Design Space Exploration for 3D Architectures

YUAN XIE
Pennsylvania State University
GABRIEL H. LOH
Georgia Institute of Technology
BRYAN BLACK
Intel Corporation
and
KERRY BERNSTEIN
IBM Corporation

As technology scales, interconnects have become a major performance bottleneck and a major source of power consumption for microprocessors. Increasing interconnect costs make it necessary to consider alternate ways of building modern microprocessors. One promising option is 3D architectures where a stack of multiple device layers with direct vertical tunneling through them are put together on the same chip. As fabrication of 3D integrated circuits has become viable, developing CAD tools and architectural techniques is imperative to explore the design space to 3D microarchitectures. In this article, we give a brief introduction to 3D integration technology, discuss the EDA design tools that can enable the adoption of 3D ICs, and present the implementation of various microprocessor components using 3D technology. An industrial case study is presented as an initial attempt to design 3D microarchitectures.

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, processor architectures, hardware, microarchitecture

Part of G. Loh's research is made possible by equipment and funding from Intel Corporation and funding from the Microelectronics Advanced Research Corporation.
Authors' addresses: Y. Xie, Computer Science and Engineering Department, Pennsylvania State University, University Park, PA, 16802; email: yuanxie@cse.psu.edu; G. Loh, College of Computing, Georgia Institute of Technology, Atlanta, GA, 30332; email: loh@cc.gatech.edu; B. Black, Intel Corporation; email: bryan.black@intel.com; K. Bernstein, IBM Corporation; email: kbernstein@us.ibm.com.
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or direct commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 1515 Broadway, New York, NY 10036 USA, fax: +1 (212) 869-0481, or permissions@acm.org.
© 2006 ACM 1550-4832/06/0400-0065 $5.00

1. INTRODUCTION

As technology scales, the International Technology Roadmap for Semiconductors projects that on-chip communications will require new design approaches to achieve system-level performance targets.

Aggressive scaling of process technologies has enabled feature sizes to shrink continuously. While the performance of gates has been improving, interconnects have become a major performance bottleneck because global interconnects do not scale accordingly with technologies. In order to keep the delays of these long wires tractable, repeaters and flip-flops are inserted to prevent performance degradation. However, these additional components have a detrimental impact on interconnect power dissipation. Consequently, intermediate and global interconnects of current microprocessors contribute to a major portion of power consumption and also serve as impediments for better performance. Hence, many research efforts are devoted to seeking solutions which can overcome the limitation of wiring requirements for present and future chip designs.

Three-dimensional integrated circuits (3D ICs) [Souri et al. 2000; Das et al. 2004] are attractive options for overcoming the barriers in interconnect scaling, offering an opportunity to continue the CMOS performance trend. In a three-dimensional (3D) chip, multiple device layers are stacked together with direct vertical interconnects tunneling through them (Figure 1 shows a conceptual 2-layer 3D integrated circuit [Deng et al. 2004]). The direct vertical interconnects are called *interwafer vias* or *die-to-die (d2d) vias*. Consequently, one of the most important benefits of a 3D chip over a traditional two-dimensional (2D) design is the reduction on global interconnect. Other benefits of 3D ICs include: (i) higher packing density and smaller footprint due to the addition of a third dimension to the conventional two-dimensional layout; (ii) higher performance due to reduced average interconnect length; (iii) lower interconnect power consumption due to the reduction in total wiring length; and (iv) support for realization of mixed-technology chips.

Even though 3D integrated circuits show great benefits, there are several challenges for the adoption of 3D architectures. First, there are few commercially available EDA tools and design methodologies for 3D integrated circuits. Second, the move from 2D to 3D architecture could accentuate the thermal concerns due to the increased power densities that result from placing one logic block over another in the multilayered 3D stack. Third, design space exploration at the architectural level is essential to fully take advantage of the 3D integration technologies and build a high performance microprocessor.

This article provides an introduction to the design space exploration for 3D architectures. It begins with an overview of the 3D integration technologies and discusses the EDA design tools for designing 3D microarchitectures. We then present the implementation of various processor components as well as an industrial case study on mapping a Pentium 4™ derivative microprocessor onto a 3D design space.

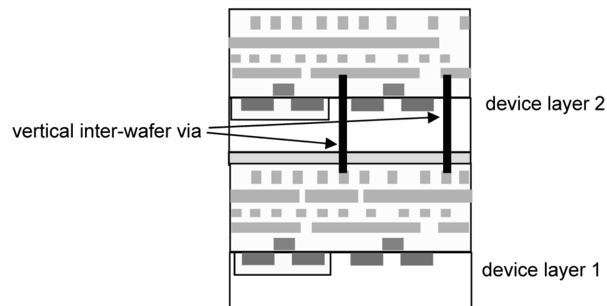


Fig. 1. A conceptual 3D IC: two device layers are stacked together with direct vertical interconnects tunneling through them [Deng et al. 2004].

2. 3D INTEGRATION TECHNOLOGIES

There are various vertical interconnect technologies that have been explored, including wire bonded, microbump, contactless (capacitive or inductive), and through-via vertical interconnect. A comparison in terms of vertical density and practical limits can be found in Davis et al. [2005]. Through-via interconnection has the potential to offer the greatest vertical interconnect density and therefore is the most promising among the vertical interconnect technologies. Most of the through-via technologies can be classified into one of the two following categories.

(1) *Bottom-up approach.* This approach involves sequential device process. An example of this approach is Multilayer Buried Structures (MLBS) technology [Xue et al. 2001; Jung et al. 2004]. The frontend processing (to build the device layer) is repeated on a single wafer to build multiple active device layers before the backend processing builds interconnects among devices.
(2) *Top-down approach.* This approach processes each active device layer separately, using conventional fabrication techniques. These multiple device layers are then assembled to build up 3D IC, using wafer-bonding technology [Reif et al. 2002]. Wafers can be bonded face-to-face (F2F) or face-to-back (F2B). Figure 1 illustrates a face-to-back wafer bonding approach. The through wafer via in face-to-face wafer bonding does not go through a thick buried Si layer and can be fabricated with smaller via sizes. However, for 3D IC's with more than two active layers, back-to-back (B2B) wafer-bonding technology is inevitable.

Compared to MLBS technology, wafer-bonding integration requires minimal changes to the manufacture process steps. Another key difference between these two technologies that can influence the architecture partitioning strategy is the size of the vertical 3D vias which provide connections between different active device layers. In wafer bonding, the dimension of the 3D vias is not expected to scale at the same rate as feature size because wafer-to-wafer alignment tolerance during bonding poses limitation on the scaling of the vias [Das et al. 2004]. Current dimensions of 3D via sizes vary from 1 μm by 1 μm to 10 μm by 10 μm [Bernstein 2006]. The relatively large size of 3D vias can

hinder partitioning a design at very fine granularity across multiple device layers.

The MLBS provides more flexibility in vertical 3D connection because the vertical 3D via can potentially scale down with feature size due to the use of local wires for connection [Xue et al. 2001]. Availability of such technologies makes it possible to partition the cache at the granularity of individual cache cells [Kang et al. 2004]. However, wafer bonding requires fewer changes in the manufacturing process and is more popular in industry [Mayega et al. 2003; Lee et al. 2000] than MLBS technology. Therefore, the integration approach we adopt in this study is the wafer-bonding technology.

3. EDA DESIGN TOOLS AND METHODOLOGIES TO ENABLE 3D TECHNOLOGY

3D integration technology will not be commercially viable without the support of EDA tools and methodologies that allow architects and circuit designers to develop new architectures or circuits using this technology. To efficiently exploit the benefits of 3D technologies, design tools and methodologies to support 3D designs are imperative. This section give examples of CAD techniques which could be the key enablers for the adoption of 3D technology for microarchitecture design. EDA design tools that are essential for 3D microarchitecture design adoptions can be divided into two different categories; *early design analysis tools* and *physical design tools*. In the following sections, we first describe 3DCacti which explores the architectural design space of cache memories using 3D structures at the early design stage; we then present a thermal-aware floorplanning tool for 3D microprocessor design.

3.1 3DCacti: An Early Analysis Tool for 3D Cache Design

To justify 3D cost overhead, it is essential to study the benefits of 3D integration at the early design cycle. It usually requires a strong link between architectural analysis tools and 3D physical planning tools. The early analysis tools study the trade-offs among the number of layers, power density, and performance.

In this section we describe an example of early design analysis tools for 3D IC design, 3DCacti which explores the architectural design space of cache memories using 3D structures. Since interconnects dominate the delay of cache accesses which determines the critical path of a microprocessor, the exploration of benefits from advanced technologies is particularly important. The regular structure and long wires in a cache make it one of the best candidates for 3D designs. A tool to predict the delay and energy of a cache at the early design stage is crucial because the timing profile and the optimized configurations of cache depend on the number of active device levels available as well as the way a cache is partitioned into different active device layers. This section examines possible partitioning techniques for caches designed using 3D structures and presents a delay and energy model to explore different options for partitioning a cache across different device layers.

3.1.1 *3D Cache Partitioning Strategies.* In this section, we discuss different approaches to partition a cache into multiple device layers.

The finest granularity of partitioning a cache is at the SRAM cell level. At this level of partitioning, any of the six transistors of a SRAM cell can be assigned to any layer. For example, the pull-up PMOS transistors can be in one device layer, and the access transistors and the pull-down NMOS transistors can be in another layer. The benefits of cell-level partitioning include the reduction in footprint of the cache arrays and, consequently, the routing distance of the global signals. However, the feasibility of partitioning at this level is constrained by the 3D via size as compared to the SRAM cell size. Assuming a limitation that the aspect ratio of 3D via size cannot be scaled less than $1\ \mu m \times 1\ \mu m$, the 3D via has a comparable size to that of a 2D 6T SRAM cell in 180nm technology and is much larger than a single cell in 70nm technology. Consequently, when the 3D via size does not scale with feature size (which is the case for wafer-bonding 3D integration), partitioning at the cell level is not feasible. In contrast, partitioning at the SRAM cell level is feasible in technologies such as MLBS because no limitation is imposed on via scaling with feature size. However, it should be noted that even if the size of a 3D via can be scaled to as small as a nominal contact in a given technology, the total SRAM cell area reduction (as compared to a 2D design) due to the use of additional layers is limited because metal routing and contacts occupy a significant portion of the 2D SRAM cell area [Zhang et al. 2004]. Consequently, partitioning at a higher level of granularity is more practical. For example, individual subarrays in the 2D cache can be partitioned across multiple device layers. The partitioning at this granularity reduces the footprint of cache arrays and routing lengths of global signals. However, it also changes the complexity of the peripheral circuits. In this section, we consider two options for partitioning the subarray into multiple layers: 3D divided wordline (3DWL) strategy and 3D divided bit line strategy (3DBL).

3D Divided Wordline (3DWL). In this partitioning strategy, the wordlines in a subarray are divided and mapped onto different active device layers (See Figure 4). The corresponding local wordline decoder of the original wordline in 2D subarray is placed on one layer and is used to feed the wordline drivers on different layers through the 3D vias. we duplicate word line drivers for each layer. The duplication overhead is offset by the resized drivers for a smaller capacitive load on the partitioned wordline. Further, the delay time of pulling a wordline decreases as the number of pass transistors connected to a wordline driver is smaller. The delay calculation of the 3DWL also accounts for the 3D via area utilization. The area overhead due to 3D vias is small compared to the number of cells on a wordline.

Another benefit from 3DWL is that the length of the address line from the periphery of the core to the wordline decoder decreases proportionally to the number of device layers. Similarly, the routing distance between the output of the predecoder to the local decoder is reduced. The select lines for the writes and muxes as well as the wires from the output drivers to the periphery also have shorter length.

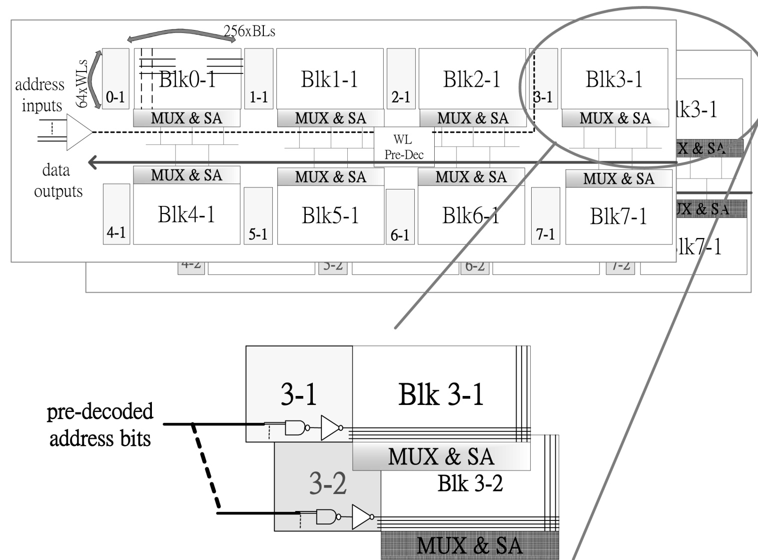


Fig. 2. Cache with 3D divided bitline partitioning and mapped into 2 active device layers.

3D Divided Bitline (3DBL). This approach is akin to the 3DWL strategy and applies partitioning to the bitlines of a subarray (See Figure 2). The bitline length in the subarray and the number of the pass transistors connected to a single bitline are reduced. In the 3DBL approach, the sense amplifiers can either be duplicated across different device layers or shared among the partitioned subarrays in different layers. The former approach is more suitable for reducing access time, while the latter is preferred for reducing the number of transistors and leakage. In the latter approach, the sharing increases complexity of multiplexing of bitlines and reduces performance as compared to the former. Similar to 3DWL, the length of the global lines are reduced in this scheme.

3.1.2 *3D Cache Delay-Energy Estimator (3DCACTI).* In order to explore the 3D cache design space, a 3D cache delay-energy estimation tool called 3DCacti was developed [Tsai et al. 2005]. The tool was built on top of the Cacti 3.0 2D cache tool (Cache Access and Cycle Time) [Shivakumar et al. 2001]. 3DCacti searches for the optimized configuration that explores the best delay, power, and area efficiency trade-off according to the cost function for a given number of 3D device layers.

In the original Cacti tool [Shivakumar et al. 2001], several configuration parameters (see Table I) are used to divide a cache into subarrays to explore delay, energy, and area efficiency trade-offs. In 3DCacti implementation, two additional parameters, Nx and Ny, are added to model the intrasubarray 3D partitions. The additional effects of varying each parameter other than the impact on length of global routing signals are listed in Table I. Note that the tag array is optimized independently of the data array and the configuration parameters for tag array: Ntwl, Ntbl, and Ntspd are not listed.

Table I. Design Parameters for 3DCacti and Their Impact on Cache Design

Parameter	Definition	Effect on Cache Design
Ndbl	the number of cuts on a cache to divide bitlines	1. the bitline length in each sub-array 2. the number of sense amplifiers 3. the size of wordline driver 4. the decoder complexity 5. the multiplexors complexity in data output path
Ndwl	the number of cuts on a cache to divide wordlines	1. the wordline length in each sub-array 2. the number of wordline drivers 3. the decoder complexity
Nspd	the number of sets connected to a wordline	1. the wordline length in each sub-array 2. the size of wordline drivers 3. the multiplexors complexity in data output path
Nx	the number of 3D partitions by dividing wordlines	1. the wordline length in each sub-array 2. the size of wordline driver
Ny	the number of 3D partitions by dividing bitlines	1. the bitline length in each sub-array 2. the complexity in multiplexors in data output path

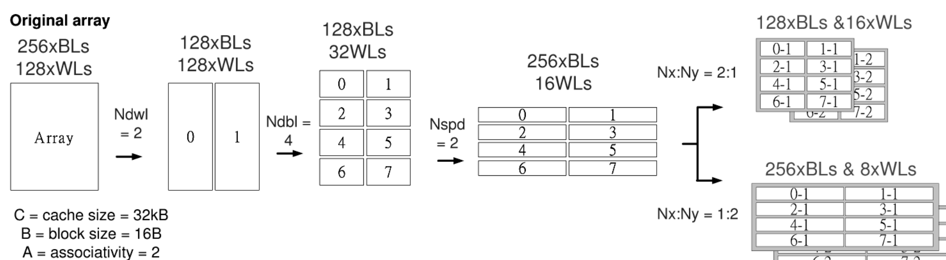


Fig. 3. Cache with 3D divided bitline partitioning and mapped into 2 active device layers.

Figure 3 shows an example of how these configuration parameters used in 3DCacti affect the cache structure. The cell-level partitioning approach (using MLBS) is implicitly simulated using a different cell width and height within Cacti.

3.1.3 *Design Exploration Using 3DCacti.* By using 3DCacti, we can explore various 3D partitioning options of caches to understand their impact on delay and power at the very early design stage. Note that the data presented in this section is in 70nm technology, assuming 1 read/write port and 1 bank in each cache unless otherwise stated.

First, we explore the best configurations for various degrees of 3DWL and 3DBL in terms of delay. Figure 5 and Figure 6 show the access delay and energy consumption per access for 4-way set associative caches of various sizes and different 3D partitioning settings. Recollect that Nx (Ny) in the configuration refers to the degree of 3DWL (3DBL) partitioning. First, we observe that delay reduces as the number of layers increase. From Figure 7, we observe that the reduction in global wiring length of the decoder is the main reason for delay reduction benefit. We also observe that for the 2-layer case, the partitioning of

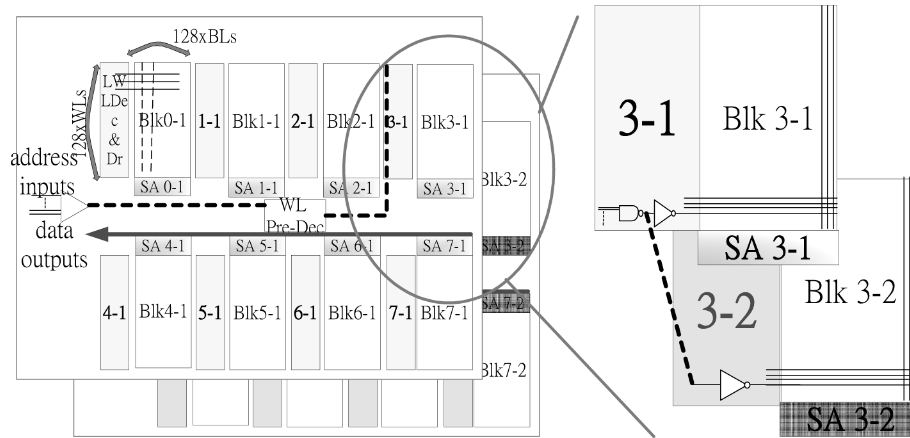


Fig. 4. An example showing how each configuration parameter affects a cache structure. Each box is a subarray associated with an independent decoder.

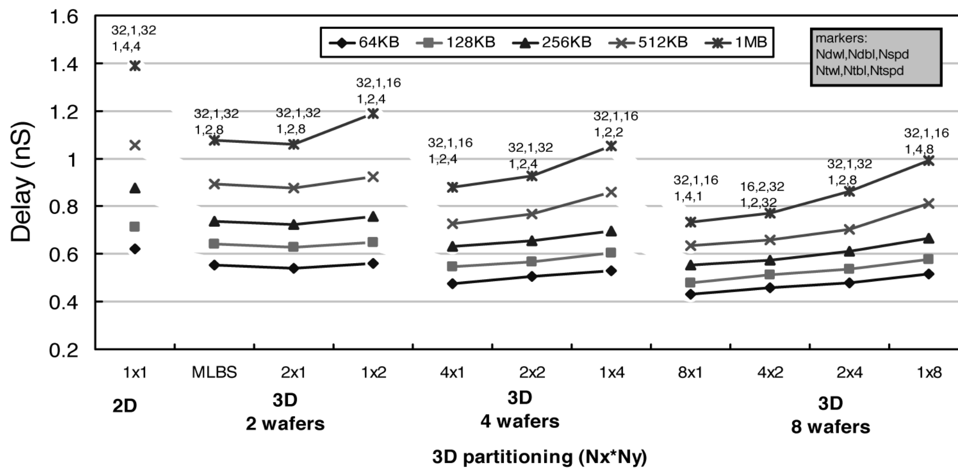


Fig. 5. Access time for different partitioning. Data of caches of associativity = 4 are shown.

a single cell using MLBS provides delay reduction benefits similar to the best intrasubarray partitioning technique compared to the 2D design.

Another general trend observed for all cache sizes is that partitioning more aggressively using 3DWL results in faster access time. For example, in the 4-layer case, the configuration 4*1 has an access time which is 16.3% less than that of the 1*4 configuration for a 1MB cache. We observed that the benefits from more aggressive 3DWL stem from the longer length of the global wires in the X direction compared to the Y direction before 3D partitioning is performed. The preference for shorter bitlines for delay minimization in each of the subarrays and the resulting wider subarrays in optimal 2D configuration is the reason for the difference in wire lengths along the two directions. For example, in Figure 9(a), the best subarray configuration for the 1MB cache in 2D design results in a longer global wire length in the X direction. Consequently, when

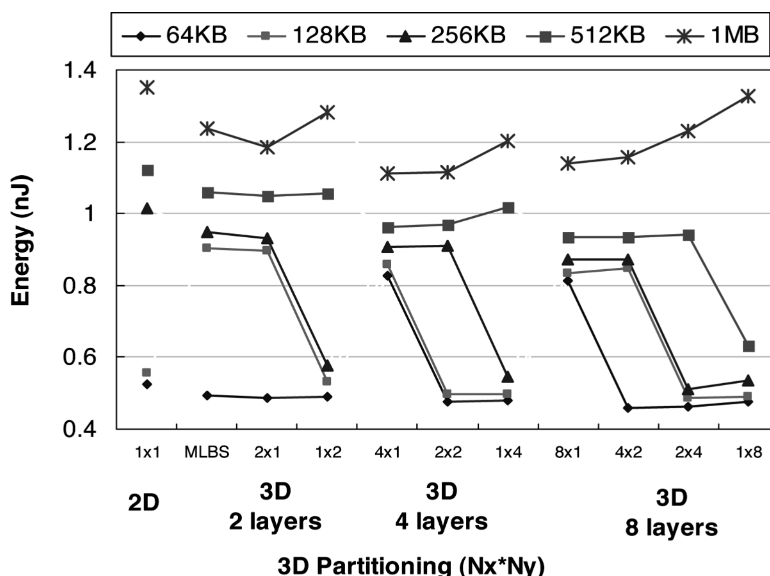


Fig. 6. Energy for different partitioning when setting the weightage of delay higher. Data of caches of associativity = 4 are shown.

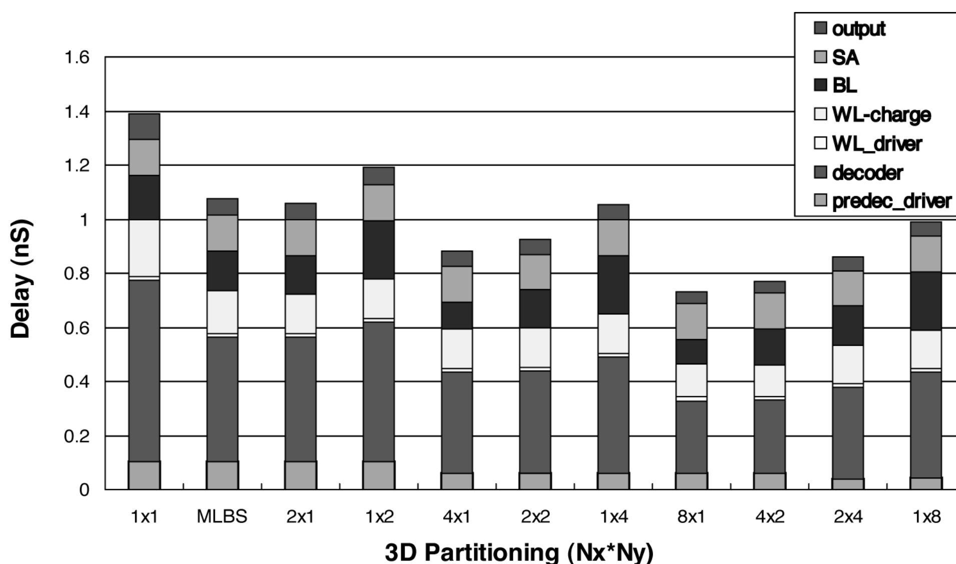


Fig. 7. Access time breakdown of an 1MB cache corresponding to the results shown in Figure 7.

wordlines are divided along the third dimension, more significant reduction in critical global wiring lengths can be achieved. Note that because 3DCacti is exploring partitioning across the dimensions simultaneously, some configurations can result in 2D configurations that have wirelengths greater in the Y directions (See Figure 9(c)) as in the 1MB cache 1*2 configuration for 2 layers. The 3DBL helps in reducing the global wire length delays by reducing the Y direction

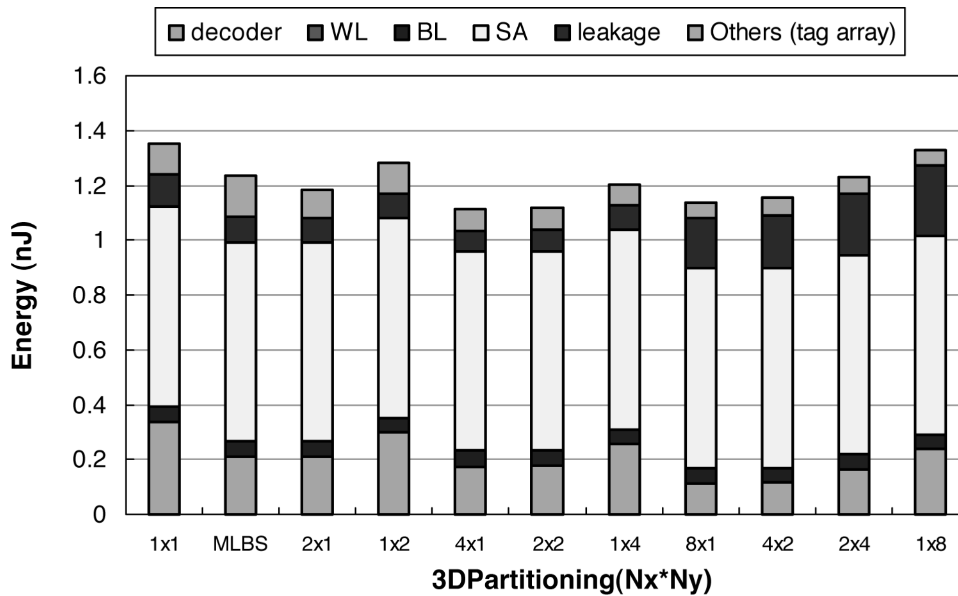


Fig. 8. Energy breakdown of a 1MB cache corresponding to the results shown in Figure 8.

length. However, it is still not as effective as the corresponding 2*1 configuration since both the bitline delays in the core and the routing delays are larger (See Figure 7 and Figure 8). These trends are difficult to analyze without the help of a tool to partition across multiple dimensions simultaneously. The energy reduction for the corresponding best delay configurations tracks the delay reduction in many cases. For example, the energy of 1MB cache increases when moving from a 8*1 configuration to a 1*8 configuration. In these cases, the capacitive loading that affects delay also determines the energy trends. However, in some cases, the energy reduces significantly when changing configurations and does not track performance behavior. For example, for the 512KB cache using 8 layers, the energy reduces when moving from 2*4 to 1*8 configuration. This stems from the difference in the number of sense amplifiers activated in these configurations due to the different number of bitlines in each subarray in the different configurations and the presence of the column decoders after the sense amplifiers. Specifically, the optimum (Ndwl,Ndbl,Nspd) for the 512KB case is (32,1,16) for the 2*4 case and (32,1,8) for the 1*8 configuration. Consequently, the number of sense amplifiers activated per access for the 1*8 configuration is only half as much as that of the 2*4 configuration, resulting in a smaller energy.

3.2 Thermal-Aware 3D Floorplanning: A Physical Design Tool for 3D Microprocessors

3D IC design is fundamentally related to the topological arrangement of logic blocks. Therefore, physical design tools play an important role in the adoption of 3D technologies. New placement and routing tools are necessary to optimize 3D instantiations, and new design tools are required to optimize interlayer

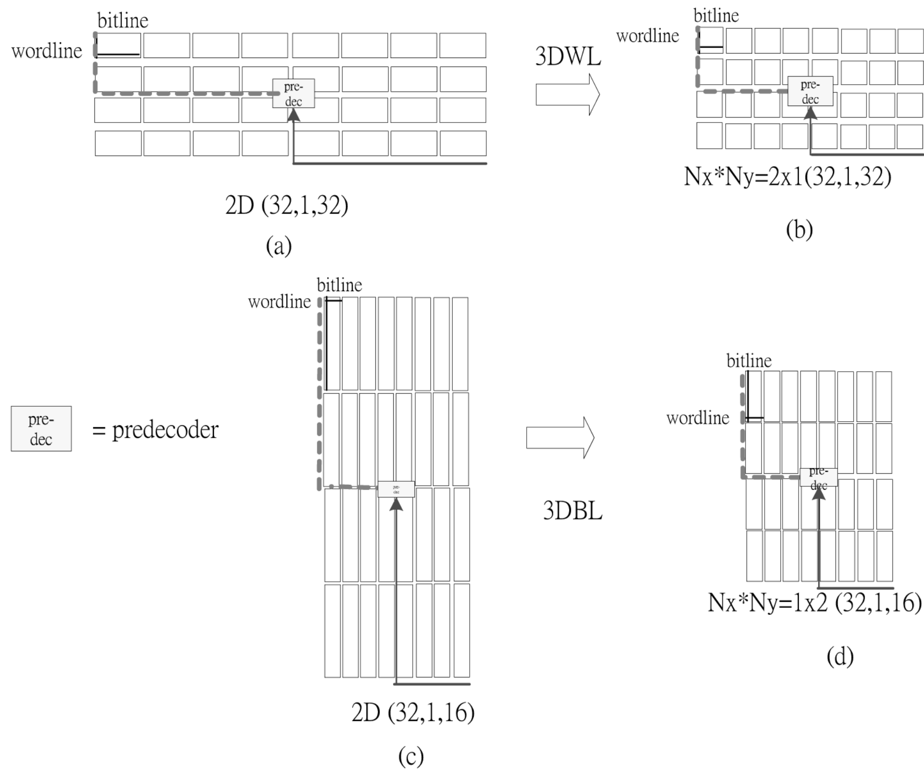


Fig. 9. Critical paths in 3DWL and 3DBL for a 1MB cache. Dashed lines represent the routing of address bits from pre-decoder to local decoder while the solid arrow lines are the routing paths from the address inputs to predecoders.

connections. A major concern in the adoption of 3D architecture is the increased power densities that can result from placing one computational block over another in the multilayered 3D stack. Since power densities are already a major bottleneck in 2D architectures, the move to 3D architectures could accentuate the thermal problem. Even though 3D chips could offer some respite due to reduced interconnect power consumption (as a result of the shortening of many long wires), it is imperative to develop thermally aware physical design tools. For example, partition design to place highly-loaded, active gates in a layer close to the heat-sink.

Tools for modeling thermal effects on chip-level placement have been developed. Recently, Cong et al. [2004] proposed a thermal-driven floorplanning algorithm for 3D ICs. Chu and Wong [1997] proposed using a matrix synthesis problem (MSP) to model the thermal placement problem. A standard cell placement tool to even thermal distribution has been introduced by Tsai and Kang [2000] with their proposed compact finite difference method-based (FDM) temperature modeling. In Goplen and Sapatnekar [2003], thermal effect was formulated as another force in a force-directed approach to direct the placement procedure for a thermally even standard cell placement. Another design metric, reliability, was taken care of in Shiu and Lim [2004] when

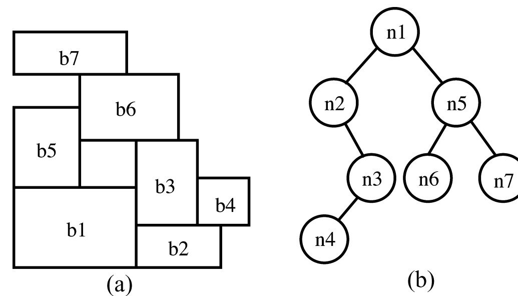


Fig. 10. (a) an example floorplan; (b) the corresponding B*-tree.

doing a multilayer System-on-Package floorplanning, though thermal issue was neglected.

In this section, we present a thermal-aware floorplanner for a 3D microprocessors [Hung et al. 2006]. Our floorplanner is unique in that it accounts for the effects of the interconnect power consumption in estimating the peak temperatures. We describe how to estimate temperatures for 3D microprocessors and show the effectiveness of the thermal-aware tool in reducing peak temperatures using one microprocessor design and four MCNC benchmarks.

3.2.1 *Temperature Estimation.* In order to consider the thermal impact, a compact thermal model is needed to provide the temperature profile. Numerical computing methods (such as FEM and FDM [Tsai and Kang 2000]) are most accurate but computationally intensive, while the simplified close-form formula is the fastest but inaccurate.

Skadron et al. [2003] proposed a thermal modeling tool called HotSpot which provides temperature estimation of a microprocessor at the functional module level by employing the principle of thermal-electrical duality. An RC network of thermal capacitances and resistances of functional modules are constructed and then temperatures at the center of functional modules are calculated by using circuit-solving techniques. The inputs to HotSpot are the floorplan and the power consumption of individual modules, and the specifications of heat spreader and heat sink are also provided to define the heat-removing ability.

While the HotSpot tool was originally intended to be a fast means of modeling temperatures of 2D architectures, we use a new tool called *HS3D* [Link and Narayanan 2006], which is an extension of the original HotSpot tool, by including a variable number of additional levels, each composed of both a silicon layer and an inter-silicon glue material. To validate the multilayer modeling, HS3D was compared to a commercial FEM tool, Flotherm, and showed an average temperature misestimation of $3^{\circ}C$ and a maximum deviation of $5^{\circ}C$.

3.2.2 *B*-tree Floorplan Model.* The floorplanner used in this work is based on the B*-tree representation which was proposed by Chang et al. [2000]. A B*-tree is an ordered binary tree which can represent a nonslicing admissible floorplan. Figure 10 shows a B*-tree representation and its corresponding floorplan. The root of a B*-tree is located at the bottom-left corner. A B*-tree of an admissible floorplan can be obtained by depth-first search procedure in a

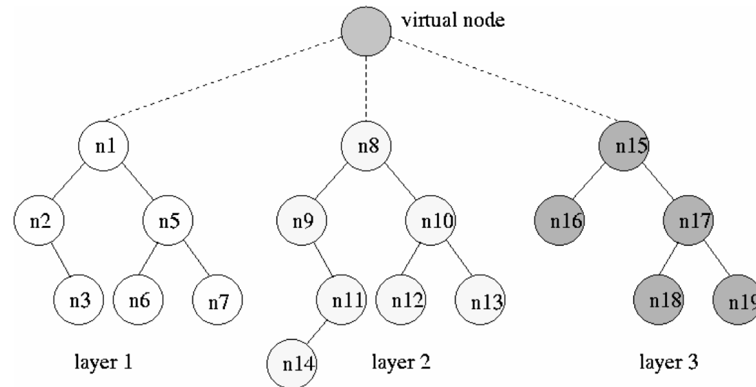


Fig. 11. Extending a B*-tree model to represent 3D floorplanning.

recursive fashion. Starting from the root, the left subtree is visited first, followed by the right subtree recursively. The left child of node n_i is the lowest unvisited module in R_i which denotes the set of modules adjacent to the right boundary of module b_i. On the other hand, the module, representing the right child of n_i, is adjacent to and above module b_i. We assume the coordinates of the root node are always (0,0) residing at bottom-left corner. The geometric relationship between two modules in a B*-tree is maintained as follows. The x-coordinate of node n_j is $x_i + w_i$, if node n_j is the left child of node n_i. That is, b_j is located and is adjacent to the right-hand side of b_i. For the right child n_k of n_i, its x-coordinate is the same as that of n_i, with module b_k sitting adjacent to and above b_i. While the original B*-tree structure was developed and used for the 2D floorplanning problem, we extend this model to represent a multi-layer 3D floorplanning (see Figure 11) and modify the perturbation function to handle 3D floorplans in this work.

There are six perturbation operations used in our algorithm and they are:

(1) node swap, which swaps two modules;
(2) rotation, which rotates a module;
(3) move, which moves a module;
(4) resize, which adjusts the aspect-ratio of a soft module;
(5) interlayer swap, which swaps two modules at different layers;
(6) interlayer move, which moves a module to a different layer.

The first three perturbations are the original moves defined in Chang et al. [2000]. Since these moves only have influence on the floorplan in a single layer, more interlayer moves, (5) and (6), are needed to explore the 3D floorplan solution space.

3.2.3 *Simulated Annealing Engine.* A simulated annealing engine is used to generate floorplanning solutions. The inputs to our floorplanning algorithm are the area of all functional modules and interconnects among modules. However, the actual dimension of each module is unknown a priori except for its

area before placement. That is, we have to treat them as soft modules. Thus, we provide the choice of adjusting aspect-ratio as one perturbation operation. During the simulated process, each module dynamically adjusts its aspect-ratio to fit closely with the adjacent modules, that is, with no dead space between two modules. A traditional weighted cost representing optimization costs (area and wire length) is generated after each perturbation.

Different from 2D floorplanning, our 3D floorplanner uses a two-stage approach. The first stage tries to partition the blocks to appropriate layers and minimize the packed area difference between layers and total wire length using all perturbation operations ((1)–(6) listed in previous section). However, because the first stage tries to balance the packed areas of the different layers, the floorplan of some layers may not be packed compactly. The second stage is intended to overcome this problem. Thus, in the second stage, we start with the partitioning solution generated by the first stage and focus on adjusting the floorplan of each layer simultaneously with the first four operations. At this point, there are no interlayer operations to disturb the module partition of each layer obtained from stage one.

One problem with 3D floorplanning is that the final packed area of each layer must match to avoid penalties of chip area. For example, if the final width of packed modules of layer $L1$ is larger than the final width of packed modules in layer $L2$, and the height of L1 is smaller than that of L2, a significant portion of chip area is wasted due to the need for the layer dimensions to match for manufacturing. To make sure the dimension of each layer will be compatible, we adopt the concept of dimension deviation *dev(F)* in [Shiu and Lim 2004]. The goal is to minimize *dev(F)* which gives the deviation of the upper-right corner of a floorplan from the average Ave_x, Ave_y values. The value Ave_x can be calculated by $\sum ux(f_i)/k$, where $ux(f_i)$ is the x-coordinate of upper-right corner of floorplan i, and k indicates the number of layers. The value Ave_y can be obtained in a similar manner. Thus, *dev(F)* is formulated as $\sum_i^{layers} |Ave_x - ux(f_i)| + |Ave_y - uy(f_i)|$. The modified cost function for 3D floorplanner can be written as

$$cost = \alpha * area + \beta * wl + \gamma * dev(F), \tag{1}$$

where *area* and wl are chip area and wire length, respectively.

3.2.4 *Temperature Approximation.* Although HS3D can be used to provide temperature feedbacks, when evaluating a large number of solutions during simulated procedure, it is not wise to involve the time-consuming temperature calculation every time. Other than using the actual temperature values, we have adopted the power density metric as a thermal-conscious mechanism in our floorplanner. The temperature is heavily dependent on power density based on a general temperature-power equation: $T = P * R = P * (t/k * A) = (P/A) * (t/k) = d * (t/k)$, where t is the thickness of the chip, k is the thermal conductivity of the material, R is the thermal resistance, and d is the power density. Thus, we can substitute the temperature and adopt the power density, according to the previous equation, to approximate the 3-tie temperature function, $C_T = (T - T_o)/T_o$, proposed in Cong et al. [2004] to reflect the

thermal effect on a chip. As such, the 3-tie power density function is defined as $P = (P_{\max} - P_{avg}))/P_{avg}$, where $P_{\max}$ is the maximum power density, while P_{avg} is the average power density, of all modules. The cost function for 2D architecture used in simulated annealing can be written as

$$cost = \alpha * area + \beta * wl + \gamma * P. \tag{2}$$

For 3D architectures, we also adopt the same temperature approximation for each layer as horizontal thermal consideration. However, since there are multiple layers in 3D architecture, the horizontal consideration alone is not enough to capture the coupling effect of heat. The vertical relation among modules also needs to be involved and is defined as $OP(TPm) = \sum (Pm + Pm_i) * overlap_area$, where *OP(TPm)* stands for the summation of the power density of module Pm and all overlapping modules m_i with module m and their relative power densities multiplying their corresponding overlapped area.

The rationale behind this is that, for a module with relatively high power density in one layer, we want to minimize its accumulated power density from overlapping modules located in different layers. We can define the set of modules to be inspected so the total overlap power density is $TOP = \sum OP(TPi)$ for all modules in this set. The cost function for 3D architecture is modified as follows:

$$cost = \alpha * area + \beta * wl + \phi * dev(F) + \gamma * P + \delta * TOP. \tag{3}$$

At the end of algorithm execution, the actual temperature profile is reported by our HS3D tool.

3.2.5 *Experimental Results.* We implemented the proposed floorplanning algorithm in C++. The thermal model is based on the HS3D. In order to effectively explore the architecture-level interconnect power consumption of a modern microprocessor, we need a detailed model which can act for the current generation high-performance microprocessor designs. We have used IVM (http://www.crhc.uiuc.edu/ACS/tools/ivm), a Verilog implementation of an Alpha-like architecture (denoted as Alpha in the rest of this section) at register-transfer-level, to evaluate the impacts of both interconnect and module power consumptions at the granularity of functional module level. A diagram of the processor is shown in Figure 12. Each functional block in Figure 12 represents a module used in our floorplanner. The registers between pipeline stages are also modeled but not shown in the figure.

The implementation of the microprocessor has been mapped to a commercial 160nm standard cell library by Design Compiler and placed and routed by First Encounter under a 1GHz performance requirement. There are a total of 24 functional modules and 168 netlists extracted from the processor design. The area and power consumptions from the actual layout served as inputs to our algorithm. Other than Alpha processor, we have also used MCNC benchmarks to verify our approach. A similar approach in Tsai and Kang [2000] is used to assign the average power density for each module in the range of $2.2*10^4$ (W/m^2) and $2.4*10^6$ (W/m^2). The total net power is assumed to be 30% of the total power of the modules due to lack of information for the MCNC benchmarks and the total wire length used to be scaled during floorplanning is the average

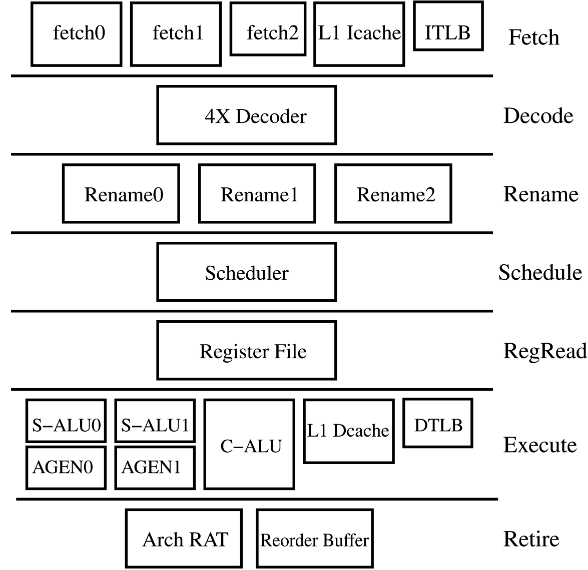


Fig. 12. Processor model diagram.

Table II. Floorplanning Results of 2D Architecture

Circuit	**2D**			**2D(thermal)**		
	wire (um)	area (mm^2)	peakT	wire (um)	area (mm^2)	peakT
Alpha	339672	29.43	114.50	381302	29.68	106.64
xerox	542926	19.69	123.75	543855	19.84	110.45
hp	133202	8.95	119.34	192512	8.98	116.91
*ami*33	44441	1.21	128.21	51735	1.22	116.97
*ami*49	846817	37.43	119.42	974286	37.66	108.86

Table III. Floorplanning Results of 3D Architecture

Circuit	**3D**			**3D(thermal)**		
	wire (um)	area (mm^2)	peakT	wire (um)	area (mm^2)	peakT
Alpha	210749	15.49	135.11	240820	15.94	125.47
xerox	297440	9.76	137.51	294203	9.87	127.31
hp	124819	4.45	137.90	110489	4.50	134.39
*ami*33	27911	0.613	165.61	27410	0.645	155.57
*ami*49	547491	18.55	137.71	56209	18.71	132.69

number from 100 test runs with the consideration of area factor alone. The widely used method of the half-perimeter bounding box model is adopted to estimate the wire length. Throughout the experiments, two-layer 3D architecture was assumed due to a limited number of functional modules and excessively high power density beyond two layers; however, our approach is capable of dealing with multiple-layer architecture.

Table II and III show the experimental results of our approach when considering traditional metrics (area and wire) and thermal effect. When taking

thermal effect into account, our thermal-aware floorplanner can reduce the peak temperature by 7% on average, while increasing wirelength by 18% and providing a comparable chip area compared to the floorplan generated using traditional metrics.

When we move to 3D architectures, the peak temperatures increased by 18% (on average) compared to the 2D floorplan due to the increased power density. However, the wire length and chip area reduced by 32% and 50%, respectively. The adverse effect of the increased power density in the 3D design can be mitigated by our thermal-aware 3D floorplanner which lowers the peak temperature by an average of 8 degrees with little area increase compared to the 3D floorplanner that does not account for thermal behavior. As expected, the chip temperature is higher when we step from 2D to 3D architecture without thermal consideration. Although the wire length is reduced when moving to 3D and thus reduces interconnect power consumption accordingly, the temperature for 3D architecture is still relatively high due to the accumulated power densities and smaller chip footprints. After applying our thermal-aware floorplanner, the peak temperature is lowered to a moderate level through the separation of high-power density modules in different layers.

Note that in this experiment, each function unit block is still in 2D implementation which is automatically synthesized by using a traditional 2D logic synthesis tool. Our thermal-aware floorplanner just simply stacks these 2D blocks together, only reducing interblock latency and power consumption by global interconnects among these 2D blocks. It does not explore the 3D design space for an individual function unit block. Section 3.1 has described how to split an on-chip cache in 3D. In the next section, we describe how to customize and reimplement other components of a processor in 3D. In Section 5, we will present an industrial case study on a 3D implementation of a Pentium 4™derivative microprocessor and show that the reimplementation of some components in 3D can further reduce power consumption so that the thermal impact is under control.

4. IMPLEMENTING 3D MICROPROCESSOR COMPONENTS

One obvious usage for 3D integration is to stack a large amount of cache on top of a conventional microprocessor. However, this does not make full use of the high-bandwidth, low-latency die-to-die interconnect. Another approach is to use 3D to implement each of the individual components in a 3D fashion. For example, Section 3.1 described how an on-chip cache can be folded in 3D to reduce the latency and power of the cache itself. In this section, we briefly describe how some other common microprocessor components can be reimplemented in 3D, and then we use these techniques to reimplement an Alpha 21364 processor in a 3D technology.

4.1 SRAM Arrays

We first consider static random access memory (SRAM) arrays which are typically wire-dominated circuits that show great promise for 3D implementations. While we have already discussed the 3D implemementation of on-chip caches, many other structures on the processor are basically variations of

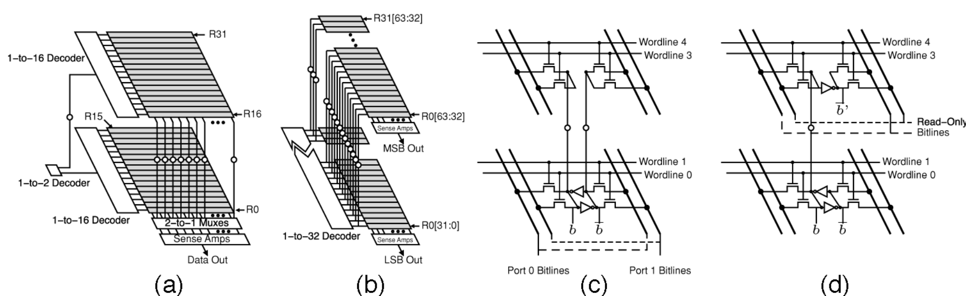


Fig. 13. 3D register file organizations achieved through (a) register-partitioning, (b) bit-partitioning, (c) port-splitting, and (d) a port-splitting alternative that only uses one die-to-die via per bitcell. A ○ represents a die-to-die via.

conventional SRAM arrays. These modules include branch predictor history tables, the branch target buffer, register alias tables, physical and/or architected register files, the reorder buffer, and the payload RAM portion of the instruction issue logic. Each instance differs in the capacity (both number of entries and the bits per entry), timing constraints, and bandwidth requirements for both reading and writing. Other table-like structures such as the decoder PLAs, the microcode ROM in x86 processors, and the lookup table for SRT dividers may also observe similar benefits from a 3D implementation.

The two primary 3D organizations described by the previous 3D cache studies are partitioning of the array in the horizontal or vertical directions [Puttaswamy and Loh 2005; Tsai et al. 2005]. The first approach is to stack rows on rows, which has the benefit of approximately halving the bitline length (for a two-layer stack) which in turn improves latency by reducing the associated bitline resistance and capacitance (RC). The second approach is to partition by stacking columns on columns which reduces the length of the wordlines and their associated RC.

For multiported SRAMs, such as register files and register alias tables (RATs) for superscalar processors, there are similar approaches to the partitioning problem. We will refer to all of these structures collectively as register files for the remainder of this section. A two-die register-partitioned (RP) 3D register file takes half of the register entries and places them on the second die. Figure 13(a) illustrates a 32-entry register file where the bottom die contains registers R0 through R15, and the top die contains R16 through R31. A result of this topology is that the vertical distance (along the bitlines) has been halved which can greatly reduce the latency and power associated with toggling the bitlines. The row decoder's height has also been halved which reduces the length of the critical path associated with accessing the farthest entry in the register file. The overall footprint of the register file has also been halved which may enable more compact processor floorplans. To implement a 4-die RP 3D register file, the register entries would simply be partitioned such that one quarter of the entries reside on each die.

The bit-partitioned (BP) 3D register file stacks different bits of the same register across the different dies. Figure 13(b) shows a 2-die 64-bit 32-entry

bit-partitioned register file where the bottom die stores the least significant bits of the register values, and the top die stores the most significant bits. The bit-partitioned register file reduces the wire length and gate loading on the wordline which provides both latency and energy benefits. While Figure 13(b) shows the bits of each register partitioned by significance, one could instead store the bits in odd positions on one die and the bits in even positions on the other die. Choosing one over the other does not impact the area, latency, or power of the 3D BP register file. However, the choice should be made to match the datapaths throughout the rest of the processor. For example, if one implements a 3D integer ALU partitioned by significance (X[0:31] + Y[0:31] on one die, X[32:63]+Y[32:63] on the second die) [Mayega et al. 2003], then the register file bit-partitioning should also be arranged by significance to avoid unnecessary die-to-die routing between the register file outputs and the ALU inputs.

For on-chip caches, the individual SRAM cells are very small to maximize the capacity of the cache, while the area-per-bit for a register file cell is dominated by the wordlines and bitlines for implementing multiple read and write ports. In Section 3.1, we have shown that the relative size of a 6T SRAM cell and a d2d via make it difficult to take an individual 6T cell and split it across multiple dies. However, register file SRAM cells have a substantially larger footprint (due to the high port count) which may provide the opportunity to allocate one or two d2d vias for each cell. Figure 13(c) shows a 2-die port-split (PS) SRAM cell where each die contains the bitlines, wordlines, and access transistors for half of the ports (either read or write). Two d2d vias are required per bit-cell to route the outputs of the chained inverters to the second die.

The PS register file provides substantial benefits in terms of area footprint reduction. Stacking the wordlines on top of each other halves the height, while stacking the bitlines halves the width. A 50% reduction in both dimensions leads to an overall footprint reduction of 75% for the SRAM array. The total register file reduction is slightly less because structures like the row decoder and sense amps may not observe as large a compaction benefit. This substantial area reduction also translates into latency and energy savings because both bit-line and wordline lengths have been halved. Figure 13(d) shows an alternative implementation of a 2-die port-split (PS) 3D register file cell where only a single d2d via is used to route the data bit b to the second die. On the upper die, an extra inverter is required to recompute the complement bit $\overline{b}$. This shows how in some situations, logic duplication may be used to trade off against excessive interdie communication. A limitation of the single-via configuration is that the ports on the top die can only support read operations because there is no path to access the true $\overline{b}$ storage node. This limitation is likely not critical as the number of write ports is typically much less than the number of read ports.

Register files implemented across four (or more) dies can use a combination of the partitioning strategies described. This may be particularly useful in an alternating F2F/B2B die-stacking organization where the available d2d via density changes between pairs of die. In a 4-die stack with alternating F2F interfaces, one could first use register-partitioning to assign half of the registers to dies 0/1 and the other half to dies 2/3, which limits the usage of the coarser

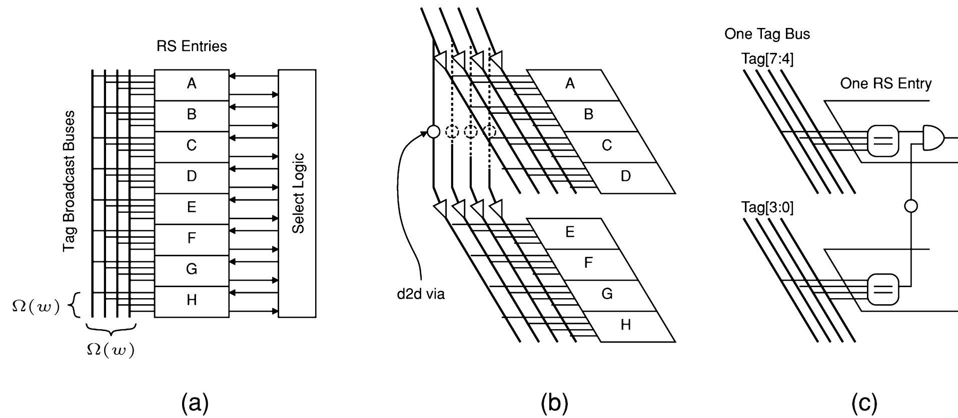


Fig. 14. (a) 2D scheduler layout, (b) entry-partitioned 3D scheduler layout, (c) single RS entry and single tag broadcast bus for a tag-partitioned 3D scheduler, and (d) cross-sectional view of cross-die tag fanout for a 4-die 3D scheduler.

B2B vias to the periphery of the main SRAM array. Then among each pair of F2F die, port-partitioning could be employed to exploit the denser F2F interface within the SRAM array.

4.2 CAM Arrays

Several components in a modern out-of-order processor require content addressable memories (CAMs) to perform fully-associative searches. CAMs are central to the implementation of the dynamic instruction scheduling logic, load and store queues, fully-associative TLBs, certain types of register renaming implementations [Lipasti et al. 2004], and several other smaller buffers and queues such as stream buffers and victim caches [Jouppi 1990]. In the following discussion, we will use the dynamic instruction scheduler (also known as the issue queues or reservation stations) as a running example of a typical CAM-based microarchitectural module. The techniques presented for implementing 3D versions of the scheduler easily translate to the other CAM structures.

The instruction scheduling logic is responsible for tracking register data dependencies between instructions, determining when instructions are ready to issue, and arbitrating among the ready instructions for limited structural resources. The scheduler contains separate entries for each in-flight instruction. The scheduler latency and power is dominated by long wire delays [Palacharla et al. 1997], primarily due to the long tag broadcast buses shown in Figure 14(a). The lengths of these buses are determined by the height of each entry and the total number of entries. The scheduler is a wire-bound structure, and so the wires entering and leaving an RS entry dictate the entry pitch. For a w-issue scheduler, there are w tag broadcast buses, and each entry must compare its register tags with those on the buses. For this implementation, the requirement forces the height of the RS entry to be at least $\Omega(w)$. The width of the broadcast buses also require $\Omega(w)$ distance, and so the broadcasts buses alone consume $\Omega(w^2)$ area. A fully-associative TLB has a very similar

structure with the tags replaced by page numbers, and w equal to the number of ports.

For a two-layer 3D stack, we consider two possible organizations of the scheduler. The first is called the *entry-partitioned* (EP) organization and simply places half of the RS entries on one layer, and the other half on the other layer as illustrated in Figure 14(b). A broadcast bus on one die will require d2d vias to connect to the RS entries on the other die. Alternatively, the buses can be replicated across the two dies to reduce the d2d via usage. The primary benefit of this organization is that the length of the broadcast bus has been halved and each leg of the bus is now only loaded by half as many comparators.

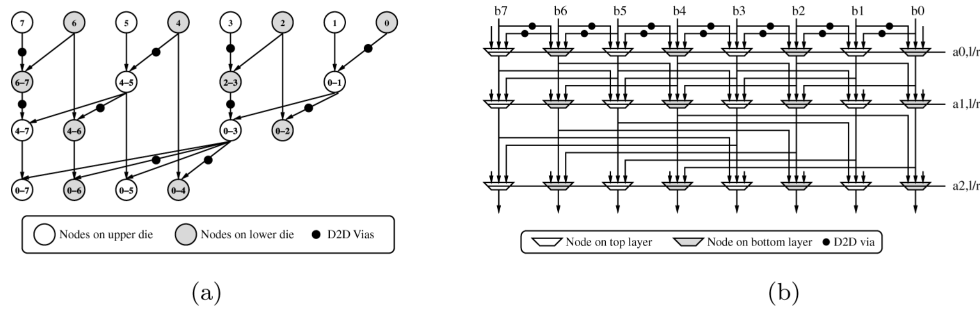
The second proposed 3D CAM organization is *address-partitioning* (AP). The address is simply the identifier used to determine a hit: a tag in the case of the scheduler, a register number for the RAT, a page number for the TLB, etc. Broadcast buses and individual RS entries are partitioned across the two dies. Figure 14(c) shows the AP organization for a single RS entry, illustrating only a single broadcast bus for the sake of clarity. This approach effectively bit-slices the tag buses with one half of each tag comparison occurring on one die. A d2d via is required to combine the two halves of the comparison together to compute the final tag match signal. Figure 14(c) assumes an 8-bit tag, and partitions the broadcast bus wires by bit-significance. Referring back to Figure 14(a), if only half of the bits are on one die, then the height of the RS entry can actually be halved. Furthermore, the width of the broadcast buses also decreases by one half.

Similar to the 3D SRAM organizations, a four-layer CAM enables the combination of different 3D stacking techniques. For example, one could use entry-partitioning to place half of the entries on die 1 and die 2 and the other half on 3 and 4. Then within each pair of die, the individual entries and buses could be address-partitioned.

4.3 Arithmetic Units

While SRAMs and CAMs account for a large portion of the circuits in a modern processor, there are still many other types of components. In some units, such as functional units, the intragroup dependency checking logic for register renaming, hardwired-decode logic, and other forms of random logic are often dominated by transistor/gate delay rather than wire delay. In this section, we consider 3D implementations of an integer adder and a barrel shifter. The critical paths through addition circuits are typically dominated by gate or logic delay rather than wire delay. For wide adders (64-bit), only a few of the upper levels of the propagate-generate logic make use of long wires. In contrast, a 64-bit barrel shifter contains many very long wires because input bits may need to be shifted 64 positions to the left or right. The comparison of circuits that are either logic-bound or wire-bound helps to illustrate the relative benefits of 3D for different situations and provides insight into what other components in a processor would benefit the most from a 3D implementation.

We evaluated several adder implementations including the Brett-Kung, Sklansky, and Kogge-Stone variants [Brent and Kung 1982; Sklansky 1960;



(a) (b)

Fig. 15. (a) A 3D implementation of a Sklansky adder, and (b) a 3D implementation of a barrel shifter.

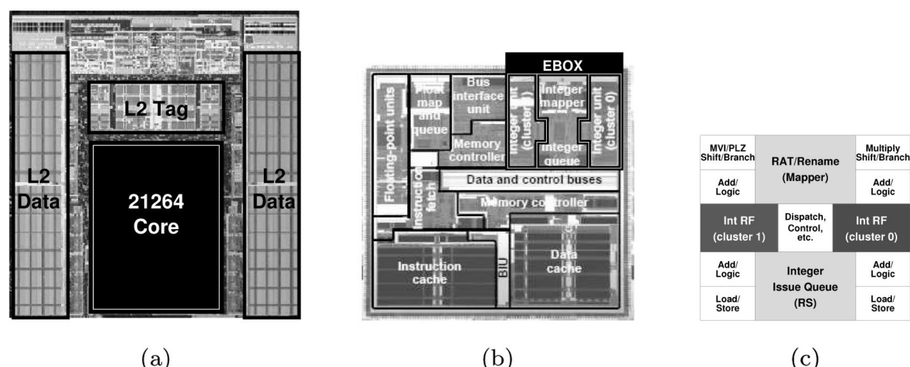
Kogge and Stone 1973]. In this article, we only discuss the Sklansky adder as the conclusions are similar for the other two circuits. Figure 15(a) shows a simplified view of the carry-propagate logic for an 8-bit Sklansky adder. In a two-die 3D implementation, we stack odd operand bits on one die and even operand bits on the other die. A similar bit-interleaving organization can be generalized to a larger number of layers. Every other node of the propagate-generate logic has been stacked on top of an adjacent node as indicated by the shading in Figure 15(a). This in turn reduces the total width of the circuit by one half which reduces the length of each internode wire by one half as well. While all of the critical wires have been halved in length, some now contain D2D vias. The additional RC overhead of the D2D via is relatively small and is more than offset by the corresponding wire length reduction.

Similar to the adders, we 3D-partitioned the barrel shifter such that adjacent nodes are vertically stacked on different dies, thus shrinking the circuit to half its original size. Figure 15(b) shows the 3D partitioning of odd and even bit positions across two dies. This organization halves the lengths of wires in every multiplexing level which, in turn, provides greater latency and power savings with every additional layer of the circuit. The logic delays remain approximately the same as in the planar shifter design, but the wire delay component decreases substantially. Note that die-to-die interconnections are only required in the first level of shifting. All subsequent levels of logic communicate only within the same die.

For circuits such as barrel shifters, which are dominated by wire-delay, 3D can potentially provide simultaneous benefits for performance and power. On the other hand, logic-bound circuits like adders do not gain as much from a 3D implementation, and 3D may be better utilized to reduce the wire-lengths between such circuits rather than within the circuits.

4.4 Other Components

Not all circuits provide substantial latency or power improvements when implemented in 3D. However, the 3D implementation of a logic-dominated block may still provide a global benefit even if its own latency and/or power remains unaffected. By reducing the size of a block, the length of global routes that pass over that block may be reduced, providing latency and power benefits for



(a) (b) (c)

Fig. 16. The floorplan of (a) the Alpha 21364, (b) the 21264 core, and (c) our default floorplan of the 21264 integer execution core (EBox).

other circuit paths. In some situations, this type of floorplan-related wire reduction may be able to remove entire pipeline stages such as the pipestages in the Intel® NetBurst microarchitecture that are primarily for driving a signal from one part of the chip to another [Hinton et al. 2001]. Another important floorplan-dependent datapath is the execution result bypass network which we will discuss in greater detail in the next section.

4.5 3D Processor Floorplans

Having described how many of the microarchitectural components of a processor could be implemented in 3D, we now discuss how this can affect the overall organization of the Alpha 21364 processor shown in Figure 16(a). The 21364 has a large on-chip L2 cache that flanks the 21264-based core. Figure 16(b) shows a closeup of the floorplan of the main processor core. We do not have access to the detailed floorplan beyond this previously published micrograph of the 21264, and so we made some assumptions about the layout of the integer datapath (EBox). Following the module boundaries of the EBox as depicted in the micrograph, we assumed the placement of submodules as shown in Figure 16(c). While the absolute results presented in this study depend on these floorplan assumptions, the general trends will hold for other floorplans and microarchitectures.

The first 3D processor configuration that we evaluate is microarchitecturally identical to the 21364, that is, it has the same number of physical registers, the same functional units, the same issue queue sizes, etc. We use 3D to reimplement all of the modules such that their critical paths are reduced. As a result, each module has half the footprint of its original planar implementation. Figure 17(a) shows the original planar floorplan, and Figure 17(b) shows the 3D version. After reducing the footprint of each module, we manually refloorplanned the processor. Since the original floorplan was not designed with 3D in mind, we end up with a few places where there is some leftover white space. Most of the floorplan translated to 3D in a fairly straightforward manner. However, the integer execution box (EBox) was specially optimized to

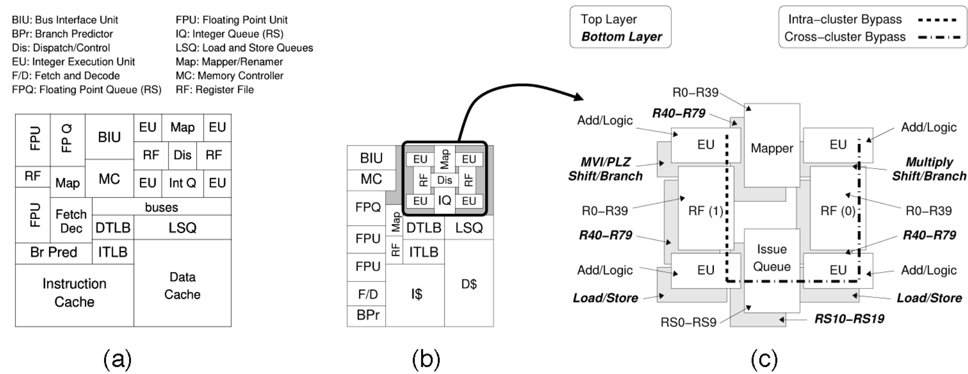


Fig. 17. (a) Our baseline planar floorplan for the 21364 core, L2 cache not shown, (b) a compacted 3D two-layer floorplan, and (c) 3D detail of the integer execution core.

provide latency improvements both within individual units as well as between the units (bypass).

Figure 17(c) shows a close up of the 3D EBox and details the 3D partitioning of all of the units. The dispatch and control logic that is located at the center of the EBox has been omitted for clarity. The issue queue (scheduler), mapper, and register files have each been self-stacked to reduce their intrablock latency and power consumption. For example, the tag broadcast buses are assumed to run horizontally across the issue queue and so folding it lengthwise reduces the length of the buses. The footprint reduction of each of these modules allows the EBox as a whole to be compressed. This in turn reduces the length of the bypass paths both within and between execution clusters. To further reduce the bypass path lengths, we stacked the execution units on top of each other as shown in the four corners of Figure 17(c).

The 3D processor illustrated in Figure 17 provides enhanced performance by enabling a faster clock frequency while maintaining approximately the same IPC rates as the 2D version. An alternative to the use of 3D is to keep the original planar floorplan as shown in Figure 17(a) but make use of the additional integration capacity to increase the processor resources (more issue queue entries, larger branch predictor tables, etc.).

We also consider a four-layer version of the 21364. In this situation, the floorplanning is similar to the two-layer version shown in Figure 17(b), except that the modules are now split across four layers. As a result, the footprint of each module is further decreased, and the overall processor floorplan can be further compacted as depicted in Figure 18(a). Note that this figure is drawn to scale with respect to the two-layer floorplans of Figure 17. Note that in the two-layer version, within each execution cluster, we stacked the ALUs on top of each other. In the four-layer version, each execution unit has been split between two layers so that a total of two units are still stacked on top of each other across all four layers. Figure 18(b) shows the organization of the arithmetic units in more detail. Note that the register file is actually partitioned across all four layers. Another alternative is shown in Figure 18(c). This 3D floorplan is identical to the previous four-layer plan except for the layout of the EBox. A 3D-view of

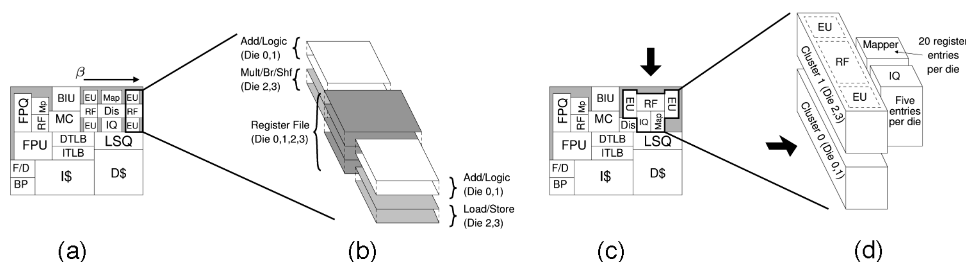


Fig. 18. (a) Our four-layer floorplan for the 21364 core drawn to scale with respect to Figure 17, (b) 3D detail of one integer execution cluster, (c) an alternative four-layer floorplan with cluster stacking, and (d) 3D detail of the stacked clusters.

the stacked clusters is shown in Figure 18(d) with the orientation indicated by the bold arrows. In this floorplan, we implement each cluster in only one pair of dies, and then stack the clusters one on top of the other. The advantage of this floorplan is that the horizontal cross-cluster bypass, indicated by the arrow ($\beta\rightarrow$) in Figure 18(a), has now been replaced by a relatively short vertical route. Note that the z-axes of Figures 18(b) and (d) are not drawn to scale. Similar to the two-die case, the resources of a four-die stack can be utilized to implement even larger structures to expose more ILP.

4.6 Performance Results

There are many alternatives for how 3D technology can affect microprocessor performance. In this section, we evaluate two approaches of extracting performance through higher clock frequencies or through higher IPC. Another approach that we do not consider in this section is to use the latency benefits to change the overall number of pipeline stages. The Alpha 21264 core has a relatively short pipeline which would not benefit as much from such an approach; furthermore, this type of repipelining approach requires implementation details about the processor pipeline that are not publicly available. We evaluate this approach in Section 5 in relation to a 3D implementation of a deeply-piped Intel® Pentium 4™-class processor.

Compared to the 3D 21364, it is likely that designing a new microarchitecture specifically targeting 3D would result in a more balanced design that provides greater overall value. However, this only implies that our results are conservative and should serve as an informal lower bound on what level of performance one could extract from a 3D technology. The goal is to demonstrate that even conventional microarchitectures can benefit from 3D, and these results should motivate the community to further explore the possibilities of new 3D microarchitectures.

4.6.1 *Experimental Methodology and Assumptions.* To quantify the circuit-level benefits of 3D, we used HSpice to simulate many critical processor circuits to determine their latencies. All circuits employ 70nm transistors using the BSIM models from Berkeley. For the d2d vias, currently available technologies already support $\sim 2\mu$m F2F vias and $\sim 4\mu$m B2B vias [Gupta et al. 2004]. Our 3D circuit designs use d2d via sizes of 0.75μm for F2F and 1.5μm

Table IV.
(Spice timing results for various microarchitectural modules for planar, 3D two-layer and 3D four-layer implementations, and the largest size implementable for each module without exceeding the latency of the corresponding planar implementation.)

Module/ Circuit	**2D**	**3D 2-layer**		
	Latency (ps)	Latency (ps)	% Speedup	Largest Size
L1 Cache (64KB)	1536	1159	24.5%	128KB
L2 Cache (1MB)	3551	2834	20.2%	1MB
BPred - (Local/Global/Meta:2/1/1KB)	760	667	12.3%	(2/2/2)
Load/Store Queue (32 each)	252	185	26.5%	44 entry
Int RAT (80 regs)	347	241	30.5%	120 regs
Int Issue Queue (20 entry)	467	416	11.0%	40 entry
Intra-Cluster Int Bypass	224	201	10.3%	—
Cross-Cluster Int Bypass	447	310	30.7%	—

Module/ Circuit	**3D 4-layer**		
	Latency (ps)	% Speedup	Largest Size
L1 Cache (64KB)	979	36.3%	256KB
L2 Cache (1MB)	2129	40.1%	2MB
BPred—(Local/Global/Meta:2/1/1KB)	615	19.1%	(2/4/4)
Load/Store Queue (32 each)	154	38.8%	80 entry
Int RAT (80 regs)	189	45.6%	160 regs
Int Issue Queue (20 entry)	396	15.1%	80 entry
Intra-Cluster Int Bypass	190	15.3%	—
Cross-Cluster Int Bypass	246	45.0%	—

for B2B. While these sizes are somewhat aggressive for current technologies, we believe that such fine via pitches will be available within a couple of process generations. We assume that the F2F via must travel 5μm to cross between the two die faces and 20μm to cross the B2B interface. These distances are quite conservative since currently available 3D technologies already thin the die down to only 12μm [Tezzaron Semiconductors 2005].

For all processor configurations, we used SimpleScalar to quantify the impact on IPC. We extensively modified the MASE cycle-level simulator [Larson et al. 2001] from SimpleScalar 4.0 [Austin et al. 2002] for the Alpha instruction set architecture. For our IPC studies, we assume a clock frequency of 2GHz for the baseline planar 21364. We use a collection of 100 application traces from SpecCPU, MediaBench [Lee et al. 1997], the Michigan embedded benchmarks (MiBench) [Guthaus et al. 2001], the Wisconsin pointer-intensive benchmarks [Austin et al. 1994], assorted graphics programs from the SimpleScalar Web site, and the BioBench bioinformatics benchmark suite [Albayraktaroglu et al. 2005].

4.6.2 *3D Circuit Evaluation.* We first present the circuit-related performance results. The numbers reported in this section are all from our Spice circuit simulations. Corresponding to our two high-level strategies for obtaining performance, we analyzed the circuits to see (1) how fast 3D can make a particular circuit, and (2) how large 3D can make a particular circuit assuming no change in clock frequency relative to the planar base case. Table IV shows the latency of our baseline 2D implementations as well as the latencies for two-layer and four-layer implementations of several important microarchitectural

structures. We also list the largest-sized 3D circuit that can be implemented assuming no increase in latency over the planar baseline. Most of our circuit implementations make use of static CMOS logic, therefore, some of the absolute values of circuit latencies may be slower than expected. This results in conservative speedup estimates since wire delay would dominate a larger fraction of the clock period in dynamic logic implementations. More aggressive custom logic design would likely result in an even greater relative 3D benefit.

The different modules exhibit different amounts of latency improvement due to differences in the impact of wire delay. Some circuits are more wire-dominated than others due to differences in total size or capacity, port requirements, etc. In determining the clock frequency of a processor, not all of the components are limiting factors. For example, since there are no single-cycle floating point instructions, the FP Issue Queue (scheduler) can be pipelined and will not directly help or hinder the overall cycle time. Palacharla [1998] identified the integer scheduling logic and the bypass network as critical cycle-time limiters. These are highlighted in bold in Table IV. We focused more attention on optimizing our circuit implementations for these two modules (for both the planar and 3D cases) and use these as an approximation of the overall processor clock frequency benefit[1]. We take the smaller of the two improvements as the expected cycle time reduction. The two-layer 3D processor would observe a 10.3% frequency boost, and the four-layer processor would have a 15.1% improvement.

4.6.3 *IPC Impact and Overall Performance.* The iron law for processor performance states that a program's overall runtime is determined by the total number of instructions, the clock frequency of the processor, and the amount of IPC that the processor can extract [Hennessy and Patterson 2003]. We are not changing the number of instructions; the previous section detailed the impact on clock frequency; now we address the IPC impact. For the 3D processor floorplans that implement larger modules, we expect some IPC benefit from having better branch prediction rates, fewer cache misses, a larger instruction window for exposing more ILP, etc. For the configurations that are targeted toward high clock frequency, we still evaluate the IPC impact because the higher clock frequency changes the number of cycles to access main memory (we assume that memory latency in absolute time is unchanged).

In our IPC simulations, we evaluated the planar baseline 21364, two 2-layer 3D versions, and three 4-layer 3D versions. For the 2-layer case, we call the first one $21364f_2$ (f denotes fast) which is the exact same microarchitecture as the base case except that the L2 cache is one cycle faster as previously discussed, and the memory latency (in cycles) is longer due to the faster processor frequency and rounding because the new frequency not an exact multiple of the frontside bus speed. The second 2-layer configuration is the $21364{+}{+}_2$, which maintains the same clock frequency as the base case but uses 3D to

[1]We do realize that the absolute latencies differ as is a result of a variety of assumptions that were made in the designs, layout, and floorplanning of the circuits. We believe that the relative benefit is still representative of the cycle time improvements achievable in practice.

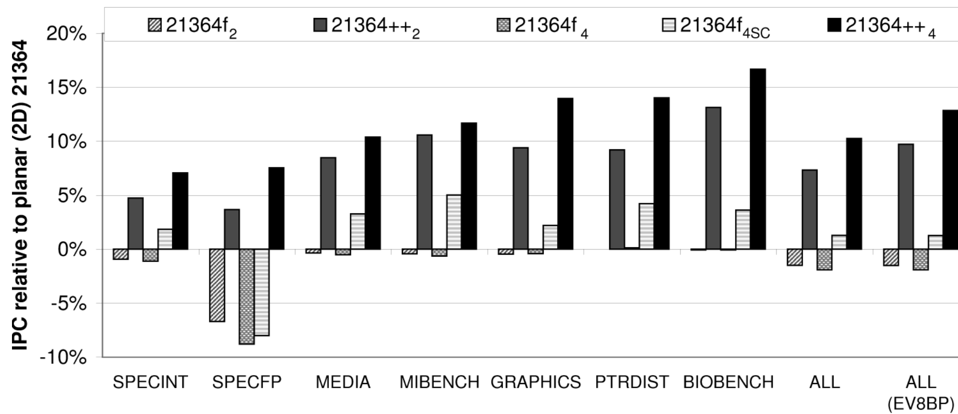


Fig. 19. The IPC impact of our different 3D microarchitectures relative to the baseline planar 21364.

implement larger microarchitectural structures in an attempt to expose more instruction-level parallelism. Similarly, we evaluate 4-layer versions that we call $21364f_4$ and $21364{+}{+}_4$. For the 4-layer processor, we also consider the stacked-cluster 3D floorplan depicted in Figure 18(c) where we assume that the 1-cycle penalty to bypass results between clusters has been eliminated which is denoted $21364f_{4SC}$.

We expect that the IPC rate will decrease for the faster clock frequency configurations due to the increase in the number of cycles to access memory, and we expect an overall IPC increase for the ++ configurations. Figure 19 shows the relative IPC rates with respect to the baseline 2D 21364 processor. The overall IPC impact varies depending on the application suite, however, the overall trends are similar. Except for SpecFP, the higher clock frequency configurations do show a slight IPC degradation as expected. SpecFP has much more traffic to main memory than the other applications, and therefore these programs are more adversely affected by the increase in the number of cycles to main memory. Note that the IPC reduction for the f configurations does not imply an overall performance decrease because these configurations are clocked at a higher frequency.

For the ++ configurations, the larger structures result in higher IPC rates. The $21364{+}{+}_2$ provides a 7.33% improvement and $21364{+}{+}_4$ yields 10.3%. The stacked-cluster floorplanning approach is a better alternative with respect to IPC than the nonstacked version. $21364f_4$ improves from −1.9% to 1.28%, and $21364{+}{+}_4$ goes from 10.26% to 11.65%.

The 21264 hybrid branch predictor performed reasonably well for the original EV6 microarchitecture. However, the predictor does not scale well for our larger machine configurations and results in a poor utilization of the additional resources. We ran one extra set of simulations using the EV8/21464 branch predictor scaled down to the hardware budgets of our 21364 configurations [Seznec et al. 2002]. These results are in the right-most data sets in Figure 19, labeled “ALL (EV8BP)”. For this last group only, the results are normalized to a planar 21264 implementation that also makes use of the EV8 branch predictor. In this

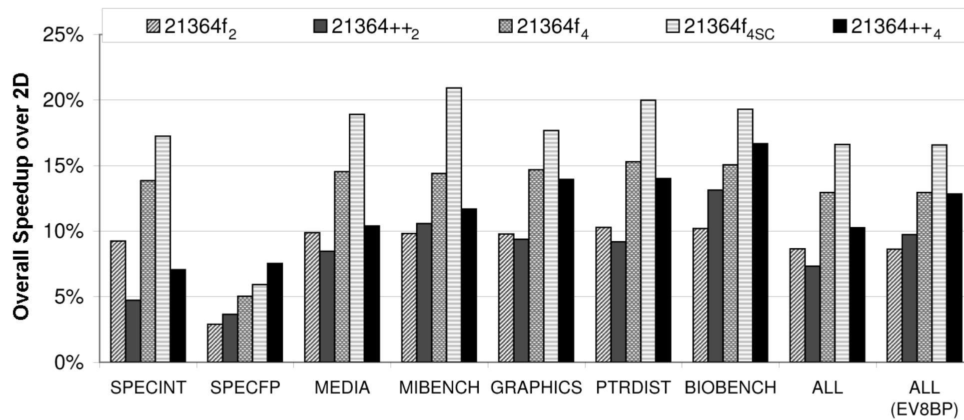


Fig. 20. The overall performance impact when both IPC and clock frequency are accounted for.

manner, all data in the plot represent the IPC impact due to 3D only. Better branch prediction rates enables the ++ configurations to make more effective use of the larger resources, leading to IPC increases of 9.73% and 12.84% for the $21364{+}{+}_2$ and $21364{+}{+}_4$, respectively. To fully exploit the performance opportunities of 3D, additional microarchitecture changes and retuning may be necessary.

The overall performance benefit can be determined by taking both IPC and clock frequencies into account. Figure 20 shows the overall performance benefit relative to the 2D 21364 baseline. For the 2-layer case, the 10.3% clock frequency improvement yields a final performance benefit of 8.65%, while $21364{+}{+}_2$ provides a respectable 7.33% benefit. In the 4-layer case, the difference is more pronounced (16.61% vs. 11.65%). However, this gap is partially due to the fact that the ++ configurations' branch predictor performances have not scaled which, in turn, leads to underutilization of the larger resources. When using the more accurate EV8 branch predictor, the 2-layer ++ configuration (9.73%) is actually able to extract enough ILP to outperform the higher-frequency approach (8.62%). For four-layers without cluster stacking, both approaches deliver nearly the same performance benefit (12.84% for $21364f_4$, 12.95% for $21364{+}{+}_4$). When cluster stacking is used, the improved frequency configuration still comes out ahead (16.59% for $21364f_{4SC}$).

At first sight, the performance results may seem modest for such a radical technology. However, one should keep in perspective that the real conclusion is that, by simply porting an existing microarchitecture to a 3D technology, a reasonable performance benefit can be had. We believe that to fully utilize the performance potential of 3D integration, a new microarchitecture should be designed from square one with the goal to exploiting the strengths of 3D (namely, vertical routing and transistor density).

5. INTEL CASE STUDY

This section focuses on exploiting the transistor density benefits of 3D die stacking. As a demonstration vehicle, we converted a deeply pipelined high frequency

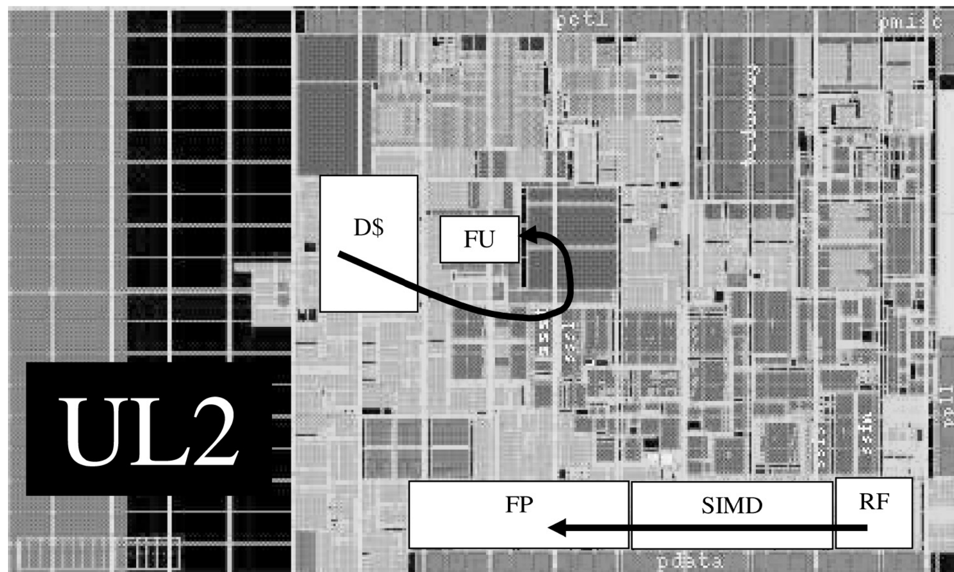


Fig. 21. intel-2dfloorplanPlanar floorplan (2D) of a deeply-pipelined x86 microprocessor.

iA32 microprocessor from Intel®Corporation from a traditional 2D or planar floorplan to a 2-die 3D floorplan. Figure 21 illustrates the planar floorplan of our test vehicle. The microarchitecture of this engine is a derivative of the Intel® Pentium 4™ microprocessor. The branch miss-prediction penalty is around 20 clock cycles.

Using 2-die 3D stacking, we developed a new floorplan that requires only 50% of the original footprint, while reducing inter-FUB (functional unit block) or FUB-to-FUB interconnect by stacking different blocks, and reducing intra-block or within-FUB interconnect by splitting individual FUBs across the two die. Figure 22 illustrates the result of our stacking. 3D stacking is intrinsically more complicated than planar floorplanning. There are physical restrictions including thermals, limited die-to-die interconnects, and active silicon region interrupts from backside through-silicon vias (TSVs). In particular, thermals play an important role in this stacking experiment. A risk of 3D stacking is the accidental doubling of power density and the thermal consequences. In a design such as this that already has a handful of peak hot areas and many other very hot areas close to the peak temperature, the task of keeping a 3D implementation within a given thermal envelope is challenging but not impossible. In this experiment, we used a simple iterative process of placing blocks, observing the new power densities and die-to-die interconnect demand, and repairing outliers.

The goal of this experiment is to reduce metal use across the microarchitecture and subsequently realize a performance and/or power advantage. To improve performance with 3D stacking, latency must be reduced. This can be accomplished by increasing frequency and/or eliminating pipe stages in the microarchitecture. Since this microarchitecture has hundreds of thousands of

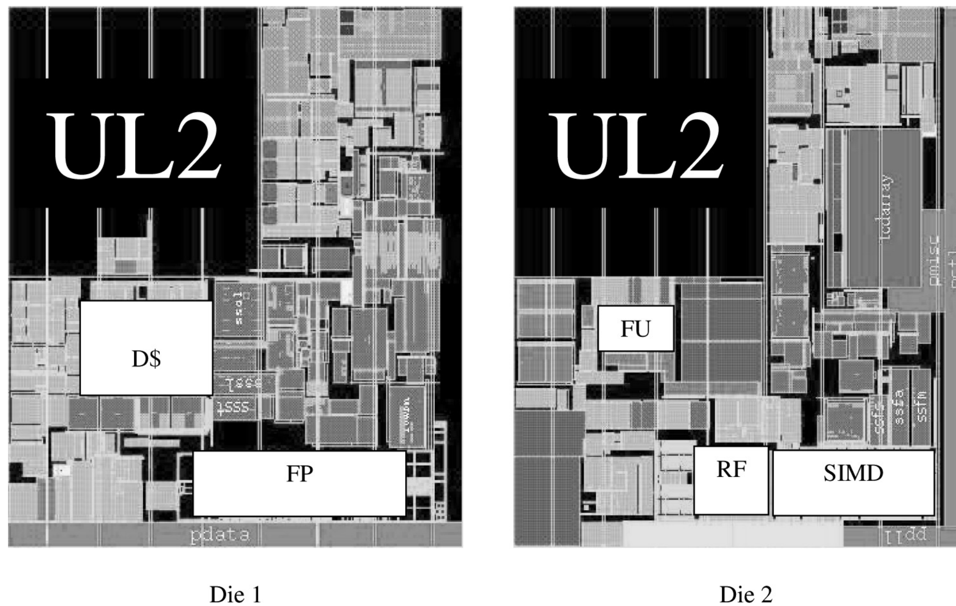


Fig. 22. 3D floorplan of a deeply-pipelined x86 microprocessor.

nets that would all require optimization to demonstrate frequency, this work assumes a constant frequency and focuses on eliminating pipe stages in the microarchitecture. (Note: The term pipe stage is used to refer to all pipe stages in the microarchitecture including the cache hierarchy, store retirement, post-completion resource recovery, etc. The number of pipe stages in this microarchitecture is much greater than the miss-prediction clocks.) We reduce power by reducing total metal capacitance, driver strengths, the number of pipe stage latches, repeating latches, and repeaters.

During the process of generating this 3D floorplan, we observed that the criteria for determining which blocks are stacked and which ones are split varies with block size, thermally sensitive regions, performance sensitivities, and power consumption. The following sections discuss the stacking and splitting methods used to optimize the new 3D floorplan, the effects on the pipeline, and realized power and latency reductions.

5.1 Stacking

Stacking simply moves blocks closer together reducing interblock latency and power. Much of our effort concentrated on known performance sensitive pipelines. For example, the load-to-use delay is critical to the overall performance of most benchmarks. The path between the first-level data cache (D$) and the data input to the functional units (FU) is illustrated in Figure 21. The worst case path occurs when load data must travel from the far edge of the data cache, across the data cache to the farthest functional unit, yielding at least 1 clock cycle of wire delay all due to planar floorplan limitations. This wire delay is in addition to the access time of the data cache itself. Figure 22

Table V. List of Pipe-Stages Eliminated and Related Performance Gain

Functionality	Stage Eliminated	Performance Gain
Frontend pipeline	12.5%	~0.2%
Trace cache read	20%	~0.33%
Rename allocation	25%	~0.66%
FP inst latency	Variable	~4.0%
Int register file read	25%	~0.5%
L1 data cache read	25%	~1.5%
Instruction loop	17%	~1.0%
Retire to deallocation	20%	~1.0%
FP load latency	35%	~2.0%
Store lifetime	30%	~3.0%
Total	~25%	~15%

shows that a 3D floorplan can overlap the data cache and functional units. In the 3D floorplan, the load data only travels to the center of the D$ at which point it is routed to the other die to the center of the FUs. Now that the same worst case path contains half as much routing distance because the data is only traversing half of the data cache and half of the functional units, we were able to eliminate 1 clock cycle in the load-to-use delay. This stacking is also favorable for thermals because the D$ is relatively low power and the 3D power density is lower than the planar floorplan's hottest area over the instruction scheduler. That is, even though the power density increases slightly, it is still significantly less than the power density of the worst case hotspot. Another example illustrated in Figure 21 is the register file (RF) data out to the floating point (FP) unit input. The single instruction multiple data (SIMD) unit is intentionally between the FP and RF because the planar floorplan is optimized for the more critical SIMD applications. This placement adds 2 cycles to the latency of all FP instructions. In the 3D floorplan in Figure 22, it is possible to optimize for both FP and SIMD, eliminating the two cycles previously allocated for wire delay, and thus improving the performance of all FP applications, while not hurting SIMD applications.

Using this stacking strategy, we eliminated 25% of all pipe stages in the microarchitecture with the 3D floorplan simply by reducing metal runs. Table V enumerates various macrofunctional modules of the machine and the percent of the planar pipe stages eliminated with the new 3D floorplan. One notable improvement is a 30% reduction in the lifetime of store instructions after retirement. This greatly reduces the energy per store instruction and improves performance significantly by more efficiently utilizing the limited capacity of the store queues. Previous studies [Rahman and Reif 2000; Ha et al. 2000] use Rent's Rule to mathematically demonstrate metal reduction resulting from 3D wiring. This work demonstrates the microarchitecture impact of reducing wire delay.

5.2 Splitting

It is also possible to split individual blocks between the two dies in a 3D stack which reduces intrablock interconnect. In fact, large blocks prefer splitting to

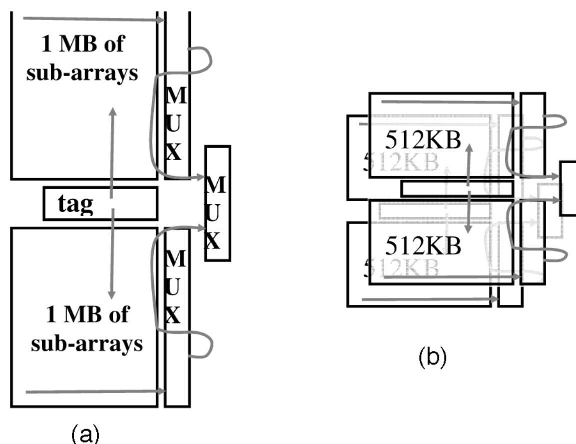


Fig. 23. 1MB UL2 block diagrams for (a) planar and (b) 3D implementations.

stacking because internal wire and latencies can be eliminated that would otherwise be unaffected if stacked. Splitting large blocks also reduces the distance traveled by external signals that cross the block. Splitting a block is an involved process that requires a new circuit implementation and block floorplan. In this study, we split a handful of sample blocks including the UL2 cache and scheduler components in order to demonstrate further power and latency reductions.

The 1MB UL2 is a large portion of the die and contains very long wires making it a good candidate for splitting. A high-level block diagram of the planar UL2 cache is illustrated in Figure 23(a). The cache consists of two subarray groups with the tags running vertically in the middle. There is a critical data path called the loop back which loops through to the final MUX. An obvious but noneffective splitting option is to simply stack the two subarray groups on top of each other. The footprint would be reduced 50% but there would be no change in wire lengths. The tag lines, wordlines, and loop backs will be unchanged. As suggested in Section 3.1, another approach is to split the subarrays. Figure 24(a) illustrates the planar array which is x rows by y columns. The y columns indicate the wordline length and contribute significantly to the subarray delay. An interesting split is to divide the subarray into an odd and even subarray stack cutting the wordline RC in half, reducing read latency, see Figure 24(b). There is further gain at the next level; referring to Figure 23(b), the new shape of the subarrays cuts the RC of the taglines and loop backs by 50%. The result is a cache read latency reduction of 25%, and a cache power reduction of 20%.

While generating the 3D floorplan described in the previous section, the thermal restriction of not increasing power density prevents blocks from being stacked above the hottest blocks in the planar design which in this case is the instruction scheduler. Splitting a block dramatically changes the timing of internal interconnect. In the case of this scheduler, it was possible to split along two long-delay M5 routes. The result is a 15% shorter access latency. Next, we used the additional timing slack to reduce the power of the very aggressive

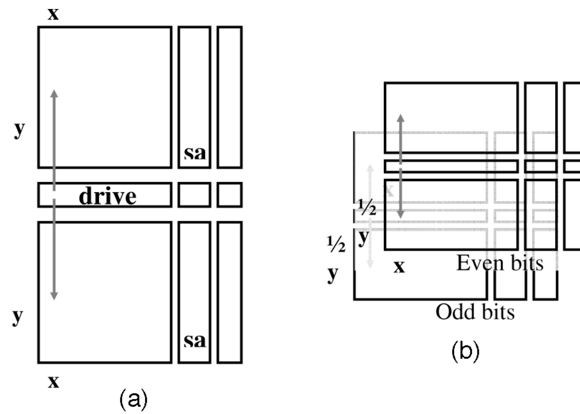


Fig. 24. UL2 subarray block diagrams for (a) planar and (b) 3D implementations.

dynamic circuit implementation by converting it to a static implementation. The final split scheduler was slightly smaller and had 50% of the original power consumption at the same frequency and performance. Such a successful split yielded a 3D floorplan that does not increase the peak power density.

5.3 Performance

We only consider single-thread performance in this work. We estimate performance based on partial results from an internal Intel® cycle-accurate fully functional performance model that is used for microprocessor development. We used over 650 benchmark traces that include SPECINT, SPECFP, hand written kernels, multimedia, Internet, productivity, server, and workstation applications. Unfortunately, the complexity of this tool prevented a full simulation of all 3D changes. Consequently, only a part of the 3D changes could be simulated directly, while a handful of the others were approximated from historical model behavior, and still others were not included in the performance gain such as the faster UL2.

As described earlier and listed in Table V, the new 3D floorplan reduces the total number of pipe stages in the microarchitecture by 25%. Fewer pipe stages result in shorter function latencies and better use of limited resources. A 15% higher IPC at the same frequency is achieved by eliminating piped wire stages, reducing delay between blocks and eliminating wire within blocks. Table V shows the harmonic mean average performance increase for each latency improvement.

5.4 Power

It is important to note that this increase in IPC does not increase activity and power because the energy per instruction was reduced with the reduction in total metal and latches. Normally an IPC increase causes more activity and increases the power. Baseline power data for the planar design is gathered using performance model activities and detailed power data from each block in the design. We estimated 3D power from the baseline by scaling according to

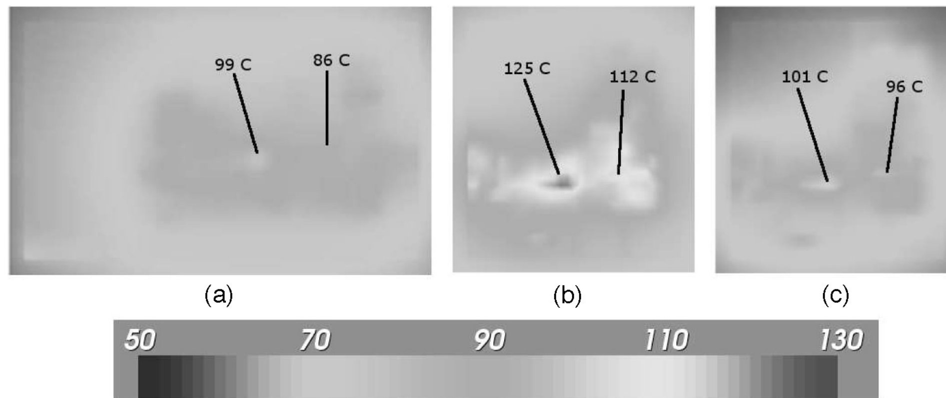


Fig. 25. Thermal maps for (a) the baseline 2D processor, (b) a 3D processor with no reduction in total power, and (c) a 3D processor with lower power due to decreased interconnect.

the proposed design modifications. The pipe stages that were removed, outlined in Table V, were dominated by long global metal. As a result, the number of repeaters and repeating latches in the implementation is reduced by 50%. We believe that more latches and repeaters can be removed with further tuning. We also assume that the two dies in the 3D floorplan share a common clock grid. The 3D clock grid will have 50% less metal RC than the planar design because the 3D floorplan footprint is 50% smaller. Clock grid details are beyond the scope of this section. Fewer repeaters, a smaller clock grid, and significantly less global wire yield a 15% power reduction. By splitting individual FUBs such as the last-level cache and instruction scheduler, we can reduce total power by another 10% for an overall chipwide power reduction of 25%. These power improvements reduce energy per instruction with no negative impact to performance. Therefore, the new 3D floorplan yields a 25% power reduction while simultaneously increasing performance by 15%. In addition to an improved performance-per-Watt ratio, this power reduction is critical for managing 3D thermals as we will discuss in the following section.

5.5 Thermals

We measured thermals using an internally developed tool that accurately models all aspects of the 3D structure for thermal dissipation. The thermal plots shown in Figure 25 were generated from power profiles of over 1000 FUBs. Figure 25(a) shows the thermal map for the baseline 2D processor before splitting into 3D. The total power for this planar configuration is 147W, and the hottest spot (highlighted in the figure) is at 99°C. Figure 25(b) shows the resulting thermals for the bottom die (farthest from the heatsink) when the processor is implemented in a two-die stack, however, no power benefit due to 3D has been accounted for. The total power consumption is still 147W (86W/61W on the bottom/top die), but the peak power density has increased by as much as $1.4\times$ which results in a worst case temperature of 125°C. Note that, in addition to exacerbating existing hotspots, naive 3D stacking with no power reduction also introduced new hotspots, one of which is highlighted in Figure 25(b).

A 26°C increase in the worst case temperature would likely make 3D an unattractive technology. However, we have not yet accounted for the power savings that 3D provides through reduction of power throughout the processor. As discussed in the previous section, our 3D processor consumes 25% less power or 110W (63W/47W on the bottom/top die). Figure 25(c) shows the thermal map for our 3D processor when we properly account for the power reduction. The worst case hotspot is now only 101°C, which is nearly the same as in the original 2D case of 99°C. 3D integration can simultaneously reduce wire delay and power, thereby providing performance benefits while keeping the thermal problem under control.

We observed that the temperature differences between the two die were relatively small. The difference between the worst case hotspot on the bottom die and the worst case hotspot on the top die is only 3°C for the 147W 3D processor in Figure 25(b), and 2°C different for the 110W 3D processor in Figure 25(c). This suggests that the 3D structure itself is not the primary cause of thermal problems and that power density and total power are still the primary thermal factors.

While the worst case hotspot has only increased by 2°C, we observed another hotspot of 96°C to the right of the worst case hotspot in Figure 25(c). This hotspot is approximately 10°C hotter than the corresponding location in the 2D implementation. The 3D organization can increase both power density and the number of hotspots but by using 3D to decrease the total power consumption, we can still keep the worst case chip temperature under control while increasing performance by 15%.

5.6 Summary

This section demonstrated the potential value of 3D technology to a real iA32 high-end microprocessor. We have shown that there are distinct advantages to a 3D structure that can be exploited to increase the performance and/or decrease the power of a heavily-pipelined machine. The example in this work shows that a 3D implementation of a real iA32 microprocessor compared to a planar implementation can improve performance by 15%, while simultaneously decreasing power by 25% with a neutral impact on worst case chip temperature. These results are a generational improvement yielding over 50% improvement in performance/power efficiency. From our study, we conclude that an effective 3D design must specifically target wire-length reductions to maximize performance, minimize power, and keep thermals under control. RC due to metal dominates the performance and power of all designs and implementations, and therefore it is expected that similar gains can be found from a variety of microarchitectures from small to large, fast to slow, graphics to DSP.

6. CONCLUSION

Three-dimensional integrated circuits (3D ICs) are attractive options to overcome the barriers in interconnect scaling for future microprocessor design. This article gives a brief introduction to 3D integration technology, discusses example EDA design tools that can enable the adoption of 3D ICs, and presents the

implementation of various microprocessor components using 3D technology. An industrial case study is presented as an initial attempt to design 3D microarchitectures. It is demonstrated that the 3D microprocessor design shows promising benefits in terms of performance and power. Other benefits of 3D microarchitecture designs, such as the support of the realization of heterogeneous substrate integration and the bandwidth improvement, are not explored in this article. However, these benefits could also be exploited by the future implementations of 3D microarchitectures.

Even though 3D microarchitecture design promises great benefits, there are several challenges that need to be addressed to enable a wide adoption of such technology. It is imperative to develop novel approaches to thermal management due to the higher power density in 3D chips. A complete commercial CAD flow for 3D design is also essential to allow the 3D technology to be widely adopted in microprocessor designs. Even though a smaller footprint in 3D design can improve the yield for individual dies, achieving high yield for wafer bonding 3D technology is essential to justify the cost of moving from a 2D to a 3D design paradigm. Nevertheless, with the fast progress in the design of 3D circuits, we expect to see 3D technology becomes a reality for commercial microprocessor implementations in the future.

ACKNOWLEDGMENTS

The EDA tool implementation in Section 3 was done by Yuh-fang Tsai and Wei-lun Hung. Much of the circuit analysis in Section 4 was conducted by Kiran Puttaswamy. Nick Samra from Intel Corporation worked on the microarchitecture of the microprocessor described in Section 5.

REFERENCES

ALBAYRAKTAROGLU, K., JALELL, A., WU, X., FRANKLIN, M., JACOB, B., TSENG, C.-W., AND YEUNG, D. 2005. Biobench: A benchmark suite of bioinformatics applications. In *Proceedings of the International Symposium on Performance Analysis of Systems and Software*. Austin, TX. 2–9.

AUSTIN, T., LARSON, E., AND ERNST, D. 2002. Simplescalar: An infrastructure for computer system modeling. *IEEE Micro Magazine*, 59–67.

AUSTIN, T. M., BREACH, S. E., AND SOHI, G. S. 1994. Efficient detection of all pointer and array access errors. In *Proceedings of the SIGPLAN Conference on Programming Language Design and Implementation*. Orlando, FL. 290–301.

BERNSTEIN, K. 2006. Introduction to 3d integration. In *International Solid State Circuits Conference Tutorial*.

BRENT, R. P. AND KUNG, H. T. 1982. A regular layout for parallel adders. *IEEE Trans. Comput.*, 260–264.

CHANG, Y., CHANG, Y., WU, G.-M., AND WU, S.-W. 2000. B*-trees: A new representation for nonslicing floorplans . In *Proceedings of the Annual ACM/IEEE Design Automation Conference*.

CHU, C. AND WONG, D. 1997. A matrix synthesis approach to thermal placement. In *Proceedings of the International Symposium on Physical Design (ISPD'97)*.

CONG, J., WEI, J., AND ZHANG, Y. 2004. A thermal-driven floorplanning algorithm for 3d ics. In *Proceedings of the International Conference on Computer Aided Design (ICCAD)*.

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 (ISPD'97)*. ACM Press, New York, NY. 108–115.

Davis, W. R., Wilson, J., Mick, S., Xu, J., Hua, H., Mineo, C., Sule, A. M, Steer, M., and Franzon, P. D. 2005. Demystifying 3d ics: The pros and cons of going vertical. *IEEE Design and Test of Comput. 22*, 498–510.

Deng, Y. and Maly, W. 2004. 2.5D system integration: A design driven system implementation schema. In *Proceedings of the Conference on Asia South Pacific Design Automation*.

Goplen, B. and Sapatnekar, S. 2003. Efficient thermal placement of standard cells in 3D ICs using a force directed approach. In *Proceedings of the International Conference on Computer Aided Design (ICCAD)*.

Gupta, S., Hilbert, M., Hong, S., and Patti, R. 2004. Techniques for producing 3d ics with high-density interconnect. In *Proceedings of the 21st International VLSI Multilevel Interconnection Conference*. Waikoloa Beach, HI.

Guthaus, M. R., Ringenberg, J. S., Ernst, D., Austin, T. M., Mudge, T., and Brown, R. B. 2001. Mibench: A free, commerically representative embedded benchmark suite. In *Proceedings of the 4th Workshop on Workload Characterization*. Austin, TX. 83–94.

Ha, P. Z., Davis, J., and Meindl, J. 2000. Prediction of net length distribution for global interconnects in a heterogeneous soc. *IEEE Trans. VLSI Syst. 8*, 6 (Dec.), 649–659.

Hennessy, J. and Patterson, D. 2003. *Computer Architecture: A Quantitative Approach* 3rd Ed. Morgan Kaufmann.

Hinton, G., Sager, D., Upton, M., Boggs, D., Carmean, D., Kyler, A., and Roussel, P. 2001. The microarchitecture of the pentium 4 processor. *Intel Techn. J.*

Hung, W., Link, G., Xie, Y., Narayanan, V., and Irwin, M. J. 2006. Interconnect and thermal-aware floorplanning for 3d microprocessors. In *Proceedings of the International Symposium of Quality Electronic Devices*.

Jouppi, N. P. 1990. Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffers. In *Proceedings of the 17th International Symposium on Computer Architecture*. Seattle, WA. 364–373.

Jung, S. M., Jang, J., Cho, W., Moon, J., Kwak, K., Choi, B., Hwang, B., Lim, H., Jeong, J., Kim, J., and Kim, K. 2004. The revolutionary and truly 3-dimentional 25F2 SRAM technology with the smallest S3 cell, 0.16um2 and SSTFF for ultra high density SRAM. *VLSI Techn. Dig. Techn. Papers*, 228–229.

Kang, Y. H. Jung, S. M., Jang, J. H., Moon, J. H., Cho, W. S., Yeo, C. D., Kwak, K. H., Choi, B. H., Hwang, B. J., Jung, W. R., Kim, S. J., Kim, J. H., Na, J. H., Lim, H., Jeong, J. H., and Kim, K. 2004. Fabrication and characteristics of novel load PMOS SSTFT (stacked single-crystal thin film transistor) for 3-dimentional SRAM memory cell. In *Proceedings of the IEEE Silicon-on-Insulator Conference (SOI)*. 127–129.

Kogge, P. M. and Stone, H. S. 1973. A parallel algorithm for the efficient solution of a general class of recurrence equations. *IEEE Trans. Comput.*, 786–793.

Larson, E., Chatterjee, S., and Austin, T. 2001. Mase: A novel infrastructure for detailed microarchitectural modeling. In *Proceedings of the International Symposium on Performance Analysis of Systems and Software*. Tucson, AZ. 1–9.

Lee, C., Potkonjak, M., and Mangione-Smith, W. H. 1997. Mediabench: A tool for evaluating and synthesizing multimedia and communication systems. In *Proceedings of the 30th International Symposium on Microarchitecture*. Research Triangle Park, NC. 330–335.

Lee, K. W. Nakqmura, T., Ono, T., Yamada, Y., Mozukusa, T., Hashimoto, H., Park, K. T., Kuring, H., and Koyanag, N. 2000. Three-dimensional shared memory fabricated using wafer stacking technology. *International Electron Devides Meeting (IEDM). Technical Digest*, 165–168.

Link, G. and Narayanan, V. 2006. Thermal trends in emergent technologies. In *Proceedings of the International Symposium of Quality Electronic Devices*.

Lipasti, M. H., Mestan, B. R., and Gunadi, E. 2004. Physical register inlining. In *Proceedings of the 31st International Symposium on Computer Architecture*. München, Germany. 325–335.

Mayega, J., Erdogan, O., Belemjian, P. M., Zhou, K., McDonald, J. F., and Kraft, R. P. 2003. 3d direct vertical interconnect microprocessors test vehicle. In *Proceedings of the ACM Great Lakes Symposium on VLSI*. Washington, DC. 141–146.

Palacharla, S. 1998. Complexity-effective superscalar processors. Ph.D. thesis, University of Wisconsin.

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

PUTTASWAMY, K. AND LOH, G. H. 2005. Implementing caches in a 3d technology for high performance processors. In *Proceedings of the International Conference on Computer Design*. San Jose, CA.

RAHMAN, A. AND REIF, R. 2000. System level performance evaluation of three-dimensional integrated circuits. *IEEE Trans. VLSI Syst. 8*, 6 (Dec.), 671–678.

REIF, R., FAN, A., CHEN, K., AND DAS., S. 2002. Fabrication technologies for three-dimensional integrated circuits. In *Proceedings of the International Symopsium on Quality Electronic Devices*. 33–37.

SEZNEC, A., FELIX, S., KRISHNAN, V., AND SAZEIDES, Y. 2002. Design tradeoffs for the alpha ev8 conditional branch predictor. In *Proceedings of the 29th International Symposium on Computer Architecture*. Anchorage, AK.

SHIU, P. AND LIM, S. K. 2004. Multi-layer floorplanning for reliable system-on-package. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*.

SHIVAKUMAR, P. SHIVAKUMAR, P., AND JOUPP, N. P. 2001. Cacti 3.0: An integrated cache timing, power, and area model. *Western Research Lab Research Report*.

SKADRON, K., STAN, M. R., HUANG, W., VELUSAMY, S., SANKARANARAYANAN, K., AND TARJAN, D. 2003. Temperature-Aware Microarchitecture. In *Proceedings of the International Sumposium on Computer Architecture 30*, 2, 2–13.

SKLANSKY, J. 1960. Conditional sum addition logic. *IRE Trans. Electron. Comput. 9*, 2 (June), 226–231.

SOURI, S. J., BANERJEE, K., MEHROTRA, A., AND SARASWAT, K. C. 2000. Multiple si layer ics: motivation, performance analysis, and design implications. In *Proceedings of the 37th Conference on Design Automation (DAC '00)*. ACM Press, New York, NY. 213–220.

TEZZARON SEMICONDUCTORS. 2005. Tezzaron unveils 3d SRAM. `http://www.tezzaron.com`.

TSAI, C. AND KANG, S. 2000. Cell-level placement for improving substrate thermal distributio. *IEEE Trans. Comput.-Aided Design Integrat. Circuits Syst*.

TSAI, Y., XIE, Y., NARAYANAN, V., AND IRWIN, M. J. 2005. Three-dimensional cache design exploration using 3dcacti. In *Proceedings of the IEEE International Conference on Computer Design (ICCD'05)* 519–524.

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.

ZHANG, K., BHATTACHARYA, U., CHEN, Z., HAMZAOGIU, F., MURRAY, D., VALLEPALLI, N., WANG, Y., ZHENG, B., AND BOHR, M. 2004. A SRAM Design on 65nm CMOS technology with Integrated Leakage Reduction Scheme. *IEEE Symposium On VLSI Circuit. Digest of Technical Papers*, 294–295.

Received April 2006; revised May 2006; accepted May 2006