Appears in the Proceedings of the 27^{th} Annual International Symposium on Computer Architecture

Clock Rate versus IPC: The End of the Road for Conventional Microarchitectures

Vikas Agarwal M.S. Hrishikesh Stephen W. Keckler Doug Burger
Computer Architecture and Technology Laboratory
Department of Computer Sciences
The University of Texas at Austin
cart@cs.utexas.edu — www.cs.utexas.edu/users/cart

Abstract

The doubling of microprocessor performance every three years has been the result of two factors: more transistors per chip and superlinear scaling of the processor clock with technology generation. Our results show that, due to both diminishing improvements in clock rates and poor wire scaling as semiconductor devices shrink, the achievable performance growth of conventional microarchitectures will slow substantially. In this paper, we describe technology-driven models for wire capacitance, wire delay, and microarchitectural component delay. Using the results of these models, we measure the simulated performance—estimating both clock rate and IPC—of an aggressive out-of-order microarchitecture as it is scaled from a 250nm technology to a 35nm technology. We perform this analysis for three clock scaling targets and two microarchitecture scaling strategies: **pipeline scaling** *and* **capacity scaling**. *We find that no scaling strategy permits annual performance improvements of better than 12.5%, which is far worse than the annual 50-60% to which we have grown accustomed.*

1 Introduction

For the past decade, microprocessors have been improving in overall performance at a rate of approximately 50–60% per year. These substantial performance improvements have been mined from two sources. First, designers have been increasing clock rates at a rapid rate, both by scaling technology and by reducing the number of levels of logic per cycle. Second, designers have been exploiting the increasing number of transistors on a chip, plus improvements in compiler technology, to improve instruction throughput (IPC). Although designers have generally opted to emphasize one over the other, both clock rates and IPC have been improving consistently. In Figure 1, we show that while some designers have chosen to optimize the design for fast clocks (Compaq Alpha), and others have optimized their design for high instruction throughput (HP PA-RISC), the past decade's performance increases have been a function of both.

Achieving high performance in future microprocessors will be a tremendous challenge, as both components of performance improvement are facing emerging technology-driven limitations. Designers will soon be unable to sustain clock speed improvements at the past decade's annualized rate of 50% per year. We find that the rate of clock speed improvement must soon drop to scaling linearly with minimum gate length, between 12% and 17% per year.

Compensating for the slower clock growth by increasing sustained IPC proportionally will be difficult. Wire delays will limit the ability of conventional microarchitectures to improve instruction throughput. Microprocessor cores will soon face a new constraint, one in which they are *communication bound* on the die instead of *capacity bound*. As feature sizes shrink, and wires become slower relative to logic, the amount of state that can be accessed in a single clock cycle will cease to grow, and will eventually begin to decline. Increases in instruction-level parallelism will be limited by the amount of state reachable in a cycle, not by the number of transistors that can be manufactured on a chip.

For conventional microarchitectures implemented in future technologies, our results show that, as wire delays grow relative to gate delays, improvements in clock rate and IPC become directly antagonistic. This fact limits the performance achievable by any conventional microarchitecture. In such a world, designers are faced with a difficult choice: increase the clock rate aggressively at the cost of reducing IPC, or mitigate the decline in IPC by slowing the rate of clock speed growth.

In this paper, we explore the scalability of microprocessor cores as technology shrinks from the current 250nm feature sizes to the projected 35nm in 2014. With detailed wire and component models, we show that today's designs scale poorly with technology, improving *at best* 12.5% per year over the next fourteen years. We show that designers must select among deeper pipelines, smaller structures, or slower clocks, and that none of these choices, nor the best combination, will result in scalable performance. Whether designers choose an aggressive clock and lower IPC, or a slower clock and a higher IPC, today's designs cannot sustain the performance improvements of the past decades.

In Section 2, we describe trends in transistor switching and wire transmission time, as well as our analytical wire delay model. The delay model is derived from capacitance extracted from a 3D field solver, using technology parameters from the Semiconductor Industry Association (SIA) technology roadmap [22]. We use the model to estimate microarchitectural wiring delays in future technologies.

In Section 3, we describe our microarchitecture component models, which are based on the Cacti cache delay analysis tool [30]. These models calculate access delay as a function of cache parameters and technology generation. We model most of the major components of a microprocessor core, such as caches, register files, and queues. We show that the inherent trade-off between access time and capacity will force designers to limit or even decrease the size of the structures to meet clock rate expectations. For example, our models show that in a 35nm implementation with a 10GHz clock, accessing even a 4KB level-one cache will require 3 clock cycles.

In Section 4, we report experimental results that show how the projected scaling of microarchitectural components affects overall performance. Using the results of our analytical models as inputs to SimpleScalar-based timing simulation, we track the performance of current microarchitectures when scaled from 250nm to 35nm technology, using different approaches for scaling the clock

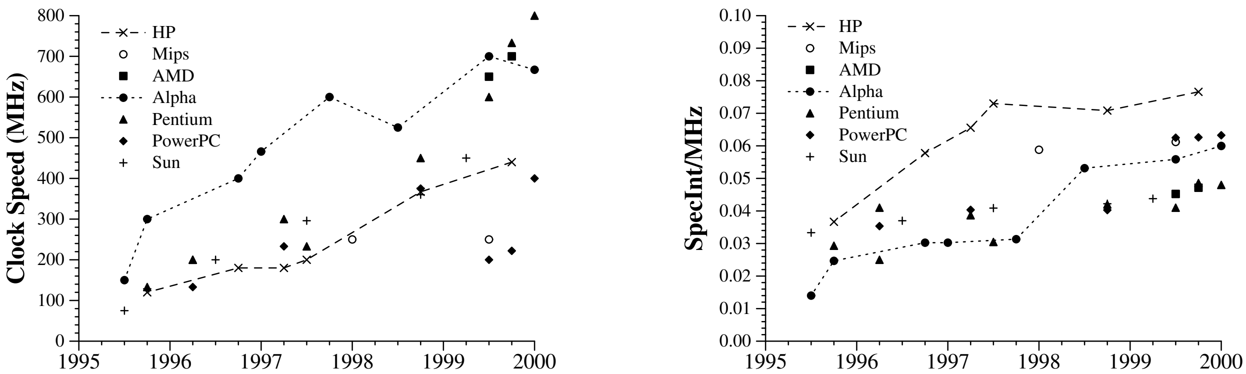

Figure 1: Processor clock rates and normalized processor performance (SpecInt/Clock rate), 1995-2000.

and the microarchitecture. We measure the effect of three different clock scaling strategies on the microarchitecture: setting the clock at 16 *fanout-of-four* (FO4) delays, setting the clock at 8 FO4 delays, and scaling the clock according to the SIA roadmap projections. We also measure two different microarchitecture scaling strategies: *pipeline scaling* and *capacity scaling*. In pipeline scaling, we hold the size of the microarchitectural structures to be constant across technology generations, and measure a deeper pipe caused by longer delays to access core structures. In capacity scaling, we reduce the sizes of the microarchitectural structures at each technology generation, attempting to limit the depth of the pipeline.

Our results show that no scaling strategy permits more than a factor of 7.4 improvement in performance over a seventeen-year period, compared to the 1720-fold improvement that would result from 55% annual performance improvements over the same period. Finally, in Section 5, we draw conclusions from our results and describe the implications for future microprocessor designs.

2 Technology Trends

Microprocessor performance improvements have been driven by developments in silicon fabrication technology that have caused transistor sizes to decrease. Reduced feature sizes have provided two benefits. First, since transistors are smaller, more can be placed on a single die, providing area for more complex microarchitectures. Second, technology scaling reduces transistor gate length and hence transistor switching time. If microprocessor cycle times are dominated by gate delay, greater quantities of faster transistors contribute directly to higher performance.

However, faster clock rates and slower wires will limit the number of transistors reachable in a single cycle to be a small fraction of those available on a chip. Reducing the feature sizes has caused wire width and height to decrease, resulting in larger wire resistance due to smaller wire cross-sectional area. Unfortunately, wire capacitance has not decreased proportionally. Even though wire surface area is smaller, the spacing between wires on the same layer is also being reduced. Consequently, the decreased parallel-plate capacitance is offset by increased coupling capacitance to neighboring wires. In this section we use simple first-order models to demonstrate the effect of technology scaling on chip-wide communication delays and clock rate improvements. We use these models to reason about how future microarchitectures can expect to be scaled.

2.1 Wire Scaling

Our source for future technology parameters pertinent to wire delay is the 1999 International Technology Roadmap for Semiconductors [22]. Although the roadmap outlines the targets for future technologies, the parameters described within are not assured. Nonetheless, we assume that the roadmap's aggressive technology scaling predictions (particularly those for conductor resistivity ρ and dielectric permittivity κ) can be met. We also use the roadmap's convention of subdividing the wiring layers into three categories: (1) *local* for connections within a cell, (2) *intermediate*, or mid-level, for connections across a module, and (3) *global*, or top-level, for chip-wide communication. To reduce communication delay, wires are both wider and taller in the mid-level and top-level metal layers. In our study of wire delay, we focus on mid-level and top-level wires, and use the the wire width, height, and spacing projected in the roadmap.

Since the delay of a wire is directly proportional to the product of its resistance and capacitance, we developed models for these parameters across all of the technology generations of interest. To compute wire resistance per unit length, ($\Omega/\mu m$) we use the simple equation, $R = \frac{\rho}{W \times H}$, where ρ is wire resistance, W is wire width, and H is wire height. Computing capacitance per unit length ($fF/\mu m$) is more complex; we use a model based on empirical results obtained from Space3D, a three-dimensional field solver [28]. Wire capacitance includes components for conductors in lower and higher metal layers as well as coupling capacitance to neighboring wires in the same layer. For each fabrication technology, we provided Space3D with the geometry for a given wire with other wires running parallel to it on the same layer and perpendicular on the layers above and below. We vary wire height, width, and spacing in the Space3D input geometries, and use least-mean-squared curve fitting to derive the coefficients for the model. By assuming that all conductors other than the modelled wire are grounded, and thus not accounting for Miller-effect coupling capacitance, our capacitance model is optimistic compared to the environment of a wire in a real system.

In Table 1, we display the wire parameters from 250nm to 35nm technologies. Our derived wire resistance per unit length (R_{wire}) and capacitance per unit length (C_{wire}) are shown for both mid-level and top-level metal layers. R_{wire} increases enormously across the technology parameters, with notable discontinuities at the transition to 180nm, due to copper wires, and 70nm, due to an antici-

Gate Length (nm)	Dielectric Constant κ	Metal ρ ($\mu\Omega$-cm)	Mid-Level Metal Width (nm)	Aspect Ratio	R_{wire} ($m\Omega/\mu m$)	C_{wire} ($fF/\mu m$)	Top-Level Metal Width (nm)	Aspect Ratio	R_{wire} ($m\Omega/\mu m$)	C_{wire} ($fF/\mu m$)
250	3.9	3.3	500	1.4	107	0.215	700	2.0	34	0.265
180	2.7	2.2	320	2.0	107	0.253	530	2.2	36	0.270
130	2.7	2.2	230	2.2	188	0.263	380	2.5	61	0.285
100	1.6	2.2	170	2.4	316	0.273	280	2.7	103	0.296
70	1.5	1.8	120	2.5	500	0.278	200	2.8	164	0.296
50	1.5	1.8	80	2.7	1020	0.294	140	2.9	321	0.301
35	1.5	1.8	60	2.9	1760	0.300	90	3.0	714	0.317

Table 1: Projected fabrication technology parameters.

pated drop in resistivity from materials improvements projected in the SIA roadmap. However, to keep pace with shrinking wire width, wire aspect ratio (ratio of wire height to wire width) is predicted to increase up to a maximum of three. Larger aspect ratios increase the coupling capacitance component of C_{wire}, which is somewhat mitigated by reductions in the dielectric constant of the insulator between the wires. Even with the advantages of improved materials, the intrinsic delay of a wire, $R_{wire} \times C_{wire}$, is increasing with every new technology generation. These results are similar to those found in other studies by Horowitz [11] and Sylvester [27].

The derived values for R_{wire} and C_{wire} form the core of our wire delay model. Given the fabrication technology, and the wire length, width, and spacing, our model computes the end-to-end wire transmission delay. For the load on the remote end of the wire, we assume a minimum-size inverter, which has a small gate capacitance relative to the wire capacitance. We assume optimal repeater placement in our model to reduce the delay's dependence on wire length from quadratic to linear. Each repeater is an inverter with PFET and NFET sizes chosen to minimize overall wire delay. We use a π circuit to model each wire segment in a repeated wire, as described in [3], and calculate the overall delay as a function of wire length L using Equation 1.

$$D_{wire} = \frac{L}{l_0}(R_0(C_g + C_w) + p + R_w(\frac{C_w}{2} + C_g)) \qquad (1)$$

R_0 is the on-resistance of the repeater, C_g is the gate capacitance of the repeater, l_0 is the length of a wire segment between repeaters, p is the intrinsic delay of a repeater, and R_w and C_w are the resistance and capacitance of the wire segment between two repeaters. Using this equation, the transmission delay for a 5mm top-level wire more than doubles from 170ps to 390ps over the range of 250nm to 35nm technologies. When possible, increasing the wire width is an attractive strategy for reducing wire delay. Increasing the wire width and spacing by a factor of four for top level metal reduces the delay for a 5mm wire to 210ps in a 35nm process, at a cost of four times the wire tracks for each signal. In this study, we assume the wire widths shown in Table 1.

2.2 Clock Scaling

While wires have slowed down, transistors have been getting dramatically faster. To first order, transistor switching time, and therefore gate delay, is directly proportional to the gate length. In this paper we use the *fanout-of-four* (FO4) delay metric to estimate circuit speeds independent of process technology technologies [11]. The FO4 delay is the time for an inverter to drive four copies of itself. Thus, a given circuit limited by transistor switching speed has the same delay measured in number of FO4 delays, regardless of technology. Reasonable models show that under typical conditions, the FO4 delay, measured in picoseconds (ps) is equal to $360 \times L_{drawn}$, where L_{drawn} is the minimum gate length for a technology, measured in microns. Using this approximation, the FO4 delay decreases from 90ps in a 250nm technology to 12.6ps in 35nm technology, resulting in circuit speeds improving by a factor of seven, just due to technology scaling.

The FO4 delay metric is important as it provides a fair means to measure processor clock speeds across technologies. The number of FO4 delays per clock period is an indicator of the number of levels of logic between on-chip latches. Microprocessors that have a small number of FO4 delays per clock period are more deeply pipelined than those with more FO4 delays per clock period. As shown by Kunkel and Smith [14], pipelining to arbitrary depth in hopes of increasing the clock rate does not result in higher performance. Overhead for the latches between pipeline stages becomes significant when the number of levels of logic within a stage decreases too much. Pipelining in a microprocessor is also limited by dependencies between instructions in different pipeline stages. To execute two dependent instructions in consecutive clock cycles, the first instruction must compute its result in a single cycle. This requirement can be viewed as a lower bound on the amount of work that can be performed in a useful pipeline stage, and could be represented as the computation of an addition instruction. Under this assumption, a strict lower bound on the clock cycle time is 5.5 FO4 delays, which is the minimal computation time of a highly optimized 64-bit adder, as described by Naffziger [17]. When accounting for latch overhead and the time to bypass the output of the adder back to the input for the next instruction, reducing the clock period to eight FO4 delays will be difficult. Fewer than eight may be impossible to implement.

In Figure 2, we plot microprocessor clock periods (measured in FO4 delays) from 1992 to 2014. The horizontal lines represent the eight FO4 and 16 FO4 clock periods. The clock periods projected by the SIA roadmap shrink dramatically over the years and reach 5.6 FO4 delays at 50nm, before increasing slightly to 5.9 FO4 delays at 35nm. The Intel data represent five generations of x86 processors and show the reduction in the number of FO4 delays per pipeline stage from 53 in 1992 (i486DX2) to 15 in 2000 (Pentium III), indicating substantially deeper pipelines. The isolated circles represent data from a wider variety of processors published in the proceedings of the International Solid State Circuits Conference (ISSCC) from 1994 to 2000. Both the Intel and ISSCC data demonstrate that clock rate improvements have come from a combination

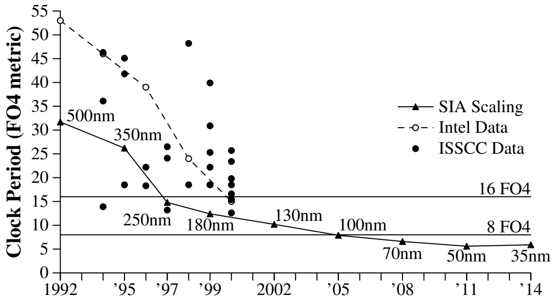

Figure 2: Clock scaling measured in FO4 inverter delays. The aggressive (8 FO4) and conservative (16 FO4) clocks are constant across technologies, but the SIA roadmap projects less than 6 FO4 delays at 50nm and below.

Gate (nm)	Chip Area (mm^2)	16FO4 Clk f_{16} (GHz)	8FO4 Clk f_8 (GHz)	SIA Clk f_{SIA} (GHz)
250	400	0.69	1.35	0.75
180	450	0.97	1.93	1.25
130	567	1.34	2.67	2.10
100	622	1.74	3.47	3.50
70	713	2.48	4.96	6.00
50	817	3.47	6.94	10.00
35	937	4.96	9.92	13.50

Table 2: Projected chip area and clock rate.

of technology scaling and deeper pipelining, with each improving approximately 15-20% per year. While the trend toward deeper pipelining will continue, reaching eight FO4 delays will be difficult, and attaining the SIA projected clock rate is highly unlikely.

In this paper, we examine microprocessor performance when scaling the clock conservatively at 16 FO4 delays (f_{16}), aggressively at eight FO4 delays (f_8), and to absolute limits with the SIA projections (f_{SIA}). In Table 2, we show the resulting clock rates across the spectrum of technologies we measure.

2.3 Wire Delay Impact on Microarchitecture

The widening gap between the relative speeds of gates and wires will have a substantial impact on microarchitectures. With increasing clock rates, the distance that a signal can travel in a single clock cycle decreases. When combined with the modest growth in chip area anticipated for high-performance microprocessors, the time (measured in clock periods) to send a signal across one dimension of the chip will increase dramatically. Our analysis below uses the clock scaling described above and the projected chip areas from the SIA Roadmap, as shown in Table 2.

Based on the wire delay model, we compute the chip area that is reachable in a single clock cycle. Our unit of chip area is the size of a six-transistor SRAM cell, which shrinks as feature size is reduced. To normalize for different feature sizes across the technologies, we measure SRAM cell size in λ, which is equal to one-half the gate length in each technology. We estimate the SRAM cell area to be $700\lambda^2$, which is the median cell area from several recently published SRAM papers [4, 23, 31]. Our area metric does not include overheads found in real SRAM arrays, such as the area required for decoders, power distribution, and sense-amplifiers. Additionally,

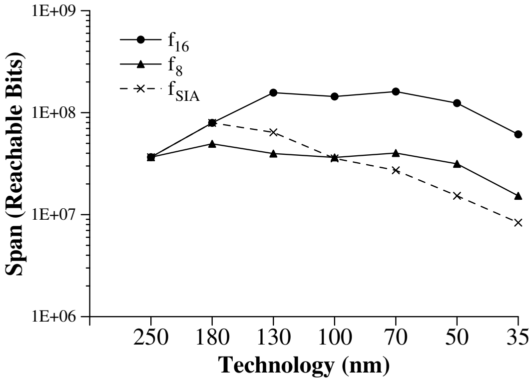

Figure 3: Reachable chip area in top-level metal, where area is measured in six-transistor SRAM cells.

it does not reflect the size of a single-cycle access memory array; the area metric includes all bits reachable within a one-cycle, one-way transmission delay from a fixed location on the chip, ignoring parasitic capacitance from the SRAM cells.

Figure 3 shows the absolute number of bits that can be reached in a single clock cycle, which we term *span*, using top-level wires for f_{16}, f_8, and f_{SIA} clock scaling. The wire width and spacing is set to the minimum specified in the SIA Roadmap for top-level metal at each technology. Using f_{16} clock scaling, the span first increases as the number of bits on a chip increases and the entire chip can still be reached in a single cycle. As the chip becomes communication bound at 130nm, multiple cycles are required to transmit a signal across its diameter. In this region, decreases in SRAM cell size are offset equally by lower wire transmission velocity, resulting in a constant span. Finally, the span begins to decrease at 50nm when the wire aspect ratio stops increasing and resistance becomes more significant. The results for f_8 are similar except that the plateau occurs at 180nm and the span is a factor of four lower than that of f_{16}. However, in f_{SIA} the span drops steadily after 180nm, because the clock rate is scaled superlinearly with decreasing gate length. These results demonstrate that clock scaling has a significant impact on architectures as it demands a trade-off between the size and partitioning of structures. Using high clock rates to meet performance targets limits the size of pipeline stages and microarchitectural structures, while tightly constraining their placement. If lower clock rates can be tolerated, then microarchitects can give less consideration to the communication delay to reach large and remote structures.

Figure 4 shows the fraction of the total chip area that can be reached in a single clock cycle. Using f_8 in a 35nm technology, less than 0.4% of the chip area can be reached in one cycle. Even with f_{16}, only 1.4% of the chip can be reached in one cycle. Similar results have been observed in prior work [16]. If microarchitectures do not change over time, this phenomenon would be unimportant, since the area required to implement them would decrease with feature size. However, microarchitectures have become more complex because architects acquired more transistors with each new fabrication technology, and used them to improve overall performance. In future technologies, substantial delay penalties must be paid to reach the state or logic in a remote region of the chip, so microarchitectures that rely on large structures and global communication

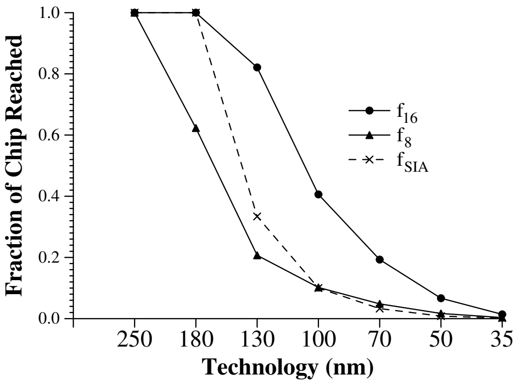

Figure 4: Fraction of total chip area reachable in one cycle.

will be infeasible.

2.4 Summary

While transistor speeds are scaling approximately linearly with feature size, wires are getting slower with each new technology. Even assuming low-resistivity conductors, low-permittivity dielectrics, and higher aspect ratios, the *absolute* delay for a fixed-length wire in top-level metal with optimally placed repeaters is increasing with each generation. Only when the wire width and spacing is increased substantially can the wire delay be kept constant. Due to increasing clock frequencies, wire delays are increasing at an even higher rate. As a result, chip performance will no longer be determined solely by the number of transistors that can be fabricated on a single integrated circuit (capacity bound), but instead will depend upon the amount of state and logic that can be reached in a sufficiently small number of clock cycles (communication bound).

The argument made by Sylvester and Keutzer [27] that wire delays will not affect future chip performance holds only if wire lengths are reduced along with gate lengths in future technologies. Traditional microprocessor microarchitectures have grown in complexity with each technology generation, using all of the silicon area for a single monolithic core. Current trends in microarchitures have increased the sizes of all of the structures, and added more execution units. With future wire delays, structure size will be limited and the time to bypass results between pipeline stages will grow. If clock rates increase at their projected rates, both of these effects will have substantial impact on instruction throughput.

3 Component Models

In addition to reducing the chip area reachable in a clock cycle, both the widening gap between wire and gate delays and superlinear clock scaling has a direct impact on the scaling of microarchitectural structures in future microprocessors. Clock scaling is more significant than wire delay for small structures, while both wire delay and clock scaling are significant in larger structures. The large memory-oriented elements, such as the caches, register files, instruction windows, and reorder buffers, will be unable to continue increasing in size while remaining accessible within one clock cycle. In this section, we use analytical models to examine the access time of different structures from 250nm to 35nm technologies based on the structure organization and capacity. We demonstrate the trade-offs between access time and capacity that are necessary for the various structures across the technology generations.

3.1 Analytical Model

To model the various storage-oriented components of a modern microprocessor, we started with ECacti [19], an extended version of the original Cacti cache modeling tool [30]. Given the capacity, associativity, number ports, and number of data and address bits, ECacti considers a number of alternative cache organizations and computes the minimum access time. ECacti automatically splits the cache into banks and chooses the number and layout of banks that incurs the lowest delay. When modeling large memory arrays, ECacti presumes multiple decoders, with each decoder serving a small number of banks. For example with a 4MB array, ECacti produces 16 banks and four decoders in a 35nm technology. Note that this model is optimistic, because it does not account for driving the address from a central point to each of the distributed decoders.

We extended ECacti to include technology scaling, using the projected parameters from the SIA roadmap. SRAM cell sizes and transistor parasitics, such as source and drain capacitances, are scaled according to their anticipated reduction in area for future technologies. We assume that the word-lines are run from a decoder across its neighboring banks in mid-level metal, and that the bit-rline in mid-level metal does not increase the size of the SRAM cell. Unlike Amrutur and Horowitz [3] we further make the optimistic assumption that the sense-amplifier threshold voltage will decrease linearly with technology. The access times from the analytical model were verified against those obtained from a SPICE simulation of the critical path, and matched within 15% for all technologies. This level of accuracy is comparable to the accuracy of the original CACTI model. A full description of the modeling and validation can be found in [1].

Apart from modeling direct-mapped and set associative caches, we used our extended version of ECacti to explore other microarchitectural structures. For example, a register file is essentially a direct mapped cache with more ports, but fewer address and data bits than a typical L1 data cache. We use a similar methodology to examine issue windows, reorder buffers, branch prediction tables, and TLBs.

3.2 Caches and Register Files

Using our extended ECacti, we measured the memory structure access time, while varying cache capacity, block size, associativity, number of ports, and process technology. While cache organization characteristics do affect access time, the most critical characteristic is capacity. In Figure 5, we plot the access time versus capacity for a dual-ported, two-way set associative cache. The maximum cache capacities that can be reached in 3 cycles for the f_{16}, f_8 and f_{SIA} clocks are also plotted as "isobars". Note that the capacity for a three cycle access cache decreases moderately for f_{16} and f_8, but falls off the graph for f_{SIA}.

We compared our analytical model to other models and related implementations. In a 250nm technology, we compute the access time for a 64KB L1 data cache to be 2.4ns. This access time is comparable to that of the 700MHz Alpha 21264 L1 data cache. Furthermore, for a 4Mb cache in a 70nm technology, our model predicts an access time of 33 FO4 delays which matches the 33

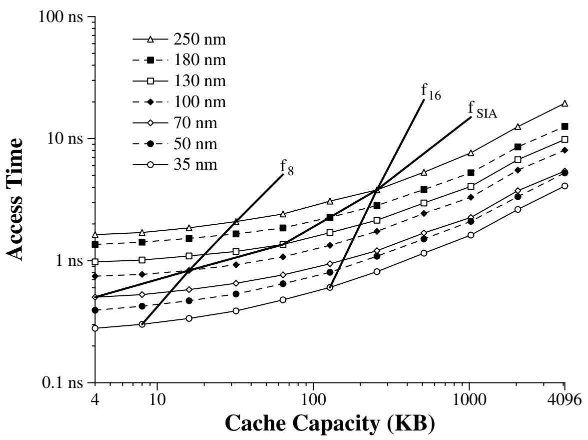

Figure 5: Access time for various L1 data cache capacities.

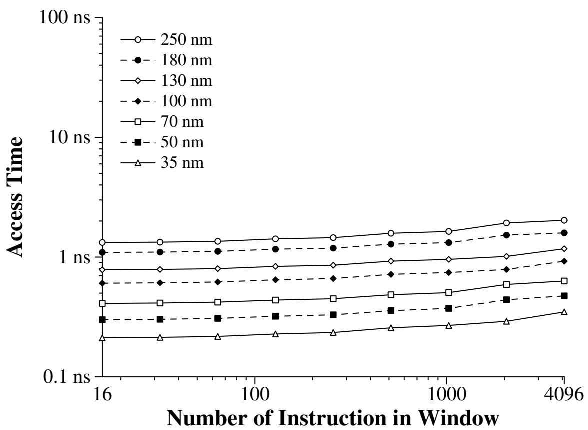

Figure 6: Access time vs. issue window size across technologies.

FO4 access time generated by Amrutur and Horowitz for a similar cache [3].

For each technology, the access time increases as the cache capacity increases. Even with substantial banking, the access time goes up dramatically at capacities greater than 256KB. For a given cache capacity, the transition to smaller feature sizes decreases the cache access time, but not as fast as projected increases in clock rates. In a 35nm technology, a 32KB cache takes one to six cycles to access depending on the clock frequency. One alternative to slower clocks or smaller caches is to pipeline cache accesses and allow each access to complete in multiple cycles. Due to the non-linear scaling of capacity with access time, adding a small number of cycles to the cache access time substantially increases the available cache capacity. For example, increasing the access latency from four to seven cycles increases the reachable cache capacity by about a factor of 16 in a 35nm technology. The results shown in Figure 5 apply to all of the cache-like microarchitectural structures that we examine in this study, including L1 instruction and data caches, L2 caches, register files, branch target buffers, and branch prediction tables.

While our cache model replicates current design methodologies, our register file model is more aggressive in design. Although register files have traditionally been built using single-ended full swing bit lines [20], larger capacity register files will need faster access provided by differential bit-lines and low-voltage swing sense-amplifiers similar to those in our model. For our register file modeling, the cache block size is set to the register width, which in our case is eight bytes. For a large ported register file, the size of each cell in the register file increases linearly in both dimensions with the number of ports.

Our capacity results for the register file are similar to those seen in caches. The most significant difference between a cache and a register file is the number of ports. Our results show that register files with many ports will incur larger access times. For example, in a 35nm technology, going from ten ports to 32 ports increases the access time of a 64-entry register file from 172ps to 274ps. Increased physical size and access time makes attaching more execution units to a single global register file impractical. The alternatives of smaller register files versus multi-cycle access times are examined quantitatively in Section 4.

3.3 Content Addressable Memories

The final set of components that we model are those that require global address matching within the structure, such as the instruction window and the TLB. These components are typically implemented as content addressable memories (CAMs) and can be modelled as a fully associative cache. Our initial model of the instruction window includes a combination of an eight-bit wide CAM and a 40-bit wide direct mapped data array for the contents of each entry. The issue window has eight ports, which are used to insert four instructions, write back four results, and extract four instructions simultaneously. Since we assume that the eight-ported CAM cell and corresponding eight-ported data array cell are port-limited, we compute the area of these cells based on the width and number of bit-lines and word-lines used to access them. Note that we model only the structure access time and do not consider the latency of the instruction selection logic.

Figure 6 shows the access time for this configuration as a function of the number of instructions in the window. As with all of the memory structures, the access time increases with capacity. The increase in access time is not as significant as in the case of the caches, because the capacities considered are small and all must pay an almost identical penalty for the fully associative match on the tag bits. Thus, in this structure, once the initial tag match delay has been computed, the delay for the rest of the array does not increase significantly with capacity. A 128-entry, eight-port, eight-bit tag instruction window has an access time of 227ps in a 35nm process, while a 12-bit tag raises the access time of a same size window to 229ps. A 128-entry, 32-port, eight-bit tag instruction window (as might be required by a 16-issue processor) has an access time of 259ps in a 35nm technology. Note that all of these results ignore the increase in complexity of the selection logic as we increase the issue window size and the port count in the issue window. We anticipate that the capacity and port count of the register file and the complexity of the selection logic will ultimately place a limit on the issue width of superscalar microarchitectures [18].

3.4 Summary

Because of increasing wire delays and faster transistors, memory-oriented microarchitectural structures are not scaling with technology. To access caches, register files, branch prediction tables, and

Structure Name	f_{SIA}	f_8	f_{16}
L1 cache 64K (2 ports)	7	5	3
Integer register file 64 entry (10 ports)	3	2	1
Integer issue window 20 entry (8 ports)	3	2	1
Reorder buffer 64 entry (8 ports)	3	2	1

Table 3: Projected access time (cycles) at 35nm.

instruction windows in a single cycle will require the capacity of these structures to decrease as clock rates increase. In Table 3, we show the number of cycles needed to access the structures from the Compaq Alpha 21264, scaled to a 35nm process for each of the three methods of clock scaling. With constant structure capacities, the L1 cache will take up to seven cycles to access, depending on how aggressively the clock is scaled.

Another factor not explored in our analysis is the communication delay between execution units in wider-issue superscalar processors. If the number of execution units is increased, the distance between the extreme units will also increase. Consequently, the bypass logic complexity and bypass data transmission delay will be substantial barriers to improved performance [18, 20].

4 Performance Analysis

As communication delays increase relative to computation delays, superlinear clock rate scaling and today's techniques for exploiting ILP work against one another. Aggressively scaling the clock reduces the amount of state that can be used to exploit ILP for a fixed pipeline depth. The designer is faced with two interacting choices: how aggressively to push the clock rate by reducing the number of levels of logic per cycle, and how to scale the size and pipeline depth of different microarchitectural structures. For a given target frequency, we define two approaches for scaling the microarchitecture to smaller technologies:

- *Capacity scaling*: Shrink the microarchitectural structures sufficiently so that their access penalties are constant across technologies. We define *access penalty* as the access time for a structure measured in clock cycles.
- *Pipeline scaling*: Hold the capacity of a structure constant and increase the pipeline depth as necessary to cover the increased latency across technologies.

While these trade-offs of clock versus structure size scaling manifest themselves in every design process, their importance will grow as communication delay creates more interference between clock speed and ILP optimizations.

In this section, we explore the effect of capacity and pipeline scaling strategies upon IPC and overall performance. For each scaling strategy, and at three different clock scaling rates (f_{16}, f_8, and f_{SIA} projections), we measure IPC for our target microarchitecture at technologies ranging from 250nm to 35nm. The methodology uses microarchitectural simulation that incorporates the results of our technology and structure models described in Sections 2 and 3. Our goal is to find the balance among capacity, pipeline, and clock rate scaling that achieves the best overall performance.

4.1 Experimental Methodology

We perform our microarchitectural timing simulations with an extended version of the SimpleScalar tool set [6], version 3.0. We also use the SimpleScalar memory hierarchy extensions, which simulate a one-level page table, hardware TLB miss handling, finite miss status holding registers (MSHRs) [13], and simulation of bus contention at all levels [7].

4.1.1 Target Microprocessor

The SimpleScalar tools use a Register Update Unit (RUU) [24] to track out-of-order issue of instructions. The RUU acts as a unified reorder buffer, issue window, and physical register file. Because of the large number of ports required for the RUU in a four-wide machine, implementing it is not feasible at high clock frequencies. We therefore split the structures logically in SimpleScalar, effectively simulating a separate reorder buffer, issue window, and physical register file.

We also modified SimpleScalar's target processor core to model a four-wide superscalar pipeline organization roughly comparable to the Compaq Alpha 21264 [12]. Our target is intended to model a microarchitecture typical of those found in current-generation processors; it is *not* intended to model the 21264 microarchitecture itself. However, we chose as many parameters as possible to resemble the 21264, including the pipeline organization and microarchitectural structure capacities. Our target processor uses a seven-stage pipeline for integer operations in our simulated base case, with the following stages: fetch, decode, map, queue, register read, execute, and writeback. As in the 21264, reorder buffer accesses occur off of the critical path, physical registers are read after instruction issue, and instructions are loaded into the issue queues in program order.

Our target differs from the 21264 in the following respects. We assume that the branch target buffer is a distinct structure, as opposed to a line predictor embedded in the instruction cache. We simulate a two-level gshare predictor instead of the 21264's local history, global history, and choice predictors. We remove instructions from the issue queue immediately after they are issued, rather than implementing load-use speculation and waiting two cycles for its resolution. We simulate a combined floating-point and integer issue queue (they are split in the 21264), but model them from a timing perspective as if they were split. We do not implement the functional unit and register file clustering found in the 21264. Instead of the six instructions per cycle that the 21264 can issue (four integer and two floating-point), we permitted only four. We use the default SimpleScalar issue prioritization policy, which issues ready loads with highest priority, followed by the oldest ready instructions. The 21264 always issues the oldest ready instruction regardless of its type. Finally, we do not simulate the 21264 victim cache, which contains an eight-entry victim buffer.

4.1.2 Simulation Parameters

Our baseline processor parameters include the following: a four-way issue superscalar processor with a 40-entry issue window for both integer and floating point operations, a 64-entry load/store queue, commit bandwidth of eight instructions per cycle, an eight-entry return address stack, and a branch mispredict rollback latency equal to the reorder buffer access delay plus three cycles. We set the number and delays of the execution units to those of the 21264 [12].

	Capacity (bits)	# entries	Bits/entry	Ports
Branch pred.	32K	16K	2	1
BTB	48K	512	96	1
Reorder buffer	8K	64	128	8
Issue window	800/160	20	40	8
Integer RF	5K	80	64	10
FP RF	5K	80	64	10
L1 I-Cache	512K	1K	512	1
L1 D-Cache	512K	1K	512	2
L2 Cache	16M	16K	1024	2
I-TLB	14K	128	112	2
D-TLB	14K	128	112	2

Table 4: Capacities of structures used in delay calculations

We show the default sizes of the remaining structures in Table 4.

In our baseline memory system, we use separate 64KB, two-way set associative level-one instruction and data caches, with a 2MB, four-way set-associative, unified level-two cache. The L1 caches have 64-byte lines, and the L2 cache has 128-byte lines. The L1/L2 cache bus is 128 bits wide, requires one cycle for arbitration, and runs at the same speed as the processing core. Each cache contains eight MSHRs with four combining targets per MSHR.

We assume a DRAM system that is aggressive by today's standards. The L2/memory bus is 64 bits wide, requires one bus cycle for arbitration, and runs at half the processor core speed. That speed ratio, assuming a tightly coupled electrical interface to memory, is similar to modern Rambus memory channels. We assumed that the base DRAM access time is 50ns, plus a somewhat arbitrary 20ns for the memory controller. That access time of 70ns is typical for what was available in 1997. For each year beyond 1997, we reduce the DRAM access time by 10%, using that delay and the clock rate for a given year to compute the DRAM access penalty, in cycles, for the year in question. We model access time and bus contention to the DRAM array, but do not model page hits, precharging overhead, refresh cycles, or bank contention. The benchmarks we use exhibit low miss rates from the large L2 caches in our simulations.

We simulate each of the SPEC95 benchmarks [26] for 500 million instructions. We use the `std` input set; a combination of different SPEC inputs that run for less time than the reference data sets, but have equivalent data sets and are not dominated by initialization code for runs of this length [5].

4.1.3 Modeling Deeper Pipelines

To simulate pipeline scaling in SimpleScalar, we added the capability to simulate variable-length pipelines by specifying the access latency for each major structure as a command-line parameter. We assume that all structures are perfectly pipelined, and can begin m accesses every cycle, where m is the number of ports. We do not account for any pipelining overheads due to latches or clock skew [14]. Furthermore, we assume that an n-cycle access to a structure will cause an $n-1$ cycle pipeline stall as specified below. We perform the following accounting for access delays in which $n > 1$:

- *I-Cache*: pipeline stalls affect performance only when a branch is predicted taken. We assume that fetches with no changes in control flow can be pipelined.

- *Issue window*: additional cycles to access the issue window cause delays when instructions are removed from the queue (wakeup and select), not when instructions are written into the issue window.

- *Reorder buffer*: as in the 21264, writes to the reorder buffer and commits occur off the critical path. We therefore add reorder buffer delays only to the rollback latency, which in turn is incurred only upon a branch misprediction.

- *Physical register file*: register access penalties are paid only on non-bypassed register file reads. Register file writes are queued and bypassed to dependent instructions directly.

- *Branch predictor and BTB*: multi-cycle predictor accesses create pipeline bubbles only when a prediction must be made. Multi-cycle BTB accesses cause the pipeline to stall only when a branch is predicted taken. We assume that the two structures are accessed in parallel, so that pipeline bubbles will be the maximum, rather than the sum, of the two delays.

4.2 Pipeline versus Capacity Scaling

We use the simulated microarchitecture described above to measure IPC across the benchmark suite. The access latencies for the microarchitectural structures are derived from the models described in Section 3. From the access latencies, we compute the access penalties for each of the three clock scaling rates (f_{16}, f_8, and f_{SIA}).

In Table 4, we show both the number of bits per entry and the number of ports for each of the baseline structures. These parameters are used to compute the delay as a function of capacity and technology as well as the capacity as a function of access penalty and technology. In the issue window entry, we show two bit capacities, one for the instruction queue and one for the tag-matching CAM.

In the third column of Table 4, we show the baseline structure sizes that we use for our pipeline scaling experiments. In Table 5, we show the actual access penalty of the structures for the fixed baseline capacities. Note that as the feature size decreases, the access penalty to the fixed size structures increases, and is dependent on the clock rate. Consequently, deeper pipelines are required as a fixed-capacity microarchitecture is scaled to smaller technologies.

In Table 6, we show the parameters for the capacity scaling experiments. Because the access penalties are held nearly constant, the capacities of the structures decrease dramatically as smaller technologies and higher clock rates are used. For each technology generation, we set each structure to be the maximum size possible while ensuring that it remains accessible in the same number of cycles as our base case (one cycle for most of the structures, and ten cycles for the L2 cache).

In future technologies, some of the structures become too small to be useful at their target access penalty. In such cases, we increase the access penalty slightly, permitting a structure that is large enough to be useful. The access penalties are shown in the subscripts in Table 6. Note that for technologies smaller than 130nm, no structure can be accessed in less than two cycles for the f_8 and f_{SIA} frequency scales. Note that all of the L2 cache access penalties are ten cycles except for f_8 and f_{SIA}, for which we increased the access penalty to 15 cycles at 50nm and 35nm.

Structure	250nm $f_{16}/f_8/f_{SIA}$	180nm $f_{16}/f_8/f_{SIA}$	130nm $f_{16}/f_8/f_{SIA}$	100nm $f_{16}/f_8/f_{SIA}$	70nm $f_{16}/f_8/f_{SIA}$	50nm $f_{16}/f_8/f_{SIA}$	35nm $f_{16}/f_8/f_{SIA}$
Branch pred.	1/2/2	2/3/2	2/3/2	2/3/3	2/3/3	2/3/4	2/3/4
BTB	1/2/1	1/2/2	1/2/2	1/2/2	1/2/3	1/2/3	1/2/3
Reorder buffer	1/2/1	1/2/2	1/2/2	1/2/2	1/2/3	1/2/3	1/2/3
Issue window	1/2/1	1/2/2	1/2/2	1/2/2	1/2/3	1/2/3	1/2/3
Integer RF	1/2/1	1/2/2	1/2/2	1/2/2	1/2/2	1/2/3	1/2/3
FP RF	1/2/1	1/2/2	1/2/2	1/2/2	1/2/2	1/2/3	1/2/3
L1 I-Cache	2/3/2	2/3/2	2/3/3	2/3/3	2/3/4	2/4/5	2/4/5
L1 D-Cache	2/4/2	2/4/3	2/4/3	2/4/4	2/4/5	3/5/7	3/5/7
L2 Cache	11/21/11	10/19/12	11/21/17	11/23/23	12/24/29	17/34/49	19/38/52
I-TLB	2/3/2	2/3/2	2/3/3	2/3/3	2/3/4	2/3/4	2/3/4
D-TLB	2/3/2	2/3/2	2/3/3	2/3/3	2/3/4	2/3/4	2/3/4

Table 5: Access times (in cycles) using pipeline scaling with f_{16}, f_8, and f_{SIA} clock scaling.

Structure	250nm $f_{16}/f_8/f_{SIA}$	180nm $f_{16}/f_8/f_{SIA}$	130nm $f_{16}/f_8/f_{SIA}$	100nm $f_{16}/f_8/f_{SIA}$	70nm $f_{16}/f_8/f_{SIA}$	50nm $f_{16}/f_8/f_{SIA}$	35nm $f_{16}/f_8/f_{SIA}$
BPred	$8K_1/8K_2/8K_1$	$8K_1/8K_2/1K_1$	$4K_1/8K_2/256_1$	$4K_1/4K_2/4K_2$	$4K_1/4K_2/2K_2$	$4K_1/4K_2/256_2$	$4K_1/4K_2/512_2$
BTB	$1K_1/1K_2/512_1$	$512_1/512_2/4K_2$	$512_1/512_2/2K_2$	$512_1/512_2/512_2$	$512_1/512_2/128_2$	$256_1/256_2/512_3$	$256_1/256_2/512_3$
ROB	$256_1/512_2/128_1$	$128_1/128_2/8K_2$	$128_1/128_2/2K_3$	$128_1/128_2/128_2$	$128_1/128_2/2K_3$	$64_1/64_2/256_4$	$64_1/64_2/256_3$
IW	$512_1/512_2/128_1$	$64_1/64_2/8K_2$	$64_1/64_2/2K_2$	$64_1/64_2/64_2$	$64_1/64_2/2K_3$	$64_1/64_2/128_3$	$64_1/64_2/256_3$
Int. RF	$256_1/256_2/128_1$	$256_1/256_2/35_1$	$128_1/128_2/512_2$	$128_1/128_2/128_2$	$128_1/128_2/64_2$	$128_1/128_2/128_3$	$128_1/128_2/128_3$
FP RF	$256_1/256_2/128_1$	$256_1/256_2/35_1$	$128_1/128_2/512_2$	$128_1/128_2/128_2$	$128_1/128_2/64_2$	$128_1/128_2/128_3$	$128_1/128_2/128_3$
L1 I$	$256K_2/64K_3/256K_2$	$256K_2/64K_3/64K_2$	$256K_2/64K_3/16K_2$	$256K_2/64K_3/64K_3$	$128K_2/64K_3/16K_3$	$128K_2/32K_3/32K_4$	$128K_2/32K_3/32K_4$
L1 D$	$64K_2/32K_3/64K_2$	$64K_2/16K_3/32K_2$	$64K_2/16K_3/2K_2$	$64K_2/16K_3/16K_3$	$64K_2/16K_3/4K_3$	$32K_2/8K_3/4K_4$	$32K_2/8K_3/8K_4$
$L2_{10}$	2M/256K/1M	2M/256K/1M	1M/256K/1M	1M/256K/256K	1M/256K/128K	$512K/256K_{15}/128K_{15}$	$512K/256K_{15}/128K_{15}$
I-TLB	$32K_2/512_3/32K_2$	$32K_2/512_3/4K_2$	$32K_2/512_3/4K_3$	$16K_2/512_3/2K_3$	$16K_2/512_3/4K_4$	$16K_2/256_3/1K_4$	$16K_2/256_3/1K_4$
D-TLB	$32K_2/512_3/32K_2$	$32K_2/512_3/4K_2$	$32K_2/512_3/4K_3$	$16K_2/512_3/2K_3$	$16K_2/512_3/4K_4$	$16K_2/256_3/1K_4$	$16K_2/256_3/1K_4$

Table 6: Structure sizes and access times (in subscripts) using capacity scaling with f_{16}, f_8, and f_{SIA} clock scaling.

4.3 Performance Measurements

Table 7 shows the geometric mean of the measured IPC values across all 18 of the SPEC95 benchmarks. The results include both pipeline scaling and capacity scaling experiments at each of the three clock scaling targets.

In general, the IPC decreases as the technology is scaled from 250nm to 35nm. For linear clock scaling (f_{16} and f_8), this effect is caused by longer structure access penalties due to sublinear scaling of the wires. For f_{SIA}, the superlinear reduction in cycle time causes a larger increase in access penalty than f_{16} or f_8, resulting in a sharper drop in IPC.

There are several cases in which IPC increases when moving from a larger technology to a smaller one. These small increases are caused by discretization limitations in our model: if the structure capacity at a given technology becomes too small to be useful, we increase the access penalty by one cycle. This increase occasionally results in a larger structure than in the previous technology, causing a small increase in IPC. One example of this effect is evident for capacity scaling at f_{SIA} clock rates when moving from 130nm to 100nm. In Table 6, the branch predictor, L1 I-Cache, and L1 D-Cache all become noticeably larger but one cycle slower. A similar effect occurs for f_{SIA} capacity scaling at the transition from 70nm to 50nm.

With the slower clock rates (f_{16} and f_{SIA}), the IPCs at 250nm for capacity scaling are considerably higher than those for pipeline scaling. While the capacity scaling methodology permits structure sizes larger than chips of that generation could contain, the pipeline scaling methodology sets the sizes to be roughly equivalent to the 21264. The capacity scaling results at 250nm and 180nm thus show the IPCs if the chip was not limited by area. As the wires grow slower in the smaller technologies, and the core becomes communication bound rather than capacity bound, the capacity scaling strategy loses its advantage. The pipeline scaling shows higher IPC than capacity scaling for fast clocks at the smallest technologies. The highest IPC at 35nm, unsurprisingly, is for capacity scaling at the slowest clock available–that point is the one at 35nm for which the microarchitectural structures are the largest. Even with capacity scaling at the f_{16} clock scale, however, IPC decreases by 20% from 250nm to 35nm.

For either scaling strategy, of course, IPC decreases as clock rates are increased for smaller technologies. However, performance estimates must include the clock as well. In Table 8, we show the geometric mean performance of the SPEC95 benchmarks, measured in billions of instructions per second (BIPS), for each of the microarchitectural and clock scales. Most striking about the results is the similarity in performance improvements across the board, especially given the different clock rates and scaling methodologies. Also remarkable are the small magnitudes of the speedups, ranging from five to seven for 35nm (for all our normalized numbers, the base case is pipeline scaling performance at f_{16} and 250nm).

For all technologies through 50nm and for both scaling strategies, faster clocks result in uniformly greater performance. For capacity scaling at 35nm, however, the faster clocks show worse performance: f_{16} has the highest BIPS, f_8 is second, and f_{SIA} has the lowest performance. This inversion is caused mainly by the memory system, since the 35nm cache hierarchy has considerably less capacity for the faster clocks.

For pipeline scaling, the caches all remain approximately the same size regardless of clock scaling, and the benefit of faster clocks overcome the setbacks of higher access penalties. Thus at 35nm, the best-performing clock scale is the most aggressive scale (f_{SIA}).

Scaling	Clock Rate	250nm	180nm	130nm	100nm	70nm	50nm	35nm
Pipeline	f_{16}	1.25	1.16	1.15	1.15	1.17	1.08	1.06
	f_8	0.77	0.73	0.72	0.72	0.71	0.64	0.63
	f_{SIA}	1.18	0.89	0.83	0.73	0.62	0.49	0.48
Capacity	f_{16}	1.63	1.55	1.48	1.48	1.46	1.30	1.30
	f_8	0.89	0.82	0.81	0.81	0.80	0.68	0.63
	f_{SIA}	1.52	1.03	0.69	0.86	0.49	0.50	0.45

Table 7: Geometric mean of IPC for each technology across the SPEC95 benchmarks.

Scaling	Clock Rate	250nm	180nm	130nm	100nm	70nm	50nm	35nm	Speedup
Pipeline	f_{16}	0.87	1.11	1.54	2.01	2.90	3.73	5.25	6.04
	f_8	1.07	1.41	1.93	2.49	3.54	4.44	6.23	7.16
	f_{SIA}	0.89	1.11	1.74	2.58	3.70	4.85	6.49	7.46
Capacity	f_{16}	1.12	1.49	1.98	2.58	3.63	4.50	6.42	7.38
	f_8	1.24	1.58	2.18	2.81	3.99	4.71	6.28	7.21
	f_{SIA}	1.14	1.28	1.44	3.02	2.92	4.97	6.04	6.95

Table 8: Geometric mean of performance (billions of instructions per second) for each technology across the SPEC95 benchmarks.

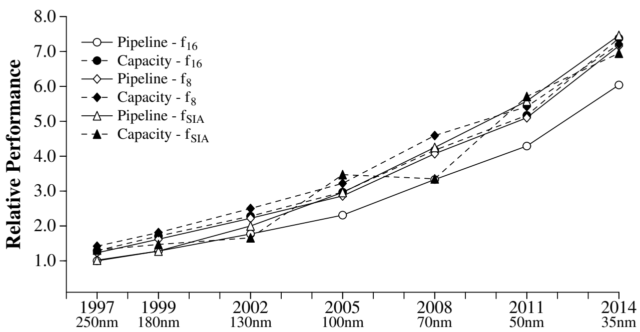

Figure 7: Performance increases for different scaling strategies.

Figure 7 graphs total performance scaling with advancing technology. The six lines represent pipeline or capacity scaling for each of the three clock scaling rates. All performance numbers represent the geometric mean of the SPEC95 benchmarks, and are normalized to our baseline.

Although the various scaling strategies perform differently for the intermediate technologies, the overall performance at 35nm is remarkably consistent across all experiments. Capacity scaling at f_{SIA} shows a high variance; this effect is due to the oscillating size of the L1 instruction cache, caused by the discretization issue described earlier. The pipeline scaling at f_{16} is the only strategy that performs qualitatively worse than the others. The f_{SIA} and f_{16} clocks differ by nearly a factor of three at 35nm, and yet the overall performance for both clocks at both scaling methodologies is nearly identical. The maximal performance increase is a factor of 7.4, which corresponds to a 12.5% annual improvement over that 17-year span.

While careful selection of the clock rate, specific structure size, and access penalty would result in small performance improvements above what we have shown here, the consistency of these results indicates that they would not be qualitatively superior. For a qualitative improvement in performance growth, microarchitectures significantly different than those we measured will be needed.

5 Conclusion

In this study, we examined the effects of technology scaling on wire delays and clock speeds, and measured the expected performance of a modern aggressive microprocessor core in CMOS technologies down to 35nm. We found that communication delays will become significant for global signals. Even under the best conditions, the latency across the chip in a top-level metal wire will be 12–32 cycles, depending on clock rate. In advanced technologies, the delay (in cycles) of memory oriented structures increases substantially due to increased wire latencies and aggressive clock rates. Consequently, even a processor core of today's size does not scale well to future technologies. In Figure 8, we compare our *best* measured performance over the next 14 years with projected scaling at recent historical rates (55% per year). While projected rates for 2014 show performance exceeding one TRIPS, our best-performing microarchitecture languishes at 6.5 BIPS. To reach a factor of thousand in performance improvement at an aggressive clock of 8 FO4 (10GHz in 35nm), a chip must sustain an execution rate of 150 instructions per cycle.

While our results predict that existing microarchitectures do not scale with technology, we have in fact been quite generous to potential microprocessor scaling. Our wire performance models conservatively assume very low-permittivity dielectrics, resistivity of pure copper, high aspect ratio wires, and optimally placed repeaters. Our models for structure access time further assume a hierarchical decomposition of the array into sub-banks, word-line routing in mid-level metal wires, and cell areas that do not depend on word-line wire width. In our simulation experiments of sample micoarchitectures, we further assumed that all structures could be perfectly pipelined, that routing delay between structures is insignificant, and that latch and clock skew overheads are negligible.

With these assumptions, the best performance we were able to obtain was a speedup of 7.4 from a 250nm chip to 35nm chip, which corresponds to an annual gain of 12.5%. Over the same period, the clock improves by either 12%, 17%, or 19% annually, depending on whether a clock period at 35nm is 16, 8, or 5.9 FO4 delays, respectively. That result means that the performance of a conventional, out-of-order microprocessor is likely to scale *worse* than the clock

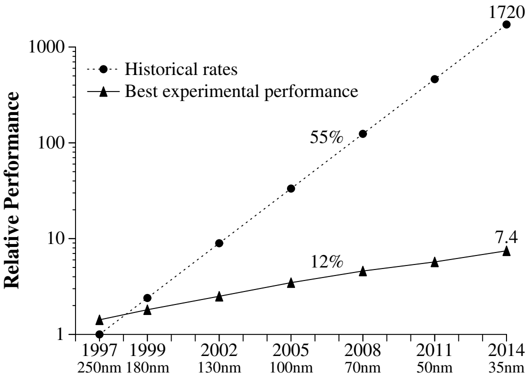

Figure 8: Projected performance scaling over a 17-year span for a conventional microarchitecture.

rate, even when given an effectively unlimited transistor budget. If any of the optimistic scaling assumptions of our models are not met, actual microprocessor scaling will be poorer than we report.

Our models show that dense storage structures will become considerably slower relative to projected clock rates, and will adversely affect instruction throughput. While structure access time remains effectively constant with the clock rate up to 70nm technologies, at 50nm and below, wire delays become significant. If clocks are scaled superlinearly relative to decreases in gate length, access times for these structures increases correspondingly. For example, when designing a level-one data cache in a 35nm technology, an engineer will be faced with several unattractive choices. First, the engineer may choose an aggressive target clock rate, and attempt to design a low access penalty cache. At the aggressive SIA projection of 13.5 GHz (which is likely unrealistic), even a single-ported *512 byte* cache will require three cycles to access. Second, the designer may opt for a larger cache with a longer access time. Given our conservative assumptions about cache designs, a 64KB L1 data cache would require at least seven cycles to access at the aggressive clock rate. Finally, the designer may choose a slower clock but a less constrained cache. At 5 GHz (16 FO4 delays), a 32KB cache can be accessed in two cycles.

None of these choices are ideal. The first two alternatives reduce IPC substantially, while the third incurs a 2.7-fold penalty in clock rate. Optimizing for any one of clock rate, pipeline depth, or structure size will force significant compromises in the other design points for future ultra-small gate-length technologies. While other work has proposed to vary the clock rate and effective structure capacity dynamically [2], those trade-offs are still within the context of a conventional microarchitecture, which is unscalable no matter which balance between clock rate and instruction throughput is chosen.

Based on this study, future microprocessors will be subject to much more demanding constraints than today if performance is to improve faster than technology scaling rates. We draw the following four conclusions from our results:

- Large monolithic cores have no long-term future in deep submicron fabrication processes. Microarchitectures that require increases in the sizes of their components—such as register files in wide-issue superscalar processors or high-performance SMT processors—will scale even more poorly than the microarchitecture used in this study.

- Of the structures we studied, the *on-chip* memory system is likely to be a major bottleneck in future processors. With aggressive clock rates, level-one caches that are not substantially smaller than those of today will have access times of three to seven cycles. For level-two caches with capacities up to 2MB, access delays ranged from thirty to fifty cycles with aggressive clock rates, even with ideal partitioning into sub-banks. For caches that use an even larger fraction of the die that the area of a 2MB structure, the access penalties will be substantially higher. Because future workloads will certainly have a larger memory footprint than SPEC95, the drop in IPC due to longer average memory access times will be larger than that measured in this study.

- Technology constraints for high-performance microprocessors will affect future designs much more so than those of today. Technology-based analysis will become a necessity for all architecture research. Distance, area, and delay models must be incorporated into the high-level performance simulations that are currently the norm. Research that fails to consider these factors will grow increasingly inaccurate as technology progresses.

- Future microprocessors must be partitioned into independent physical regions, and the latency for communicating among partitions must be exposed to the microarchitecture and possibly to the ISA. This observation is not new; a number of researchers and product teams have proposed or implemented partitioned architectures [8, 9, 10, 12, 15, 21, 25, 29]. However, many of these architectures use conventional communications mechanisms, or rely too heavily on software to perform the application partitioning. The best combination of static and dynamic communication and partitioning mechanisms, which lend themselves to the high-bandwidth, high-latency substrate, has yet to be demonstrated.

The results of this study paint a bleak picture for conventional microarchitectures. Clock scaling will soon slow precipitously to linear scaling, which will force architects to use the large transistor budgets to compensate. While it is likely that research innovations will allow conventional microarchitectures to scale better than our results show, we believe that the twin challenges—of recent diminishing returns in ILP and poor scaling of monolithic cores with technology—will force designers to consider more radical alternatives.

One challenge is to design new ways of mapping applications with varying granularities of parallelism onto the partitioned substrate, and to tolerate the variance in both application and total workload behavior with efficient dynamic mechanisms. The other key challenge is to design cores within each partition that can sustain high ILP at fast clock rates. These cores will need structures that maximize information density, different logical clocks for different components of the microarchitecture, and a physical layout that supports fast execution of multiple independent instructions along short wires.

Acknowledgments

Many thanks to the anonymous referees for their valuable suggestions on an earlier version of this paper. We thank Mark Horowitz for his technical comments on multiple drafts, and particularly for pointing out the realities of clock scaling. We also thank Ravi Rajwar and Jim Goodman for loaning us machine cycles, and other members of the CART group at UT for their help in setting up the experiments. This research is supported by IBM University Partnership Program awards for Doug Burger and Steve Keckler.

References

[1] Vikas Agarwal, Stephen W. Keckler, and Doug Burger. Scaling of microarchitectural structures in future process technologies. Technical Report TR2000-02, Department of Computer Sciences, The University of Texas at Austin, April 2000.

[2] David H. Albonesi. Dynamic ipc/clock rate optimization. In *Proceedings of the 25th Annual International Symposium on Computer Architecture*, pages 282–292, June 1998.

[3] B.S. Amrutur and M.A. Horowitz. Speed and power scaling of SRAMs. *IEEE Journal of Solid State Circuits*, 35(2):175–185, February 2000.

[4] Geordie Braceras, Alan Roberts, John Connor, Reid Wistort, Terry Frederick, Marcel Robillard, Stu Hall, Steve Burns, and Matt Graf. A 940MHz data rate 8Mb CMOS SRAM. In *Proceedings of the IEEE International Solid-State Circuits Conference*, pages 198–199, February 1999.

[5] Doug Burger. *Hardware Techniques to Improve the Performance of the Processor/Memory Interface*. PhD thesis, University of Wisconsin-Madison, December 1998.

[6] Doug Burger and Todd M. Austin. The simplescalar tool set version 2.0. Technical Report 1342, Computer Sciences Department, University of Wisconsin, June 1997.

[7] Doug Burger, Alain Kägi, and M.S. Hrishikesh. Memory hierarchy extensions to simplescalar 3.0. Technical Report TR99-25, Department of Computer Sciences, The University of Texas at Austin, April 2000.

[8] Keith Diefendorff. Power4 focuses on memory bandwidth. *Microprocessor Report*, 13(13), October 1999.

[9] Marco Fillo, Stephen W. Keckler, William J. Dally, Nicholas P. Carter, Andrew Chang, Yevgeny Gurevich, and Whay S. Lee. The M-Machine Multicomputer. In *Proceedings of the 28th International Symposium on Microarchitecture*, pages 146–156, December 1995.

[10] Lance Hammond, Basem Nayfeh, and Kunle Olukotun. A single-chip multiprocessor. *IEEE Computer*, 30(9):79–85, September 1997.

[11] Mark Horowitz, Ron Ho, and Ken Mai. The future of wires. In *Seminconductor Research Corporation Workshop on Interconnects for Systems on a Chip*, May 1999.

[12] R.E. Kessler. The alpha 21264 microprocessor. *IEEE Micro*, 19(2):24–36, March/April 1999.

[13] David Kroft. Lockup-free instruction fetch/prefetch cache organization. In *Proceedings of the Eighth International Symposium on Computer Architecture*, pages 81–87, May 1981.

[14] S. R. Kunkel and J. E. Smith. Optimal pipelining in supercomputers. In *Proceedings of the 13th Annual International Symposium on Computer Architecture*, pages 404–411, June 1986.

[15] K. Mai, T. Paaske, N. Jayasena, R. Ho, and M. Horowitz. Smart memories: A modular reconfigurable architecture. In *Proccedings of the 27th Annual International Symposium on Computer Architecture*, June 2000.

[16] Doug Matzke. Will physical scalability sabotage performance gains? *IEEE Computer*, 30(9):37–39, September 1997.

[17] S. Naffziger. A subnanosecond $0.5\mu m$ 64b adder design. In *Digest of Technical Papers, International Solid-State Circuits Conference*, pages 362–363, February 1996.

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

[19] Glenn Reinman and Norm Jouppi. Extensions to cacti, 1999. Unpublished document.

[20] Scott Rixner, William J. Dally, Brucek Khailany, Peter Mattson, Ujval J. Kapasi, and John D. Owens. Register organization for media processing. In *Proceedings of the Sixth International Symposium on High-Performance Computer Architecture*, January 2000.

[21] Eric Rotenberg, Quinn Jacobson, Yiannakis Sazeides, and Jim Smith. Trace processors. In *Proceedings of 30th Annual International Symposium on Microarchitecture*, pages 138–148, December 1997.

[22] The national technology roadmap for semiconductors. Semiconductor Industry Association, 1999.

[23] Hiroshi Shimizu, Kenji Ijitsu, Hideo Akiyoshi, Keizo Aoyama, Hirotaka Takatsuka, Kou Watanabe, Ryota Nanjo, and Yoshihiro Takao. A 1.4ns access 700MHz 288Kb SRAM macro with expandable architecture. In *Proceedings of the IEEE International Solid-State Circuits Conference*, pages 190–191, 459, February 1999.

[24] Gurindar S. Sohi. Instruction issue logic for high-performance, interruptible, multiple functional unit, pipelined computers. *IEEE Transactions on Computers*, 39(3):349–359, March 1990.

[25] Gurindar S. Sohi, Scott E. Breach, and T. N. Vijaykumar. Multiscalar processors. In *Proceedings of the 22nd Annual International Symposium on Computer Architecture*, pages 414–425, June 1995.

[26] Standard Performance Evaluation Corporation. *SPEC Newsletter*, September 1995.

[27] Dennis Sylvester and Kurt Keutzer. Rethinking deep-submicron circuit design. *IEEE Computer*, 32(11):25–33, November 1999.

[28] A. J. van Genderen and N. P. van der Meijs. Xspace user's manual. Technical Report ET-CAS 96-02, Delft University of Technology, Department of Electrical Engineering, August 1996.

[29] Elliot Waingold, Michael Taylor, Devabhaktuni Srikrishna, Vivek Sarkar, Walter Lee, Victor Lee, Jang Kim, Matthew Frank, Peter Finch, Rajeev Barua, Jonathan Babb, Saman Amarasinghe, and Anant Agarwal. Baring it all to software: Raw machines. *IEEE Computer*, 30(9):86–93, September 1997.

[30] Steven J.E. Wilton and Norman P. Jouppi. An enhanced access and cycle time model for on-chip caches. Technical Report 95/3, Digital Equipment Corporation, Western Research Laboratory, 1995.

[31] Cangsang Zhao, Uddalak Bhattacharya, Martin Denham, Jim Kolousek, Yi Lu, Yong-Gee Ng, Novat Nintunze, Kamal Sarkez, and Hemmige Varadarajan. An 18Mb, 12.3GB/s cmos pipeline-burst cache SRAM with 1.54Gb/s/pin. In *Proceedings of the IEEE International Solid-State Circuits Conference*, pages 200–201,461, February 1999.