

# Thermal Herding: Microarchitecture Techniques for Controlling Hotspots in High-Performance 3D-Integrated Processors

Kiran Puttaswamy<sup>†</sup>      Gabriel H. Loh<sup>‡</sup>  
Georgia Institute of Technology

<sup>†</sup>School of Electrical and Computer Engineering

<sup>‡</sup>College of Computing

kiranp@ece.gatech.edu      loh@cc.gatech.edu

## Abstract

*3D integration technology greatly increases transistor density while providing faster on-chip communication. 3D implementations of processors can simultaneously provide both latency and power benefits due to reductions in critical wires. However, 3D stacking of active devices can potentially exacerbate existing thermal problems. In this work, we propose a family of Thermal Herding techniques that (1) reduces 3D power density and (2) locates a majority of the power on the top die closest to the heat sink. Our 3D/thermal-aware microarchitecture contributions include a significance-partitioned datapath that places the frequently switching 16-bits on the top die, a 3D-aware instruction scheduler allocation scheme, an address memoization approach for the load and store queues, a partial value encoding for the L1 data cache, and a branch target buffer that exploits a form of frequent partial value locality in target addresses. Compared to a conventional planar processor, our 3D processor achieves a 47.9% frequency increase which results in a 47.0% performance improvement (min 7%, max 77% on individual benchmarks), while simultaneously reducing total power by 20% (min 15%, max 30%). Without our Thermal Herding techniques, the worst-case 3D temperature increases by 17 degrees. With our Thermal Herding techniques, the temperature increase is only 12 degrees (29% reduction in the 3D worst-case temperature increase).*

## 1. Introduction

The semiconductor industry faces an increasing number of challenges that must be overcome to keep pace with Moore's Law and industry projections. Some of these problems include poor scaling of wire delays [1, 18, 37], increasing power consumption [7, 43], and limits in manufacturing techniques. 3D die-stacked integration has the potential to address many of these problems for future high-performance microprocessors. Processor companies

are actively researching the technology [35, 6, 14], and the embedded domain is already producing products in 3D [15, 53].

While 3D integration provides increased device density, reduced latency and lower power [35, 47, 30, 51, 31, 33, 49], stacking multiple die increases power density. The increased power density exacerbates existing hotspots and can create new hotspots [49, 34]. In this work, we propose several microarchitecture-level techniques to address the challenge of 3D thermals. While we present a variety of methods, they can all be categorized under the general theme of *Thermal Herding*. Thermal Herding techniques herd or steer the majority of the processor's switching activity to the die that is closest to the heat sink. Overall, we show that it is possible to keep 3D thermals under control through a combination of reducing total processor power, local power density, and effective thermal resistance while simultaneously increasing performance by a significant amount.

The rest of this paper is organized as follows. Section 2 provides an overview of 3D integration technology. Section 3 explains our Thermal Herding techniques for designing a 3D-thermal-aware microarchitecture. Section 4 describes our evaluation framework and technology assumptions for circuit latency, IPC performance, power, and thermal estimations. Section 5 presents performance, power, and thermal results. Section 6 concludes the paper.

## 2. Overview of 3D Integration

This section describes the die-stacked 3D integration technology, outlines the critical parameters that are of interest to a microarchitect designing for a 3D process, and discusses a possible roadmap for the evolution of 3D processors.

### 2.1. Technologies and Topologies

There are currently several proposed methods for vertically integrating multiple die such as multi-layer buried

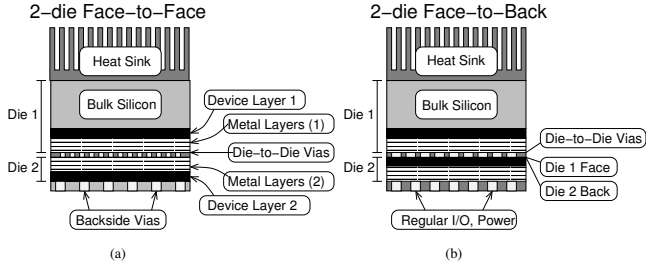


Figure 1. 2-die 3D stack with (a) face-to-face and (b) face-to-back bonding topologies. Note that the figures are not drawn to scale.

structures (MLBS) [50, 19] and die-bonding [26, 36]. The die-bonding approach takes two die and deposits via stubs on the top metal layer of the die and/or etches vias through the backside of the die, aligns the two die, and bonds them together. Under thermo-compression, the via stubs fuse together providing both the die-to-die (d2d) interconnects as well as the physical mechanism to hold the die together. After bonding, one die is thinned with chemical-mechanical polishing (CMP) down to only  $\sim 10\mu\text{m}$  allowing low impedance backside vias to be etched through, which provide input/output and power/ground connections.

There are many organizations for stacking multiple die. Figure 1 shows two-die face-to-face (a) and face-to-back (b) bonding topologies. The bonding process may be repeated to stack more than two die in any combination of face-to-face, face-to-back and back-to-back organizations.

The signal propagation delay between adjacent die is largely determined by the physical characteristics of the d2d vias. Since the individual die are thinned to only  $\sim 10\mu\text{m}$ , the d2d via between adjacent die may be  $< 5\mu\text{m}$  to  $\sim 20\mu\text{m}$  [12]. Prior research has reported the d2d via delay to be less than one FO4 [30].

## 2.2. Our Place on the 3D Roadmap

In this paper, we propose a 3D microarchitecture with functional block partitioning that requires a redesign of functional blocks at the circuit level. Figure 2 shows one possible path from current planar designs (a) to future 3D microarchitectures such as that proposed in this work. The first incarnations of 3D high-performance processors may primarily leverage the technology for device density. A likely design (b) would integrate multiple cores with a large 3D-stacked L2 or L3 cache [6]. This can provide performance benefit from reducing the wire delay between the cores and the L2 cache and increasing the overall size of the cache, but it does not fully exploit the benefits of 3D technology (i.e., the clock speed, area, and power of individual cores are not different from a planar/2D implementation). As the technology matures, more sophisticated 3D imple-

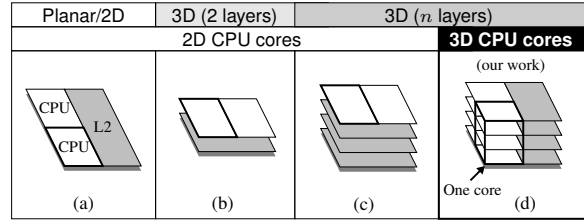


Figure 2. A progression from 2D to 3D processors with varying numbers of layers and CPU core implementation styles.

mentations will evolve by first adding more layers of cache (c), and then implementing circuit blocks on multiple die leading to the implementation of a 3D processor as shown in Figure 2(d). We categorize microarchitectures such as the 3D CMP [22] and the IntroSpective 3D processor [27] as belonging to Figure 2(b) and (c) since each core is still implemented on a single die. The scope of our research is 3D CPU cores. In this paper, we focus on the 3D implementation of a 64-bit processor using a 4-die stack.

## 3. Thermal Herding for 3D Microarchitectures

For a high-performance microarchitecture, the primary design goal is the maximization of performance within given power and thermal envelopes. With 3D technology, we can convert wire-length reduction into both performance improvements and power savings. Although 3D reduces wire latency and power, stacking active devices may increase power *density* leading to thermal problems. This section proposes several techniques based on *Thermal Herding* to reduce total power and power density while still maintaining the performance benefits of 3D.

Past research has observed that many integer instructions use data that require only a few of their least significant bits [8, 48]. In particular, many 64-bit integer values require only 16 or fewer bits to represent. Furthermore, past research has also observed that an instruction's usage of low-width values is highly predictable [24, 13]. We organize our datapath by assigning each word (16 bits) of the datapath to a separate die. We place the least significant bits on the top die that is closest to the heat sink, and then make use of the instruction's width prediction to save power on the other three die that do not require access.

For each instruction, we make a prediction whether to use low-width ( $\leq 16$ -bit) or full-width ( $> 16$  bits) values. We use a simple program counter (PC)-indexed two-bit saturating counter predictor [13]. When the predictor predicts an instruction to be low-width but the data is actually full-width, the result is an *unsafe* misprediction. An unsafe misprediction requires pipeline stalls in relevant pipeline

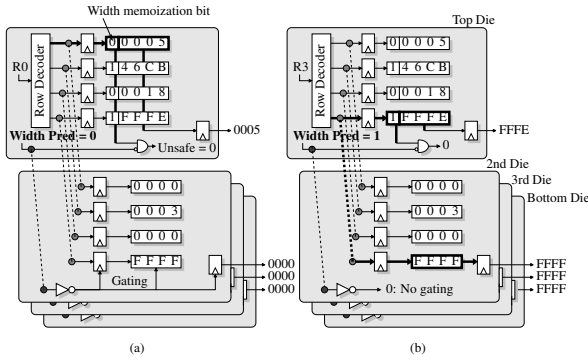


Figure 3. Thermal Herding in register files: examples where (a) a low-width value only requires circuit activity on the top die, and (b) a full-width value requiring reading state from all four die.

stages. The complementary case of a *conservative* or safe misprediction does not cause pipeline stalls, although it is a missed opportunity to reduce processor switching activity.

In the next few sections, we discuss some critical microarchitecture components of our Thermal Herding 3D microarchitecture.

### 3.1. Register Files

We partition each 64-bit entry in the register file such that each word (16-bits) reside on a separate die. This word-partitioned 3D register file organization reduces the access latency and the dynamic power consumption [32]. We use width *prediction* to enable early determination of gating control signals in advance of the actual register file access. On a predicted low-width instruction (Figure 3(a)) shows such an access for R0), only the top die portion of the register file is active. When we access only the top die, the power density characteristics are similar to that of a planar register file, and the activity is isolated to the top die (adjacent to the heat sink). In the case of a predicted full-width access (R3 in Figure 3(b)), all four die are active.

The top die (LSB's) contains a *width memoization* bit for each entry that indicates whether the remaining three die contain non-zero values. On reading the width memoization bit, the processor compares it to the predicted width. If the width prediction is low and the actual width is full, then the processor performs two actions: (1) it gates (stalls) the previous stages of the register file and enables the logic on the remaining three die, and (2) it corrects the instruction's width prediction to prevent any further stalls in the rest of the pipeline.

In a superscalar processor, the register file must provide operands for many instructions in parallel. To maintain an in-order dispatch process, any register file stall due to an unsafe misprediction prevents all later (in program order) instructions from dispatching to the out-of-order backend

of the core. All instructions that suffer from unsafe mispredictions in a *group* can be serviced in parallel in the next cycle, and therefore any group of instructions (those accessing the RF in the same cycle) can induce at most one stall for the entire group regardless of whether one or all of them had unsafe mispredictions.

Note that the register file access latency not only impacts the number of cycles in the conventional “branch mispredict detection” pipeline, but it also affects post-commit latencies such as the time required to copy committed values from the physical register file to the architected register file.

### 3.2. Arithmetic Units

We explain the 3D integer adder; however, the concepts presented here can be extended to the design of other arithmetic units. Figure 4 shows a 4-die implementation of a standard tree-based adder. The portion of the adder that adds the least significant 16-bits resides on the top die closest to the heatsink. In the cycle prior to execution, a conventional processor can make use of issue information to decide whether to clock gate the higher-order bits of the adder to save dynamic power (① in Figure 4). Even though the register file provides memoization bits that indicate if an instruction's operands are low-width, a full-width prediction initiates access to the entire adder because two low-width operands may generate a full-width result (e.g., adding two 16-bit values may result in a 17-bit sum). Figure 4-② shows the additional input to the clock-gating logic to gate the bottom three die of the adder.

There are two possible unsafe width misprediction scenarios. The first is a misprediction on an instruction's input operands. If the predictor predicts that an instruction is low-width but its operands are actually full width, then the instruction's arithmetic unit is not fully enabled at the start of execution. This results in a one cycle stall to re-enable the upper 48 bits of the arithmetic unit. The second type of misprediction is on the output of the arithmetic unit. In the case of output width misprediction, the width misprediction may not be known until several cycles into the computation (for pipelined functional units) and so we force any instruction with an unsafe output width misprediction to re-execute. While these mispredictions can induce a performance penalty, the accuracy of the width predictor prevents this from being a serious problem.

When the adder deals with low-width values, our approach not only reduces total power but also maintains comparable power density as the original planar adder implementation. Arithmetic units that are wire-intensive, e.g., shifters and multipliers, will benefit even more from the wire reduction. Hence, 3D functional units with Thermal Herding are simultaneously faster and lower power while having a similar power density profile as the planar functional units when handling low-width values.

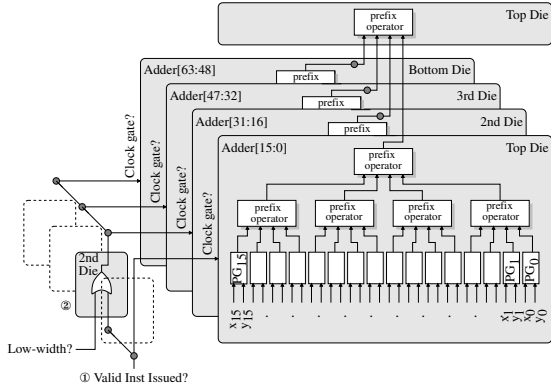


Figure 4. Thermal Herding in an integer adder with the most active least significant bits placed on the top die

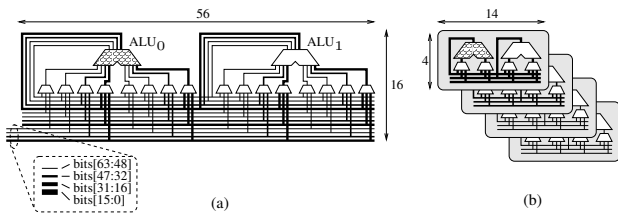


Figure 5. (a) A planar/2D bypass network with a register file path and two ALU outputs, and (b) the equivalent 3D bypass network.

### 3.3. Bypass Network

We organize the bypass network using significance-partitioning with 16-bits per die. Since the unsafe mispredictions have already been handled by the arithmetic units, the bypass network does not need additional circuitry to handle the width predictions. A correctly predicted low-width output will only cause the drivers/wires on the top die to dissipate dynamic power. A full-width output will cause activity on all die. In addition to the power reduction due to Thermal Herding, the wire-intensive nature of the bypass network allows for a substantial reduction in wire-related area, latency and power. Figure 5(a) shows a bypass network between two ALUs. Figure 5(b) shows the same ALUs and bypass implemented in a 4-die organization. Note that the dimensions of *both* the width and height of the bypass network have been reduced to a quarter of their original sizes.

### 3.4. Instruction Scheduler

The instruction scheduler contains one reservation station (RS) entry for each instruction dispatched but not yet executed. Each RS entry contains identifiers or tags for an instruction’s outstanding register input dependencies. When an instruction is ready to issue (all operand dependencies satisfied), the instruction bids for an execution unit,

and, if successful, broadcasts its destination identifier (tag) to notify dependent instructions.

We partition the instruction scheduler based on the RS entries, with one quarter of the entries placed on each die [31]. Although there is a slight overhead to fan-out the tag broadcasts to all four die, this organization greatly reduces the length of the broadcast buses which results in net power and latency reductions. We combine this entry-stacked scheduler organization with a modified allocation algorithm that herds instructions toward the top die to keep the active entries close to the heat sink. If there are no available entries in the top die, then the allocator starts allocating on the die that is next closest to the heat sink. To further reduce power consumption, the RS entries can make use of the allocator’s information regarding the occupancy of each die. If there are no occupied RS entries on a given die, then the tag broadcast for that die can be gated, leading to further power reductions.

### 3.5. Load and Store Queues

The load and store queues track both the data and addresses of memory instructions. The data loaded from and stored to memory exhibit value-width properties similar to that of register values. As a result, we propose to word-partition these queues similar to the main datapath. This provides the additional advantage that values (addresses or data) propagating between these structures already have their bits located on the correct die, avoiding the need for extra d2d vias.

Load and store addresses are almost always full-width values. However, we make the observation that the upper bits of the addresses do not frequently change. For example, loads and stores to and from the stack are likely to have identical upper address bits. To exploit this phenomenon, we use *partial address memoization* (PAM). On the top die, we always broadcast the low-order 16-bits of a load or store’s address. In addition, we broadcast an extra bit that indicates whether the remaining 48 bits are identical to those of the most recent store address. So long as there is sufficient locality in the types of memory references (e.g., stack vs. heap), our PAM approach will herd most of the address broadcasts and comparisons to the top die. Our address memoization is inspired from instruction scheduler tag memoization [41], although we use this in a different context to target 3D power density.

### 3.6. Data Cache

The data cache values have similar low-width characteristics as register values. As a result, we organize the data arrays of the L1 data cache in a word-partitioned manner. On the prediction of a low-width load, we access only the top die. The organization is analogous to the register file in that a small amount of extra state (memoization bits) on the top die provides fast detection of unsafe width mispredictions.

On an unsafe misprediction, we stall the cache pipeline. Since the tag match occurs in parallel with the misprediction detection, the processor knows the set-associative way of the cache hit and therefore only needs to access a single set-associative way when retrieving the remaining 48 bits. A store in the commit stage already knows its data-width, and therefore stores will not cause unsafe width mispredictions when writing to the cache.

To increase the frequency of low-width values, we broaden the definition of a “low-width” value for load and store instructions. Instead of storing a single width memoization bit, we store two bits that encode the upper 48 bits. When the encoding bits are 00, that means the upper 48 bits are all zeros; 01 means the bits are all ones (encodes negative numbers); 10 means the upper bits are identical to the upper bits of the referencing address, which occurs when heap data structures store pointers to other nearby objects [11]; 11 means the upper bits are not trivially encodable and should be read from the remaining three die. Previous work on *frequent value locality* [52] has observed that there are frequently occurring data values. When we can ignore the lowest 16 bits, the remaining *partial value* exhibits even stronger frequent value locality.

It is important to note that we only gate the bottom three die on a low-width predicted load or store. For a fill from or a spill to the L2 cache, we do not have a corresponding width prediction. Therefore, all spill/fill interactions between the L1 data cache and the L2 cache always access all four die of the respective caches.

### 3.7. Front End

Since the functional blocks of the front-end do not deal with data values, a data-centric approach to activity partitioning is not effective here. Figure 6(a) shows a four-die 3D organization of the processor front end. We implement the instruction cache (IS) and instruction translation lookaside buffer (ITLB) using previously proposed 3D stacking organizations [30, 47] which provide latency and power benefits, but there is no explicit Thermal Herding.

After instructions enter the decoding pipeline, they will not move between die until they dispatch into the RS entries. We implement the register alias table (RAT) by placing the ports corresponding to each instruction on different die [32]. A single instruction’s RAT read and write ports are all located on the same die as the decoded instruction itself, thereby avoiding the need for unnecessary d2d vias (the two R’s and one W in Figure 6(a) show these RAT ports for the top die).

Having instructions located on different die forces the intra-group rename dependency checking to use d2d vias. However, we can partition the logic across multiple die to place more of the activity closer to the top die. A given instruction in a rename group only needs to check whether

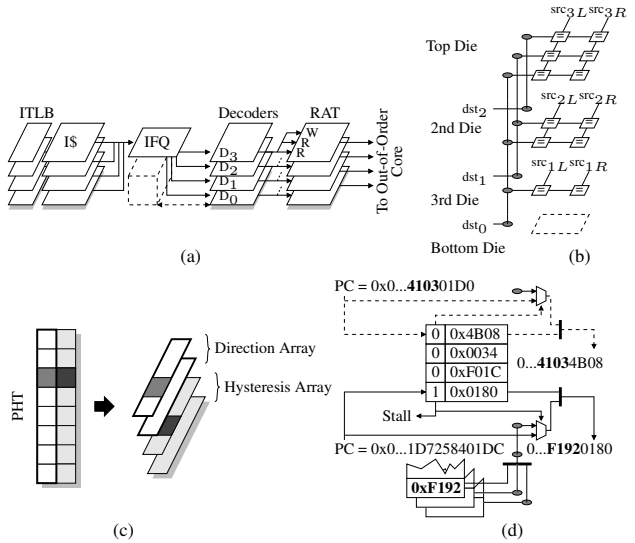


Figure 6. (a) 3D organization of the front-end pipeline components, (b) 3D register rename intra-group dependency checking logic, (c) branch predictor saturating counter array partitioned into two separate 3D sub-tables, and (d) Thermal Herding in the BTB.

one of its input operands matches the output of a previous instruction in the same group. This implies that the first instruction does not require any dependency checks, and the last instruction requires the most checks. We place the instruction that requires the most register name comparisons on the top die, thereby herding more of the switching activity to the top die as illustrated by the dependency checking logic in Figure 6(b).

The last major parts of the frontend are the control-flow predictors. In particular, we consider the branch direction predictor and the branch target buffers. For branch direction predictors based on two-bit saturating counters, we first partition the counters into two separate arrays: one array to store the direction bit (msb) and the other to store the hysteresis bit (lsb) [40]. We implement the two arrays by partitioning them across two die each, as shown in Figure 6(c). The processor needs the direction bit for making the initial prediction as well as during the update/training phase, while the hysteresis bit is only needed for the update. Therefore, we place the more frequently accessed direction-bit array on the top two die closer to the heatsink.

For the branch target buffers (BTBs), we make use of the observation that most branch targets are located relatively close to the originating branch itself. This is particularly true for PC-relative branch targets. Based on this observation, we organize the BTBs like our data cache, where we store the low-order 16 bits on the top die, and then store one

additional *target memoization* bit that indicates if the bits on the remaining three die should be accessed. The top of Figure 6(d) shows a target prediction example (dashed lines) that reuses the upper bits of the branch’s program counter (PC). The target memoization bit causes the selection of the upper PC bits. The bottom of Figure 6(d) shows an example (solid lines) of the infrequent case where the target memoization bit is set to one. In this case, we need to stall the prediction pipeline for one cycle to retrieve the upper bits from the remaining three die of the array. Similar to reading the upper 48 bits of the data cache, the BTB tag match in the first cycle enables the front-end to only access the single set-associative way that had a hit.

### 3.8. Microarchitecture Summary

The majority of our 3D functional units make use of a compact 3D organization to reduce wires which results in simultaneous latency and power reductions. When possible, we attempt to herd most of the switching activity to the top die of the 3D stack. We observed that 97% of all instructions fetched have their widths correctly predicted, thereby avoiding any severe IPC degradation due to our microarchitectural Thermal Herding.

There are other 3D optimizations that yield IPC improvements. In particular, a 3D implementation of the L2 cache results in a significantly faster access time which can reduce the number of clock cycles for an L2 access, even at a faster clock frequency. A variety of “global” signals may have shorter wire lengths in a 3D implementation because each of the individual units now has a reduced footprint, thereby compressing the overall processor floorplan. Some microarchitectures impose an extra cycle of latency to load values into floating point registers due to the extra distance required to route from the cache to the floating point units [6, 49]. The area reduction due to the 3D bypass network may sufficiently reduce wire distances to remove this extra cycle. Another example is the extra pipeline stage(s) that may be necessary to communicate the branch misprediction from the execution stage in the backend to the processor frontend [17].

## 4. Experimental Methodology

Our experimental analysis includes the quantification of performance in terms of both clock frequency improvements and IPC rates, power consumption, and the overall thermal impact. We used HSpice to simulate many of the processor components (circuit designs primarily in static CMOS) to determine their latencies and energy consumptions. The latency reports the worst-case path through the circuit, whereas the energy includes the energy consumption from all active circuits and wires. All circuits employ 65nm transistors using the Predictive Technology Models (formerly BPTM) [10]. We used 130nm wire parameters

Parameter	Value
Fetch/Decode/Commit	4 insts/cycle
Issue	Max. 6/cycle
<i>Int</i>	3 ALU, 2 shift, 1 mult/complex
<i>FP</i>	1 add, 1 mult, 1 div/sqrt
<i>Memory</i>	1 Ld/St port, 1 Ld-only port
ROB size	96 entries
RS size	32 entries
LQ/SQ size	32/20 entries
I/D L1 caches	32KB, 8-way, 3-cycle
Branch Predictor	10KB Bimodal/Local/Global hybrid
Unified L2 cache	4MB, 16-way, 12-cycle
Br Mispred Latency	Min. 14 cycles
I/D TLBs	128/256-entry, 4-way
BTB/iBTB	2K/512-entry, 4-way
Inst Fetch Queue	16 entry

Table 1. Microarchitecture parameters of the baseline processor.

from Intel and extrapolated down to 65nm [46]. Our 3D circuit designs use d2d via pitches of  $1\mu\text{m}$  for the face-to-face and  $2\mu\text{m}$  for backside interfaces. We model a distance of  $5\mu\text{m}$  to cross between the two die faces and  $20\mu\text{m}$  to cross the B2B interface, which is conservative since current 3D technologies already thin the die down to  $12\mu\text{m}$  [45].

For planar and 3D processor configurations, we used SimpleScalar/MASE for the Alpha ISA to quantify the impact on IPC [20, 4]. We model a processor loosely based on the Intel Core 2 microarchitecture. Table 1 lists the microarchitectural parameters. Figure 7 shows our floorplans for (a) the baseline 2D processor and (b) the top die of our 3D 4-die stacked processor. Note that our baseline parameters and floorplan are best-effort estimates of a Core 2-class microarchitecture, and should not be construed as an official representations of the Core 2 microarchitecture.

The majority of the 3D processor layout is identical to the planar processor except for an  $\sim 4\times$  footprint reduction due to the partitioned implementation of individual circuit blocks on four die. We reoriented and realigned the cores and the L2 cache to reduce empty regions (whitespace) in the floorplan of the 3D processor.

We assume a clock frequency of 2.66GHz for the baseline planar processor. We use a collection of 106 application traces including all benchmarks from SpecInt2000 and SpecFP2000 with the reference inputs, and a variety of programs from MediaBench [21], the Michigan embedded benchmarks [16], the Wisconsin pointer-intensive benchmarks [3], assorted graphics programs from the SimpleScalar website (includes games such as Doom and Quake, ray-tracing, and mpeg and avi video playback), and the BioBench [2] and BioPerf [5] bioinformatics benchmark suites. In all cases, we use SimPoint 2.0 to choose representative simulation points [29].

For our thermal analysis, we use HotSpot 3.0.2 from the

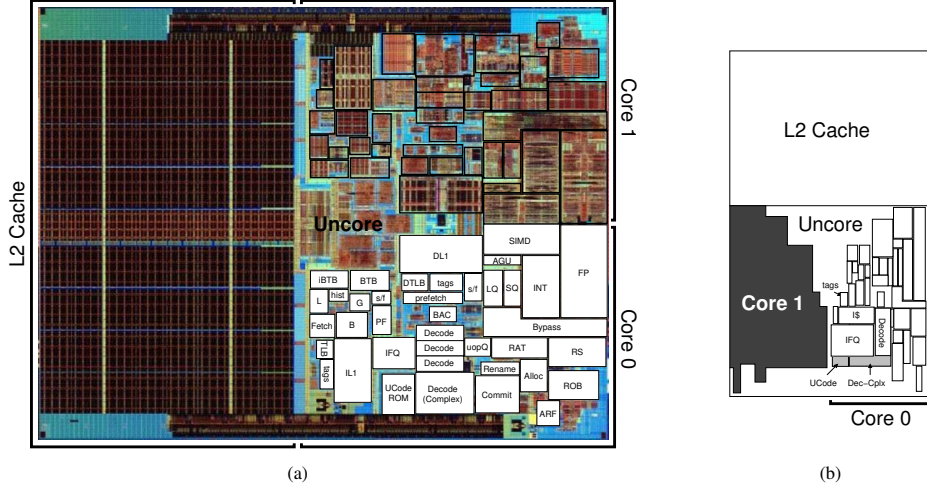


Figure 7. (a) Floorplan for the baseline 2D microarchitecture, and (b) the top die of our 3D floorplan. The gray blocks show the locations of modules that are not stacked on top of themselves; refer to Figure 6(a).

University of Virginia [42]. We perform a fine-grained simulation of the entire 3D stack. For the interface between die, we compute the average thermal resistance and specific heat capacity assuming the d2d vias are fully populated and the d2d via width is half of the via pitch, which results in a 25% copper occupancy (75% air). We assume a phase change metallic alloy [38] for the thermal interface material (TIM) between the last die and the heat spreader. For each module, we compute the power by combining our HSpice results, the activity factor of the module as reported by MASE, and the clock frequency. We assume that the clock network footprint is reduced by  $\frac{1}{4}$  since it is distributed across four die, but we conservatively reduce its power consumption by  $\frac{1}{2}$  for the 3D processor configurations. We assume that the baseline 2D processor dissipates 35% of its power in the clock network [9] and 20% in leakage, and that the 3D organization and Thermal Herding do not reduce the leakage power consumption.

## 5. Experimental Evaluation

In this section, we present our experimental results to quantify the impact of our 3D Thermal Herding microarchitecture on performance, power, and thermals.

### 5.1. Performance

3D microarchitecture influences both the processor’s clock frequency as well as its IPC rates. In the following sections, we discuss each of them in detail.

#### 5.1.1. Clock Frequency

As discussed in Section 3, the 3D implementation of many of the processor’s blocks reduces the wire-delay internal to those blocks. By implementing all of a processor’s critical

Block/Path	Latency			
	Name	2D (ps)	3D (ps)	(% Reduction)
<b>Wakeup+Select</b>		<b>347.8</b>	<b>235.3</b>	<b>32.4%</b>
<b>ALU+Bypass</b>		<b>641.8</b>	<b>410.9</b>	<b>36.0%</b>
ROB/PRF		787.0	378.4	51.9%
Arch. RF		524.8	248.8	52.6%
L1 Cache		1019.5	707.9	30.6%
L2 Cache		4140.7	2008.7	51.5%
ITLB		663.1	369.9	44.2%
DTLB		788.8	504.6	36.0%
RAT		594.0	378.4	36.3%
Load Queue		345.8	253.1	26.8%

Table 2. Critical path latencies for several microprocessor blocks. We consider the Wakeup-Select and the ALU-Bypass loops (in bold) to be the clock limiting paths.

paths in 3D, we can increase the overall clock frequency to gain more performance. Table 2 shows the latencies of some of the processor’s blocks for both 2D and 3D implementations.

Previous work has identified the instruction scheduling logic (wakeup-select loop) and the arithmetic unit and result bypass loops (highlighted in bold in Table 2) to be particularly important in determining a processor’s maximum clock frequency [28]. While we recognize that reducing a processor’s cycle time may require speeding up hundreds or thousands of critical timing paths, we believe that these two fundamental loops are representative of the achievable speedups of a 3D microarchitecture. We observe a 32% improvement in the latency of the wakeup-select loop. This benefit derives from the reduction in the wire length and wire loading of the wakeup logic’s tag broadcast bus as

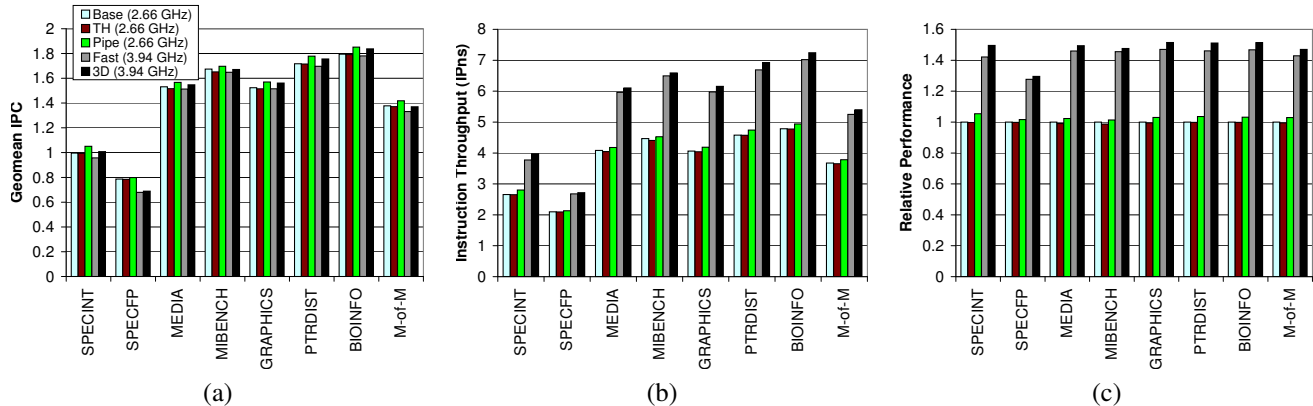


Figure 8. Performance impact of Thermal Herding and the 3D reorganization of the processor on (a) IPC, (b) instruction throughput in instructions per nanosecond, and (c) the overall relative performance speedup. M-of-M is the “mean of means” (geometric mean across all benchmark groups).

well as the wire delay in the select logic. Our 3D word-partitioned datapath results in a 36% latency improvement in the *ALU+Bypass* loop. The adder only accounts for 3% out of the 36% benefit because our 3D-partitioned adder only reduces the wire delay of the last levels of the carry logic. However, the compaction of the ALUs results in a substantial reduction in the distance traversed by the bypass network. Overall, wire latency reduction translates into a 47.9% increase in clock frequency, from 2.66 GHz to 3.93 GHz.

While we base our overall clock frequency increase on the two critical loops of scheduling and execution, Table 2 demonstrates that 3D provides substantial latency improvements across a wide variety of other blocks. In agreement with prior research on 3D cache designs [35, 47, 30, 25], we observe that large arrays (caches, register files, TLBs) observe substantial latency improvements. Most of these structures can be fairly easily pipelined and therefore we do not consider them as frequency limiters.

It is important to understand that the clock frequency increase provided by 3D microarchitecture is directly from reducing the wire delay component of the cycle time, whereas the conventional microarchitectural approach for increasing clock frequencies is to increase the number of pipeline stages. Adding more pipeline stages increases pipeline complexity (e.g., requiring more inter-stage bypassing) and it usually decreases IPC (e.g., needing aggressive speculative scheduling resulting in more replays). 3D’s frequency benefits do not require changes to the overall pipeline organization at the microarchitecture level. With careful design, 3D can actually *remove* pipeline stages, as described in Section 3.8, which provides a microarchitecturally simpler pipeline organization *and* higher clock speeds.

### 5.1.2. IPC and Overall Performance

Different features of our 3D Thermal Herding microarchitecture affect overall IPC in different ways. For example, our 3D processor can improve IPC by reducing the pipeline depth and reducing the L2 latency in clock cycles (this is on top of frequency increase). However, our Thermal Herding microarchitecture can reduce IPC rates due to the different width-misprediction related stalls and increases in the average number of *cycles* to access main memory due to the frequency increase. Figure 8(a) shows the geometric mean IPC rates for each of our benchmark classes, and the overall mean of the per-group means (M-of-M). *Base* is the baseline 2.66GHz processor, *TH* applies the Thermal Herding techniques and *Pipe* applies the pipeline optimizations described earlier in Section 3. For both *TH* and *Pipe*, we do not change the clock frequency to isolate the IPC impact directly attributable to these changes. The *Fast* configuration is microarchitecturally identical to the baseline processor, but the higher clock frequency (3.91 GHz) increases the average number of cycles to access main memory which in turn decreases IPC. Finally, the IPC rates for 3D are for our 3D processor that simultaneously accounts for the impacts of Thermal Herding, the pipeline optimizations, and increased clock frequency.

Overall, the pipeline reduction benefits slightly outweigh the effects of width mispredictions and the higher clock speed, and provides a small IPC benefit. However, overall performance is determined by both IPC rates and the clock frequency. Figure 8(b) shows the overall performance in instructions per nanosecond (IPNs) that accounts for both of these factors, and Figure 8(c) shows the relative performance speedup over the baseline. Because the pipeline optimizations cancel out the IPC degradation of increased clock speeds, performance tends to scale directly with the overall

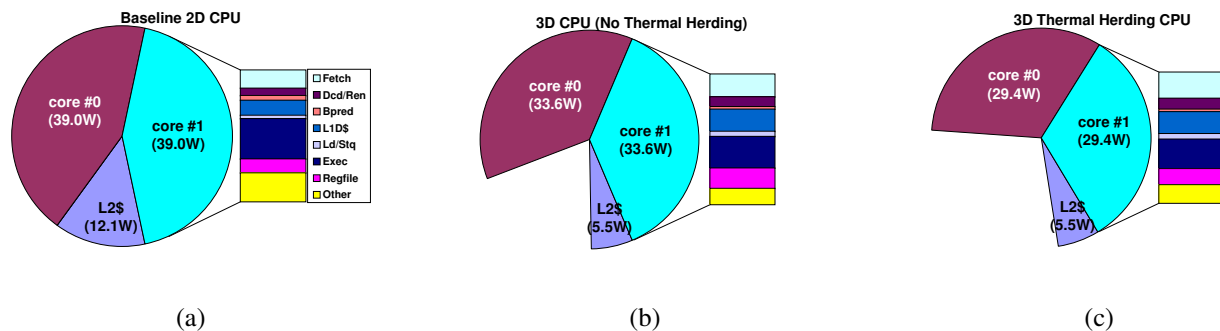


Figure 9. Power consumption distribution for (a) the baseline 2D processor, (b) a 3D processor without Thermal Herding, and (c) our 3D processor with Thermal Herding (running the Mpeg2 encoding benchmark from MediaBench).

frequency improvement of the 3D processor. Performance improvements ranged from 7% (mcf/SPECInt2000) to 65% (crafty/SPECInt2000) and 77% (patricia/MiBench). Except for SPECFP2000, all of the other benchmark groups exhibit 49.4% to 51.5% mean performance improvements. Our 3D processor only provides a 29.5% benefit for SPECFP2000 because these benchmarks have a large number of accesses to main memory, and our 3D processor has not reduced the latency (in seconds) of DRAM accesses. One could potentially use 3D integration to further stack DRAM on top of our already 3D-stacked processor to help reduce main memory latency [23].

## 5.2. Power

Our 3D processor provides overall power reduction in two ways: first, the 3D implementation of all of the processor blocks substantially reduces the amount of wires in these units; second, our Thermal Herding allows the processor to sometimes gate approximately 75% of a block’s switching activity. However, when we increase clock frequency to improve performance, we also increase power. For the baseline processor, the Mpeg2 encoding benchmark from the MediaBench suite [21] resulted in the highest power consumption. Figure 9(a) shows the power distribution for two identical instances of the Mpeg2 encoder application, each running on one of the two processor cores (90W total). Figure 9(b) shows the power impact for the 3D processor accounting for wire reduction and frequency increase, but not Thermal Herding. Despite the frequency increase, 3D microarchitecture aggressively reduces wires, resulting in a net reduction of total power to 72.7W (19% reduction over the planar baseline). Finally, Figure 9(c) shows the total power of our 3D Thermal Herding design, which consumes a total power of only 64.3W (29% reduction over the planar baseline). Note that the total power saving for our 3D processor over the planar baseline ranges from 15% (Yacr2 benchmark from the pointer applications [3]) to 30% (Susan im-

age processing benchmark applying a smoothing filter from MiBench [16]) depending on the characteristics of the application. Applications such as Yacr2 are memory-intensive and thus derive less benefit from 3D CPU cores. Image processing applications are computation-intensive and thus derive larger benefit from the Thermal Herding techniques in 3D CPU cores.

## 5.3. Thermals

Figure 10(a-c) shows the thermal maps for the applications that induced the worst-case temperatures among our 106 application traces. Different applications were responsible for these worst-case scenarios (Mpeg2 for planar and 3D with no Thermal Herding; Yacr2 for Thermal Herding 3D). Figure 10 includes annotations for the locations and magnitudes for some notable hotspots, with the worst-case hotspot marked by a box around the temperature. Figure 10(a) shows the baseline 2D processor, which has a peak temperature of 360K located at the instruction scheduling logic (RS entries). Figure 10(b) shows the 3D processor without Thermal Herding, which has a worst-case hotspot of 377K (17K increase). Figure 10(c) shows the 3D processor with Thermal Herding which has the hottest spot at 372K in the data cache area (only 12K increase from the planar baseline, a net reduction of 29% from the 3D processor without Thermal Herding). Since Yacr2 is a memory-intensive application, the high frequency of data cache accesses combined with additional cache accesses to handle width mispredictions causes the data cache to become the hottest spot.

To understand the interaction between 3D die-stacking and temperature, we performed a thermal analysis to explore the power density effect. We analyzed the 3D processor at the same total power (90 W) and same frequency (2.66 GHz) as the planar processor. Note that this configuration mimics a quadrupling of the power density due to 3D stacking while ignoring the latency and power benefits

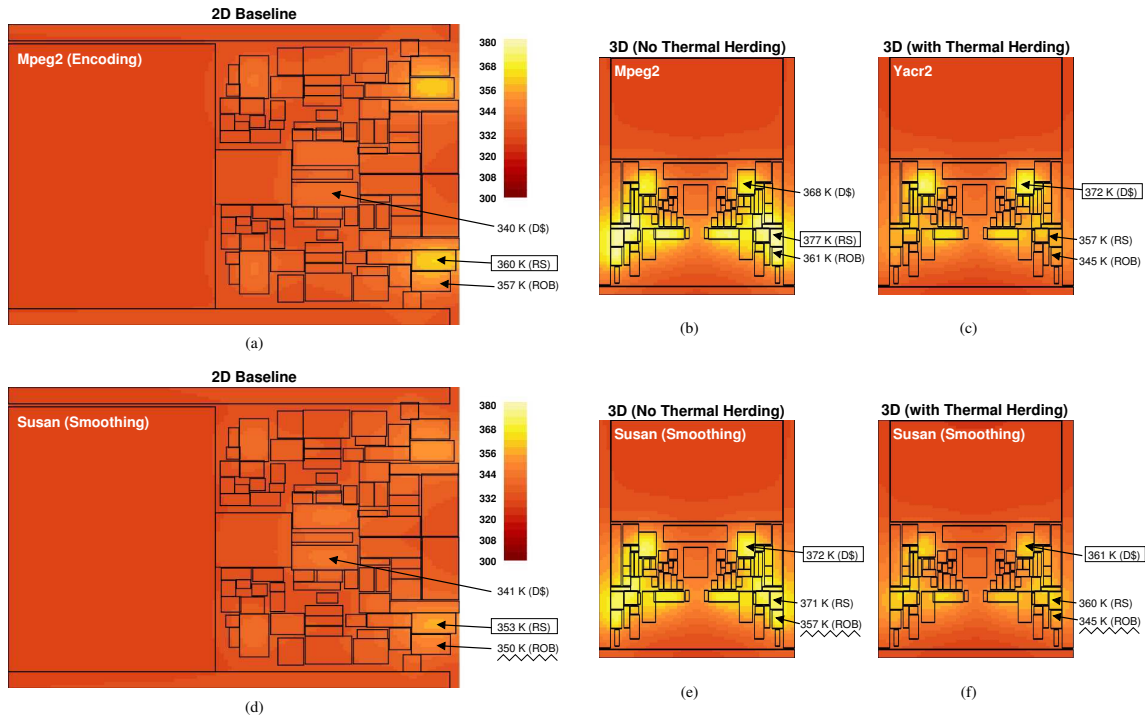


Figure 10. Worst-case thermal plots of (a) the 2D baseline processor, and 3D processors (b) without and (c) with Thermal Herding. (d-f) Thermal plots for the Susan benchmark from MiBench. Boxes indicate hottest blocks. Lighter is hotter, darker is cooler.

of a 3D organization. The worst case temperature increased to 418K (an increase of 58K over planar processor). The reason why 3D processors are nowhere nearly as hot as the  $4\times$  power density 3D stacking is that the decrease in the *total* power consumption provides relief to the increase in the power density.

The worst-case examples determine the cooling requirements for the overall system design. However, more insights can be gained by examining the thermal behavior of the processors across a single benchmark. Figure 10(d-f) shows the same three processor configurations, except that each one is now running the same application. For several blocks, Thermal Herding provides a substantial reduction in the local temperature of 3D configuration. This is largely due to a combination of overall power reduction as well as vertical herding effects. In fact, we observed some blocks in our 3D processor to have a lower local temperature than the planar processor. For example, the reorder buffer (ROB) which contains the physical registers exhibits a large number of low-width accesses (approximately  $5\times$  ( $2\times$ ) more low-width reads (writes) than full-width). The net result is a 5K temperature reduction for the ROB over the planar baseline. The data cache is more typical, where the Thermal Herding reduces the temperature to slightly above the mid-point between the 2D baseline and the 3D implementation

without Thermal Herding. Note that Black et al. [6] have demonstrated that we can further reduce the temperature by converting a small part of our performance gains into power reduction.

## 6. Conclusions and Future Directions

3D integration has the ability to increase transistor density and to perhaps delay the end of Moore’s Law for a few more generations. Through detailed circuit analysis, we have demonstrated that a conventional superscalar processor implemented in 3D technology can provide improvements in circuit latency and power, resulting in substantial performance benefits. We have demonstrated that microarchitecture-level techniques can control power density and mitigate 3D thermal issues. 3D technology provides more performance for less power, thereby providing excellent performance-per-Watt ratios.

This paper demonstrates that 3D technology applied to the design of high-performance microprocessors can provide significant value. However, at the heart of this study lies a conventional ROB/RS-based microarchitecture originally designed for a planar fabrication technology. We believe that a new microarchitecture designed from the ground up with 3D technology in mind will be much more effective at exploiting the strengths of 3D integration.

Other non-traditional architectures such as TRIPS [39] or WaveScalar [44] may also be very interesting candidates for 3D implementations. There are many possibilities for 3D processor design and there is a great need for more 3D microarchitecture research.

## Acknowledgments

Funding and equipment for this project have been provided by Intel Corporation and a grant from the Microelectronics Advanced Research Corporation (MARCO).

## References

- [1] V. Agarwal, M. Hrishikesh, S. Keckler, and D. Burger. Clock Rate Versus IPC: The End of the Road for Conventional Microarchitectures. In *Proc. of the 27th Intl. Symp. on Comp. Arch.*, pages 248–259, Vancouver, Canada, June 2000.
- [2] K. Albayraktaroglu, A. Jaleel, X. Wu, M. Franklin, B. Jacob, C.-W. Tseng, and D. Yeung. BioBench: A Benchmark Suite of Bioinformatics Applications. In *Proc. of the Intl. Symp. on Perf. Analysis of Systems and Software*, pages 2–9, Austin, TX, USA, March 2005.
- [3] T. Austin, S. E. Breach, and G. S. Sohi. Efficient Detection of All Pointer and Array Access Errors. In *Proc. of the ACM SIGPLAN Conf. on Programming Language Design and Implementation*, pages 290–301, Orlando, FL, USA, June 1994.
- [4] T. Austin, E. Larson, and D. Ernst. SimpleScalar: An Infrastructure for Computer System Modeling. *IEEE Micro Magazine*, pages 59–67, February 2002.
- [5] D. A. Bader, Y. Li, T. Li, and V. Sachdeva. BioPerf: A Benchmark Suite to Evaluate High-Performance Computer Architecture of Bioinformatics Applications. In *Proc. of the IEEE Intl. Symp. on Workload Characterization*, pages 163–173, Austin, TX, USA, October 2005.
- [6] B. Black, M. Annaram, N. Brekelbaum, J. DeVale, L. Jiang, G. H. Loh, D. McCauley, P. Morrow, D. W. Nelson, D. Pantuso, P. Reed, J. Rupley, S. Shankar, J. Shen, and C. Webb. Die Stacking (3D) Microarchitecture. In *Proc. of the Intl. Symp. on Microarchitecture*, Orlando, FL, December 2006.
- [7] D. Brooks, P. Bose, S. E. Schuster, H. Jacobson, P. N. Kudva, A. Buyuktosunoglu, J.-D. Wellman, V. Zyuban, M. Gupta, and P. W. Cook. Power-Aware Microarchitecture: Design and Modeling Challenges for Next-Generation Microprocessors. *IEEE Micro Magazine*, 20(6):26–44, November 2000.
- [8] D. Brooks and M. Martonosi. Dynamically Exploiting Narrow Width Operands to Improve Processor Power and Performance. In *Proc. of the 5th Intl. Symp. on High Perf. Comp. Arch.*, pages 13–22, Orlando, FL, USA, January 1999.
- [9] D. Brooks, V. Tiwari, and M. Martonosi. Watch: A Framework for Architectural-Level Power Analysis and Optimizations. In *Proc. of the 27th Intl. Symp. on Comp. Arch.*, pages 83–94, Vancouver, Canada, June 2000.
- [10] 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 the 2000 Custom Integrated Circuits Conf.*, pages 201–204, Orlando, FL, USA, May 2000.
- [11] J. Collins, S. Sair, B. Calder, and D. M. Tullsen. Pointer Cache Assisted Prefetching. In *Proc. of the 35th Intl. Symp. on Microarchitecture*, pages 62–73, Istanbul, Turkey, November 2002.
- [12] S. Das, A. Fan, K.-N. Chen, and C. S. Tan. Technology, Performance, and Computer-Aided Design of Three-Dimensional Integrated Circuits. In *Proc. of the Intl. Symp. on Physical Design*, pages 108–115, Phoenix, AZ, USA, April 2004.
- [13] O. Ergin, D. Balkan, K. Ghose, and D. Ponomarev. Register Packing: Exploiting Narrow-Width Operands for Reducing Register File Pressure. In *Proc. of the 37th Intl. Symp. on Microarchitecture*, pages 304–315, Portland, OR, USA, December 2004.
- [14] K. Guarini, A. Topol, M. Leong, R. Yu, L. Shi, M. R. Newport, D. J. Frank, D. V. Singh, G. M. Cohen, S. V. Nitta, D. C. Boyd, P. A. O’Neill, S. L. Tempest, H. B. Pogge, S. Purushothaman, and W. E. Haensch. Electrical Integrity of State-of-the-Art 0.13 $\mu$ m SOI CMOS Devices and Circuits Transferred for Three-Dimensional (3D) Integrated Circuit (IC) Fabrication. In *Proc. of the Intl. Electron Devices Meeting*, pages 943–945, December 2002.
- [15] S. Gupta, M. Hilbert, S. Hong, and R. Patti. Techniques for Producing 3D ICs with High-Density Interconnect. In *Proc. of the 21st Intl. VLSI Multilevel Interconnection Conf.*, Waikoloa Beach, HI, USA, September 2004.
- [16] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown. MiBench: A Free, Commercially Representative Embedded Benchmark Suite. In *Proc. of the 4th Work. on Workload Characterization*, pages 83–94, Austin, TX, USA, December 2001.
- [17] G. Hinton, D. Sager, M. Upton, D. Boggs, D. Carmean, A. Kyker, and P. Roussel. The Microarchitecture of the Pentium 4 Processor. *Intel Technology Journal*, Q1 2001.
- [18] R. Ho, K. W. Mai, and M. A. Horowitz. The Future of Wires. *Proceedings of the IEEE*, 89(4):490–504, April 2001.
- [19] S. M. Jung, J. Jang, W. Cho, J. Moon, K. Kwak, B. Choi, B. Hwang, H. Lim, J. Jeong, J. Kim, and K. Kim. The revolutionary and truly 3-dimensional 25f2 sram technology with the smallest s3 cell, 0.16 $\mu$ m<sup>2</sup> and sstff for ultra high density sram. *VLSI Techn. Dig. Techn. Papers*, pages 228–229, 2004.
- [20] E. Larson, S. Chatterjee, and T. Austin. MASE: A Novel Infrastructure for Detailed Microarchitectural Modeling. In *Proc. of the 2001 Intl. Symp. on Perf. Analysis of Systems and Software*, pages 1–9, Tucson, AZ, USA, November 2001.
- [21] C. Lee, M. Potkonjak, and W. Mangione-Smith. MediaBench: A Tool for Evaluating and Synthesizing Multimedia and Communication Systems. In *Proc. of the 30th Intl. Symp. on Microarchitecture*, pages 330–335, Research Triangle Park, NC, USA, December 1997.
- [22] F. Li, C. Nicopoulos, T. Richardson, Y. Xie, V. Narayanan, and M. Kandemir. Design and Management of 3D Chip Multiprocessors Using Network-in-Memory. In *Proc. of the 33rd Intl. Symp. on Comp. Arch.*, pages 130–141, Boston, MA, USA, June 2006.

- [23] C. C. Liu, I. Ganusov, M. Burtscher, and S. Tiwari. Bridging the Processor-Memory Performance Gap with 3D IC Technology. *IEEE Design and Test of Comp.*, 22(6):556–564, November–December 2005.
- [24] G. H. Loh. Exploiting Data-Width Locality to Increase Superscalar Execution Bandwidth. In *Proc. of the 35th Intl. Symp. on Microarchitecture*, pages 395–405, Istanbul, Turkey, November 2002.
- [25] J. Mayega, O. Erdogan, P. M. Belemjian, K. Zhou, J. F. McDonald, and R. P. Kraft. 3D Direct Vertical Interconnect Microprocessors Test Vehicle. In *Proc. of the ACM Great Lakes Symp. on VLSI*, pages 141–146, Washington, DC, USA, April 2003.
- [26] P. Morrow, M. Kobrinsky, et al. Wafer-Level 3D Interconnects Via Cu Bonding. In *Proc. of the 21st Advanced Metallization Conf.*, San Diego, CA, USA, October 2004.
- [27] S. Mysore, B. Agarwal, N. Srivastava, S.-C. Lin, K. Banerjee, and T. Sherwood. Introspective 3D Chips. In *Proc. of the 12th Symp. on Arch. Support for Prog. Lang. and Operating Systems*, San Jose, CA, USA, October 2006.
- [28] S. Palacharla. *Complexity-Effective Superscalar Processors*. PhD thesis, University of Wisconsin, 1998.
- [29] E. Perelman, G. Hamerly, and B. Calder. Picking Statistically Valid and Early Simulation Points. In *Proc. of the 2003 Intl. Conf. on Par. Arch. and Compilation Techniques*, pages 244–255, New Orleans, LA, USA, September 2004.
- [30] K. Puttaswamy and G. H. Loh. Implementing Caches in a 3D Technology for High Performance Processors. In *Proc. of the Intl. Conf. on Comp. Design*, San Jose, CA, USA, October 2005.
- [31] K. Puttaswamy and G. H. Loh. Dynamic Instruction Schedulers in a 3-Dimensional Integration Technology. In *Proc. of the ACM Great Lakes Symp. on VLSI*, pages 153–158, Philadelphia, PA, USA, May 2006.
- [32] K. Puttaswamy and G. H. Loh. Implementing Register Files for High-Performance Microprocessors in a Die-Stacked (3D) Technology. In *Proc. of the Intl. Symp. on VLSI*, pages 384–389, Karlsruhe, Germany, March 2006.
- [33] K. Puttaswamy and G. H. Loh. The Impact of 3-Dimensional Integration on the Design of Arithmetic Units. In *Proc. of the Intl. Symp. on Circuits and Systems*, Kos, Greece, May 2006.
- [34] K. Puttaswamy and G. H. Loh. Thermal Analysis of a 3D Die-Stacked High-Performance Microprocessor. In *Proc. of the ACM Great Lakes Symp. on VLSI*, pages 19–24, Philadelphia, PA, USA, May 2006.
- [35] P. Reed, G. Yeung, and B. Black. Design Aspects of a Microprocessor Data Cache using 3D Die Interconnect Technology. In *Proc. of the Intl. Conf. on Integrated Circuit Design and Tech.*, pages 15–18, Austin, TX, USA, May 2005.
- [36] R. Reif, A. Fan, K.-N. Chen, and S. Das. Fabrication Technologies for Three-Dimensional Integrated Circuits. In *Proc. of the 3rd Intl. Symp. on Quality Electronic Design*, pages 33–37, San Jose, CA, USA, March 2002.
- [37] R. Ronen, A. Mendelson, et al. Coming Challenges in Microarchitecture and Architecture. *Proceedings of the IEEE*, 89(3):325–340, March 2001.
- [38] E. Samson, S. Machiroutu, et al. Interface Material Selection and a Thermal Management Technique in Second-Generation Platforms Built on Intel Centrino Mobile Technology. *Intel Technology Journal*, 9(1), February 2005.
- [39] K. Sankaralingam, R. Nagarajan, H. Liu, C. Kim, J. Huh, D. Burger, S. W. Keckler, and C. R. Moore. Exploiting ILP, TLP, and DLP with the Polymorphous TRIPS Architecture. In *Proc. of the 30th Intl. Symp. on Comp. Arch.*, pages 422–433, San Diego, CA, USA, May 2003.
- [40] A. Seznec, S. Felix, V. Krishnan, and Y. Sazeides. Design Tradeoffs for the Alpha EV8 Conditional Branch Predictor. In *Proc. of the 29th Intl. Symp. on Comp. Arch.*, Anchorage, AK, USA, May 2002.
- [41] J. Sharkey, D. Ponomarev, K. Ghose, and O. Ergin. Power-Efficient Wakeup Tag Broadcast. In *Proc. of the Intl. Conf. on Comp. Design*, San Jose, CA, USA, October 2005.
- [42] K. Skadron, M. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan. Temperature-Aware Microarchitecture: Modeling and Implementation. *ACM Trans. on Arch. and Code Optimization*, 1(1):94–125, March 2004.
- [43] V. Srinivasan, D. Brooks, M. Gschwind, P. Bose, V. Zyuban, P. N. Strenski, and P. G. Emma. Optimizing Pipelines for Power and Performance. In *Proc. of the 35th Intl. Symp. on Microarchitecture*, pages 333–344, Istanbul, Turkey, November 2002.
- [44] S. Swanson, K. Michelson, A. Schwerin, and M. Oskin. WaveScalar. In *Proc. of the 36th Intl. Symp. on Microarchitecture*, pages 291–302, San Diego, CA, USA, May 2003.
- [45] Tezzaron Semiconductors. Tezzaron Unveils 3D SRAM. Press Release from <http://www.tezzaron.com>, January 24 2005.
- [46] S. Thompson, M. Alavi, et al. 130nm Logic Technology Featuring 60nm Transistors, Low-K Dielectrics, and Cu Interconnects. *Intel Technology Journal*, 6(2), May 2002.
- [47] Y.-F. Tsai, Y. Xie, N. Vijaykrishnan, and M. J. Irwin. Three-Dimensional Cache Design Using 3DCacti. In *Proc. of the Intl. Conf. on Comp. Design*, San Jose, CA, USA, October 2005.
- [48] L. Villa, M. Zhang, and K. Asanović. Dynamic Zero Compression for Cache Energy Reduction. In *Proc. of the 33rd Intl. Symp. on Microarchitecture*, Monterey, CA, USA, December 2000.
- [49] Y. Xie, G. H. Loh, B. Black, and K. Bernstein. Design Space Exploration for 3D Architecture. *To appear in the ACM Jour. of Emerging Technologies in Comp. Systems*, July 2006.
- [50] L. Xue, C. Liu, and S. Tiwari. Multi-layers with buried structures (mlbs): An approach to three-dimensional integration. *IEEE International Conference on Silicon On Insulator*, pages 117–118, 2001.
- [51] A. Zeng, J. Lü, K. Rose, and R. J. Gutmann. First-Order Performance Prediction of Cache Memory with Wafer-Level 3D Integration. *IEEE Design and Test of Comp.*, 22(6):548–555, November–December 2005.
- [52] Y. Zhang, J. Yang, and R. Gupta. Frequent Value Locality and Value-Centric Data Cache Design. In *Proc. of the 9th Symp. on Arch. Support for Prog. Lang. and Operating Systems*, pages 150–159, Cambridge, MA, USA, November 2000.
- [53] Ziptronix Corporation. <http://www.ziptronix.com>.