

# Cache Implications of Aggressively Pipelined High Performance Microprocessors

Timothy J. Dysart, Branden J. Moore, Lambert Schaelicke, Peter M. Kogge  
Department of Computer Science and Engineering  
University of Notre Dame  
384 Fitzpatrick Hall  
Notre Dame, IN 46556, USA  
Telephone: 574-631-8320  
{tdysart,bmoore,lambert,kogge}@cse.nd.edu

## Abstract

*One of the major design decisions when developing a new microprocessor is determining the target pipeline depth and clock rate since both factors interact closely with one another. The optimal pipeline depth of a processor has been studied before, but the impact of the memory system on pipeline performance has received less attention. This study analyzes the affect of different level-1 cache designs across a range of pipeline depths to determine what role the memory system design plays in choosing a clock rate and pipeline depth for a microprocessor. The pipeline depths studied here range from those found in current processors to those predicted for future processors. For each pipeline depth a variety of level-1 cache sizes are simulated to explore the relationship between clock rate, pipeline depth, cache size and access latency. Results show that the larger caches afforded by shorter pipelines with slower clocks outperform longer pipelines with smaller caches and higher clock rates.*

## 1. Introduction

Determining the target clock frequency and pipeline organization are some of the most critical decisions to be made during the design of a microprocessor. Both aspects are tightly intertwined with semiconductor technology, workload characteristics, memory performance and power consumption. Long pipelines reduce the amount of work per stage to achieve high clock rates and result in high instruction throughput. On the other hand, insufficient instruction level parallelism as well as practical limits on the minimum amount of work performed per stage limit the pipeline depth in practical implementations. Depending on the clock rate, microarchitectural structures of a given size

will exhibit different access latencies. One fundamental structure with significant impact on overall performance is the level-1 cache. Generally, small caches are faster, but the higher miss rate may offset the performance gain from the clock rate increase. This paper presents a study that trades cache size for access latency for a wide variety of pipeline depths.

The optimal pipeline depth of a microprocessor has been explored before in a variety of contexts, but the influence of the memory hierarchy in this tradeoff has received less attention. The study presented here is designed to close that gap by varying pipeline depths, level-1 cache sizes and access latencies. In particular, there are two endpoints of interest. Short pipelines operate at lower clock rates and afford larger level-1 caches than deep, high clock rate pipelines. To explore the tradeoff between pipeline depth and cache access latency, pipeline depth and level-1 instruction and data cache size are varied independently. For each pipeline depth, a projected clock rate is calculated. Although a specific 65 nm process is used for these projections, the underlying methodology is technology-independent and hence the results of this work are generally applicable. For each clock period, access latencies in cycles are computed for various level-1 cache sizes. Each of the resulting configurations is simulated running SPEC2000 benchmarks. The primary performance measure in this context is the number of instructions completed per second (IPS). Unlike miss rates or IPC, this metric accounts for different cycle times and represents predicted absolute performance.

Results show that performance decreases for longer pipelines. A short 7-stage pipeline consistently outperforms all other organizations. Although operating at slower clock rates, it allows large caches to be accessed with low latency. The higher clock rate of deeper pipelines is not able to compensate for the lower hit rates of smaller caches, in part due to the relatively modest degree of instruction level parallelism found in the benchmarks.

Section 2 discusses related work. The simulation methodology is described in Section 3. Section 4 discusses the method to derive clock frequencies, pipeline depth and cache characteristics from technology projection. The results are presented in Section 5 and Section 6 draws conclusions and outlines future work.

## 2. Related Work

Several groups have analyzed the problem of determining the ideal pipeline length. Kunkel and Smith analyze the Cray-1S processor [9]. Hartstein and Puzak develop a formula and analyze the results of different workload types to determine the optimum depth [5]. Similarly, Sprangle and Carmean start with a Pentium IV type architecture, and determine that the optimal pipeline depth is greater than the twenty stage pipeline currently implemented [13]. Agarwal *et al.* show that processor speedups will not continue at the same rate due to the wire delays in semiconductor devices and slowing processor clocks [1]. Lastly, Hrishikesh *et al.* take a slightly different approach and determine the optimum amount of logic for a pipeline stage independent of the length of the pipeline [7].

Hartstein and Puzak [5] take an analytical approach and develop a formula to determine the optimum pipeline depth based on the total logic depth, latch overhead, width, workload, and hazard penalties of the processor. Intuitively, reducing the latch overhead, hazard penalties, and hazards in the workload will allow for a deeper processor. Increasing the width of the processor will reduce the optimal pipeline depth since hazards occur more frequently. This formula is validated with simulation results that vary each of the different parameters in the formula individually. Interestingly, each type of workload has its own optimal pipeline depth. These optimal depths range from 13 to 36 stages with a nearly Gaussian distribution. Cache miss rates and miss penalties represent one aspect of hazard frequency and penalty, but are not separately addressed in this work.

Sprangle and Carmen analyze many of the same aspects [13], with a particular emphasis on the effect of branch misprediction. This work also identifies the different effects of various workloads on processor performance which are mainly due to branch behavior. This work does not directly consider memory hierarchy effects, but identifies the assumption of a fixed cache latency as a modeling limitation.

Agarwal *et al.* predict that the 50 – 60% annual processor improvement seen over the past decade will soon cease [1]. Instead, the maximum annual performance improvement is predicted to shrink to 12.5%. These improvements are attributed to a combination of pipeline depth scaling and structure capacity scaling. This study identifies on-chip memory as a significant bottleneck, and predicts that monolithic processor cores will become inefficient.

The approach taken by Hrishikesh *et al.* in [7] is similar to that taken by Kunkel and Smith in that it determines the optimal amount of work that should be done during each pipeline stage. To compensate for changes in manufacturing technology, logic depth in terms of fan-out-of-four (FO4) is used. This metric removes the fabrication process as a variable from consideration. Using the Alpha 21264 as a baseline processor, Hrishikesh *et al.* vary the number of stages needed for each component in the pipeline. The number of stages is determined by dividing the total amount of logic in that component by the amount of logic in FO4s per stage. This results in an optimal amount of logic per stage of 6 FO4's for integer code and 4 FO4's for vector floating point code. These results translate into a pipeline of about thirty stages in length.

Combined, previous studies provide a comprehensive view of many pipeline design aspects. However, the interaction of the memory system with pipeline depth and clock frequency has not received full attention. The work presented here is designed to quantify this effect by varying the size of both level-1 caches and adjusting access latencies based on technology projections for various pipeline depth and clock frequencies. Additionally, the sensitivity of the observed trends to the level-2 cache organization and branch prediction scheme is analyzed.

## 3. Methodology

Using SimpleScalar 3.0 [3] and the SPEC2000 benchmark suite [6], an environment is created to explore the interaction between pipeline depth, cache size and latency. SimpleScalar is a flexible execution-driven cycle-level processor simulation tool that has been modified for this study to support varying pipeline depth. The SPEC2000 benchmarks are a commonly used and well-understood suite of computationally demanding performance benchmarks.

Simulations are run for a number of pipeline depths ranging from 7 to 50 stages, thus representing current as well as predicted future organizations. For a given process technology and an estimated total amount of logic, a maximum clock frequency can be calculated for each pipeline depth, as further described in Section 4.1. Then, for a variety of level-1 cache sizes ranging from 1 kbyte to 512 kbyte, access latencies in cycles are calculated. Note that although absolute cache access times are independent of pipeline depth, the number of cycles until a load value can be used changes due to varying clock frequencies.

### 3.1. Baseline Processor Configuration

All aspects of the simulated processor except pipeline depth and the level-1 cache configuration are kept constant across the experiments, thus isolating the interaction effect

of these two factors from other variables. The baseline system is closely related to the Alpha 21264, which is a modern dynamically-scheduled superscalar microprocessor, capable of decoding, dispatching and graduating four instructions per cycle. The simulated processor contains four integer ALUs, one integer multiply / divide unit, four floating point ALUs, and one floating point multiply / divide unit. All of these functional units are fully pipelined. Their operational latencies are varied according to pipeline depth, and are further discussed in Section 4.2.

This processor organization corresponds to a number of currently available microprocessors but may be aggressive for deeply-pipelined, high-frequency designs. Conditional branches are predicted by a 2-level branch predictor, configured similarly to the local branch prediction strategy discussed in [11]. The performance of the branch prediction scheme is discussed in Sec. 5.4. The predictor is sized with 8192 entries and 512 entries for the level one and level two predictors, respectively. Both predictors keep eight bits of history. These values are used because of their reasonable size and high prediction rate. The Register Update Unit (RUU) is 512 entries deep, and the Load/Store Queue holds 64 instructions. Both structures are intentionally kept large to minimize the number of structural stalls.

The processor uses 4-way set associative level-1 instruction and data caches with 32 byte blocks. The size and latency of both caches is varied with the pipeline depth. The level-2 cache is held constant at 2 Mbyte, 4-way set associative, with 64 byte blocks. The main memory bus is 8 bytes wide, and its speed is held constant at 800MHz. The access times for all caches and main memory, along with the level-1 cache sizes are discussed in Section 4.1.

### 3.2. Simulator Modifications

In order to simulate a number of different pipeline depths, modifications to two parts of the Sim-OutOrder simulator from SimpleScalar version 3.0 [3] are needed.

The issue latency of all functional units is set to one cycle, thus representing fully-pipelined units. The operational latency determines how long it takes to complete a single instruction and have the result ready for use in another instruction. By varying the operational latency of the functional units, the pipeline length of the simulated processor is increased.

In addition to increasing functional unit latency, changes made to the Decode/Dispatch stages of the pipeline allow a configurable “front-end” length to the pipeline. The Decode and Dispatch stages in the standard Sim-OutOrder simulator are implemented as a single stage. By splitting this function into two functions (Decode and Dispatch), and inserting a queue to hold instructions between these functions, the length of these functions can be increased to approximate

a variable-length pipeline. The Decode stage contains the instruction decode logic and the branch predictors, while placement of the instruction into the RUU and LSQ is delayed until the Dispatch stage. Since the branch prediction logic is in the early Decode stage, before instructions are put on the queue, the simulator is more optimistic with branch prediction than real processors might be.

Generally, this simulation model is overly optimistic about the ability to pipeline all parts of the processor. Since this modeling error affects deep pipelines more than shallow ones, simulations likely overestimate the performance benefits of deep pipelines.

### 3.3. Benchmarks

The SPEC CPU2000 benchmark suite [6] is one of the most commonly used set of benchmarks to evaluate processor and system performance on computationally intensive workloads. It is well understood by researchers and generally considered representative of scientific and engineering workloads for workstations. This study considers both integer and floating point benchmarks. All benchmarks are compiled with the SimpleScalar port of gcc, version 2.6.3. Due to compiler and simulation limitations, four integer programs (175.vpr, 181.mcf, 186.crafty, 252.eon) and six floating point programs (178.galgel, 187.facerec, 188.ammp, 189.lucas, 191.fma3d, 200.sixtrack) are not included, because they can not be compiled by the SimpleScalar tool chain.

The remaining 16 benchmarks (8 integer, 8 floating point) are run for each of the processor configurations discussed in Sections 4.1 and 4.2. Using this number of benchmark programs provides a broad range of workloads that fairly compare the performance of the various processor configurations. All simulations are “fast-forwarded” 500 million instructions and then run for 1 billion instructions.

## 4. Estimating Pipeline Organizations

In order to vary the amount of logic in each stage of the pipeline, the total logic length of the pipeline must be determined. Dividing this measure by the number of pipeline stages results in the logic depth per pipeline stage. Thus, using the results from [7], the Alpha 21264 has a seven stage pipeline and a clock period that is equivalent to 17.4 FO4’s. Included in this period length is 1.8 FO4’s of pipeline overhead. Subtracting the overhead results in a usable logic depth of 15.6 FO4’s per stage, and a total logic depth of 109.2 FO4’s to process an instruction.

To determine the resulting logic depth per pipeline stage, and thus the maximum clock frequency, the total logic depth is divided by the number of stages and the pipeline overhead of 1.8 FO4’s is added to each stage. From the number

**Table 1. Tested pipeline depths, FO4’s per stage, period lengths, and clock frequencies.**

Pipeline Depth	FO4’s per stage	Period Length (ns)	Clock Rate in GHz
7	17.4	.2192	4.562
10	12.72	.1603	6.238
15	9.08	.1144	8.741
20	7.26	.09148	10.931
25	6.17	.07774	12.863
30	5.44	.06854	14.59
40	4.53	.05708	17.519
50	3.98	.05015	19.940

of FO4 delays per stage, the clock period can be computed for a given CMOS process technology by multiplying the drawn gate length by 360 picoseconds [7]. This work uses an aggressive 65 nm fabrication process generation with a drawn gate length of 35nm for high-performance microprocessors estimated in the Semiconductor Industry Association Roadmap [2]. Under these assumptions, one FO4 delay corresponds to 12.6 ps.

Table 1 summarizes the pipeline depths considered in this work, as well as the resulting per-stage logic depth and clock frequency. The SIA roadmap suggests a clock frequency of 6.739GHz for the 65 nm fabrication process, which is roughly equivalent to an 11 stage pipeline using the method developed in this work. Note that this approach to calculating pipeline depth and clock frequencies is overly optimistic in several aspects. It assumes that the processor logic can be perfectly pipelined to any depth, and it is based on technology projections rather than established processes. Both assumptions lead to an overestimate of the benefits of deep pipelines.

Given that both FO4 delays and the scaling model that Cacti [14] uses to estimate cache access latencies are process independent, only one process technology needs to be considered. Although clock speeds will increase as the fabrication process becomes smaller and more mature, performance trends remain.

#### 4.1. Memory Hierarchy Calculations

Cacti was originally developed by Wilton and Jouppi as a cache access timing model that is capable of power and area modeling as well [14]. This work uses the most recent version of Cacti, 3.2, to determine the access time for all cache designs in this study [12]. Since block size and associativity do not impact access time, they are held constant. For example, for an 8 kbyte cache in a 50 stage pipeline, access time changes by only one processor clock cycle when varying block size between 32 and 64 bytes and associativity between two and eight. Similar behavior is seen for the

**Table 2. Pipeline depth, period length, calculated access latency, and rounded access latency in cycles for a 32 kbyte cache.**

Pipeline Depth	Period Length (ns)	Calculated Latency	Rounded Latency
7	.2192	2.355	2
10	.1603	3.222	3
15	.1144	4.514	4
20	.09148	5.645	6
25	.07774	6.643	7
30	.06854	7.534	8
40	.05708	9.047	9
50	.05015	10.297	10

associativity of the level-2 cache. This study uses split 4-way set associative level-1 instruction and data caches with 32 byte blocks and a unified 2 Mbyte, 4-way set associative level-2 cache with 64 byte blocks.

For each pipeline depth, clock periods are determined using the method described previously. By dividing the access time by the clock period, a latency in cycles for the cache structure is then determined. For example, for a 64 kbyte cache in the 65 nm process, the latency is calculated to be 11.81 cycles for a 50 stage pipeline. The same cache in the 90 nm process has an access time of 0.79644 ns, which is equivalent to 10.49 cycles. Similarly, for an 8 kbyte cache at the same pipeline depth, the latency is 8.99 cycles in the 65 nm process and 8.23 cycles for the 90 nm fabrication processes. These observations confirm that Cacti’s access latency predictions are process-independent.

As an example, Table 2 shows the latency in cycles for a fixed 32 kbyte cache across all pipeline depths. Cacti calculates a 0.51641 ns access latency in a 65 nm process. For each pipeline configuration, the access time is divided by the processor cycle time, and the resulting access latency is rounded to the nearest integer value. Arithmetic rounding is used to compensate for possible advances in cache designs and process technology, while possibly overestimating the cache size and speed. This calculation is then repeated at each pipeline depth for cache sizes ranging from 1 kbyte to 512 kbyte.

Table 3 shows the calculated and rounded latencies, in cycles, for a 15-stage pipeline across all considered cache sizes. Using a 65 nm process, this pipeline is estimated to run with a 0.1144 ns cycle time. The reported cache sizes demonstrate that multiple cache configurations may exhibit the same access latency in cycles. In this case, only the largest cache is considered for further evaluation. Note that the access latency for three cache sizes (1, 2 and 32 kbyte) is rounded down to compensate for possible modeling error and to allow for implementation-specific latency optimizations such as sum-addressed memory [10]. All other

**Table 3. Cache sizes, access times estimated by Cacti and access latency in cycles for a 15 stage pipeline.**

Cache Size (KB)	Access Time (ns)	Calculated Latency	Rounded Latency
1	0.40883	3.574	3
2	0.41797	3.654	3
4	0.43826	3.831	4
8	0.45092	3.942	4
16	0.47535	4.155	4
32	0.51641	4.514	4
64	0.59522	5.203	5
128	0.68278	5.968	6
256	0.85591	7.482	7
512	1.49065	13.030	13

latencies are derived via normal arithmetic rounding.

Finally, Table 4 summarizes all pipeline and cache configurations considered for evaluation.

The 2 Mbyte, 4-way set associative, 64 byte block level-2 cache has an access time of 3.37543 ns according to Cacti. With normal rounding, an L2 access takes 15, 21, 29, 37, 43, 49, 59, and 67 cycles respectively for pipeline depths of 7, 10, 15, 20, 25, 30, 40, and 50 stages. The latency for an L2 miss is calculated based on an estimated 90 ns main memory access. Dividing that latency by the clock rate of the respective pipeline, level-2 miss penalties are calculated to be 411, 561, 787, 984, 1158, 1313, 1577, and 1795 cycles for the first word of a cache line. At a bus speed of 800 MHz, each subsequent part of a cache line takes 6, 8, 11, 14, 16, 18, 22, and 25 processor cycles.

## 4.2. Functional Unit Latencies

The modified simulation model attributes pipeline depth changes to two distinct parts of the processor: instruction decode and execution. Based on published pipeline organizations [7] approximately 60% of the processing are attributed to the execution stage, and the remaining 40% are assigned to the decode logic. Branch prediction, decoding and register renaming table are placed in the front-end, while the issue window, register file, and execution units are placed into the back-end. Using the above-mentioned ratio, the target pipeline depth is divided into a number of front-end and back-end stages. Since the simulation model does not allow a uniform lengthening of the pipeline, latencies of the execute stage and modified decode stage are changed to approximate pipelines of the desired depth.

Functional unit latencies are scaled for deeper pipelines based on published data from the Alpha 21264, with integer addition as baseline. For each pipeline depth, the latency for an integer addition is computed and the latencies of other functional units are scaled based on the original ratio. Since

**Table 4. Summary of all pipeline and level-1 cache configurations tested.**

Depth	Cache Size (KB)	Latency (cycles)
7	32	2
	128	3
	256	4
	512	7
10	32	3
	128	4
	256	5
15	2	3
	32	4
	64	5
	128	6
20	1	4
	16	5
	64	6
	128	7
25	2	5
	16	6
	32	7
	128	8
30	4	6
	16	7
	32	8
	64	9
40	2	7
	16	7
	32	9
	64	10
50	2	8
	16	9
	32	10
	64	12

the Alpha 21264 does not have a functional unit for integer division, the SimpleScalar default value of 20 cycles is used as the base latency. Table 5 summarizes the latencies of all functional units for the different pipeline depths.

## 5. Results

The results express performance in billions of instructions per second (BIPS). The data points in the graphs are the harmonic means of the respective integer and floating point benchmark program results. Additionally, a sensitivity analysis is run to analyze how stable results are as the level-2 cache and branch predictor configuration changes. The reader is reminded here that the level-1 cache sizes listed reflect the respective size of the instruction and data level-1 caches, not the sum of these caches.

### 5.1. Instruction Completion Rate

Ultimately, the speed of a processor is determined by the amount of work completed in a given time period. Figures

**Table 5. Decode and functional unit latencies for each pipeline depth.**

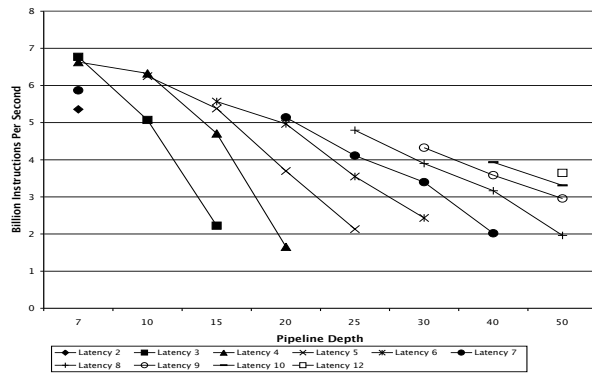
Pipeline Depth	Num. Decode Stages	Num. Int. Add Stages	Int. Mult. Latency	Int. Div Latency	FP Add/Mult. Latency	FP Div. Latency
7	1	1	7	20	4	12
10	2	3	13	33	8	20
15	4	6	21	53	13	33
20	6	9	27	64	18	41
25	8	12	33	78	22	50
30	10	15	41	99	28	63
40	14	21	56	132	39	85
50	18	27	79	193	53	123

1 and 2 show the performance of all configurations tested in terms of Instructions per Second. This metric is computed by multiplying the number of cycles required to complete 1 billion instructions with the cycle time for each pipeline organization, normalized to one second of real time. Note that both figures contain the same data, but with the data points grouped differently to highlight distinct trends. Each data point represents the harmonic mean of all benchmarks.

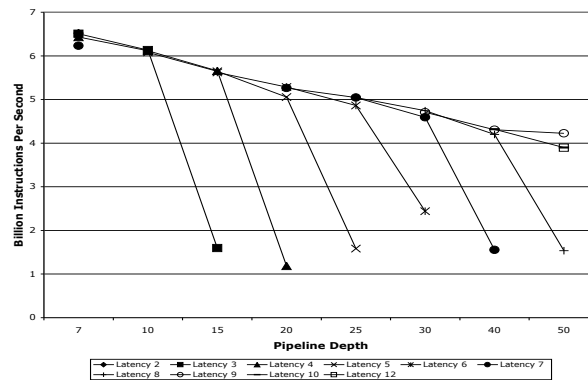
Overall, the results clearly show that the higher clock rates of deeply pipelined architectures are not able to compensate for the smaller caches that are needed to maintain acceptable access latencies. The shortest pipeline consistently outperforms all other organizations with almost identical results for all cache sizes tested. On the other hand, the slowest organization is found in a 20-stage pipeline with 1 kbyte caches, with many deeper pipelines performing significantly better.

Figure 1 connects data points with a constant level-1 cache access latency. This representation shows how deepening the pipeline while maintaining a constant access latency in cycles causes a drastic reduction in performance. The increasing clock frequency requires the use of smaller caches, and the higher clock rates are not able to compensate for the resulting higher miss rates. In fact, in all cases larger caches with a higher access latency result in a performance gain for a given pipeline depth and clock rate. The higher cache hit rates outweigh the penalty of a longer access latency. This effect is particularly pronounced for deep, high clock rate pipelines.

Figure 2 shows the same data as discussed before, but with data points for identical cache sizes connected. This representation provides a visual record of the decreasing size of the caches as pipelines are stretched to maintain reasonable hit latencies. For a fixed cache size, access latency grows as the pipeline is deepened, due to higher clock rates. Despite consistent hit rates, the longer access latency counters the benefit of faster clocks and degrades overall performance.

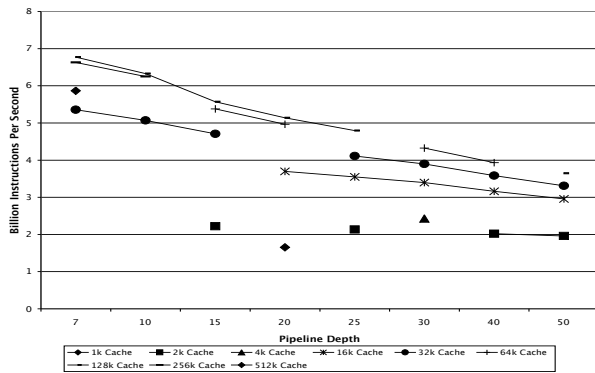


(a) Integer Benchmarks

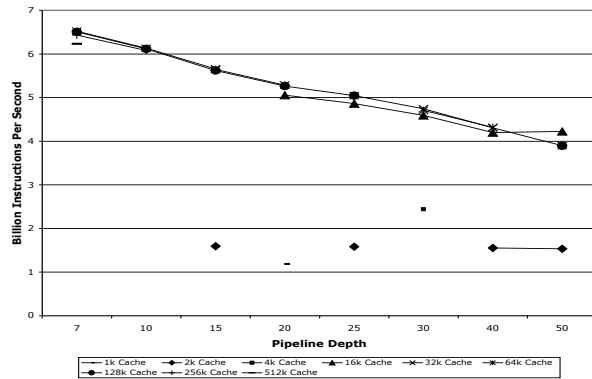


(b) Floating Point Benchmarks

**Figure 1. Instructions Per Second versus Pipeline Depth, Data Points Connected by Cache Access Latency**

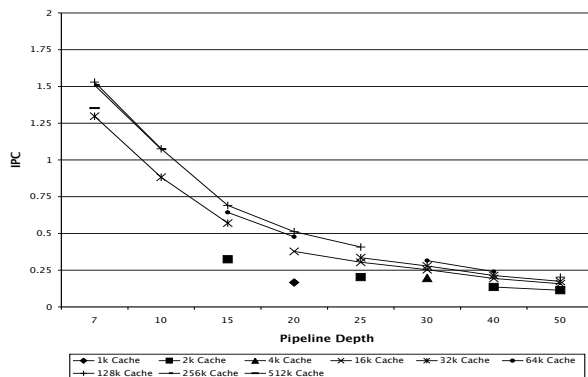


(a) Integer Benchmarks

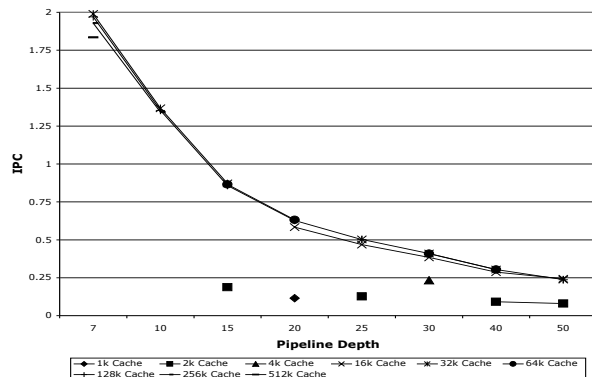


(b) Floating Point Benchmarks

**Figure 2. Instructions Per Second versus Pipeline Depth, Data points Connected by Cache Size**



(a) Integer Benchmarks



(b) Floating Point Benchmarks

**Figure 3. Instructions Per Cycle versus Pipeline Depth**

## 5.2. Instruction-level Parallelism

Instructions-per-cycle (IPC) is a commonly used processor performance measure that is independent of the implementations cycle time. On the 4-way superscalar processor modeled here, the theoretical maximum IPC is four. In reality, this peak value is negatively affected by various stall conditions caused for instance by branch mispredictions, cache misses and instruction dependencies.

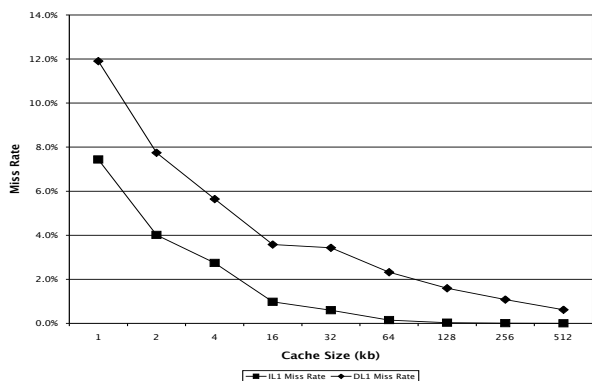
Figure 3 shows the measured IPC for each pipeline depth. As expected, IPC decreases as pipelines get deeper, since longer pipelines exhibit larger branch misprediction penalties. Furthermore, increasing functional unit latencies expose more instruction dependencies. Finally, deeper pipelines with higher clock rates lead to smaller caches with

higher access latencies, thus increasing the average memory latency.

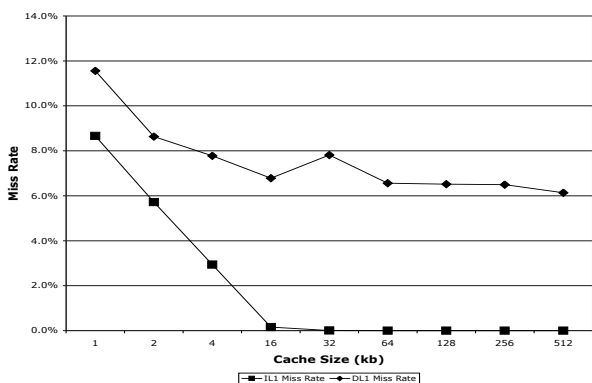
While a decreasing IPC is expected for deeper pipelines, results from the previous section show that the higher clock rate is not able to compensate for the lower IPC.

## 5.3. Level-1 Cache Miss Rate

Unlike other studies on the effect of pipeline depth on performance, this study varies the level-1 instruction cache configuration along with the data cache. Figure 4 shows both the level-1 instruction and data cache miss rates. As expected, smaller caches result in rapidly growing miss rates for both instructions and data. These results explain in part the decreasing IPC of deep pipelines, and confirm



(a) Integer Benchmarks



(b) Floating Point Benchmarks

**Figure 4. Level-1 Instruction and Data Cache Miss Rates**

that high cache miss rates can counter the benefit of faster clock rates.

#### 5.4. Branch Prediction Accuracy

Section 3.1 discussed the branch prediction scheme used by this study while Sec. 3.2 discussed how the prediction logic is placed early in the pipeline to provide a higher prediction rate, and minimize branch misprediction penalties. Using these methods, an average branch prediction accuracy (both directional and address) of 95% is achieved for the floating point benchmarks, while the directional accuracy for the integer benchmarks is 94% and the address accuracy is 91% - 92%. These prediction rates are unchanged across all pipeline depths, which practically eliminates the

branch prediction scheme as a variable in simulation. However, the branch misprediction delay still exist, and scales with processor depth.

#### 5.5. Level-2 Cache Miss Rates

Table 6 shows the level-2 cache miss rate (memory accesses divided by level-2 cache misses) as a function of the level-1 cache size and pipeline depth. The general trend for the integer benchmarks shows that the miss rate decreases as the level-1 cache size grows and as the pipeline depth increases. For the floating point benchmarks, the general trend is the same for increasing pipeline depths, but that the miss rate is approximately the same for all level-1 cache sizes. For reference, the total number of memory access for the integer benchmarks ranges from 1.92 billion to 4.52 billion, while the floating point benchmarks have from 1.5 billion to 3.28 billion memory accesses. These low miss rates show that the latency to access main memory has little effect on the performance of these benchmarks.

#### 5.6. Sensitivity Analysis

Two major architectural components that can have a substantial effect on the performance of a microprocessor are the branch prediction scheme and the level two cache. As others have pointed out, deep pipelines are more sensitive to branch prediction behavior [5, 13]. In the experiments performed here, the overall branch prediction accuracy is above 90 percent, due to the aggressive prediction scheme modeled. To test the sensitivity of the results to the branch prediction scheme, a perfect branch predictor is tested on pipeline depths of 7, 15, 25, and 50. Table 7 shows that a perfect branch predictor improves instruction throughput. However, the trend of shorter pipelines outperforming longer pipelines remains unchanged.

**Table 7. Sensitivity of results to the branch prediction scheme.**

Pipeline Depth	BIPS Perfect Prediction	BIPS Non-Perfect Prediction
7	8.0238	7.0863
15	6.9314	5.9623
25	5.9562	5.0939
50	4.5718	3.8868

The level-2 cache size and access latency can have a significant impact on performance, especially for low level-1 hit rates. All results presented here are based on a 2 Mbyte 4-way set-associative level-2 cache with an access latency of 3.37 ns. To explore the impact of this assumption on

**Table 6. Level-2 Cache Miss Rates**

Level-1 Cache Size (KB)	Depth 7		Depth 10		Depth 15		Depth 20	
	Int	FP	Int	FP	Int	FP	Int	FP
1							0.0177%	0.2935%
2					0.0172%	0.3343%		
4								
16							0.0138%	0.3069%
32	0.0221%	0.4710%	0.0180%	0.4036%	0.0149%	0.3451%		
64					0.0145%	0.3444%	0.0130%	0.3050%
128	0.0216%	0.4686%	0.0174%	0.4022%	0.0142%	0.3438%	0.0127%	0.3047%
256	0.0210%	0.4640%	0.0171%	0.3985%				
512	0.0195%	0.4583%						
Level-1 Cache Size (KB)	Depth 25		Depth 30		Depth 40		Depth 50	
	Int	FP	Int	FP	Int	FP	Int	FP
1								
2	0.0149%	0.2769%			0.0133%	0.2306%	0.0126%	0.2142%
4			0.0134%	0.2600%				
16	0.0128%	0.2829%	0.0120%	0.2605%	0.0110%	0.2330%	0.0103%	0.4370%
32	0.0123%	0.2813%	0.0116%	0.2592%	0.0106%	0.2317%	0.0100%	0.2146%
64			0.0113%	0.2592%	0.0104%	0.2315%		
128	0.0118%	0.2808%					0.0095%	0.2142%
256								
512								

the results, Figure 5 shows instruction throughput for four pipeline organizations as the level-2 cache size is varied from 256 kbyte to 16 Mbyte while the level-1 cache is held constant at 32 kbyte. Smaller level-2 caches allow faster access, but incur higher miss rates. The graphs show that to a point, larger level-2 caches improve performance, despite higher access latencies. Once this point is reached though there is a performance dropoff. However, the key point to notice is that the main conclusion of this work is unaffected by level-2 cache organization.

Previous work has shown that the miss rates vary only slightly (less than 1%) between a 4-way and fully associative, unified 1MB, 64 byte block level-1 cache [4]. As this study uses a unified 2MB, 64 byte clock level-2 cache, changing the associativity is expected to have almost no impact on performance, and is not presented here.

Other assumptions such as the ability to perfectly pipeline all parts of the processor and the optimistically modeled one-cycle branch prediction scheme give a performance advantage to deeper pipelines. Consequently, the observed trends are expected to hold under different, more realistic assumptions.

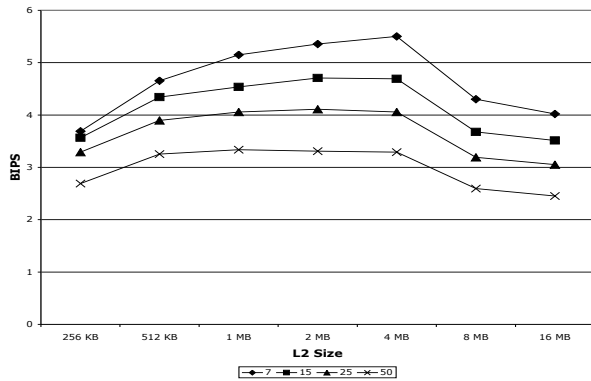
## 6. Future Work and Conclusions

There are several avenues for fruitful future work in this direction. First, a continued device analysis will be needed to determine if the common use of FO4 delays as a process independent method of scaling is accurate for aggressive, deep-submicron technologies. Advances in processor mi-

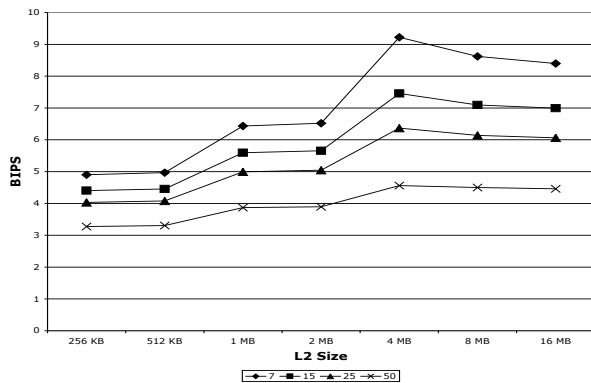
croarchitecture, including aggressive value and dependency speculation require more accurate modeling tools, but also broaden the spectrum of design decisions that affect overall performance. Furthermore, other important workloads such as commercial and e-commerce applications or large-scale scientific workloads likely exhibit different characteristics [5]. Finally, advances in VLSI design techniques and cache implementations, such as those suggested in [8], the use of a deeper hierarchy (such as three levels of on-chip cache) and pipelined caches, can have a significant impact on performance. Although this study focuses strictly on performance, area and power are significant issues facing modern processor design that must also be considered during the design of a microprocessor.

The simulation results show that, in general, lengthening the pipeline of a microprocessor tends to reduce its performance. The higher clock rates of deeper pipelines imply smaller level-1 caches. The resulting higher miss rates are not compensated for by the higher clock rates, resulting in a performance degradation for deeper pipelines. A sensitivity analysis with regard to level-2 cache organizations, and branch prediction schemes confirms that the observed trends are stable. By focusing on the interaction between pipeline depth and frequency with level-1 cache performance, this study complements previous work on pipeline depth tradeoffs, and provides insights into another important aspect of processor design.

The authors wish to thank the anonymous reviewers for their helpful comments.



(a) Integer Benchmarks



(b) Floating Point Benchmarks

**Figure 5. Instructions per Second versus Level-2 Cache Size, Data points Connected by Pipeline Depth**

## References

- [1] V. Agarwal, M. S. Hrishikesh, S. W. Keckler, and D. Burger. Clock rate versus IPC: the end of the road for conventional microarchitectures. In *27th Annual International Symposium on Computer Architecture*, pages 248–259, 2000.
- [2] S. I. Association. *International Technology Roadmap for Semiconductors*. 2001.
- [3] T. Austin et al. SimpleScalar: An infrastructure for computer system modeling. *IEEE Computer*, pages 59–67, Feb. 2002.
- [4] J. F. Cantin and M. D. Hill. Cache performance for SPEC CPU2000 benchmarks, version 3.0, 2003. <http://www.cs.wisc.edu/multifacet/misc/spec2000cache-data/>.
- [5] A. Hartstein and T. R. Puzak. The optimum pipeline depth for a microprocessor. In *29th Annual International Symposium on Computer Architecture*.
- [6] J. Henning. SPEC CPU2000: Measuring cpu performance in the new millennium. *IEEE Computer*, 33(7):28–35, July 2000.
- [7] M. S. Hrishikesh, N. P. Jouppi, K. I. Farkas, D. Burger, S. W. Keckler, and P. Shivakumar. The optimal logic depth per pipeline stage is 6 to 8 FO4 inverter delays. In *29th Annual International Symposium on Computer Architecture*.
- [8] C. Kim, D. Burger, and S. W. Keckler. An adaptive, non-uniform cache structure for wire-delay dominated on-chip caches. In *Tenth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 211–222, 2002.
- [9] S. R. Kunkel and J. E. Smith. Optimal pipelining in supercomputers. In *13th Annual International Symposium on Computer Architecture*, pages 404–411, 1986.
- [10] W. L. Lynch, G. Lauterbach, and J. I. Chamdani. Low load latency through sum-addressed memory (sam). In *25th Annual International Symposium on Computer Architecture*, pages 369–379, 1998.
- [11] S. McFarling. Combining branch predictors. Technical report, Digital Equipment Corporation Western Research Laboratory.
- [12] P. Shivakumar and N. Jouppi. CACTI 3.0: An integrated cache timing, power, and area model. Technical report, Compaq Western Research Laboratory.
- [13] E. Sprangle and D. Carmean. Increasing processor performance by implementing deeper pipelines. In *29th Annual International Symposium on Computer Architecture*.
- [14] S. J. Wilton and N. Jouppi. An enhanced access and cycle time model for on-chip caches. Technical report, Digital Western Research Laboratory.