic block $S[1]$;
Initialize the uncovered lists in three directions;
T-pointer = 0;
For i = 2 to n :
Uncovered_list = the uncovered list in $L[i\text{-}1]$ direction;
k= the length of *Uncovered_list*;
While*T*[*T_pointer*] ==*1 and k>0:*
$S[i]$ covers kth block and all blocks after kth block in uncovered_list from $L[i]$ direction and the coordinates of $S[i]$ is updated accordingly;
T_pointer++;
$k-$;
Update uncovered lists in all three directions: The blocks covered by $S[i]$ are deleted in uncovered list in $L[i\text{-}1]$ direction, $S[i]$ is added to all three uncovered lists.
End.

Based on the given 3D CBL list, we can construct the cubic floorplan accordingly in $O(n)$ time, where n is the number of blocks. However, the complexities of ST [Yamazaki et al. 2000] and 3D sub-TCG [Yuh et al. 2008] are $O(n^2)$[Kohira et al. 2006]. Therefore, 3D CBL can obtain good results in a shorter running time. And the major advantage of CBL representation is that the transformation from lists to packing is incrementally processed from lower left to upper right in linear time. Compared to graph-based representation, it is much easier to handle constraints by fixing the violations dynamically. In multilayer design, the z-dimension (height of the block) is given as a constraint. In the following section we propose a heuristic based on CBL representation to fix violations while doing the packing. This approach guarantees the feasibility of the final results and improves the converge process.

4.2.3 *Packing Optimization with Alternative 3D Block Implementations.* With the 3D modeling described in the previous section, the candidates vary in dimensions, delay, power consumption, and layer numbers according to different partitioning approaches. In 3D microarchitecture design, the number of chip layers is often given as a constraint. Our approach adopts a standard simulated annealing process. Therefore, to choose the best feasible configuration for blocks, we define the new operation, *Alternative_Selection*, to create a new solution.

Operation: Alternative_Selection.
Randomly choose a block i with multiple candidates;
Randomly choose a feasible candidatefrom the candidate list;
Update block i with the dimension of the chosen candidate.

The move used to generate a neighboring solution is based on any one of the following operations:

(1) randomly exchange the order of blocks in S;
(2) randomly choose a position in L and change the orientation;
(3) randomly choose a position in T, change “1” to “0”, or change “0” to “1”; or
(4) alternative_Selection.

The various candidates for components greatly enlarge the solution space. Especially with some layer-number constraints, parts of the solutions are infeasible. Therefore, heuristic methods are devised to speed-up the searching process.

4.2.4 *Packing with Layer Constraints.* During the packing process, the stacked blocks may violate the layer-number constraints. The traditional method penalizes the violations in the cost function. However, this method does not guarantee the feasibility of the final results and may slow down the convergence of the optimization. With 3D CBL representation, we pack the blocks in sequence. Therefore, we can dynamically change the blocks or CBL list during the packing. If some block exceeds the layer-number constraint, we can fix the violation by either lowering the block or changing the direction of the block. We take the following steps to fix the violation.

(i) To maintain the topology as much as possible, we first try to change the implementation of the block by choosing a candidate with a lower z-dimension.

(ii) If the violation cannot be fixed by changing the candidate, we try to modify the 3D CBL list to achieve a feasible packing. If block B covers previous blocks in the z direction, which means that block B will be placed on top of packed blocks, and if block B exceeds the layer-number constraint, we can change the covering direction to x or y so that block B can be placed on the right of or behind previous blocks. But if the z position of block B is still too high, we can dynamically move block B to the lower position by increasing the number of "1"s in $T_{B.}$ Since T_B signifies the number of blocks covered by B in the direction L_B, block B will be moved to lower blocks when we increase the number of "1"s in T_B. We can continue this process until block B satisfies the layer-number constraint.

Given the number of layers of the design Z_{con}, we scan the CBL list to pack the blocks from lower-left front corner to upper-right rear corner. The coordinates of the lower-left front corner for packed block B are (x_B, y_B, z_B) with the corresponding implementation c_j^B.

Hence the process can be described as the following algorithm.

Algorithm Fix_Violation

Input:
block B which exceeds the layer number constraint: $z_B + z_j^B > Z_{con}$;
3D_CBL and the candidate list for block B.
Output: New 3D_CBL with the new candidate selection c^B for B;
If $z_B < Z_{con}$
  For candidate c_j^B in candidate list of B
    If $z_B + z_j^B \leq Z_{con}$
      choose this candidate $c^B = c_j^B$ and update the positions of B;
      return;
    choose the candidate with the lowest z-height and update the information of B;
  If $L_B = Z$ // cover previous block from z-direction
    Change L_B to X or Y;

While ($z_B + z_j^B > Z_{con}$)
Increase the number of "1" in T_B^{LB} which means the number of blocks covered by B in the direction L_B is increased.
Update the position of B;
End.

The extreme case is that block B is moved to the bottom ($z_B = 0$). The candidate list should be constructed with the constraint that every block's z height should be less than Z_{con}. Block B will not exceed the layer-number constraint if $z_B = 0$. Therefore, our algorithm guarantees the feasibility of the results.

4.3 Performance Validation

Once the physical planning stage is finished, the critical-loop latencies and cycle time, along with the architectural configuration, are provided as input into our cycle-accurate simulation framework. We adapt the SimpleScalar 3.0 tool set [Burger et al. 1997], a suite of functional and timing simulation tools for the Alpha AXP ISA, for our simulation framework. We have made significant modifications to SimpleScalar to model the various speculative techniques and different configurations in this study. Our framework gives performance statistics in instructions per cycle (IPC) that can be combined with the cycle time from the floorplanning stage to give a result in BIPS.

5. EXPLORATION WITH A DESIGN DRIVER

We present the detailed evaluation results obtained for our design driver microarchitecture. Figure 12 shows the important elements of a typical high-performance superscalar processor. Table I shows the baseline processor parameters used in this study. We modified SimpleScalar [Burger et al. 1997] to model this architecture. Based on McFarland and Flynn [1995], we assume that the clock-cycle overhead is 46ps, which corresponds to roughly 1.8FO4 *(fan-out-of-four)* for 70nm technology. Thus, for a 4GHz target cycle time, we set the useful time for computation as 204ps and use this to calculate the number of pipeline stages required to cover a given path delay. The delay of interconnects is derived using the IPEM models [Cong et al. 1999], which consider several optimizations such as wire sizing, buffer insertion, buffer sizing, etc. To facilitate the insertion of repeaters, flip-flops, vias, etc., we assume that 10% of each block's area is reserved around the block in the floorplan. To perform our evaluation, results were collected for the SPEC2000 benchmarks.

5.1 Cube Packing Results

As described in previous sections, we model each critical component with different implementations. Given the layer-number constraints, our packing engine can pack the blocks successfully and choose the best implementation for each. In Figure 13 we show the packing results for 4GHz frequency. Figure 13(a) displays the best floorplan in terms of the performance we achieved for one-layer packing. The chip area is $4.9 \times 4.9 mm^2$ and BIPS is 2.34. Figure 13(b) displays a 3D view of the floorplan with the highest performance for two-layer packing

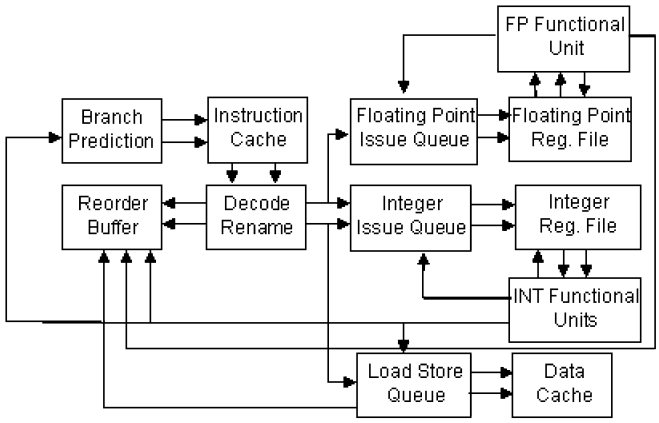

Fig. 12. Out-of-order, superscalar processor as the design driver.

Table I. Architectural Parameters for the Design Driver

Processor Width	6-way out-of-order superscalar, two integer execution clusters
Register Files	128 entry integer (two replicated files), 128 entry FP
Data Cache	8 KB 4-way set associative, 64B blocksize
Instruction Cache	8 KB 2-way set associative, 32B blocksize
L2 Cache	4 banks, each 128KB 8-way set associative, 128B blocksize
Branch Predictor	8K entry gshare and a 1K entry, 4-way BTB
Functional Units	2 IntALU+1 Int MULT/DIV in each of two clusters; 1 FPALU and 1MULT/DIV

with 3D blocks. The area is $3.6 \times 3.6 mm^2$and BIPS is 2.91. Our packing engine selects between single-layer or two-layer block architectures. For blocks such as ALU, MUL, and L2 cache units, single-layer implementation was selected. The rest of the blocks were implemented in two layers. (We use cubic blocks to represent multilayer blocks. All these multilayer blocks are placed on multiple layers.) A subset of blocks are partitioned by block folding, and the remainder by port partitioning. By choosing multilayer components, the delay along the critical path can be reduced, leading to a better performance result. Table II shows the number of cycles along critical loops in the packing results for different designs with 4GHz. Comparing the critical paths in Figures 13(a) and 13(b), the number of cycles along the branch misprediction loop is reduced from 21 to 15.

5.2 Performance Impact of 3D Integration

The intrablock delay can be reduced by implementing components in multiple layers. Therefore, to study the impact of multilayer blocks on the performance of the microarchitecture, we generated the best performance results for 2D and 3D block packing by running the floorplanning engine ten times and picking the best solution for each case. 2D architectural blocks are restricted to a single layer of silicon, whereas 3D architectural blocks span more than one layer of silicon, using the wordline or port folding techniques.

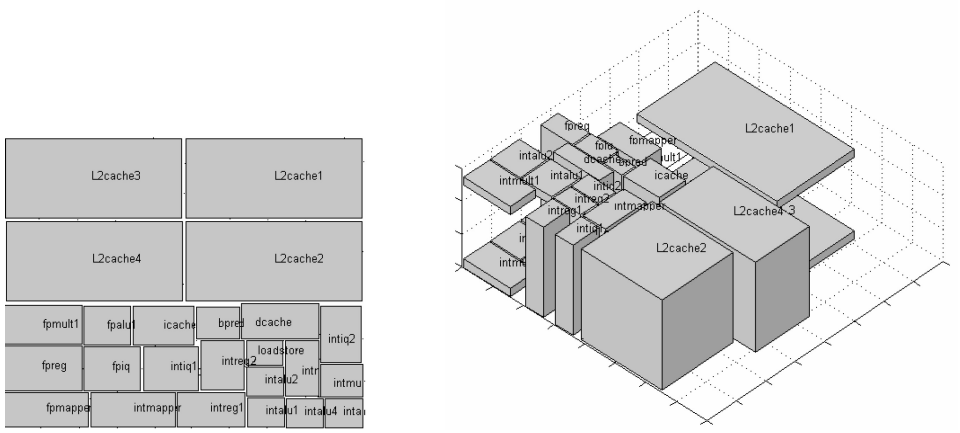

Fig. 13. Cubic packing with different layers: (a) packing on single chip layer; (b) 3D view of packing on two chip layers.

Table II. Number of Cycles along Critical Loops for 4GHz Frequency

	2D1L	2D2L	3D2L	3D3L	3D4L
Wakeup	5	4	4	3	3
DL1	6	5	4	4	4
L2	12	11	10	10	10
Branch Misprediction	21	18	15	16	14

2D1L means 2D architectural blocks packing on one layer, and 3D2L means 3D architectural blocks packing on two layers.

5.2.1 *3D Blocks versus 2D Blocks.* Figure 14 shows performance simulation results on SPEC2000 relative to a single-layer design. The first bar represents the benefit from using two layers of silicon with 2D blocks (as in Cong et al. [2006]), and the second bar represents that from using two layers of silicon with 3D blocks. All three configurations (single-layer and dual-layer 2D blocks and dual-layer 3D blocks) ran at 4GHz. On average, the use of 2D blocks in a two-layer design improves performance by 6%. Since the blocks themselves do not take advantage of vertical integration, any performance gain can only come from a reduction in the interblock wire latency. When we allow selecting 3D block alternatives, we see a performance improvement of 23% on average over the single-layer architecture. This can be attributed to the ability of 3D blocks to reduce the intrablock latency of critical processor loops (as shown in Table II). Therefore, in this 4GHz case in 3D chip design, because of greater reduction in wire latencies, using multilayer blocks can further improve the performance by about 16% over single-layer blocks.

5.2.2 *Clock Frequency and Number of Layers.* To explore the effect on the designs of different frequencies and layer numbers, in Table III we demonstrate the performance in BIPS when using different clock frequencies (3GHz–6GHz) and when using more silicon layers (1–4 layers of silicon). Vertical integration

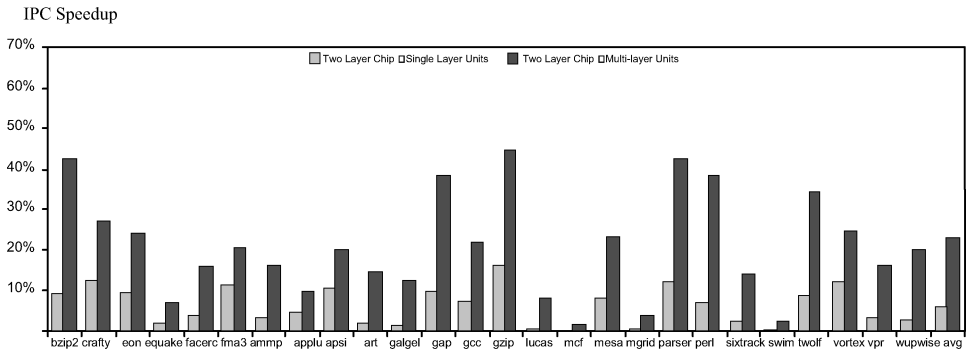

Fig. 14. Performance speedup for dual-layer architecture (2D/3D blocks) to single-layer design.

Table III. Performance Comparison of 2D/3D Blocks in BIPS for 3–6GHz, 1–4 Layers

	1L	2L		3L		4L	
Type of blocks		2D	3D	2D	3D	2D	3D
3G	2.09	2.2	2.70	2.38	2.83	2.8	2.91
4G	2.34	2.48	2.91	2.76	3.05	2.83	3.25
5G	2.48	2.65	3.19	3.01	3.40	3.2	3.58
6G	2.34	2.53	3.16	2.96	3.33	3.29	3.52
Normalized	1	1.07	1.29	1.20	1.36	1.31	1.43
Average		***2D / 1L***	1.19	***3D/1L***	1.36	***3D/2D***	1.14

with single-layer blocks can improve performance by about 14%. However, by allowing the use of multilayer blocks and optimizing the implementation with the packing process, chip performance can be improved by 36% on average and up to 43% for 4 layers. In order to evaluate the sensitivity of our approach to different frequencies, we compile results in BIPS for the designs with multilayer blocks, as shown in Figure 15. We can see better performance with an increase in the frequency and the number of layers. However, when the frequency increases to 6GHz, the BIPS drops a little. This is because the higher the frequency of the chip, the more degradation the extra latency will cause for chip performance. This trend is also true for the single-layer design and 3D design with single-layer blocks.

5.2.3 *Architecture Block Sizes.* Figure 16 illustrates the impact of block sizing on performance. On average, doubling the issue-queue size provided an additional 5% performance boost over the experimental benchmarks from the SPEC2000 suite. An additional 6% performance boost was achieved by doubling the sizes of the register file and data-cache as well. It is also important to note that the performance gain is not uniform over all experimental cases. For benchmarks such as swim and art, increasing the register file and data-cache size was quite effective, whereas doubling block sizes did not change the performance for other benchmarks such as mcf.

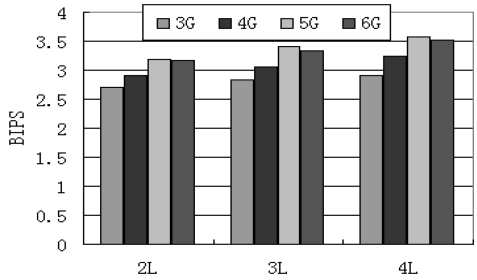

Fig. 15. Frequency impact on performance in multilayer implementations.

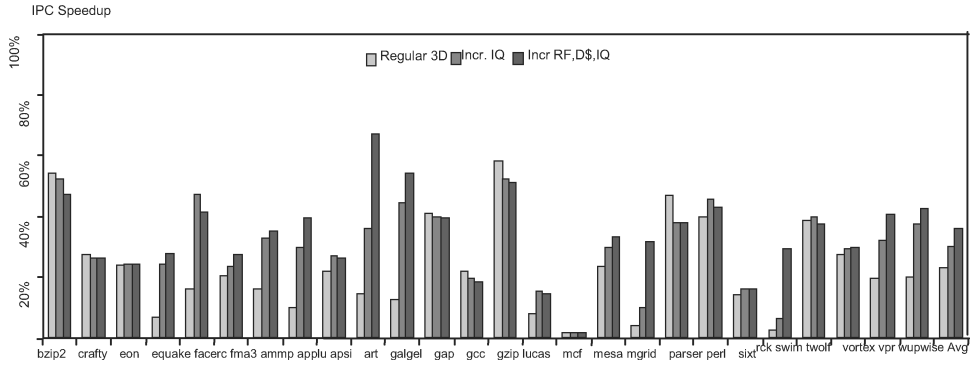

Fig. 16. Performance impact of doubling the sizes of: (i) issue queue; and (ii) register file, data cache, and issue queue.

5.3 Thermal Characteristics

The increased on-chip temperature profile of 3D integration is considered one of the greatest challenges in 3D design. Therefore, an accurate and fast thermal simulation framework is very crucial for design optimization. We use a state-of-the-art 3D thermal modeling tool, CFD ACE+ temperature simulator [Wilkerson et al. 2004], based on the finite element method (FEM), along with thermal via insertion [Cong and Zhang 2005]. Our thermal simulation infrastructure provides detailed thermal maps of individual layers with the corresponding local hotspots.

Figure 17 shows the temperature profiles of the layers in the two-layer design, where the top layer is significantly hotter than the bottom layer with a hotspot temperature of 90°C. The bottom layer, which is in contact with the heat spreader and heat sink, is cooler. Although the bottom layer has a higher power density compared to the top layer, the thermal resistance to the heat sink from the top layer is higher. Even though silicon is considered a good thermal conductor, vertical heat conduction is negatively affected by the combination of metal layers, bonding materials, and the increased distances. We look at using thermal vias that enable improved vertical heat conduction from the top layer to the heat sink. Using thermal vias seems effective in keeping the hotspot temperatures below thermal thresholds.

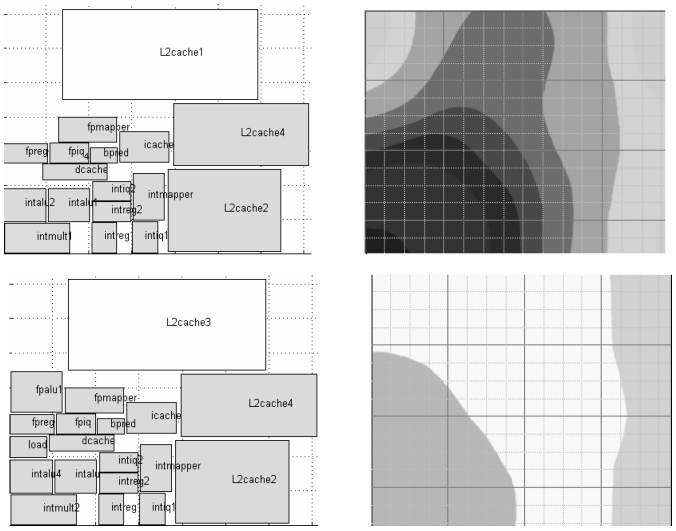

Fig. 17. Thermal maps for top and bottom layers (bottom layer is closer to the heat sink).

The results also demonstrate the importance of layer selection for thermal design of 3D integrated circuits. This is taken into consideration in our thermal-aware floorplanning algorithm. In general, layers farther away from the heat sink have considerably higher temperatures; this is due to the serial thermal resistances from heat source to heat sink in the corresponding thermal RC network.

Figure 18 illustrates the temperature comparison of the 2D and 3D architectural block technologies. The x-axis shows the different configurations with 2–4 silicon layers in the 3–5 GHz frequency range. The y-axis has the temperature in $^{\circ}$C for 3D and 2D block technologies and the results of thermal via insertion. The ambient temperature is assumed to be 27°C. On average, multilayer (3D) block configurations have 11% lower temperature due to the thermal optimization in the 3D packing process. As Section 3.2.2 shows, larger structures will dissipate more power than regular-sized 3D blocks. But despite the increase in power, the increased area of these larger designs will see an average slight change in temperature in the case where all three resources are increased for the initial area.

The previous section showed that multilayer blocks can save about 10–30% power consumption over single-layer blocks. But temperature heavily relies on the layout. To relieve hotspots, it is often necessary to keep potential hotspots away from one another. Even though single-layer blocks may seem to have advantages over multilayer blocks in this respect, our packing engine overcomes this issue by its intelligent layer selection for blocks depending on their thermal profile. Therefore, we can see that for two- and three-layer designs, temperatures can be reduced due to the power reduction of multilayer blocks and alternative selection in our engine.

Although multilayer blocks can reduce some power consumption inside blocks, the temperatures still display a nonlinear increase with an increased

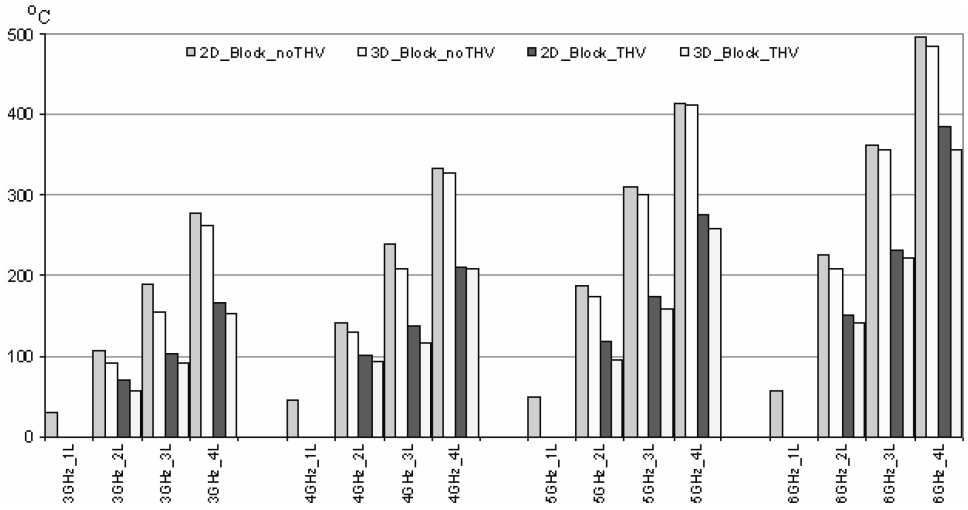

Fig. 18. Temperature comparison of 2D and 3D for 3–5GHz and 1–4-layer cases.

number of layers, as well as an elevation with higher frequencies. We see that without thermal via insertion, the temperatures are above 250°C for four-layer designs; these are out of the normal operation range of silicon. Cong et al. [2005] demonstrate the effects of thermal via insertion with floorplanning benchmarks: A four-layer design with peak temperature above 200°C can be cooled using thermal vias. In our test, through effective use of thermal via insertion, the temperatures are reduced. Therefore, by incorporating temperature-aware design planning, 3D architectures with multilayer blocks provide a 36% improvement in performance over 2D designs, a 14% improvement over a single-layer block 3D, and up to 43% improvement in the four-layer case.

6. CONCLUSIONS

Vertical integration has been shown to enable reduction in both interblock and intrablock wire latency. However, current research is limited to only exploiting interblock latency due to lack of tool infrastructure. In this study we investigate the effects of using multilayer blocks instead of constraining blocks in single-layer silicon. Our results indicate that effective use of multilayer architectural blocks reduces the impact of wires within a block through a reduction in block access time and/or power. We observed an average 36% increase performance in BIPS compared to the single-layer case (up to 43% in the four-layer case). Multilayer block integration provides a 14% improvement over the single-layer block case, along with an 11% reduction in average temperature. Experimental results show that vertical integration technology causes a significant temperature increase if blocks are restricted to single-layer silicon. However, temperature-aware design planning and use of thermal vias both enable on-chip temperatures below the critical thermal thresholds for the two-layer case.

REFERENCES

BANERJEE, K., SOURI, S., KAPUR, P., AND SARASWAT, K. 2001. 3D ICs: A novel chip design for improving deep-submicrometer interconnect performance and systems-on-chip integration. *Proc. IEEE 89*, 5, 602–633.

BLACK, B., ANNAVARAM, M., BREKELBAUM, N., DEVALE, J., JIANG, L., LOH, G. H., MCCAULEY, D., MORROW, P., NELSON, D. W., PANTUSO, D., REED, P., RUPLEY, J., SHANKAR, S., SHEN, J., AND WEBB, C. 2006. Die stacking (3D) microarchitecture. *Proceedings of the International Symposium on Microarchitecture*, 469–479.

BLACK, B., NELSON, D. W., WEBB, C., AND SAMRA, N. 2004. 3D processing technology and its impact on IA32 microprocessors. In *Proceedings of the International Conference on Computer Design*, 316–318.

BORCH, E., TUNE, E., MANNE, S., AND EMER, J. 2002. Loose loops sink chips. In *Proceedings of the IEEE International Symposium on High-Performance Computer Architecture*, 299–310.

BURGER, D. C. AND AUSTIN, T. M. 1997. The SimpleScalar tool set. Tech. Rep. CS-TR-97-1342, University of Wisconsin, Madison, Wisconsin. June.

BURNS, J. A., CHEN, C. K., KNECT, J. M., AND WYATT, P. W. 2006. A wafer-scale 3-D circuit integration technology. *IEEE Trans. Electron. Dev. 53*, 10, 2507–2516.

BURNS, J., MCLLRATH, L., KEAST, C., LEWIS, C., LOOMIS, A., WARNER, K., AND WYATT, P. 2001. Three dimensional integration for low power, high-bandwidth systems on a chip. In *Proceedings of the IEEE International Solid State Circuits Conference*, San Francisco, CA, 268–269

CAO, Y., SATO, T., SYLVESTER, D., ORSHANSKY, M., AND HU, C. 2000. New paradigm of predictive MOSFET and interconnect modeling for early circuit design. In *Proceedings of the Custom Integrated Circuit Conference*, 201–204.

CONG, J. AND PAN, D. Z. 1999. Interconnect estimation and planning for deep submicron designs. In *Proceedings of the 36th ACM/IEEE Conference on Design Automation*, 507–510.

CONG, J. AND ZANG, Y. 2005. A thermal-driven multilevel routing for 3-D ICS. In *Proceedings of the Asia Pacific Design Automation Conference*, 121–126.

CONG, J., JAGANNATHAN, A., MA, Y., REINMAN, G., WEI, J., AND ZHANG, Y. 2006. An automated design flow for 3D microarchitecture evaluation. In *Proceedings of the Asia and South Pacific Design Automation Conference*, 384–389.

CONG, J., WEI, J., AND ZHANG, Y. 2004. A thermal-driven floorplanning algorithm for 3D ICs. In *Proceedings of the ACM/IEEE International Conference on Computer Aided Design*, 306–313.

DAS, S., FAN, A., CHEN, K.-N., TAN, C. S., CHECKA, N., AND REIF, R. 2004. Technology, performance, and computer-aided design of three-dimensional integrated circuits. In *Proceedings of the International Symposium on Physical Design*, 108–115.

DAS, S., CHANDRAKASAN, A., AND REIF, R. 2003. Design tools for 3D integrated circuits. In *Proceedings of the Asia Pacific Design Automation Conference*, 53–56.

FARKAS, K., JOUPPI, N., AND CHOW, P. 1996. Register file design considerations in dynamically scheduled processors. In *Proceedings of the 2nd International Symposium on High Performance Computer Architecture*, 40–51.

FOLEGNANI, D. AND GONZALEZ, A. 2001. Energy-Effective issue logic. In *Proceedings of the 28th Annual International Symposium on Computer Architecture*, 230–239.

HONG, X. L., HUANG, G., CAI, Y. C., DONG, S. Q., CHENG, C. K., AND GU., J. 2000. Corner block list: An effective and efficient topological representation of non-slicing floorplan. In *Proceedings of International Conference on Computer Aided Design*, 8–12.

HUNG, W., LINK, G., XIE, Y., VIJAYKRISHNAN, N., AND IRWIN, M. 2006. Interconnect and thermal-aware floorplanning for 3D microprocessors. In *Proceedings of the International Symposium on Quality of Electronic Design*, 98–104.

JAGANNATHAN, A. , YANG, H. H., KONIGSFELD, K., MILLIRON, D., MOHAN, M., ROMESIS, M., REINMAN, G., AND CONG, J. 2005. Microarchitecture evaluation with floorplanning and interconnect pipelining. In *Proceedings of the Asia Pacific Design Automation Conference*, 8–15.

KLEINER, M. B., KUHN, S. A., RAMM, P., AND WEBER, W. 1996. Performance and improvement of the memory hierarchy of Risc-systems by application of 3-D technology. *IEEE Trans. Comp. Packag. Manufact. Technol. 19*, 4, 709–718.

KOHIRA, Y., KODAMA, C., FUJIYOSHI, K., AND TAKAHASHI, A. 2006. Evaluation of 3D-packing represnetations for scheduling of dynamically reconfigurable systems. In *Proceddings of the IEEE International Symposium on Circuits and Systems*, 4–8.

LI, F., NICOPOULOS, C., RICHARDSON, T., XIE, Y., NARAYANAN, V., AND KANDEMIR, M. 2006. Design and management of 3D chip multiprocessors using network-in-memory. In *Proceedings of the 33rd International Symposium on Computer Architecture*, Boston, MA, 130–141.

LI, Z., HONG, X., ZHOU, Q., ZENG, S., BIAN, J., YANG, H., PITCHUMANI, V., AND CHENG, C.-K. 2006. Integrating dynamic thermal via planning with 3D floorplanning algorithm. In *Proceedings of the ACM International Symposium on Physical Design*, 178–185.

LIU, Y., MA, Y., KURSUN, E., CONG, J., AND REINMAN, G. 2007. Fine grain 3D integration for microarchitecture design through cube packing exploration. In *Proceedings of the International Conference on Computer Design*, 259–266.

LOH, G. 2008a. 3D stacked memory architectures for multi-core processors. In *Proceedings of the International Symposium on Computer Architecture*, 453–464.

LOH, G. 2008b. A modular 3D processor for flexible product design and technology migration. In *Proceedings of the ACM International Conference on Computing Frontiers*, 157–170.

MCFARLAND, G. AND FLYNN, M. 1995. Limits of Scaling MOSFETS. Rep. CSL TR-95-62, Stanford University. November.

MA, Y., HONG, X., CHENG, C. K., AND DONG, S. 2005. 3D CBL: An efficient algorithm for general 3-dimensional packing problems. In *Proceedings of the IEEE the International Midwest Symposium on Circuits and Systems*, 1079–1082.

MIT LINCOLN LABORATORY. 2006. Mitll low-power FDSOI CMOS Process: Design guide. March.

PALACHARLA, S., JOUPPI, N. P., AND SMITH, J. E. 1997. Complexity-Effective superscalar processors. In *Proceedings of the 24th Annual International Symposium on Computer Architecture*, 206–218.

PUTTASWAMY, K. AND LOH, G. H. 2006a. Thermal analysis of a 3D die-stacked high-performance microprocessor. In *Proceedings of the ACM/IEEE Great Lakes Symposium on VLSI*, 19–24.

PUTTASWAMY, K. AND LOH, G. H. 2006b. The impact of 3-dimensional integration on the design of arithmetic units. In *Proceedings of the International Symposium on Circuits and Systems*, 4951–4954.

PUTTASWAMY K. AND LOH, G. H. 2006c. Dynamic instruction schedulers in a 3-dimensional integration technology. In *Proceedings of the ACM/IEEE Great Lakes Symposium on VLSI*, 153–158.

PUTTASWAMY, K. AND LOH, G. H. 2007. Scalability of 3D-integrated arithmetic units in high-performance microprocessors. In *Proceedings of the Design Automation Conference*, San Diego, CA, 622–625.

REINMAN, G. AND JOUPPI, N. 2000. Cacti 2.0: An integrated cache timing and power model. Tech. Rep. 2000/7. Palo Alto, Compaq, California. http://www.hpl.hp.com/techreports/Compaq-DEC/WRL-2000-7.pdf.

RONNEN, R., MENDELSON, A., LAI, K., LIU, S.-L., POLLACK, F., AND SHEN, J. P. 2001. Coming challenges in microarchitecture and architecture. *Proc. IEEE 89*, 3, 325–340.

SHIVAKUMAR, P. AND JOUPPI, N. 2001. Cacti 3.0: An integrated cache timing, power, and area model. Tech. Rep. Compaq, Palo Alto, California. www.hpl.hp.com/personal/Norman-Jouppi/eacti3.pdf.

SPRANGLE, E. AND CARMEAN, D. 2002. Increasing processor performance by implementing deeper pipelines. In *Proceedings of the 29th Annual International Symposium on Computer Architecture*, 25–34.

TOPOL, A. W., LA TULIPE, D. C., SHI, L., FRANK, D. J., BERNSTEIN, K., STEEN, S. E., KUMAR, A., SINGCO, G. U., YOUNG, A. M., GUARINI, K. W., AND IEONG, M. 2006. Three-Dimensional integrated circuits. *IBM J. Res. Develop. 50*, 4-5, 491–506.

TOPOL, A. W., LA TULIPE, D. C., SHI, L., ALAM, S. M., YOUNG, A. M., FRANK, D. J., STEEN, S. E., VICHICONTI, J., POSILLICO, D., CANAPERI, D. M., MEDD, S., CONTI, R. A., GOMA, S., DIMILIA, D., WANG, C., DELIGIANNI, L., COBB, M. A., JENKINS, K., KUMAR, A., KWIETNIAK, K. T., ROBSON, M., GIBSON, G. W., D'EMIC, C., NOWAK, E., JOSHI, R., GUARINI, K. W., AND IEONG, M. 2005. Enabling SOI-based assembly technology for three dimensional integrated circuits. In *Proceedings of the IEEE Interconnection Electron Devices Meeting*, 352–355.

TREMBLAY, M., JOY, B., AND SHIN, K. 1995. A three dimensional register file for superscalar processors. In *Proceedings of the 28th Hawaii International Conference on System Sciences*, 191–201.

TSAI, Y., XIE, Y., VIJAYKRISHNAN, N., AND IRWIN, M. 2005. Three-Dimensional cache design exploration using 3D CACTI. In *Proceedings of the International Conference on Computer Design*, 519–524.

TSUI, Y. K., LEE, S. W. R., WU, J. S., KIM, J. K., AND YUEN, M. M. F. 2003. Three-Dimensional packaging for multi-chip module with through-the-silicon via hole. In *Proceedings of the Electronics Packaging Technology*, 1–7.

WILKERSON, P., RAMAN, A., AND TUROWSKI, M. 2004. Fast, automated thermal simulation for three-dimensional integrated circuits. In *Proceedings of the Conference on Thermal and Thermomechanical Phenomena in Electronic Circuits*, 706–713.

XIE, Y., LOH, G. H., AND BLACK, B. 2007. Processor design in 3D die-stacking technologies. In *Proceedings of the IEEE Micro Conference*, 31–48.

XIE, Y., LOH, G. H., BLACK, B., AND BERNSTEIN, K. 2006. Design space exploration for 3D architectures. *ACM J. Emerg. Technol. Comput. Syst. 2*, 2, 65–103.

XUE, L., LIU, C., AND TIWARI, S. 2001. Multi-Layers with buried structures (MLBS): An approach to three-dimensional integration. In *Proceedings of the IEEE International Conference on Silicon on Insulator*, 117–118.

YAMAZAKI, H., SAKANUSHI, K., NAKATAKE, S., AND KAJITANI, Y. 2000. The 3D-packing by meta data structure and packing heuristics. *IEICE Trans. Fundam. E83-A*, 4, 639–645.

YUH, P. H., YANG, C.-L., CHANG, Y.-W. AND CHEN, H.-L. 2008. Temporal floorplanning using 3D-subTCG. In *Proceedings of the Asia Pasific Design Automation Conference*, 723–728.

Received December 2007; revised May 2008; accepted May 2008