3D Architecture Modeling and Exploration

Jason Cong[†,‡] Eren Kursun[†] Yongxiang Liu[†] Yuchun Ma[*] Glenn Reinman[†]

[†]Computer Science Department, University of California, Los Angeles
[‡] California NanoSystems Institute
[*] Tsinghua University, China

Abstract

Vertical integration (3D ICs) has demonstrated the potential to reduce inter-block wire latency through flexible block placement and routing. However, there is an untapped potential for 3D ICs to reduce intra-block wire latency through architectural designs that can leverage multiple silicon layers in innovative ways. Furthermore, it is particularly challenging to simultaneously explore the physical design space and microarchitectural space for vertical integration. The physical design space typically has no information on the microarchitectural impact of latency optimization, and the microarchitectural space has no information on the physical design impact of different architectural alternatives.

We make the following contributions in this paper: (1) the introduction of port partitioning, a new approach to constructing multi-layer blocks; (2) the extension of a microarchitectural exploration tool to include the ability to model multi-layer blocks and to consider these blocks as alternative implementations of single-layer architectural blocks on the fly, within a single floorplanning run; and (3) the evaluation of vertical integration on a design driver using this framework.

For this design driver, we see an average 36% improvement in performance (measured in BIPS) over a single-layer architecture, and a 29% improvement in performance over a multi-layer architecture with single-layer blocks. The on-chip temperature is kept below 40°C.

1. INTRODUCTION AND MOTIVATION

Vertical integration [23, 30, 35, 32, 14] leverages multiple layers of silicon to allow physical designers more flexibility in component layout. One approach to using this technology is to place single-layer (i.e., 2D) blocks in one of the silicon layers and run both horizontal and vertical interconnect between blocks. The flexibility that this design affords has the potential to dramatically reduce inter-block interconnect latency in a design [10, 1, 2, 6].

However, this approach does little to help intra-block wire latency. And despite the advantage of almost completely eliminating inter-block wire latency, we find that the placement of 2D blocks in two layers improves performance only by 6% on average for a particular architecture [6]. Additional gains from the use of vertical integration are needed from attacking the intra-block wire latency.

Furthermore, the emergence of technology like vertical integration can have a dramatic impact on microarchitecture design – a field that heavily relies on physical planning and technological innovation. However, physical planning is not meaningful without consideration of microarchitectural loop sensitivities: some loose loops [3] can tolerate latency better than others [31]. A floorplan with a 5% reduction in wirelength may actually be better than a floorplan with a 7% reduction in wirelength – if the former reduces the length of more critical microarchitectural loops than the latter. Similarly, architectural innovations are not meaningful without understanding their physical design implications.

Recently, the MEVA-3D [6] framework was proposed to bridge the gap between physical planning and microarchitectural design. The framework uses microarchitectural loop sensitivities in the floorplanning process to guide block placement. With this framework, architects can obtain accurate loop latencies to feed to a cycle-accurate simulation framework. This can help evaluate the impact of new and emerging technologies on microprocessor design.

In this paper, we explore the architectural impact and potential of finer granularity vertical integration, where individual blocks are placed across multiple layers. The challenge from the architectural side is the construction of blocks that can span multiple layers. The challenge for physical design is to automate the process of placing blocks in multiple layers.

To address these challenges, we make the following contributions:

- 3D Architectural Blocks: We propose *port partitioning*, an approach that places architectural blocks like register files, issue queues, and caches in multiple silicon layers. We compare port partitioning with wordline/bitline partitioning [33] with respect to area, timing, power, and required vertical interconnect.

- 3D Design Driver Exploration: We explore the design space of different partitioning schemes for a particular design driver architecture, using one to four layers of silicon. In addition to exploring the use of single-layer and multi-layer blocks, we consider increasing the sizes of different architectural structures, using the timing slack from vertical integration. In some cases, the timing slack can enable the use of larger instruction or scheduling windows, or larger caches.

In addition to helping latency, the reduction in wire RC delay can reduce power dissipation. However, the stacking of components can adversely impact the temperature of the microprocessor. It is therefore essential for any study using vertical integration to make use of accurate temperature modeling to demonstrate the effectiveness of any architecture. All of our explorations are enhanced with a state-of-the-art, accurate, temperature simulator tool. We also consider automated thermal via insertion to help mitigate the impact of temperature.

The remained of this paper is organized as follows: We review the prior work on 3D integration technology, microarchitectural exploration techniques, and block modeling in Sec-

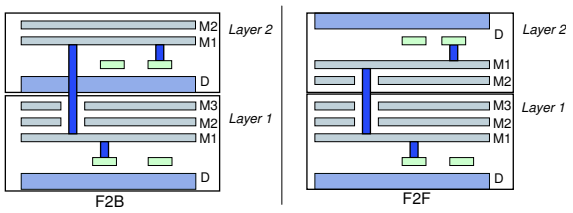

Figure 1: Face-to-Back and Face-to-Face integration technologies.

tion 2. Next, we detail and evaluate our 3D architectural blocks in Section 3. Our 3D block placement enhancements are detailed in Section 4. We finally explore a design driver microarchitecture in Section 5, and then conclude in Section 6.

2. RELATED WORK

In this section we focus on prior work that is most relevant to our study.

2.1 3D Technologies

While a number of 3D IC fabrication technologies have been proposed [20, 24, 22], we consider the use of wafer bonding [1, 2, 10] in this study. In this technology, fully processed wafers are bonded together, and devices are fabricated on these wafers. Interlayer vias that connect different layers are etched after metalization and prior to wafer bonding. Two main types of wafer bonding strategies have been evaluated in prior work [2, 10]: Face-to-Back (F2B) placement and Face-to-Face (F2F) placement (Figure 1). Vias in F2B cut through device layers in addition to metal layers. In F2F placement, the top device layer is flipped to face the lower device layer. Metal layers are placed between the facing device layers. Hence, vias cut through metal layers only. However, F2F cannot scale beyond two layers without also employing F2B layers.

2.2 3D Microarchitectural Exploration

MEVA-3D [6] is an automated exploration framework that can explore a 3D design space for an optimal placement of 2D architectural blocks into multiple device layers. MEVA-3D optimizes a cost function that is configured to weigh latencies of critical microarchitectural loops, temperature, and die area. The critical loop latency is the sum of individual block latencies along the loop and inter-block wire latencies. Critical loop latencies relate to performance (IPC) as in [31]. The algorithm returns a floorplan with the best overall performance, temperature and die area for a given target frequency. MEVA-3D leverages SimpleScalar [4] to validate its performance estimate. MEVA-3D can also perform automated thermal via insertion to help mitigate areas of high power density. MEVA-3D is further enhanced with the capability to explore 3D designs using 3D blocks [18]. It treats the 3D blocks as cubic blocks to be packed in 3D space, and packs a combination of 2D and 3D blocks using an floorplanner extended from [19]. More details will be covered in Section 4.

2.3 2D and 3D Block Modeling

Prior work provided block models for various architectural structures including caches [34], register files [12, 25], and wakeup and select logic [25]. CACTI [34, 28, 29] is an analytical model that provides timing, area, and power results for different cache configurations. CACTI models different levels of associativity, multiporting, and sub-banking, and ideally scales to different feature sizes using 0.80μm cache data. The work in [33] extended CACTI to explore 3D cache designs. However, they only considered folding blocks by wordlines or bitlines, and not by port partitioning as we do in this work. In addition, they did not explore the impact of this 3D design on the overall microarchitecture (i.e. performance, temperature, layout), or the impact of 3D stacking on area in general. Puttaswamyy et al. [16] showed the delay benefit and the reduction of power consumption in a stacked cache design by bank-stacking or array-splitting. Palacharla et al. [25] built detailed transistor-level models for critical structures in dynamically scheduled processors, analyzing critical timing paths and the scalability of these structures. However, this study was limited to single-layer structures.

Port partitioning for cache structures and tag partitioning for issue queue have been proposed independently in [9, 26]. In this paper, we review the 3D-stacking strategies of critical architectural units such as caches and issue queue. We further explore overall architectural benefits by constructing microprocessors using 3D building blocks.

3. 3D ARCHITECTURAL BLOCK DESIGN AND MODELING

To reduce intra-block interconnect latency, we evaluate two main strategies for designing blocks in multiple silicon layers: *block folding* and *port partitioning*. Block folding implies either a vertical or horizontal folding of the block - potentially shortening the wirelength in one direction. Port partitioning places the access ports of a structure in different layers - the intuition here is that the additional hardware needed for replicated access to a single block entry (i.e., a multiported cache) can be distributed in different layers, which can greatly reduce the length of interconnect within each layer. In this section, we describe the use of these strategies for the issue queues and various cache-like blocks in our design driver architecture.

3.1 Issue Queues

The issue queue is a critical component of out-of-order microprocessor performance and power consumption. Recent research [6] has shown that every additional pipeline stage of latency seen in the scheduling loop causes an average 5% performance degradation. Moreover, Folegnani and Gonzalez [13] found that the issue queue is responsible for an average 25% of a processor's total power consumption.

The issue queue stores renamed instructions and performs out-of-order instruction scheduling. For purpose of this paper, we study an issue queue based on Palacharla's implementations [25]. There are two main stages of issue queue functionality: the wakeup stage where tags from completing register values are compared against input register tags stored in issue queue entries, and a selection stage where ready instructions (as determined by the wakeup stage) are selected for execution.

Each issue queue entry must track and compare the input register tags required by a given instruction in that entry. Figure 2 shows a single CAM cell used to store one bit of a register tag for an issue queue entry. Assuming that at most four register values can be written back each cycle, and at most four new instructions can enter the issue queue each cycle, an individual cell would have four different 1-bit tags to compare against and have four write ports. In a processor with a 128-entry physical register file, register tags are 7-bits. Therefore each row would need seven CAM cells for each operand, for a total of fourteen CAM cells. In general, an N-entry issue queue has N such rows.

In the wakeup stage, the match lines for each issue queue

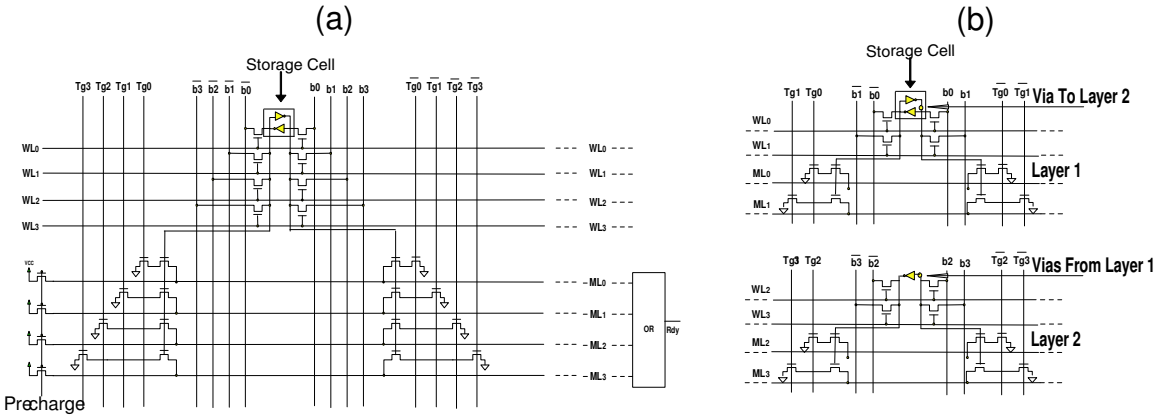

Figure 2: (a): A Single IQ Cell with Four Tag Lines and Four Access Ports. Over 99% of the area is occupied by tags and access ports. (b): Port Partitioning. Tags and access ports are distributed into two layers. Width and height of each bit are reduced by half, and area by 75%.

entry are precharged high, and the tag lines are driven with the register tags of completed instructions. A match line only remains high if the register tag stored at the issue queue entry is the same as a certain one of the register tags driven on the tag lines. If any match line for a given input register remains high, the ready bit for that operand is set in the issue queue. Once both ready bits are set, the operand is eligible for issue (i.e., has woken up). In this stage, most of the delay comes from tag broadcasting and matching.

In the selection stage, the select logic picks instructions to execute among all instructions that are eligible for issue [25].

For example, a selection tree for a 32-entry issue queue consists of three levels of arbiters. Each arbiter takes four input requests (i.e., four eligible instructions) and grants one request (i.e., selects one eligible instruction). In general, an N-entry issue queue needs a selection tree of level $L = log_4 N$.

In the issue queue, the delay due to wakeup logic contributes a large portion of the overall delay. Our simulations show that wakeup takes about 60% of the delay in a 32-entry issue queue with four incoming register tags to compare against, and four access ports. A significant contributor to delay is the wire latency of the tag bits and match lines. A 3D integrated issue queue can significantly reduce the length of these wires.

3.1.1 3D IQ Design: Block Folding

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

3.1.2 3D IQ Design: Port Partitioning

In an issue queue with four tag comparison ports and four read/write ports, as shown in Figure 2(a), most of the silicon area is allocated to ports. The wire pitch is typically five times the feature size [28, 29, 25]. For each extra port, the wire length in both X and Y directions is increased by twice the wire pitch [28, 29]. On the other hand, the storage, which consists of four transistors, is twice the wire pitch in height, and has a width equal to the wire pitch. Hence, in the cell shown in Figure 2(a), the storage area is less than 1% of the total area, while tags and access ports occupy over 99% of the total area.

One strategy for attacking the tag and port requirements is port partitioning, which places tag lines and ports on multiple layers, thus reducing both the height and width of the issue queue. The reduction in tag and matchline wire length

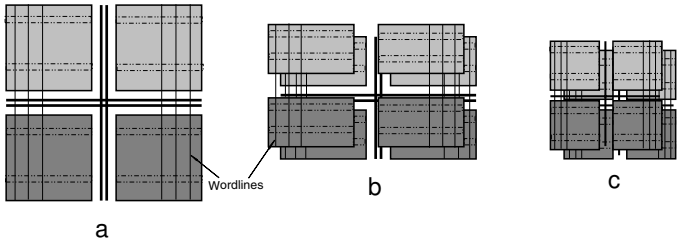

Figure 3: Cache Block Alternatives (a) A 2-Ported Cache: the two lines denote the input/output wires of two ports. (b) Wordline Folding: Only Y-direction length is reduced. Input/output of the ports are duplicated. (c) Port Partitioning: Ports are placed in two layers. Both X and Y direction lengths are reduced.

can help reduce both power and delay. The selection logic also benefits from this, as the distance from the farthest issue queue entry to the arbiter is reduced. This will speed up the comparison and also reduce power consumption.

3.2 Caches

The cache is another commonly found architectural block with regular structures - they are composed of a number of tag and data arrays. Figure 3(a) demonstrates a high level view of a number of cache tag and data arrays connected via address and data buses. Each vertical and horizontal line represents a 32-bit bus – we assume two ports on this cache, and therefore the lines are paired. Each box of the figure is a tag or data array, which is composed of a mesh of horizontal wordlines and vertical bitlines. Every port must have a wordline for each cache set and a pair of bitlines for each bit in a cache set. The regularity of caches means that their components can easily be subdivided. For example, the tag and data arrays can easily be broken down into subarrays. We make use of CACTI [29] to explore the design space of different subdivisions and find an optimal point for performance, power, and area.

3.2.1 3D Cache Design: Block Folding

Prior research [33] looks into two folding options: wordline and bitline folding. In the former, the wordlines in a cache sub-array are divided and placed onto different silicon layers. The wordline driver is also duplicated. The gain from wordline folding comes from the shortened routing distance from predecoder to decoder and from output drivers to the edge of the cache.

Similarly, bitline folding places bitlines into different layers. This approach needs to duplicate the pass transistor.

The sense amplifier can be duplicated to improve timing performance at a cost of increased power consumption. The cost is significant because sense amplifiers can make up a significant portion of total cache energy consumption. The other approach is to share sense amplifiers across layers, but this dramatically reduces the improvement in timing.

Our investigation shows that wordline folding has a better access time and lower power dissipation in most cases when compared to a realistic implementation using bitline folding. In this paper, we only present results using wordline folding.

3.2.2 3D Cache Design: Port Partitioning

The port partitioning strategy that we proposed for the issue queue can also be leveraged for caches. For example, a 3-ported structure would have a port area to cell area ratio of approximately 18:1. Hence, there is a significant advantage to partitioning the ports and placing them onto different layers. In a two layer design, we can place two ports on one layer, one port and the SRAM cells on the other layers. The width and height are both approximately reduced by a factor of two, and the area by a factor of four.

3.3 Other Cache-Like Architectural Blocks

Register files are similar to caches, sharing the regularity of a cache. We therefore adapt our CACTI to model this structure as well. However, register files are not associative and typically have more ports than caches. Register files dissipate relatively large amounts of power due to their porting requirements, and the size of the physical register file can constrain the size of the instruction window in a dynamically scheduled superscalar processor. We will consider the same folding schemes for the register files as those used for caches.

The register mapping units, load-store queue, and branch predictors can be approximated using only the data array portion of the cache.

3.4 Modeling Methodology

We assume a supply voltage of $1.0V$ and a $70nm$ process technology. Transistor and wire scaling parameters are derived from [33, 21], and we assume copper interconnect in our simulation. Further transistor parameters are obtained from [5]. The 3D via resistance is estimated to be $10^{-8}\Omega cm^2$ [33]. The height of the 3D vias is assumed to be $10\mu m$ per device layer. Current dimensions of 3D via sizes vary from $1\mu m \times 1\mu m$ to $10\mu m \times 10\mu m$ [33, 11]. As 3D technology advances, the 3D via size will decrease even further. In this study, we assume the via pitch is $1.4\mu m$. An area of $0.7\mu m \times 0.7\mu m$ is reserved for each 3D via for the upper layers in F2B technology.

We have modified 3D-CACTI [33] to model caches and cache-like structures. First, we add port partitioning to 3D-CACTI in addition to wordline/bitline folding. Second, we add area estimation, including the area impact of 3D vias on the transistor layer. Both 3D bonding technologies are available: F2B and F2F. We validated our modifications to 3D-CACTI with HSpice.

We implemented our issue queue models using HSpice to obtain accurate timing and power data. The area of the issue queue is approximated by 3D-CACTI using a similarly sized cache. Our 2D issue queue is derived from Palacharla et al.'s model [25].

3.5 3D Block Performance

Figure 4 demonstrates the effectiveness of 3D block design on area, power, and timing for dual-layer F2F blocks. The y-axis is normalized to the area of a single-layer baseline block. The x-axis represents different folding techniques for each architectural block investigated. The letters in the label of a bar

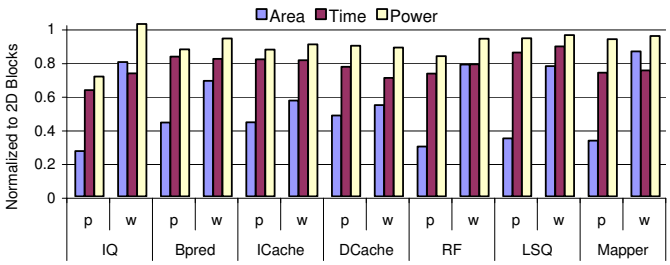

Figure 4: The improvement in area, power and timing for dual-layer vertical integration.

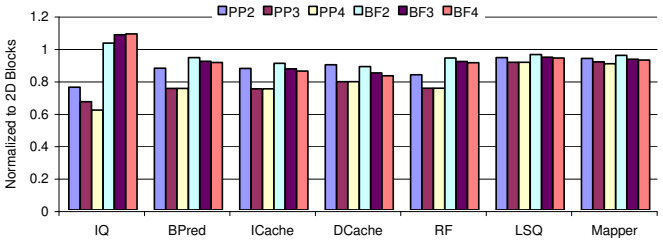

Figure 5: The improvement in power for multi-layer F2B vertical integration.

represent the type of folding: either port partitioning (PP) or block folding (BF). All results are shown normalized to the 2D implementation of the block. In F2F technology, the via starts from the surface of one layer and ends on the surface of the other layer. Therefore, vias do not impact the layout of transistors.

For the issue queue (IQ), delay is reduced by 27% with BF. PP sees even more improvement (37% reduction in delay). PP reduces both tag wire lengths and match wire lengths, and wire lengths to the selection logic. On the other hand, BF only reduces tag wire lengths. The match wire lengths are even increased due to 3D via insertions for every tag and bit line. As a result, we observe an over 70% reduction in area for PP, with only a 20% reduction for BF. Note that the area shown is the maximal area in any one layer for that block, and while the footprint of the block may be reduced, the sum of the area occupied in all layers may actually increase relative to the 2D baseline.

The power consumed in CMOS circuits is represented as $P = 0.5 * a * f * C * Vdd^2$, where f is the clock frequency, a is the activity factor, Vdd is the supply voltage and C is the switching capacitance. The power consumption rate is proportional to the switching capacitance. In BF, although tag wire lengths for each layer are reduced, the tag wires are duplicated on different layers. The aggregate wire length is still the same. In addition, there is an increase in match line lengths mentioned above. Thus, the total switching capacitance is slightly increased due to the increased total wire length. As a result of this, the power consumption of BF is slightly increased. On the other hand, PP is able to reduce power consumption by 29%.

For the caches and cache-like structures, PP is extremely effective in heavily ported structures. For example, the register file with PP sees a 27% reduction in delay, a 17% reduction in power, and an impressive 70% reduction in area. However, for structures with fewer ports, BF can be more effective. The data cache sees a 30% reduction in delay with BF, and a 23% reduction in delay with PP. While PP does reduce both word-

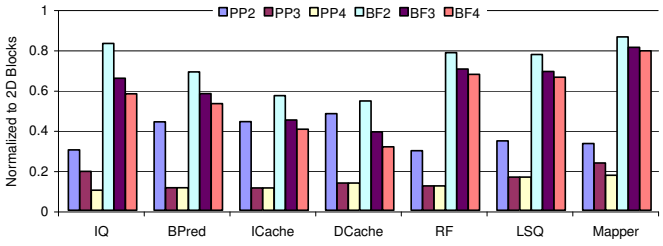

Figure 6: The improvement in area for multi-layer F2B vertical integration.

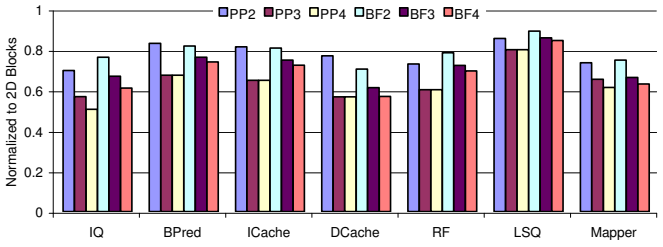

Figure 7: The improvement in timing for multi-layer F2B vertical integration.

line and bitline length, this reduction is proportional to the number of ports that can be partitioned to other silicon layers. For structures with very few ports, BF is able to reduce wordline length more than PP. Hence, in structures that have significant wordline delay, the overall reduction in delay with BF can be greater than PP.

The diversity in benefit from these two approaches demonstrates the need for a tool to flexibly choose the appropriate implementation based on the constraints of an individual floorplan.

3.5.1 Scaling to Multiple Silicon Layers

For a dual-layer implementation, F2F is able to outperform F2B since the 3D vias in F2B impact the silicon footprint in the top silicon layers. For example in the PP results, the F2B area is about 5% larger than that of F2F due to the increased silicon footprint. The delay and power consumption are larger than those of F2F as well. However, F2B allows more layers to be stacked. It may be possible to stack two F2Fs in back to back fashion; however, we do not consider this alternative in this paper.

Figures 7, 5, and 6 show timing, power, and area results (respectively) with F2B blocks for two, three, and four layers of silicon. All measurements are normalized to the performance of a single-layer block. In general, we observe that the reduction of area, power and delay is further increased as the number of layers is increased.

For the issue queue (IQ) with PP, area reduction increases to 80% with 3 layers, and to 90% with 4 layers. Reduction in issue queue delay increases to 43% with 3 layers, and to 50% with 4 layers. Reduction in power consumption grows as high as 38% with 4 layers.

For the issue queue with block folding, there is less reduction in area and delay with additional layers. However, the impact on match line wire length from stacking more layers increases the power consumption for folding to 9% with 4 layers.

3.5.2 Impact of 3D Bonding Technology

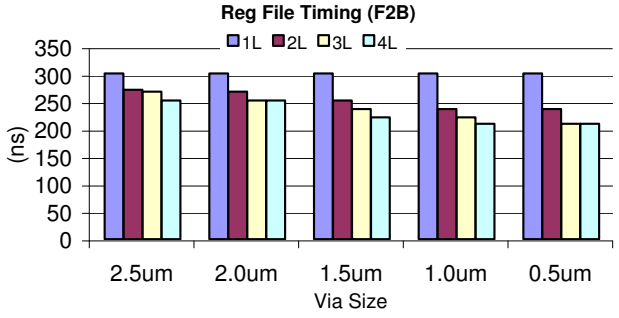

Figure 8: Impact of Via Size on Timing using F2B, Port Partitioning

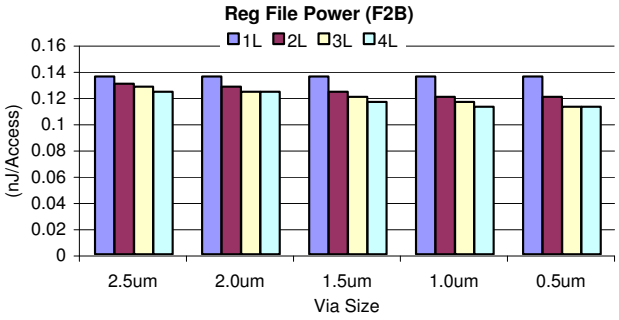

Figure 9: Impact of Via Size in Power using F2B, Port Partitioning

3D via size has rapidly scaled down as 3D bonding technology has advanced. 3D via size has been reduced from $10\mu m$ to $1.75\mu m$ in MIT Lincoln Laboratory's 3D process technology [17] at $180nm$. We expect the 3D via size to continue to scale at smaller feature sizes. In this paper, we have assumed a $0.7\mu m$ via size for a $70nm$ feature size.

To demonstrate the impact of scaling via size, we plot the performance of the register file for via sizes ranging from $2.5\mu m$ to $0.5\mu m$ in $70nm$ technology. The via pitch is twice the via size. The register file has four read ports and four write ports. A single cell size is approximately $5.6\mu m$ x $5.6\mu m$. In F2B bonding technology, 3D vias occupy silicon area in all layers except the bottom layer. Taking 2-Layer partitioning as an example, when via size is $2.5\mu m$, the best solution is to place seven ports in the bottom layer, and one port in the top layer, which only slightly reduces the wirelength. When the via size is scaled to $0.5\mu m$, the best solution places four ports in each layer. The wirelength is almost cut in half in both X and Y directions. As shown in Figures 8 and 9, the larger reduction in wirelength reduces both delay and power as the via size is scaled from $2.5\mu m$ to $0.5\mu m$.

4. 3D BLOCK PLACEMENT

Microprocessor throughput, as measured in IPC, is influenced by the latency of critical architectural loops such as the scheduling loop, branch resolution loop, inter-cluster communication loop, etc [31]. Vertical integration can help to reduce the latency of these critical loops. Critical loops differ in the magnitude of their impact on throughput, and therefore the exploration of the use of vertical integration on microprocessor design requires consideration for both physical design and architecture. Existing work on this type of co-design exploration [6] has only explored the use of vertical integration to reduce inter-block latency in these critical loops. However, as demonstrated in Section 3, there is a tremendous potential for vertical integration to reduce the latency of blocks along

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

Table 1: Architectural parameters for the design driver used in this study.

critical loops. In this section, we detail our modifications to the co-design framework of [6].

MEVA-3D [6, 18] is an automated physical design and architecture performance estimation flow for 3D architectural evaluation. It includes 3D floorplanning, routing, interconnect pipelining, automated thermal via insertion, and associated die size, performance, and thermal modeling capabilities.

First, MEVA-3D takes a microarchitectural configuration, a target frequency, architectural critical path sensitivities, and power density estimates and uses 2D/3D floorplanning to optimize for performance and temperature. Then routing and thermal via planning are performed to provide physical design information to our microprocessor simulation. Critical loop latencies are passed from the floorplanner to the simulator for accurate determination of performance. MEVA-3D makes use of the SimpleScalar [4] simulator to obtain performance in IPC and utilization counts of individual blocks.

To enable the packing of 3D components which may occupy more than one layer, we constructed a new packing engine which is a true 3D packing engine – 3D components in our design can be treated as cubic blocks to be packed in 3D space. The dimension of the block in the Z direction represents the layer information. The 3D packing algorithm is extended from the CBL floorplanner [19].

5. MICROARCHITECTURAL EXPLORATION

In this section, we use the modified MEVA framework to investigate the ability of vertical integration to reduce both intra-block and inter-block architectural latencies.

5.1 Performance of 3D Architecture

We constructed a design driver based loosely on the Alpha 21264 [15], and along with the architectural blocks from Section 3 (functional unit blocks are based on [6]), we feed this driver into our modified version of MEVA-3D. The architectural parameters are shown in Table 1. We measure architectural performance on all 26 programs of the SPEC CPU2000 suite.

Figure 11 presents performance results relative to a single-layer design driver. The first bar represents the benefit from using two layers of silicon with 2D blocks (as in [6]), and the second bar represents the benefit from using two layers of silicon with 3D blocks. All three configurations (single-layer, dual-layer 2D blocks, dual-layer 3D blocks) are running at 4GHz. On average, the use of 2D blocks in a two layer design improves performance by 6%. Since the blocks themselves do not take advantage of vertical integration, any performance gain can only come from a reduction in the inter-block wire latency. For example, the branch misprediction loop has a total latency of 815ps at 4GHz for a single-layer design – 238ps of this total latency is from inter-block wire delay. When using 2D blocks in two layers, this inter-block wire delay is reduced to only 63ps. However, the overall reduction in path delay is not enough to reduce the loop by a cycle of our 4GHz clock. Thus, while timing slack is certainly increased, the benefit of this has not been exploited in Figure 11. When we allow MEVA-3D to select 3D block alternatives, we see a performance improvement of 23% on average over the single-layer architecture. This can be attributed to the ability of 3D blocks to reduce the intra-block latency of critical processor loops.

We show floorplans for all three architectures in Figure 10. The single-layer design occupies $3.4 \times 3.4mm^2$ in one silicon layer. The dual-layer design with 2D blocks occupies $2.8 \times 2.8mm^2$ in each silicon layer. The dual-layer design with 3D blocks occupies $2.3 \times 2.3\ mm^2$ in each silicon layer.

Temperature issues are considered to be a major concern for vertical integration. Therefore, an accurate and fast thermal simulation framework was very crucial for our experimental analysis. We used the finite element method (FEM) based CFD-ACE+ temperature simulator [27]. Further details on the heat sink and thermal parameters that we used can be found in [27]. The average and maximum temperature for the single-layer architecture was 30.6°C and 32.7°C. The average and maximum temperature for the dual-layer architecture with 2D blocks was 30.6°C and 32.6°C. The average and maximum temperature for the dual-layer architecture with 3D blocks was 30.3°C and 34.1°C.

Thermal vias can help to relieve thermal problems in 3D microarchitectures. We used the algorithm proposed in [7, 8] for thermal via insertion. In our multi-layer designs, we designate 5% of the area as dead space on each layer, which provides sufficient space for thermal vias.

5.2 Scaling Architectural Sizes

Even in the 3D block architecture, there are still cases where we are able to increase the timing slack within a given cycle of a critical loop, without actually reducing the number of cycles in that critical loop. Figure 12 presents one approach to leveraging this extra slack: we double the size of the data cache, issue queue, and register file.

Figure 13 shows the timing performance when three structures are scaled from the default size to 16 times larger. As shown in the figure, with 3D integration technology, the access latency of double-sized structures is still less than in that of 2D. The register file and data cache can even quadruple their sizes while still outperforming the default blocks in 2D. In this paper, we limit our study to doubled sizes.

As shown in Figure 12, the performance is increased by an additional 5% with a doubled cache, an additional 1% with a doubled register file, and an additional 7% with a doubled IQ. The best performance is observed when doubling the size of all three structures. Overall, there is a 36% gain over the 2D architecture and a 13% gain over the 3D architecture with our default block sizes.

These larger structures will dissipate more power than regular-sized 3D blocks. But despite the increase in power, the increased area of these larger designs saw an average slight decrease in temperature of 0.8°C for the case where all three resources were doubled. The maximal temperature in this case was 34.1°C.

6. SUMMARY

Vertical integration has a tremendous potential for reducing both inter-block and intra-block wire latency. We have proposed and evaluated tag partitioning for the issue queue, for caches, and for cache-like blocks. And we have enhanced the MEVA-3D exploration framework to evaluate the use of 3D blocks in multiple layers of silicon. When using two layers of silicon with 3D blocks, we see an average 36% improvement in performance over a single-layer architecture and 29% improvement in performance over two layers with single-layer blocks, for the architectural design driver we explored. Temperature is kept below 40°C using a two heat sink F2F design.

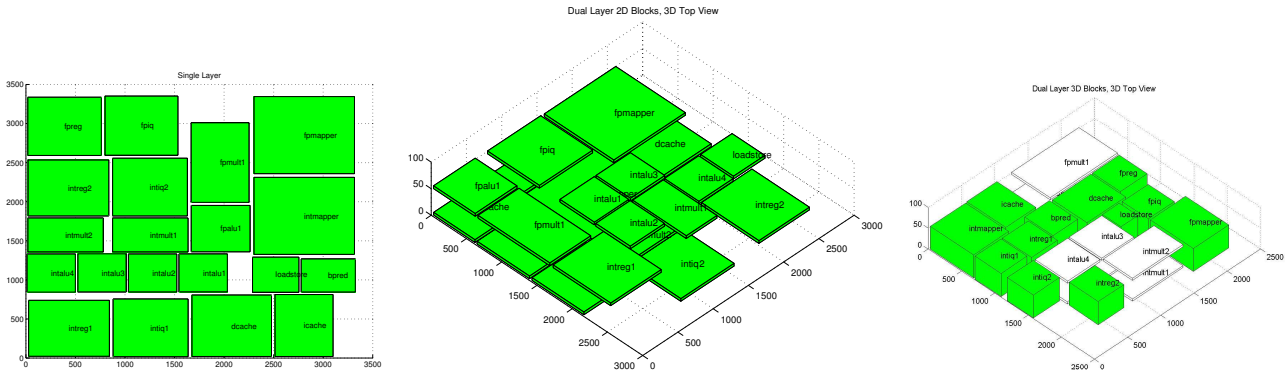

Figure 10: Floorplans for the best architectural configuration as determined by our modified version of MEVA-3D. From left to right: (a) A single Layer Architecture, (b) Dual Layer Architecture with 2D-only Blocks, (c) Dual Layer Architecture with 2D and 3D Blocks. Dark blocks occupy two layers and white blocks occupy one layer.

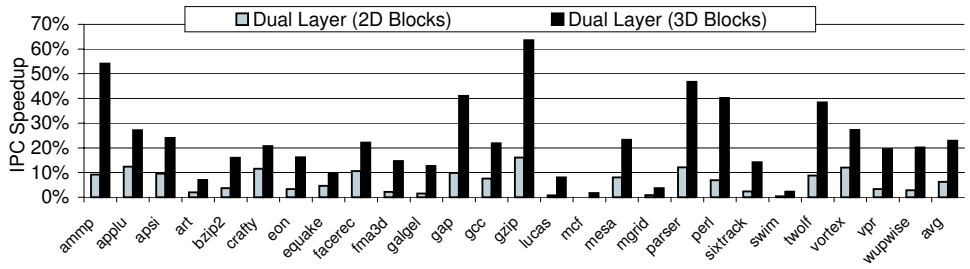

Figure 11: Performance speedup for dual silicon layer architectures relative to a single-layer architecture.

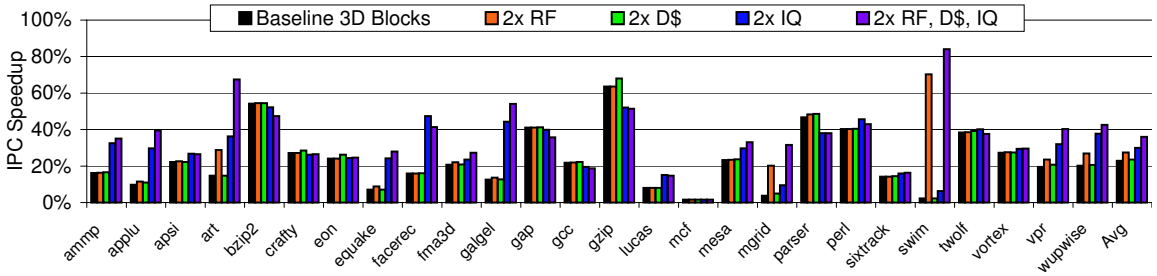

Figure 12: Performance when doubling critical resources.

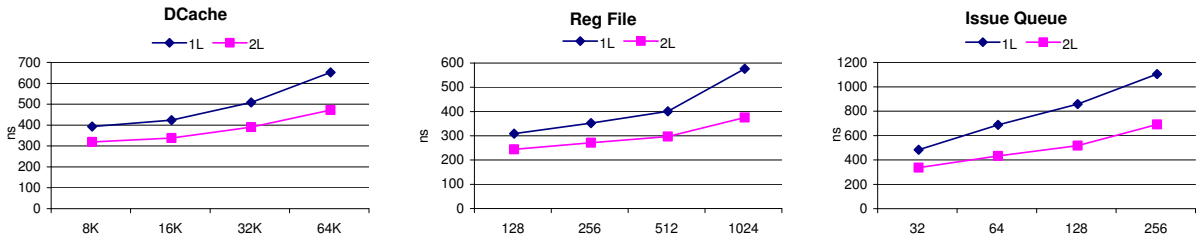

Figure 13: Latency Impact of Vertical Integration when Scaling the Size of Three Critical Blocks

Acknowledgments

This work is partially supported by the National Science Foundation under Grants CCR-0096383 and CCF0530261, and the GSRC Focus Center, one of five research centers funded under the Focus Center Research Program, a Semiconductor Research Corporation program. The authors would like to thank Dr. Yan Zhang for providing the program and assistance for thermal via insertion.

REFERENCES

[1] K. Banerjee, S. Souri, P. Kapur, and K. Saraswat. 3-D ICs: A novel chip design for improving deep-submicrometer interconnect performance and systems-on-chip integration. In *In Proc. of the IEEE, 89(5):602-633*, May 2001.

[2] B. Black, D. W. Nelson, C. Webb, and N. Samra. 3D processing technology and its impact on IA32 microprocessors. In *Proceedings of International Conference on Computer Design (ICCD), pp.316-318*, 2004.

[3] Eric Borch, Srilatha Manne, Joel Emer, and Eric Tune. Loose loops sink chips. In *HPCA '02: Proceedings of the 8th International Symposium on High-Performance Computer Architecture*, pages 299–310, Washington, DC, USA, 2002. IEEE Computer Society.

[4] D. C. Burger and T. M. Austin. The simplescalar tool set, version 2.0. Technical Report CS-TR-97-1342, U. of Wisconsin, Madison, June 1997.

[5] Y. Cao, T. Sato, D. Sylvester, M. Orshansky, and C. Hu. New paradigm of predictive MOSFET and interconnect modeling for early circuit design. In *Proc. of Custom Integrated Circuit Conference*, pages 201–204, Orlando, Florida, May 2000.

[6] J. Cong, A. Jagannathan, Y. Ma, G. Reinman, J. Wei, and Y. Zhang. An automated design flow for 3D microarchitecture evaluation. In *Proc. Asia and South Pacific Design Automation Conf*, pages 384–389, Yokohama, Japan, January 2006.

[7] J. Cong and Y. Zang. Thermal-driven multilevel routing for 3-D ICs. In *Proceedings of the Asia Pacific Design Automation Conference*, pages 121–126, 2005.

[8] J. Cong and Y. Zhang. Thermal via planning for 3-d ics. In *ICCAD '05: Proceedings of the 2005 IEEE/ACM International conference on Computer-aided design*, pages 745–752, Washington, DC, USA, 2005. IEEE Computer Society.

[9] Jason Cong, Eren Kursun, Yongxiang Liu, Yuchun Ma, and Glenn Reinman. 3D architecture modeling and exploration. In *UCLA Computer Science Department Technical Report UCLA/CSD-060032*, January 2006.

[10] S. Das, A. Chandrakasan, and R. Reif. Design tools for 3-D integrated circuits. In *Proc. Asia and South Pacific Design Automation Conf., pp. 53-56*, January 2003.

[11] Shamik Das, Andy Fan, Kuan-Neng Chen, Chuan Seng Tan, Nisha Checka, and Rafael Reif. Technology, performance, and computer-aided design of three-dimensional integrated circuits. In *ISPD '04: Proceedings of the 2004 international symposium on Physical design*, pages 108–115, New York, NY, USA, 2004. ACM Press.

[12] K. Farkas, N. Jouppi, and P. Chow. Register File Design Considerations in Dynamically Scheduled Processors. In *Proceedings of the Second International Symposium on High Performance Computer Architecture*, pages 40–51, January 1996.

[13] Daniele Folegnani and Antonio Gonzalez. Energy-effective issue logic. In *Proceedings of the 28th Annual International Symposium on Computer Architecture (ISCA'01)*, pages 230–239. ACM Press, 2001.

[14] K.Banerjee, S.Souri, P.Kapur, and K.C. Saraswat. 3-D ICs: A novel chip design for improving deep-submicrometer interconnect performance and systems-on-chip integration. In *IEEE Special Issue Interconnections - Addressing The Next Challenge of IC Technology, Vol. 89, No. 5*, pages 602–633, 2001.

[15] R.E. Kessler, E.J. McLellan, and D.A. Webb. The Alpha 21264 microprosessor architecture. In *International Conference on Computer Design*, December 1998.

[16] K.Puttaswamy and G.H.Loh. Implementing caches in a 3D technology for high performance processors. In *Proceedings of the International Conference on Computer Design*, 2005.

[17] MIT Lincoln Laboratory. Mitll low-power FDSOI CMOS process: Design guide. March 2006.

[18] Yongxiang Liu, Yuchun Ma, Eren Kursun, Jason Cong, and Glenn Reinman. Fine grain 3D integration for microarchitecture design through cube packing exploration. In *Proceedings of the International Conference on Computer Design. (To appear)*, October 2007.

[19] Y. Ma, X. Hong, and C.K. Cheng S. Dong. 3D CBL: An efficient algorithm for general 3-dimensional packing problems. In *In IEEE International Midwest Symposium on Circuits and Systems*, August 2005.

[20] M.A.Crowder, P.G.Carey, P.M.Smith, R.S.Sposili, H.S.Cho, and J.S.Im. Low temperature single crystal si tfts fabricated on si–filmw processed via sequential lateral solidification. In *IEEE Electron Device Letters, Vol. 19, No. 8*, pages 306–308, 1986.

[21] G. McFarland and M. Flynn. Limits of scaling mosfets. CSL TR-95-62, Stanford University, November 1995.

[22] M.Rodder and S.Aur. Utilization of plasma hydrogenization in stacked SRAMs with pli-Si PMOSFETs and bulk Si NMOS FETs. In *IEEE Electron Device Letters, Vol. 12*, pages 233–235, 1999.

[23] M.W.Geis, D.C.Flanders, D.A. Antoniadis, and H.I.Smith. Crystalline silicon on insulators by graphoepitaxy. In *IEDM Technical Digest*, pages 210–212, 1979.

[24] S. Pae, T.-C. Su, J.P. Denton, and G/W. Neudeck. Multiple layers of silicon-on-insulator islands fabrication by selective epitaxial growth. In *IEEE Electron Device Letters, Vol. 20, No. 5*, 1999.

[25] S. Palacharla, N. P. Jouppi, and J. E. Smith. Complexity-effective superscalar processors. In *Proceedings of the 24th Annual International Symposium on Computer Architecture*, pages 206–218, June 1997.

[26] Kiran Puttaswamy and Gabriel H. Loh. Implementing register files for high-performance microprocessors in a die-stacked (3D) technology. In *ISVLSI '06: Proceedings of the IEEE Computer Society Annual Symposium on Emerging VLSI Technologies and Architectures*, page 384, Washington, DC, USA, 2006. IEEE Computer Society.

[27] P.Wilkerson, A.Raman, and M.Turowski. Fast, automated thermal simulation for three-dimensional integrated circuits. In *Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Circuits, Itherm*, pages 706–713, 2004.

[28] G. Reinman and N. P. Jouppi. Cacti 2.0 beta. In *http ://www.research.digital.com/wrl//people/jouppi/CACTI.html, 1999.*

[29] P. Shivakumar and Norman P. Jouppi. Cacti 3.0: An integrated cache timing, power, and area model. In *Technical Report*, 2001.

[30] S.Kawamura, N.Sasaki, T.Iwai, M.Nakano, and M.Takagi. Three-dimensional cmps ics fabricated using beam recrystallization. In *IEEE Electron Devices Vol. Ed. 4*, pages 366–369, 1983.

[31] E. Sprangle and D. Carmean. Increasing processor performance by implementing deeper pipelines. In *ISCA '02: Proceedings of the 29th annual international symposium on Computer architecture*, pages 25–34, Washington, DC, USA, 2002.

[32] T.Kunio, K.Oyama, Y.Hayashi, and M.Morimoto. Three dimensional ics, having four stacked active decide layers. In *IEDM Technical Digest*, pages 837–840, 1989.

[33] Y. Tsai, Y. Xie, N. Vijaykrishnan, and M. Irwin. Three-dimensional cache design exploration using 3DCacti. In *Proceedings of the International Conference on Computer Design*, pages 519–524, October 2005.

[34] S. Wilton and N. Jouppi. Cacti: An enhanced cache access and cycle time model. In *IEEE Journal of Solid-State Circuits*, pages 677–687, May 1996.

[35] Y.Akasaka and T. Nishimura. Concept and basic technologies for 3D ic structure. In *IEDM Technical Digest*, pages 488–491, 1986.