OCTOBER 1993

WRL Research Report 93/3

Tradeoffs in Two-Level On-Chip Caching

Norman P. Jouppi and Steven J.E. Wilton

digital **Western Research Laboratory** 250 University Avenue Palo Alto, California 94301 USA

The Western Research Laboratory (WRL) is a computer systems research group that was founded by Digital Equipment Corporation in 1982. Our focus is computer science research relevant to the design and application of high performance scientific computers. We test our ideas by designing, building, and using real systems. The systems we build are research prototypes; they are not intended to become products.

There two other research laboratories located in Palo Alto, the Network Systems Laboratory (NSL) and the Systems Research Center (SRC). Other Digital research groups are located in Paris (PRL) and in Cambridge, Massachusetts (CRL).

Our research is directed towards mainstream high-performance computer systems. Our prototypes are intended to foreshadow the future computing environments used by many Digital customers. The long-term goal of WRL is to aid and accelerate the development of high-performance uni- and multi-processors. The research projects within WRL will address various aspects of high-performance computing.

We believe that significant advances in computer systems do not come from any single technological advance. Technologies, both hardware and software, do not all advance at the same pace. System design is the art of composing systems which use each level of technology in an appropriate balance. A major advance in overall system performance will require reexamination of all aspects of the system.

We do work in the design, fabrication and packaging of hardware; language processing and scaling issues in system software design; and the exploration of new applications areas that are opening up with the advent of higher performance systems. Researchers at WRL cooperate closely and move freely among the various levels of system design. This allows us to explore a wide range of tradeoffs to meet system goals.

We publish the results of our work in a variety of journals, conferences, research reports, and technical notes. This document is a research report. Research reports are normally accounts of completed research and may include material from earlier technical notes. We use technical notes for rapid distribution of technical material; usually this represents research in progress.

Research reports and technical notes may be ordered from us. You may mail your order to:

Technical Report Distribution
DEC Western Research Laboratory, WRL-2
250 University Avenue
Palo Alto, California 94301 USA

Reports and notes may also be ordered by electronic mail. Use one of the following addresses:

Digital E-net:	`DECWRL::WRL-TECHREPORTS`
Internet:	`WRL-Techreports@decwrl.dec.com`
UUCP:	`decwrl!wrl-techreports`

To obtain more details on ordering by electronic mail, send a message to one of these addresses with the word ‘‘`help`’’ in the Subject line; you will receive detailed instructions.

Tradeoffs in Two-Level On-Chip Caching

Norman P. Jouppi and Steven J.E. Wilton

10 December 1993

Abstract

The performance of two-level on-chip caching is investigated for a range of technology and architecture assumptions. The area and access time of each level of cache is modeled in detail. The results indicate that for most workloads, two-level cache configurations (with a set-associative second level) perform marginally better than single-level cache configurations that require the same chip area once the first-level cache sizes are 64KB or larger. Two-level configurations become even more important in systems with no off-chip cache and in systems in which the memory cells in the first-level caches are multiported and hence larger than those in the second-level cache. Finally, a new replacement policy called *two-level exclusive caching* is introduced. Two-level exclusive caching improves the performance of two-level caching organizations by increasing the effective associativity and capacity.

digital **Western Research Laboratory** 250 University Avenue Palo Alto, California 94301 USA

Table of Contents

List of Figures

List of Tables

1. Introduction

In recent years, increases in memory subsystem speed have not kept pace with the increase in processor speed, causing processor execution rates to become increasingly limited by the latency of accessing instructions and data. On-chip caches are a popular technique to combat this speed mismatch. As integrated circuits become denser, designers have more chip area that can be devoted to on-chip caches. Straight-forward scaling of cache sizes as the available area increases, however, may not be the best solution, since the larger a cache, the larger its access time. Using cache hierarchies (two or more levels) is a potential solution. This paper explores the tradeoffs in the design of on-chip microprocessor caches for a range of available on-chip cache areas.

There are a number of potential advantages of two-level on-chip caching with a mixed (instruction and data) second-level cache over single-level on-chip caching. First, the primary cache (also referred to as the L1 cache) usually needs to be split into separate instruction and data caches to support the instruction and data fetch bandwidths of modern processors. Many programs would benefit from a data cache that is larger than the instruction cache, while others would prefer the opposite. By having a two-level hierarchy on-chip where the majority of the cache capacity is in a mixed second-level cache (L2 cache), cache lines are dynamically allocated to contain data or instructions depending on the program's requirements, as opposed to living with a static partition given by single-level on-chip cache sizes chosen at design time.

A second and more important potential advantage of two-level on-chip caching is an improvement in cache access time. As existing processors with single-level on-chip caching are shrunk to smaller lithographic feature sizes, the die area typically needs to be held constant in order to keep the same number of bonding pads. When processors are initially designed, their on-chip cache access times are usually well matched to their cycle times. If the additional area available due to a process shrink is used to simply extend the first-level cache sizes, the caches will get slower relative to the processor datapath. Instead, if the extra area is used to hold a second-level cache, the primary caches can scale in access time along with the datapath, while additional cache capacity is still added on-chip.

A third potential advantage of two-level cache structures is that the second-level cache can be made set-associative while keeping the primary caches direct-mapped. This keeps the fast primary access time of direct-mapped caches, but reduces the penalty of first-level conflict misses since many of these can be satisfied from an on-chip set-associative cache instead of requiring an off-chip access.

When primary cache sizes are less than or equal to the page size, address translation can easily occur in parallel with a cache access. However, most modern machines have minimum page sizes of between 4KB and 8KB. This is smaller than most on-chip caches. By using two-level on-chip caching, the primary caches can be made less than or equal to the page size, with the remaining on-chip memory capacity being devoted to the second-level cache. This allows the address translation and first-level cache access to occur in parallel. By the time the second-level cache would be accessed on a primary cache miss, there will have been plenty of time to complete the translation from virtual to physical addresses for indexing a larger physically-address second-level cache. This is a fourth potential advantage of two-level cache structures.

A fifth advantage of two-level cache structures is that a chip with a two-level cache will usually use less power that one with a single-level organization (assuming the area devoted to the cache is the same). In a single-level configuration, wordlines and bitlines are longer, meaning there is a larger capacitance that needs to be charged or discharged with every cache access. In a two-level configuration, most accesses only require an access to a small first-level cache.

Perhaps the biggest potential disadvantage of two-level on-chip caching occurs if the total capacity of the first-level caches is not much smaller than that of the second-level cache. Here much of the second-level cache will consist of instructions and data which are already in the primary caches, causing most misses in the primary caches to also miss in the second-level cache. In this situation adding a second-level cache can "get in the way" by adding delay between a first-level cache miss and an off-chip access, hurting more than it helps by reducing the off-chip miss rate. In order to mitigate problems of duplication in on-chip multi-level caching a new technique called *exclusive two-level caching* will be introduced in Section 8.

On-chip cache studies involve the interaction of a number of factors which have not been combined to date:

- Miss rates for a range of system parameters, including first and second-level cache sizes, associativity, and replacement policy need to be obtained.
- Area models of on-chip memories are needed to compute the chip area required by a cache configuration as a function of cache size, layout parameters, and associativity.
- Time models are needed to determine how much of an effect various cache parameters (size, associativity) have on the cache access time, and hence overall execution time of a program.

By combining these three factors, the best choice for a cache organization for a technology and a given amount of available area can be obtained. To our knowledge, this paper is the first that encompasses all three of these issues for on-chip memory system performance modeling.

Previous work by Hill [3] has studied access times and miss rates, and recommended that first-level caches should be direct-mapped. However, he did not study on-chip RAM area, and studied only single-level caching organizations. Przybylski [7] has studied execution times of multi-level cache systems as a function of many parameters. However, no mapping of configuration to chip area was done, nor was access time computed from cache parameters. Mulder [5] modeled the area of on-chip caches, but did not consider the access time implications or the effect on miss rates. Wada [10] modeled the access time of memories, but did not consider the effects on area or miss rates.

This work extends the work by Mulder [5] and Wada [10] and combines it with miss rate data to arrive at the best performance for on-chip memory hierarchies considering all three parameters simultaneously.

2. Experimental Method

The results presented in this paper were obtained in four steps. First miss rates for various cache configurations were obtained using simulation. Second, time models were used to estimate the cache cycle time. Third, area models were used to estimate the chip area required for each cache configuration. Finally, the miss rates, cycle times, and areas were combined to obtain the overall performance as a function of chip area. This section will describe each of these steps in more detail.

2.1. Framework

A pipelined RISC architecture which allows the issue of an instruction and data reference each cycle was assumed. We further assumed that the processor cycle time is determined by the first-level cache cycle time. This results in a variation in machine cycle time of about 1.8X from processors with 1KB caches through 256KB caches. (Olukotun [6] has studied the effects of multi-cycle cache latency on processor performance, however this is currently beyond the scope of this work). We modeled the baseline CPI of the machine without cache misses as being 1. To attain this CPI in a real machine in the presence of non-unit-latency functional units, branch miss prediction penalties, and other non-memory system stalls, a modest degree of superscalar execution would be required.

We restricted the design space to systems comprising split instruction and data direct-mapped first-level caches of equal size, and to optional mixed second-level caches. Both direct-mapped and set-associative second-level caches with pseudo-random replacement were investigated. First-level cache size varied from 1KB to 256KB, and second-level cache sizes ranged from 0KB (non-existent) to 256KB. Physically-addressed lockup caches with 16 byte lines were assumed. We are currently working on extending the model to include multicycle cache access with non-blocking load instructions (see Section 10).

Off-chip miss service times of 50ns and 200ns were chosen, corresponding to systems with and without a board-level cache. These miss service times include the off-chip access time as well as the transfer time and other overhead associated with refilling a cache line from off-chip.

2.2. Miss Rates

Because we required address traces at least several tens of millions of address references long (since the second-level caches were as large as 256KB), we obtained long and accurate memory references with a tracing system that allowed on-the-fly analysis [2]. Because of the low overhead incurred by the tracing system and the cache simulator (about 10x slowdown and 6x slowdown, respectively), we have been able to simulate the cache configurations under study using a DECStation 5000. The SPEC [9] benchmarks *gcc1, espresso, fpppp, doduc, li, eqntott,* and *tomcatv* were used to gather miss rate data. Table 1 shows the number of instruction and data references made by each of these workloads.

Write traffic was modeled as read traffic (i.e., write-allocate and fetch-on-write). Effects of multiprogramming and system references were beyond the scope of this study.

Program	Instr. refs.	Data refs.	Total refs.
gcc1	22.7M	7.2M	30.0M
espresso	135.3M	31.8M	167.1M
fpppp	244.1M	136.2M	380.3M
doduc	283.6M	108.2M	391.8M
li	1247.1M	452.8M	1699.9M
eqntott	1484.7M	293.6M	1778.3M
tomcatv	1986.3M	963.6M	2949.9M

Table 1: Test program references

2.3. Cycle Time Model

The cycle time for each cache was computed using the model presented in [11] (which was based on the access time model of Wada et al in [10]). Wada's model used SPICE parameters to predict the delays due to the address decoder, word-line drivers, bit-lines, sense amplifiers, data bus drivers, and data output drivers. Their equations are written in terms of cache size, block size, associativity, and RAM organization parameters.

Wada's model was extended in [11] in several ways. First, the tag array was modeled, including its comparator and, in the case of a set-associative cache, its multiplexor driver. Precharged bit-lines were modeled, as well as column multiplexing of the bit-lines. Equations were derived for the *cycle time* of the cache (that is, the minimum time required between the start of two accesses), as opposed to just the *access time* (the time between the start and end of a single access).

The SPICE parameters in [10] and [11] are based on a 0.8μm technology. In this study we scaled the resulting access times to more closely match a high-performance 0.5μm CMOS technology. This resulted in an overall cycle time reduction to 50% of the values derived in [11].

The resulting access and cycle times for direct-mapped first-level caches are shown in Figure 1. By iterating through the delay expressions for a range of memory array organizations (such as the number of bitlines and word lines in the data and tag arrays, and the number of subarrays that the memory is broken into, if any), the minimum access and cycle times for each cache size were chosen. The vertical axis in Figure 1 shows the smallest access and cycle times obtained, while the horizontal axis shows the area required by this configuration (the area model is described in the next section).

Second level cache cycle times were computed as:

$$T_{level2\_cycle} = \lceil \frac{\text{level2 RAM cycle time}}{\text{processor cycle time}} \rceil \times \text{processor cycle time}$$

where *processor cycle time* is the first-level cache cycle time. Figure 2 shows the cycle and access times for various second-level cache sizes, assuming the first-level cache is 4KB. Although the cycle time is always larger than the access time, the graph only shows a difference below 64KB. This is because the second-level cycle and access times are rounded to the next

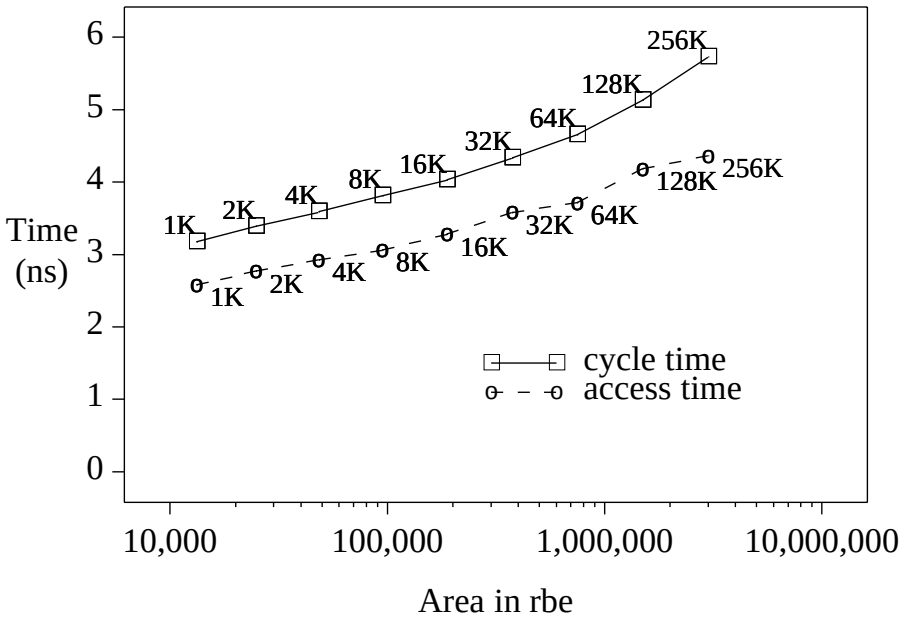

Figure 1: First level cache access and cycle times

larger multiple of the processor cycle time as described above. The right axis shows the L2 access time in terms of CPU cycles. One observation is that the difference in access time between on-chip L1 and L2 access times is much smaller than that between on-chip L1 and off-chip L2 access times.

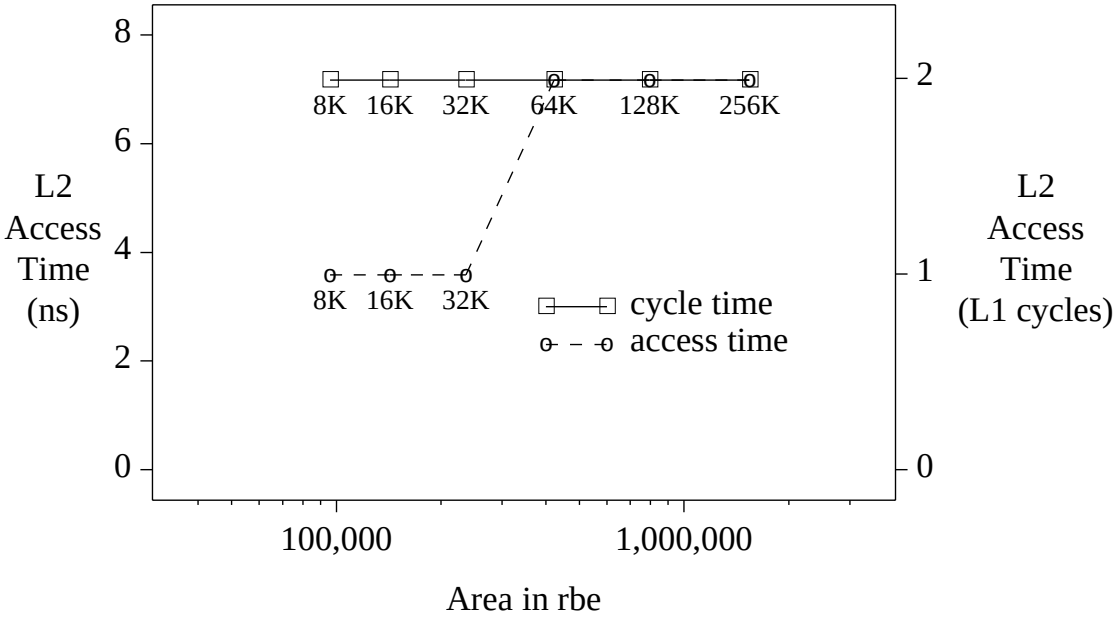

Figure 2: L2 access and cycle times with 4KB L1 caches

2.4. Cache Area Model

Estimates of the chip area required by each cache organization were obtained from a model by Mulder, Quach and Flynn which included area for memory cells, drivers, sense amps, address tags, dirty and valid bits, comparators, and control logic [5]. They defined a unit of area termed a *register-bit equivalent* (rbe) which was independent of technology. A single a 6-transistor SRAM cell was presented as 0.6 rbe. For small memories, the area required by RAM peripheral logic can significantly increase the average area required per bit. For large memories where speed is not critical, the per bit overhead of peripheral logic will be less. However, based on the memory array organization parameters from the time model, we always organized the memories to give the highest performance. In general this increases the area required per bit because it breaks up large memory arrays into smaller subarrays with lighter loading on their bitlines and wordlines. This results in faster access times but a larger ratio of peripheral to RAM core cell area.

In most of this paper, we will assume first-level RAM cells are 6-transistor single-ported cells allowing one read or write per cycle. Section 6, however, will consider larger multiported cells. In all cases, the second-level cache is assumed to consist of single-ported 6-transistor cells.

2.5. Average Time Per Instruction

Once the miss rates, cache cycle times, and chip areas were determined for a particular cache organization, the results were combined into execution time. This section describes how the execution time was determined for two-level cache systems; performance modeling of single-level caching is a straightforward subset of this model.

The execution time of a program can be written as:

$$T_{total} = \text{time spent if no L1 cache misses} + \text{time due to L2 hits} + \text{time due to L2 misses}$$

Since the first-level cache is split, both an instruction and data reference can be issued in the same clock cycle. The time for first-level data reference hits is thus included within the first-level instruction reference time. Since we have assumed an instruction is issued every cycle, we can write:

$$\text{time spent if no L1 cache misses} = (\text{Number of Instructions}) \times (\text{L1 cycle time})$$

Upon a first level cache miss, we assume that one L2 cache cycle time is needed to probe the second-level cache and transfer the first 8 bytes to the first-level cache, followed by another L2 cycle time to transfer the next 8 bytes (16 byte lines are used throughout this paper). We assume only one L1 cycle is required to write each 8 bytes. All but the last write will be overlapped with transfers of the rest of the line. Thus:

$$\text{time due to L2 hits} = (\text{Number L2 Hits}) \times (2 \times \text{L2 cycle time} + \text{L1 cycle time})$$

Recall that the L2 cycle time is rounded to the next higher multiple of the L1 cycle time. As an example, consider the miss penalty for the system parameters corresponding to Figure 2. The L1 miss penalty for references that hit in the L2 cache would be (2×2)+1=5 CPU cycles.

The time due to L2 cache misses can be calculated similarly. One L2 cache cycle time is required to initially probe the second-level cache. Off-chip access and transfer times are combined into a single off-chip time term. Once the data has been fetched, the cost of writing the two 8 byte blocks of refill data to the second-level cache and transferring them to the first level is two L2 cycle times. Finally, the write of the last 8 bytes to the first level is one more L1 cycle time. Thus,

$$\text{time due to L2 misses} = (\text{Number of L2 Misses}) \times (\text{Off-chip time} + 3 \times \text{L2 cycle time} + \text{L1 cycle time})$$

The off-chip service time is also rounded to the next higher multiple of the L1 cycle time.

The average time per instruction (TPI) is calculated as the ratio of the total execution time divided by the number of instructions executed. It can also be calculated as $TPI = cycle\_time \times CPI$. TPI is a better metric than the traditional clocks per instruction (CPI) for comparing cache configurations since it takes into account changes in processor cycle time resulting from changes in first-level cache cycle time.

3. Single-Level Caching Performance

We begin with the investigation of systems with only a single level of on-chip caching. Figures 3 and 4 show the performance of all seven workloads measured in time per instruction (TPI) as a function of chip area available for the caches (measured in register-bit equivalents). Notice that a logarithmic scale is used for each axis. The time required to service a miss from off-chip is assumed to be 50ns. The markers on each point give the cache size of the level-one instruction and data caches. For clarity, markers are only shown for one workload on each graph; points directly above the marked points all correspond to the same cache configuration.

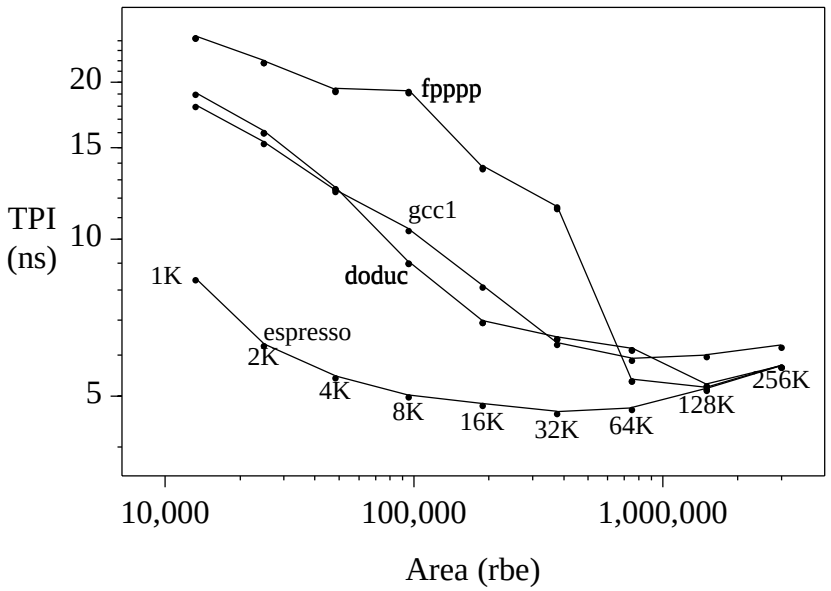

Figure 3: *gcc1, espresso, doduc, and fpppp*: 50ns off-chip service time, L1 only

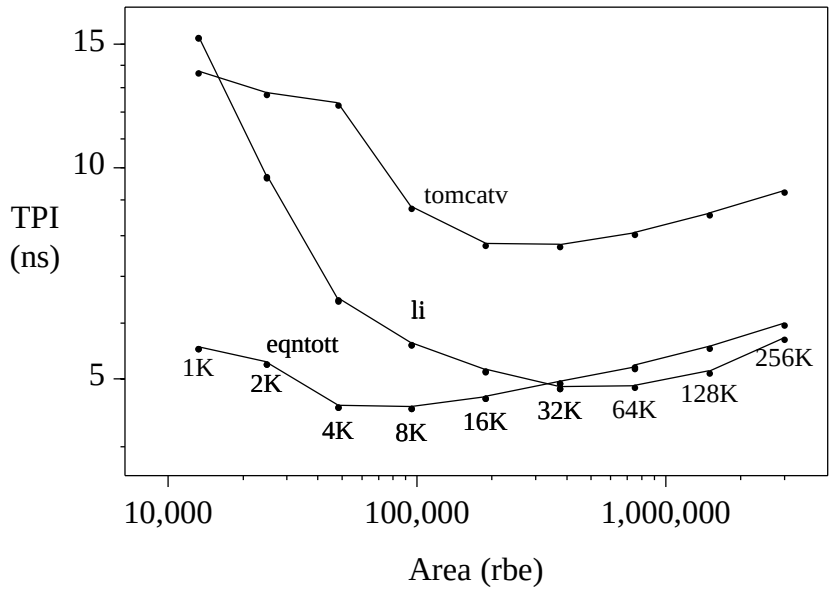

Figure 4: *li, eqntott, and tomcatv*: 50ns off-chip service time, L1 only

All seven workloads exhibit a minimum TPI between 8KB and 128KB. Although larger caches result in lower miss rates, as the cache grows, the decrease in miss rate is more than offset by the increase in cache cycle and access times. This implies that if a designer has 2,000,000 rbe's available for an on-chip cache, and only a single-level cache configuration is to be employed, better performance is obtained if only a portion of the available real-estate (about 500,000 rbe's) is devoted to the cache, even if the remaining potential area is unused. This corresponds to an optimum single-level cache size of about 32KB.

The location of the minimum for each workload depends on how much the miss rate decreases as the cache size is increased. Both *espresso* and *eqntott* have low miss rates (0.0100 and 0.0149 respectively at 32KB), so there is little potential for a larger cache to remove significantly more misses. Thus, these two workloads favor a small cache. The *tomcatv* workload as a relatively high miss rate (0.109 at 32KB), but the miss rate does not drop appreciably as the cache size is increased. Thus, this workload also favors small cache sizes.

4. Baseline Two-Level Caching Performance

In this section we consider the performance of two-level on-chip cache configurations. Although direct-mapped caches usually provide the best performance for first-level caches [3], Przybylski points out that associativity is useful in lower levels of the cache hierarchy [7]. In this section we assume the second-level cache is four-way set-associative, while the first-level cache is direct-mapped. Set-associative caches tend to result in lower miss rates, but their access and cycle times are larger than the same-sized direct mapped caches, since the tag must be read and compared in order to select the proper item from the data array. Both the first and second-level caches are assumed to consist of single-ported 6-transistor RAM cells. The time required to service an on-chip cache miss from off-chip is assumed to be 50ns.

Figure 5 shows the performance of many different cache configurations plotted against area for the *gcc1* workload. As in Section 3, the X-axis is the chip area required for the configuration, while the time per instruction is plotted on the Y-axis. Again, a logarithmic scale is used for both axes. Each point is labeled with two numbers; the first corresponds to the size (in KB) of each of the L1 caches (instruction and data), while the second indicates the size of the second-level cache. The solid line in the figure shows the *best performance envelope*. This is the best performance that can be obtained for a given cache area. The staircase appearance of this line is due to the discrete nature of the cache sizes. For example, if 3,000,000 rbe's are available, the best performance for *gcc1* could be obtained if 32KB instruction and data caches and a 256KB second level cache were used. As another example, 1KB first-level caches with a 2KB second-level cache would be a bad choice (for *gcc1*), since the "2:0" configuration occupies approximately the same area, and has a lower TPI. In the "1:2" configuration, most of the items in the 2KB secondary cache are also in one of the 1KB primary caches, so a miss in the first level usually results in a miss in the second level. Thus, there are almost as many misses that must go off-chip as in the single-level 1KB case, but the cost of probing another cache level is incurred.

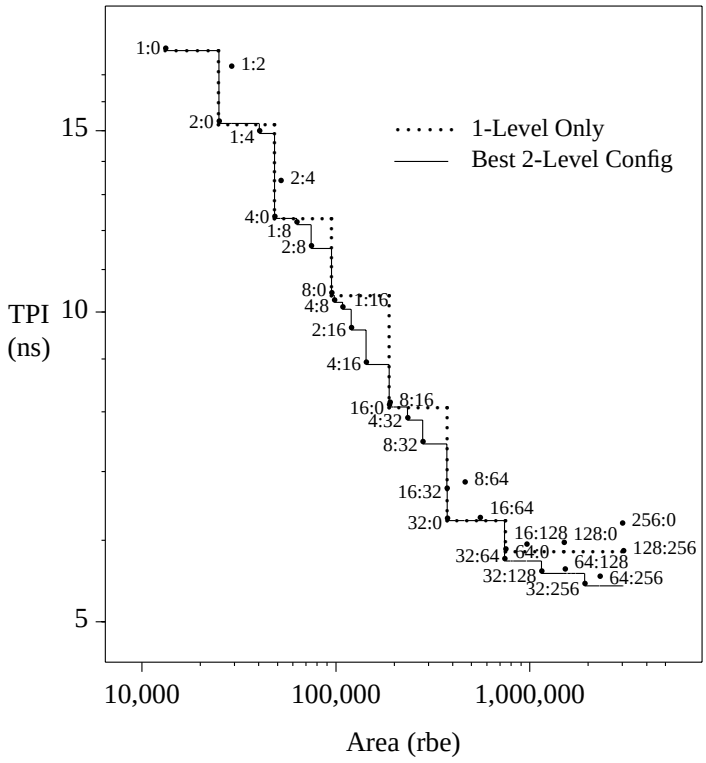

Figure 5: *gcc1*: 50ns off-chip, L2 4-way set-associative

The dotted line in the graph represents the best performance possible if only single-level cache configurations are used. Thus, the "distance" between the solid and dotted lines give an indication of the effectiveness of two-level caching compared to using single-level configurations. In Figure 5, the dotted line lies largely on top of the solid line, meaning that for most available chip areas, a single level configuration is preferable.

Figure 6, 7, and 8 show the best performance envelope as well as the single-level caching performance staircase for *doduc, espresso, fpppp, li, tomcatv,* and *eqntott* with the same system

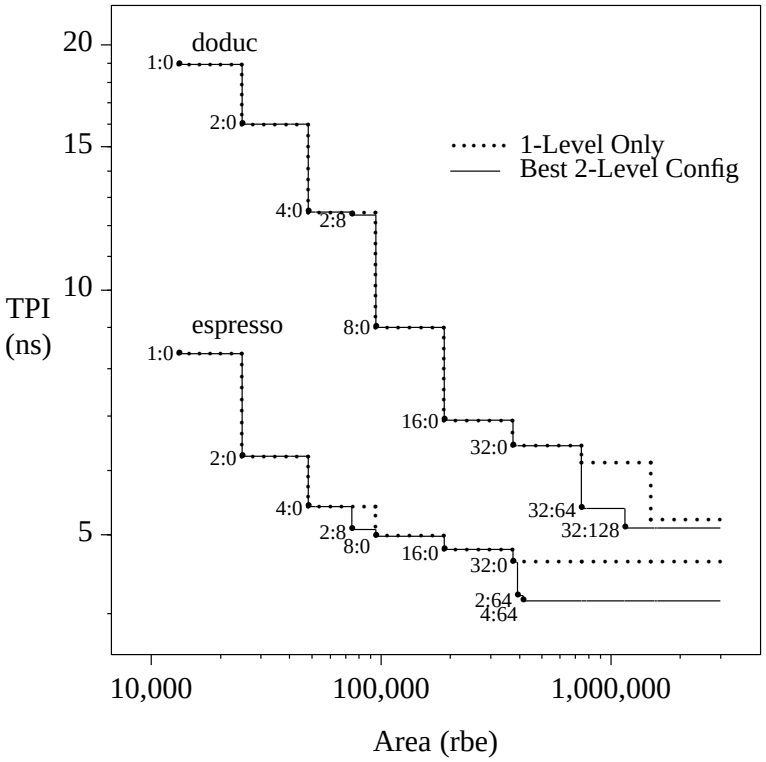

Figure 6: *doduc* and *espresso*: 50ns off-chip, L2 4-way set-associative

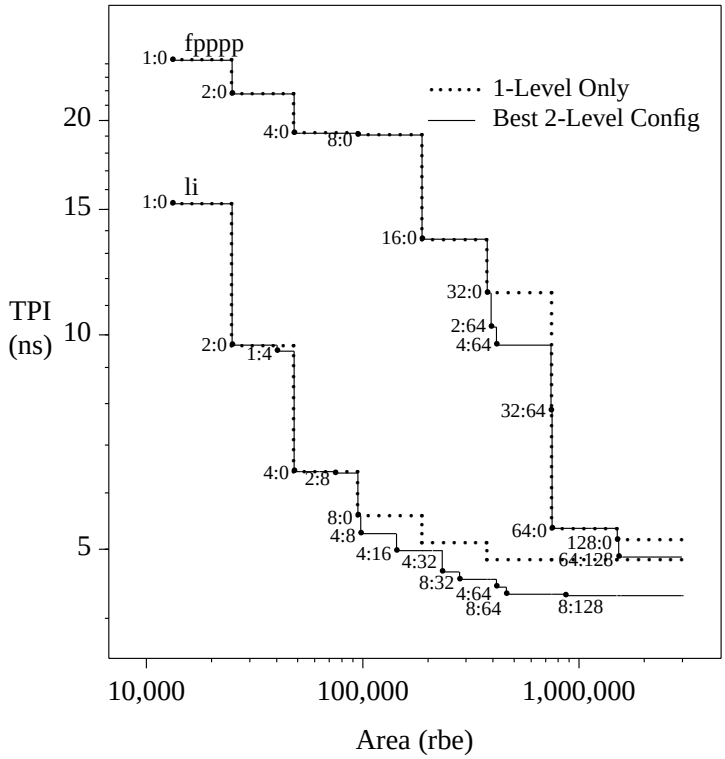

Figure 7: *fpppp* and *li*: 50ns off-chip, L2 4-way set-associative

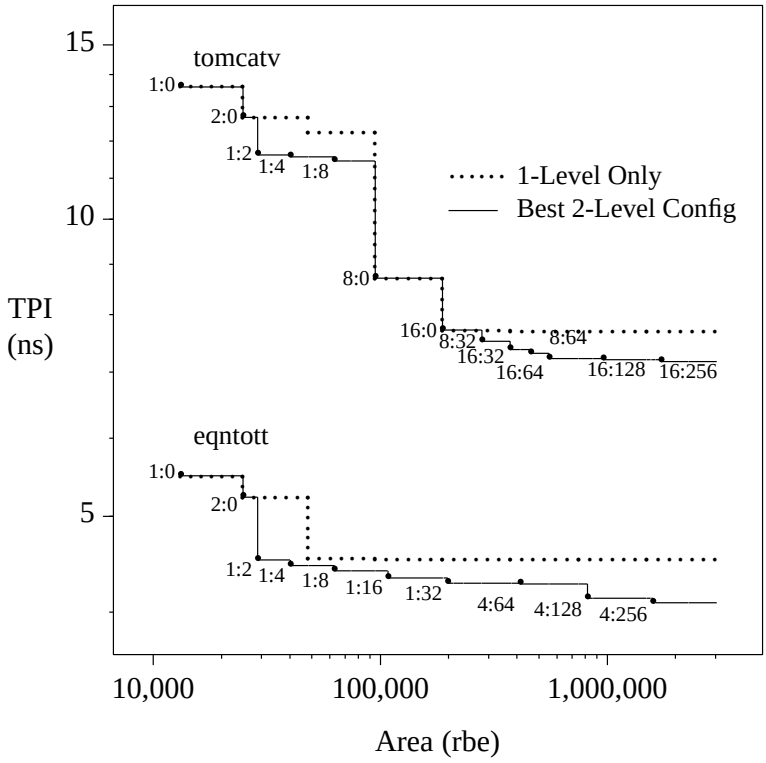

Figure 8: *tomcatv* and *eqntott*: 50ns off-chip, L2 4-way set-associative

parameters. For clarity, only those configurations that make up the best performance envelope are shown. For most of the workloads, single-level configurations tend to dominate the performance envelope for areas below about 300,000 rbe's, while for larger available areas, two-level configurations become marginally preferable.

5. Direct-Mapped Second-Level Caches

We have also analyzed the performance of direct-mapped second-level caches. For most benchmarks, 4-way set-associative caches perform slightly better than direct-mapped caches. This is because the increase in L2 access time required for set-associativity is more than offset by the decrease in miss rate. Also, because the second-level cycle time is rounded up to a multiple of the first-level cache cycle time, in many cases the number of cycles required for a second-level access is not increased by the addition of set-associativity. Finally, the extra area required by a set-associative cache does not significantly affect the performance for a given area. This is because the extra comparators needed in a set-associative cache are very small when compared to the area required by the data and tag arrays (in the area model, a comparator only occupies 6x0.6 rbe's [5]). The performance of *gcc1* with a direct-mapped second-level cache is shown in Figure 9.

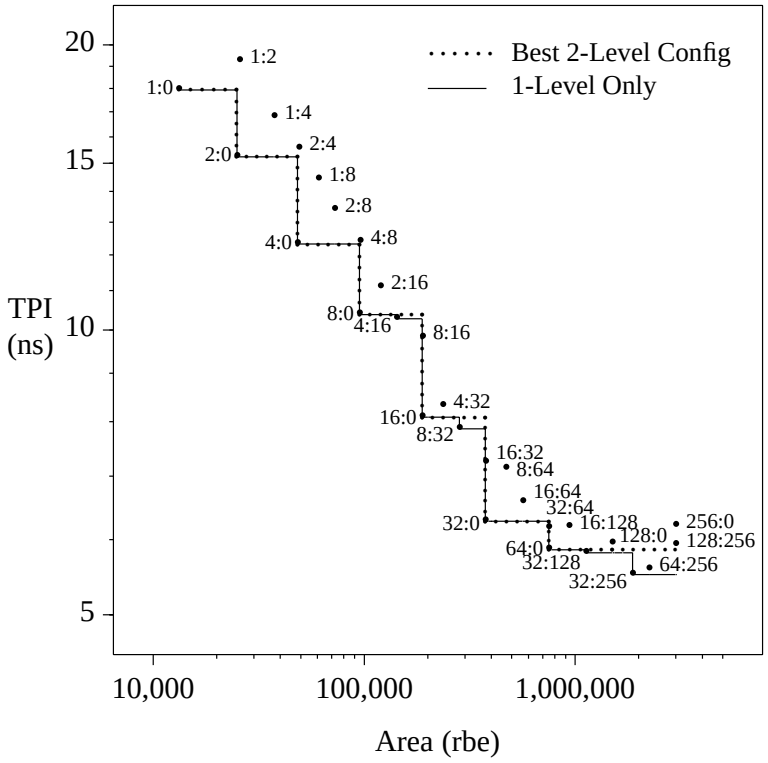

Figure 9: *gcc1*: 50ns off-chip, L2 direct-mapped

6. Dual-Ported First-Level Caches

An important additional degree of freedom in systems with two-level on-chip cache configurations is to use different RAM cells for the first and second-level caches. For example, in a superscalar machine that issues many instructions per cycle, a multi-ported first-level cache may be needed to support the issue of more than one load or store per cycle. A cache with two ports typically requires twice the area of a cache with one port. In fact, it is not uncommon to implement a memory with two read ports and one write port as two copies of a one-read-port, one-write-port memory. A banked cache can also be used to support more than one load or store per cycle; since banking requires more inputs and outputs to the cache it also increases the area required for the cache (the tradeoffs between banking and dual porting have been studied in [8]).

In this section we assume that the cell used in the first-level caches requires twice the area but can support twice the access bandwidth of the cell used in the second-level cache. We assume this results in an effective doubling of the instruction issue rate for superscalar machines which can make effective use of dual-port caches. Both the first-level instruction and data caches are assumed to grow in area to achieve higher bandwidth access rates. A level of off-chip caching is assumed as before, meaning the off-chip service time is 50ns. The second-level cache is assumed to be 4-way set-associative.

Figures 10 to 16 show the performance of the seven workloads with the first-level caches having twice the area but supporting twice the instruction issue rate as the corresponding cache in the base system. The dotted line in each graph shows the performance envelope if only a

single level of on-chip caching is used, with the base cell from the previous sections in the caches. The dashed line shows the best performance envelope if the base cell is replaced with one that is twice as big with twice the bandwidth (still a single-level cache). The solid line shows the best performance envelope if two-level cache structures are used, with the base cell used in the L2 cache, and the larger dual-ported cell used in the L1 cache.

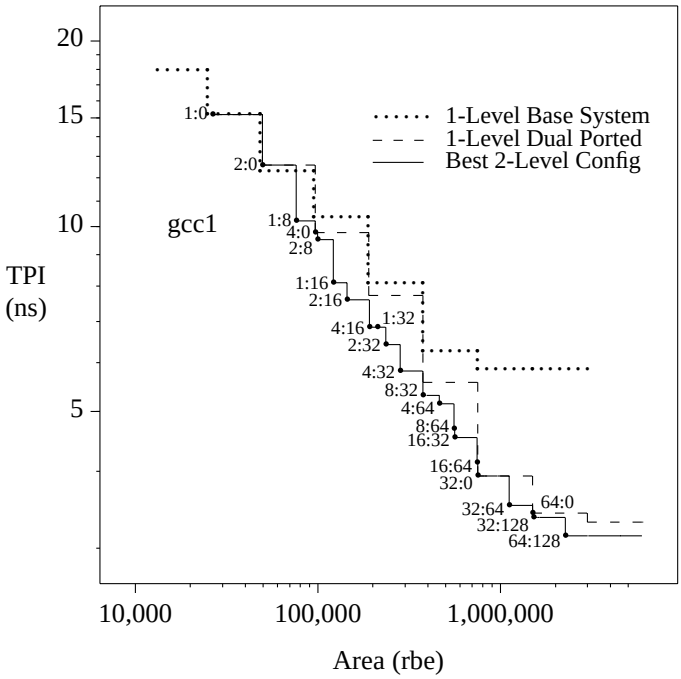

Figure 10: *gcc1*: 50ns, 4-way, 2X L1 area, 2X instruction issue rate

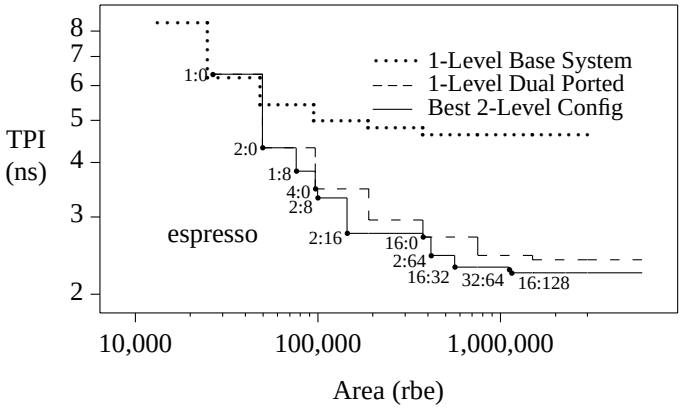

Figure 11: *espresso*: 50ns, 4-way, 2X L1 area, 2X instruction issue rate

First consider the effect of moving from the base cell to the dual-ported cell in a single-level cache configuration. Comparing the dotted and dashed lines in each figure, it is apparent that in many workloads, the base cell is preferred for small caches, while for larger caches, the dual-ported cell gives a better performance for a fixed area. The cross-over point ranges from 50,000 rbe's to 400,000 rbe's. For small caches, the performance gain when using a dual-ported cell is usually less than the performance gain that could be obtained by keeping the smaller single-

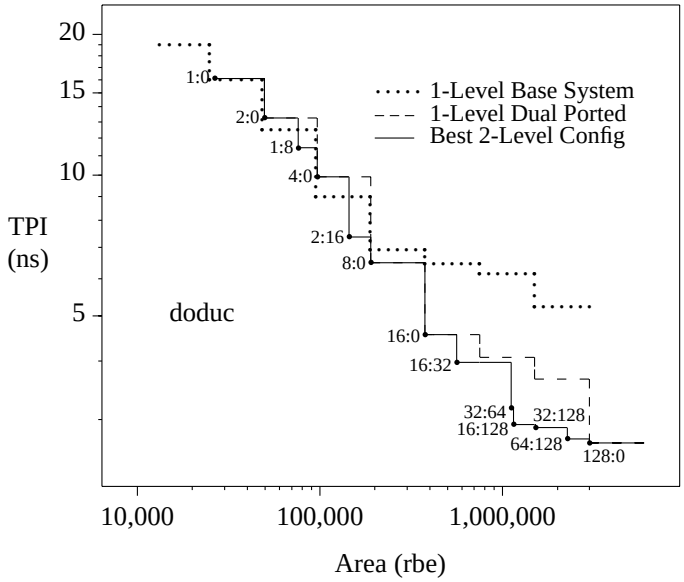

Figure 12: *doduc*: 50ns, 4-way, 2X L1 area, 2X instruction issue rate

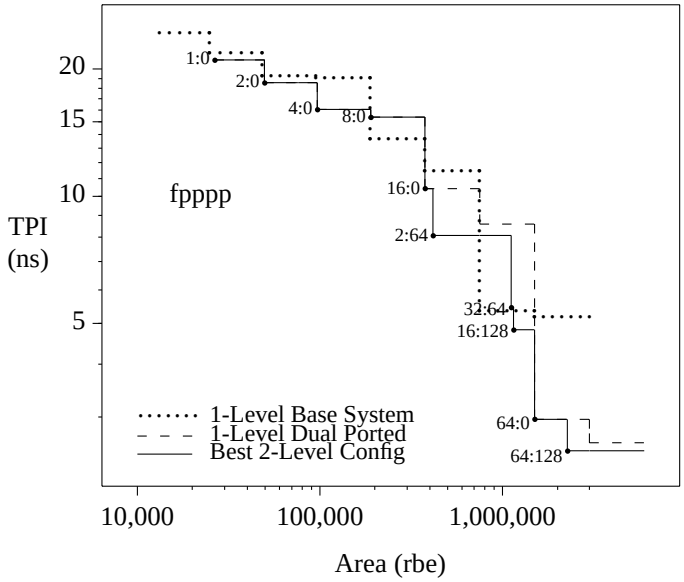

Figure 13: *fpppp*: 50ns, 4-way, 2X L1 area, 2X instruction issue rate

ported cell, but doubling the number of cells in the cache (the cache size). This is because for the small caches most of the execution time is spent in cache misses, and doubling the instruction issue rate without changing the amount of time spent in cache misses has little overall effect on performance. The opposite is true when the cache gets bigger than about 8KB (for most workloads). Here most of the execution time is due to instruction execution and not the processing of cache misses, so increasing the instruction issue rate at the expense of the miss rate is a good tradeoff. These results are consistent with Section 3 that showed for large caches increasing the single-level cache size is usually a detriment to performance. Moving from a cache with single-ported cells to the same-capacity cache with dual-ported cells, however, always improves performance. In *eqntott* and with all but 1KB caches in *espresso* the dual-ported cells are

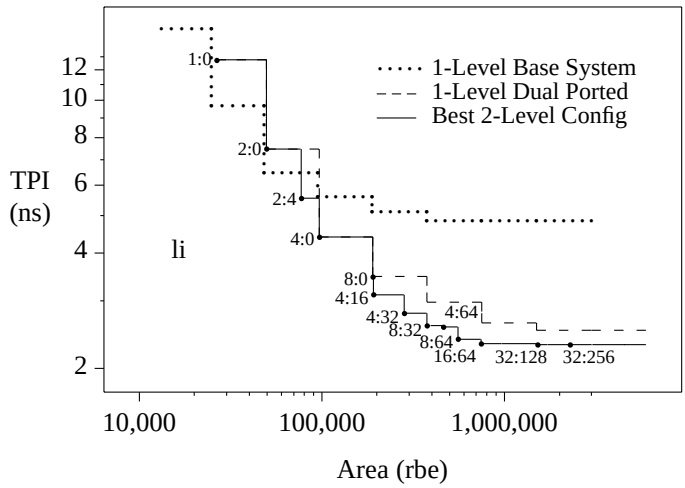

Figure 14: *li*: 50ns, 4-way, 2X L1 area, 2X instruction issue rate

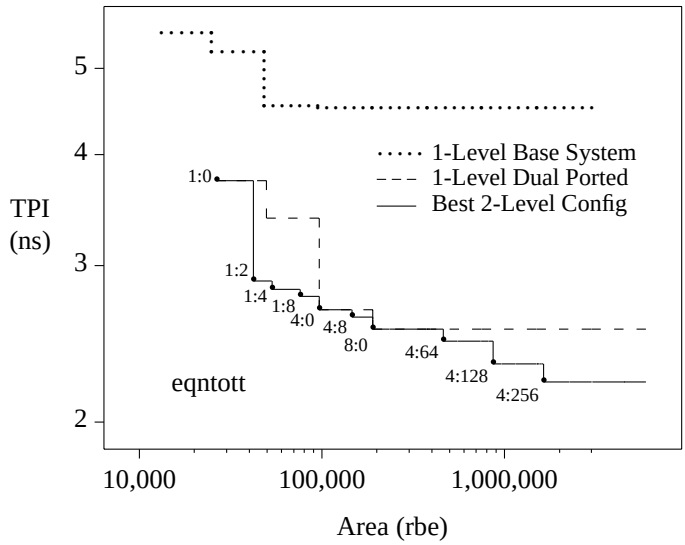

Figure 15: *eqntott*: 50ns, 4-way, 2X L1 area, 2X instruction issue rate

preferred. The low miss rate of these applications means improving the miss rate is less important than increasing the instruction issue rate.

Now consider the effects of a second-level cache, by comparing the dashed and solid lines in each graph. Comparing these graphs with Figures 5 to 8, it can be seen that using two levels is more important when the first level uses the large dual-ported cell than when it uses the base cell. In almost every workload, there are fewer single-level configurations on the best performance envelope when the dual-ported cell is used. A hybrid two-level configuration combines the advantages of high-bandwidths at level one (from the large dual-ported L1 cells) with high on-chip capacity (from the small single-ported L2 cells).

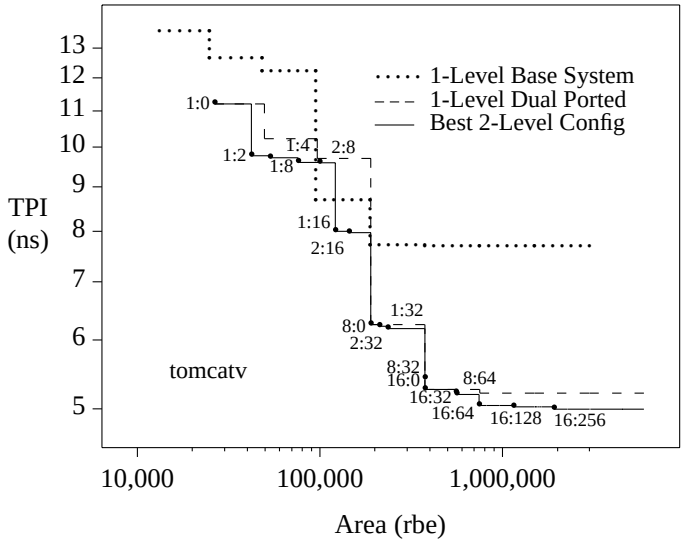

Figure 16: *tomcatv*: 50ns, 4-way, 2X L1 area, 2X instruction issue rate

7. Long Off-Chip Miss Service

In previous sections it was assumed that the processor chip would have misses serviced from an off-chip cache in 50ns. In many low-cost systems, the expense of a board-level cache may be prohibitive. In this section we assume the on-chip misses are serviced from off-chip in 200ns, corresponding to a machine without a board-level cache. The on-chip assumptions remain the same as in the baseline section: direct-mapped L1 caches, 4-way set associative L2 caches, and single-ported RAM cells for both L1 and L2.

Figure 17 shows the best performance envelope for *gcc1* along with the envelope if only single-level configurations are considered. The TPI for small on-chip caches is increased dramatically compared to systems with off-chip caches. A system with 1KB on-chip caches pays a penalty of about 3X in run time, as compared to a machine with 50ns off-chip service times. For a system with 32KB L1 caches and 256KB L2 caches, however, there is much less difference between configurations with 50ns and 200ns off-chip miss service.

More interesting than the change in TPI, however, is the effect of moving from a single-level to a two-level configuration. Comparing Figures 17 and 5, it is clear that fewer single-level configurations lie on the performance envelope for a 200ns off-chip penalty than a 50ns off-chip penalty. In fact, there are no 1-level configurations larger than "4:0" on the envelope.

Figures 18, 19, and 20 show the same trends for the other six workloads. Again, with an off-chip penalty of 200ns, the TPI for systems with small on-chip caches is considerably larger than the corresponding TPI in a system with an off-chip penalty of 50ns. Even in *eqntott* and *espresso*, with their low miss rates, the TPI is doubled when the off-chip penalty is changed from 50ns to 200ns. Comparing these figures to those in Section 4, it can be seen that for every workload, the "distance" between the single-level and two-level best-performance envelopes is larger when the off-chip time is 200ns (i.e., two-level cache hierarchies are a bigger win with a larger off-chip access time). This is not surprising, since one of the goals of a second-level cache

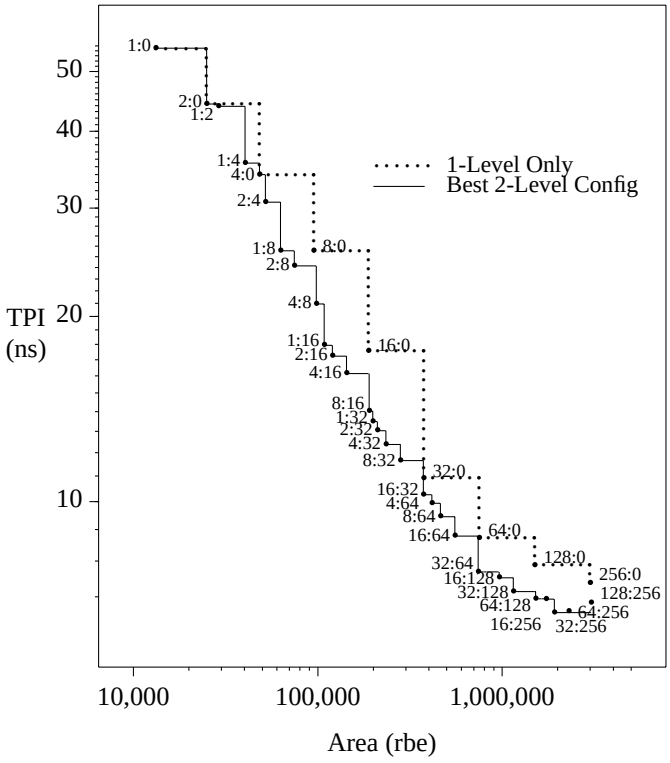

Figure 17: *gcc1*: 200ns off-chip, L2 4-way set-associative

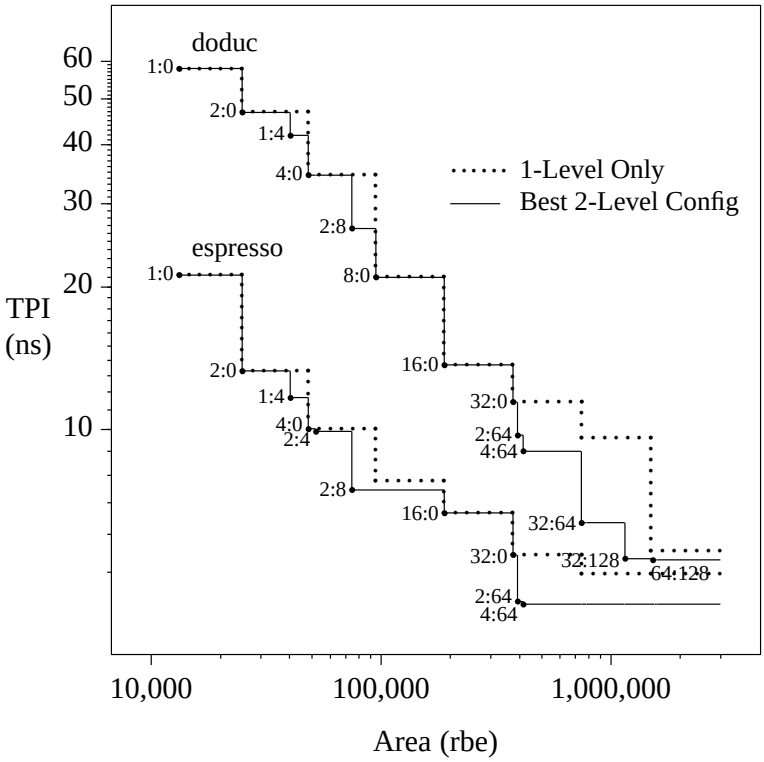

Figure 18: *doduc* and *espresso*: 200ns off-chip, L2 4-way

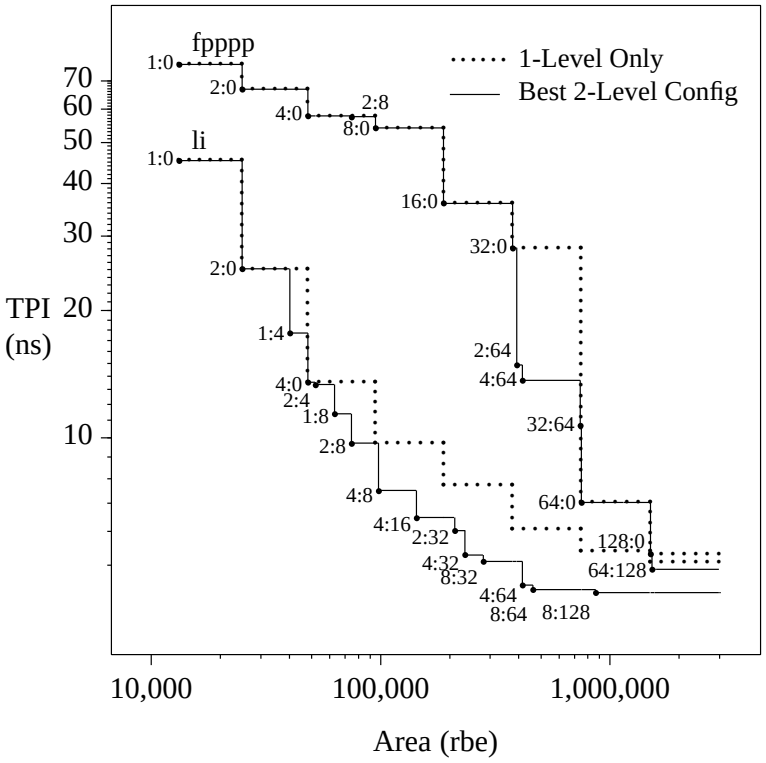

Figure 19: *fpppp* and *li*: 200ns off-chip, L2 4-way

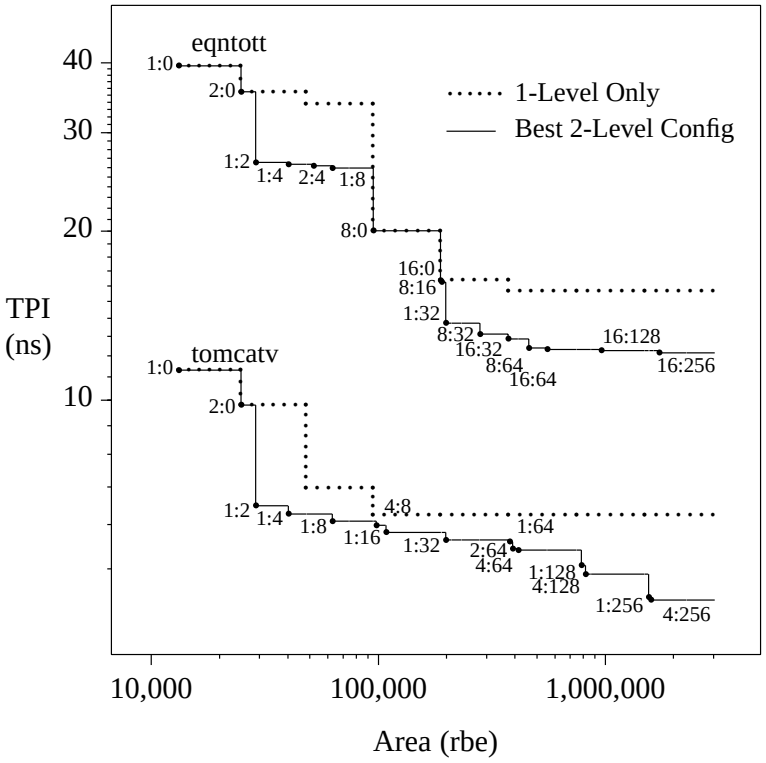

Figure 20: *tomcatv* and *eqntott*: 200ns off-chip, L2 4-way

is to reduce the first-level miss penalty. The second-level cache will be more effective at this if the off-chip miss penalty is large. Also, as the miss penalty becomes larger, the flexible allocation of instruction and data cache lines in a two-level system becomes more important.

8. Two-Level Exclusive Caching

In standard two-level caching, the same line can exist in more than one level in the cache hierarchy at once. To maximize the ratio of information to memory area, a scheme called *two-level exclusive caching* is introduced. In two-level exclusive caching, when a reference misses in the first level and hits in the second, the contents of the first-level cache line are transferred to the second-level cache while the second-level cache line is used to refill the first-level cache. This results in a swap if the current contents of the first-level cache line and the desired contents of the first-level cache line map to the same second-level cache line. When a reference also misses in the second level, the desired line is loaded directly into the first-level cache from off-chip, while the first-level victim is sent to the second-level cache.

Under this scheme, a mapping conflict in both first-level and second-level direct-mapped caches will give rise to "exclusion"; that is, the data involved in the mapping conflict will exist in one level of the hierarchy or the other, but not both. For example, consider the simple configuration shown in Figure 21-a. If a reference is made to address *A* which maps to line 13 in the second-level cache, followed by a reference to address *E* which also maps to line 13 in the second-level, then lines *A* and *E* will be swapped. If references to *A* and *E* alternate, they will repeatedly exchange places. Thus, each line would exist in exactly one level of the hierarchy. This is in contrast to a conventional two-level system, which could only store either *A* or *E*, but not both.

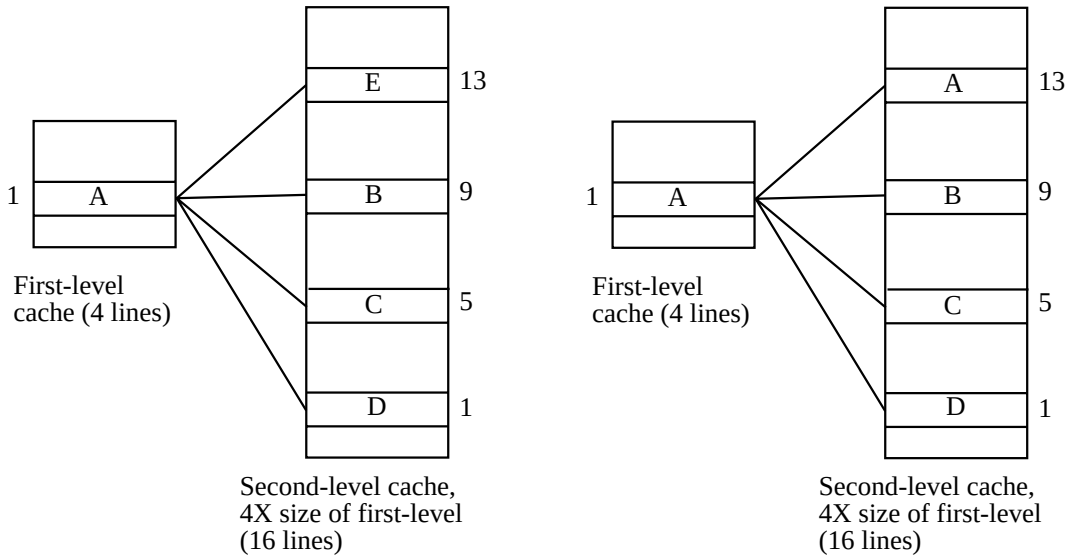

Figure 21: Exclusion vs. inclusion during swapping, direct-mapped caches

If a conflict occurs only in the first-level cache, however, exclusion will not result. Consider Figure 21-b. If address *A* is referenced, followed by a reference to address *B*, sending data at address *A* back to the second-level cache will leave the second-level cache unchanged. (If both caches are write-back, then the contents of address *A* in the second-level cache will be updated from the contents of the first, but the address mapping will stay the same). Similarly, if references are made to addresses *C* and *D*, inclusion will still occur between the first-level cache and the second-level cache.

Exclusive caching has two advantages over conventional replacement policies:

- Conflict misses in the second level cache are reduced since two lines can be present in the first two levels of the hierarchy that map to the same line in the second level cache. This provides a limited form of associativity.
- The capacity of the limited on-chip area is better utilized since there will be less duplication between the contents of the first and second level cache.

For direct-mapped split first-level caches each of size x, and a direct-mapped mixed second-level cache of size y (with $y \geq 2x$), exclusive caching holds up to $2x+y$ possible unique lines on-chip. (For $y < x$, the configuration becomes a shared direct-mapped victim cache [4]). The average increase in capacity provided by two-level exclusive caching increases as associativity is added to the system. In the limiting case with the number of L2 sets equal to the number of lines in the L1 cache, exactly $2x+y$ unique lines will always be held on-chip.

Figure 22 shows the performance of *gcc1* with exclusive two-level caching, single-ported L1 and L2 RAM cells, direct-mapped L2 cache, and 50ns off-chip miss service. Comparing this to Figure 9, it appears that exclusive caching does improve the performance of two-level cache hierarchies. Upon comparing it to the 4-way set-associative graph (Figure 5), it is apparent that for *gcc1* the exclusive caching scheme with a direct-mapped second-level cache performs about as well as a system that does not use exclusive caching, but uses a 4-way set-associative second-level cache. Both set-associativity and exclusive caching tend to improve performance; neither is found to be significantly more effective than the other.

Combining set-associativity and exclusive caching can improve performance beyond what either technique alone accomplishes. Figure 23 shows the results when the second-level is 4-way set associative, and exclusive caching is used. The best performance envelope is lower than that in either Figure 5 or 22. Since the number of conflict misses in a 4-way set-associative cache is small, this improvement is primarily due to the increased on-chip capacity provided by exclusive caching. The same trend can be seen for the other traces by comparing Figures 24 to 26 with Figures 6 to 8 (direct-mapped exclusive caching results are not shown for these benchmarks).

Although the contents of the first-level cache and the second-level cache can be mutually exclusive, inclusion between the sum of their contents and a third level of off-chip caching can still be maintained for ease of constructing multiprocessor systems [1] by eliminating on-chip cache lines which are not present off-chip.

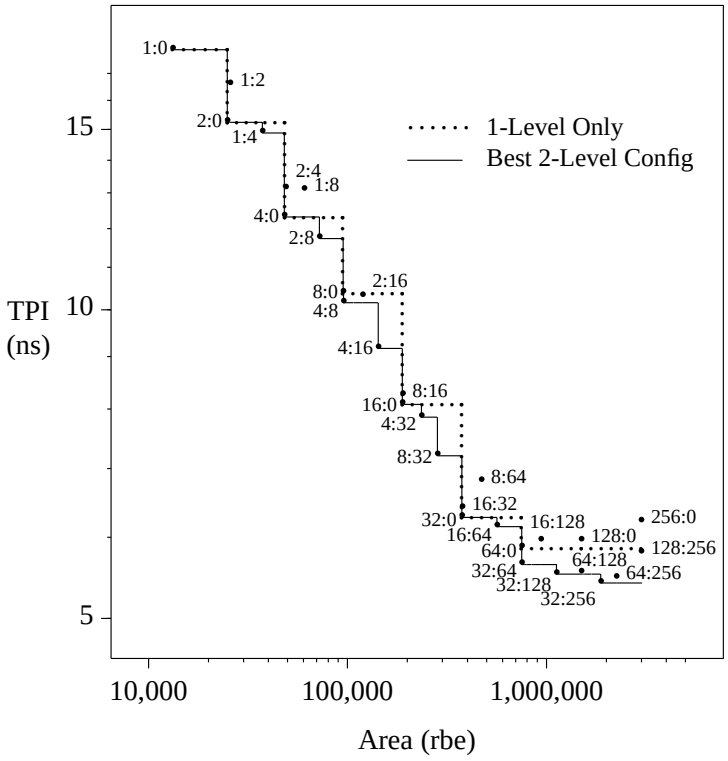

Figure 22: *gcc1*: 50ns off-chip, exclusive direct-mapped L2

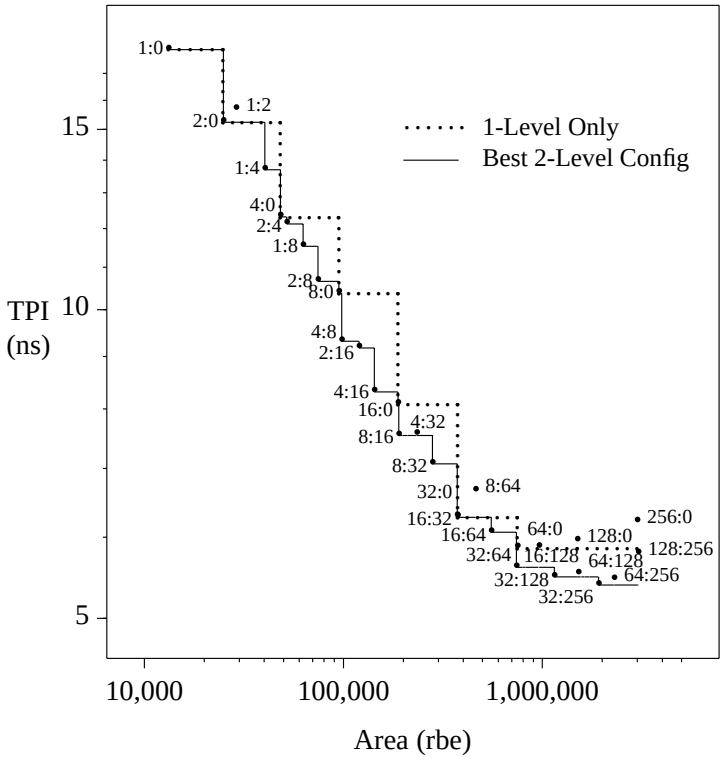

Figure 23: *gcc1*: 50ns off-chip, exclusive 4-way L2

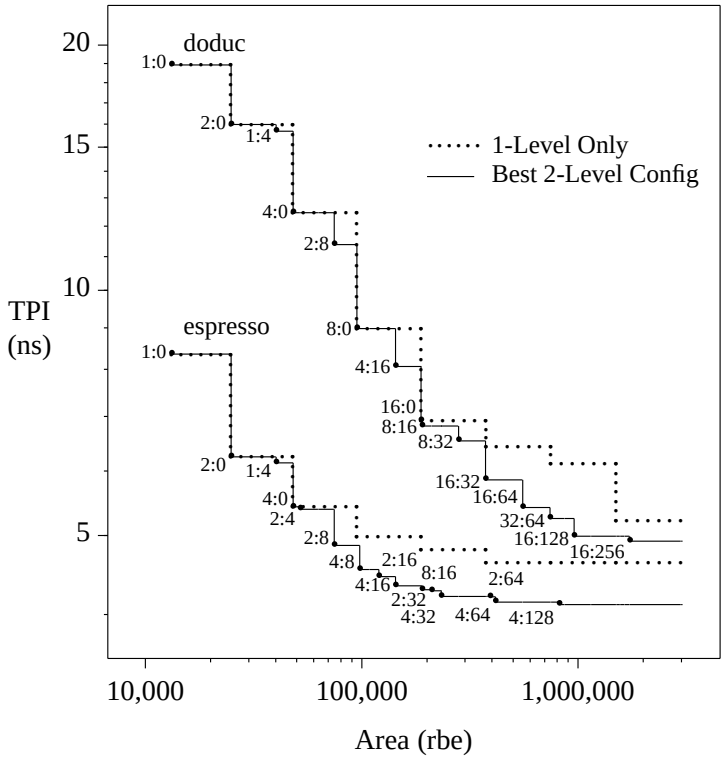

Figure 24: *doduc* and *espresso*: 50ns off-chip, exclusive 4-way L2

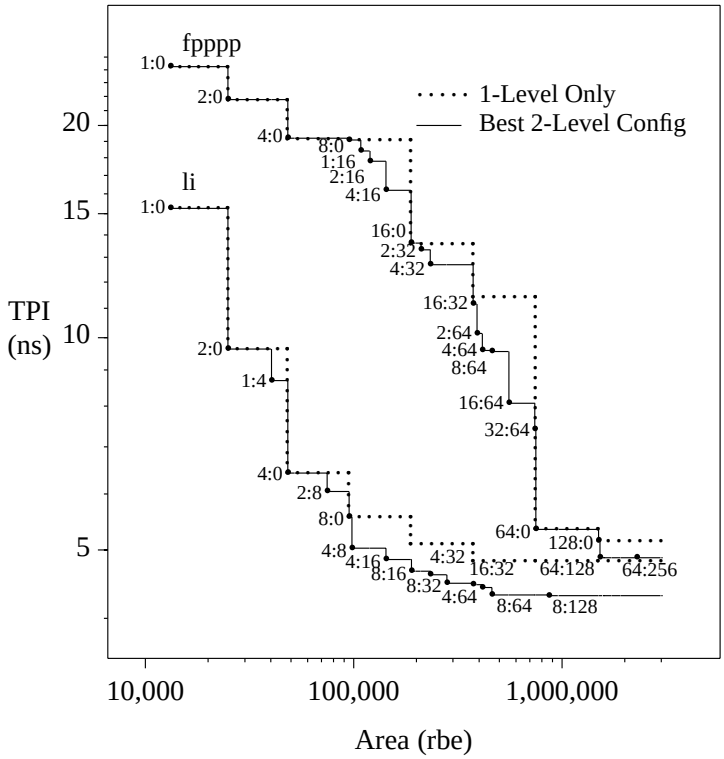

Figure 25: *fpppp* and *li*: 50ns off-chip, exclusive 4-way L2

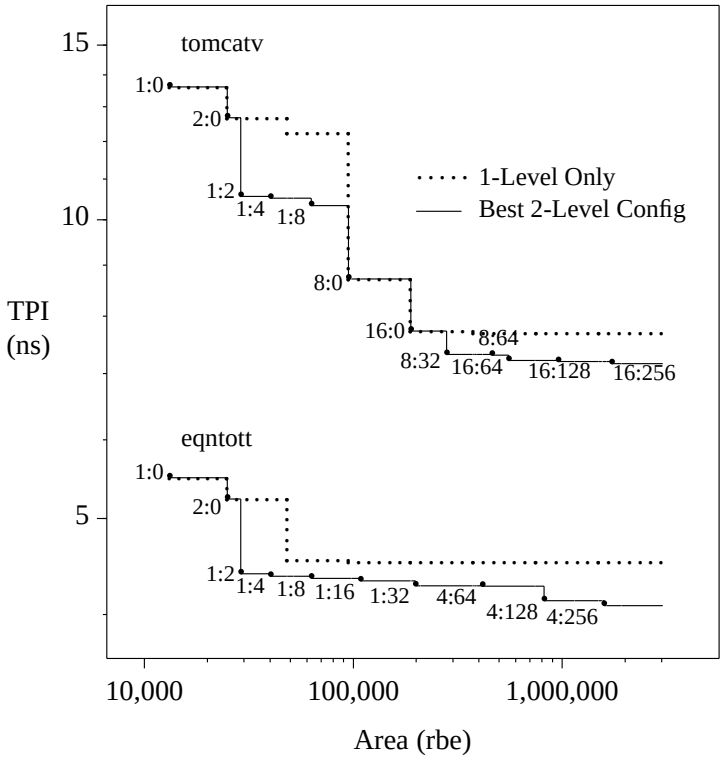

Figure 26: *eqntott* and *tomcatv*: 50ns off-chip, exclusive 4-way L2

9. Conclusions

We have modeled the miss rate, cache area, and cache access time to achieve a solid basis to study on-chip memory system tradeoffs. For simple processors with a board-level (off-chip) cache capable of servicing an on-chip miss in 50ns, our results show that a single-level on-chip cache organization is sufficient for instruction and data caches of up to about 32KB. However, if more area is available for on-chip caching, a two-level configuration with a set-associative second level should be considered.

Two-level on-chip cache hierarchies perform even better in low-cost systems without a board-level cache. In a system that requires 200ns for servicing an on-chip miss, a two-level configuration has better performance once the single-level instruction and data caches grow to about 16KB or larger, for the benchmarks we have simulated.

As more area becomes available for on-chip caching, multiported but larger memory cells become more feasible. If only a single-level cache is used, and enough chip area is present to build caches larger than 32KB, performance can be improved by using memory cells that provide twice the bandwidth but require twice the area as standard memory cells (assuming multiple-instruction issue techniques can make full use of the additional bandwidth.) Two-level caching (with the second level composed of normal small cells) is particularly effective in this case.

Two-level exclusive caching, in which lines that are replaced in the first-level caches are transferred to the second-level cache, was also found to improve the performance of two-level

on-chip caching. Two-level exclusive caching reduces duplication of data between the first-level and second-level caches, while providing additional associativity. Combining this with set-associative second-level caches improves performance even further. This is because the increase in capacity provided by two-level exclusive caching increases as the second level of caching is made more associative.

The time and area models used to gather these results were a vital part of this study. Using these models enabled us to accurately compare the performance of various caching organizations, which would not have been possible with only hit rate data. We believe that both time and area considerations are essential for meaningful on-chip cache studies.

10. Future Work

Probably the biggest limitations of this study have been the assumptions of single-cycle non-pipelined first-level caches and the blocking memory system model.

We expect the extension to multicycle first-level caches to reduce the effectiveness of two-level on-chip caching in baseline configurations since the longer latency of larger first-level cache accesses would not set the cycle time and hence directly affect the instruction issue rate. This would be especially true for applications that can tolerate large load latencies, such as numeric benchmarks.

On the other hand, the extension to non-blocking caches may increase the benefits of a two-level on-chip caching organization if many of the first-level cache misses can be overlapped with useful instruction execution. Especially in the case where the first-level cache is multiported, this can reduce the traffic to a large second-level cache to just the miss requests from the first-level cache. This should allow the second-level cache to be single-ported and hence denser than if it needed ports for all of the references made by instruction execution.

We are currently working on extending this study to systems with both multicycle cache access and non-blocking loads. It will be interesting to see if both conjectures above are true, and to what extent the two effects cancel each other out when both multicycle first-level caches and non-blocking loads are present.

Acknowledgments

The authors wish to thank Davin Wong who worked on simulations and graphs used in an early version of this paper. We would also like to thank Joel Emer and Bob Nix for their very helpful comments on a recent draft of this paper.

References

[1] Jean-Loup Baer and Wenn-Hann Wang. On the Inclusion Properties for Multi-Level Cache Hierarchies. In *The 15th Annual Symposium on Computer Architecture*, pages 73-80. IEEE Computer Society Press, June, 1988.

[2] Anita Borg, Rick E. Kessler, Georgia Lazana, David W. Wall. *Long Address Traces from RISC Machines: Generation and Analysis*. Technical Report 89/14, DEC Western Research Lab, 1989.

[3] Mark D. Hill. A Case for Direct-Mapped Caches. *Computer* 21(12), December, 1988.

[4] Norman P. Jouppi. Improving Direct-Mapped Cache Performance by the Addition of a Small Fully-Associative Cache and Prefetch Buffers. In *Proceedings of the 17th Annual International Symposium on Computer Architechure*, pages 364-373. May, 1990.

[5] Johannes M. Mulder, Nhon T. Quach, Michael J. Flynn. An Area Model for On-Chip Memories and its Application. *IEEE Journal of Solid-State Circuits* 26(2):98-106, Feb., 1991.

[6] Kunle Olukotun, Trevor Mudge, and Richard Brown. Performance Optimization of Pipelined Primary Caches. In *Proceedings of the 19th Annual International Symposium on Computer Architechure*, pages 181-190. May, 1992.

[7] Steven A. Przybylski. *Cache and Memory Hierarchy Design: A Performance-Directed Approach.* Morgan Kaufmann Publishers, Inc., 1990.

[8] Gurindar S. Sohi and Manoj Franklin. High Bandwidth Data Memory Systems for Superscalar Processors. In *Proceedings of the Fourth Conference on Architectural Support for Programming Languages and Operating Systems*, pages 53-62. April, 1991.

[9] SPEC Benchmark Suite. Release 1.0. Oct. 1989.

[10] Tomohisa Wada, Suresh Rajan, Steven A.. Przybylski. An Analytical Access Time Model for On-Chip Cache Memories. *IEEE Journal of Solid-State Circuits* 27(8):1147-1156, Aug., 1992.

[11] Steven J.E. Wilton and Norman P. Jouppi. An Enhanced Access and Cycle Time Model for On-Chip Caches. DEC Western Research Lab, Tech Report 93/5.

WRL Research Reports

''Titan System Manual.''
Michael J. K. Nielsen.
WRL Research Report 86/1, September 1986.

''Global Register Allocation at Link Time.''
David W. Wall.
WRL Research Report 86/3, October 1986.

''Optimal Finned Heat Sinks.''
William R. Hamburgen.
WRL Research Report 86/4, October 1986.

''The Mahler Experience: Using an Intermediate Language as the Machine Description.''
David W. Wall and Michael L. Powell.
WRL Research Report 87/1, August 1987.

''The Packet Filter: An Efficient Mechanism for User-level Network Code.''
Jeffrey C. Mogul, Richard F. Rashid, Michael J. Accetta.
WRL Research Report 87/2, November 1987.

''Fragmentation Considered Harmful.''
Christopher A. Kent, Jeffrey C. Mogul.
WRL Research Report 87/3, December 1987.

''Cache Coherence in Distributed Systems.''
Christopher A. Kent.
WRL Research Report 87/4, December 1987.

''Register Windows vs. Register Allocation.''
David W. Wall.
WRL Research Report 87/5, December 1987.

''Editing Graphical Objects Using Procedural Representations.''
Paul J. Asente.
WRL Research Report 87/6, November 1987.

''The USENET Cookbook: an Experiment in Electronic Publication.''
Brian K. Reid.
WRL Research Report 87/7, December 1987.

''MultiTitan: Four Architecture Papers.''
Norman P. Jouppi, Jeremy Dion, David Boggs, Michael J. K. Nielsen.
WRL Research Report 87/8, April 1988.

''Fast Printed Circuit Board Routing.''
Jeremy Dion.
WRL Research Report 88/1, March 1988.

''Compacting Garbage Collection with Ambiguous Roots.''
Joel F. Bartlett.
WRL Research Report 88/2, February 1988.

''The Experimental Literature of The Internet: An Annotated Bibliography.''
Jeffrey C. Mogul.
WRL Research Report 88/3, August 1988.

''Measured Capacity of an Ethernet: Myths and Reality.''
David R. Boggs, Jeffrey C. Mogul, Christopher A. Kent.
WRL Research Report 88/4, September 1988.

''Visa Protocols for Controlling Inter-Organizational Datagram Flow: Extended Description.''
Deborah Estrin, Jeffrey C. Mogul, Gene Tsudik, Kamaljit Anand.
WRL Research Report 88/5, December 1988.

''SCHEME->C A Portable Scheme-to-C Compiler.''
Joel F. Bartlett.
WRL Research Report 89/1, January 1989.

''Optimal Group Distribution in Carry-Skip Adders.''
Silvio Turrini.
WRL Research Report 89/2, February 1989.

''Precise Robotic Paste Dot Dispensing.''
William R. Hamburgen.
WRL Research Report 89/3, February 1989.

''Simple and Flexible Datagram Access Controls for Unix-based Gateways.''
Jeffrey C. Mogul.
WRL Research Report 89/4, March 1989.

''Spritely NFS: Implementation and Performance of Cache-Consistency Protocols.''
V. Srinivasan and Jeffrey C. Mogul.
WRL Research Report 89/5, May 1989.

''Available Instruction-Level Parallelism for Superscalar and Superpipelined Machines.''
Norman P. Jouppi and David W. Wall.
WRL Research Report 89/7, July 1989.

''A Unified Vector/Scalar Floating-Point Architecture.''
Norman P. Jouppi, Jonathan Bertoni, and David W. Wall.
WRL Research Report 89/8, July 1989.

''Architectural and Organizational Tradeoffs in the Design of the MultiTitan CPU.''
Norman P. Jouppi.
WRL Research Report 89/9, July 1989.

''Integration and Packaging Plateaus of Processor Performance.''
Norman P. Jouppi.
WRL Research Report 89/10, July 1989.

''A 20-MIPS Sustained 32-bit CMOS Microprocessor with High Ratio of Sustained to Peak Performance.''
Norman P. Jouppi and Jeffrey Y. F. Tang.
WRL Research Report 89/11, July 1989.

''The Distribution of Instruction-Level and Machine Parallelism and Its Effect on Performance.''
Norman P. Jouppi.
WRL Research Report 89/13, July 1989.

''Long Address Traces from RISC Machines: Generation and Analysis.''
Anita Borg, R.E.Kessler, Georgia Lazana, and David W. Wall.
WRL Research Report 89/14, September 1989.

''Link-Time Code Modification.''
David W. Wall.
WRL Research Report 89/17, September 1989.

''Noise Issues in the ECL Circuit Family.''
Jeffrey Y.F. Tang and J. Leon Yang.
WRL Research Report 90/1, January 1990.

''Efficient Generation of Test Patterns Using Boolean Satisfiablilty.''
Tracy Larrabee.
WRL Research Report 90/2, February 1990.

''Two Papers on Test Pattern Generation.''
Tracy Larrabee.
WRL Research Report 90/3, March 1990.

''Virtual Memory vs. The File System.''
Michael N. Nelson.
WRL Research Report 90/4, March 1990.

''Efficient Use of Workstations for Passive Monitoring of Local Area Networks.''
Jeffrey C. Mogul.
WRL Research Report 90/5, July 1990.

''A One-Dimensional Thermal Model for the VAX 9000 Multi Chip Units.''
John S. Fitch.
WRL Research Report 90/6, July 1990.

''1990 DECWRL/Livermore Magic Release.''
Robert N. Mayo, Michael H. Arnold, Walter S. Scott, Don Stark, Gordon T. Hamachi.
WRL Research Report 90/7, September 1990.

''Pool Boiling Enhancement Techniques for Water at Low Pressure.''
Wade R. McGillis, John S. Fitch, William R. Hamburgen, Van P. Carey.
WRL Research Report 90/9, December 1990.

''Writing Fast X Servers for Dumb Color Frame Buffers.''
Joel McCormack.
WRL Research Report 91/1, February 1991.

''A Simulation Based Study of TLB Performance.''
J. Bradley Chen, Anita Borg, Norman P. Jouppi.
WRL Research Report 91/2, November 1991.

''Analysis of Power Supply Networks in VLSI Circuits.''
Don Stark.
WRL Research Report 91/3, April 1991.

''TurboChannel T1 Adapter.''
David Boggs.
WRL Research Report 91/4, April 1991.

''Procedure Merging with Instruction Caches.''
Scott McFarling.
WRL Research Report 91/5, March 1991.

''Don't Fidget with Widgets, Draw!.''
Joel Bartlett.
WRL Research Report 91/6, May 1991.

''Pool Boiling on Small Heat Dissipating Elements in Water at Subatmospheric Pressure.''
Wade R. McGillis, John S. Fitch, William R. Hamburgen, Van P. Carey.
WRL Research Report 91/7, June 1991.

''Incremental, Generational Mostly-Copying Garbage Collection in Uncooperative Environments.''
G. May Yip.
WRL Research Report 91/8, June 1991.

''Interleaved Fin Thermal Connectors for Multichip Modules.''
William R. Hamburgen.
WRL Research Report 91/9, August 1991.

''Experience with a Software-defined Machine Architecture.''
David W. Wall.
WRL Research Report 91/10, August 1991.

''Network Locality at the Scale of Processes.''
Jeffrey C. Mogul.
WRL Research Report 91/11, November 1991.

''Cache Write Policies and Performance.''
Norman P. Jouppi.
WRL Research Report 91/12, December 1991.

''Packaging a 150 W Bipolar ECL Microprocessor.''
William R. Hamburgen, John S. Fitch.
WRL Research Report 92/1, March 1992.

''Observing TCP Dynamics in Real Networks.''
Jeffrey C. Mogul.
WRL Research Report 92/2, April 1992.

''Systems for Late Code Modification.''
David W. Wall.
WRL Research Report 92/3, May 1992.

''Piecewise Linear Models for Switch-Level Simulation.''
Russell Kao.
WRL Research Report 92/5, September 1992.

''A Practical System for Intermodule Code Optimization at Link-Time.''
Amitabh Srivastava and David W. Wall.
WRL Research Report 92/6, December 1992.

''A Smart Frame Buffer.''
Joel McCormack & Bob McNamara.
WRL Research Report 93/1, January 1993.

''Recovery in Spritely NFS.''
Jeffrey C. Mogul.
WRL Research Report 93/2, June 1993.

''Tradeoffs in Two-Level On-Chip Caching.''
Norman P. Jouppi & Steven J.E. Wilton.
WRL Research Report 93/3, October 1993.

''Unreachable Procedures in Object-oriented Programing.''
Amitabh Srivastava.
WRL Research Report 93/4, August 1993.

''Limits of Instruction-Level Parallelism.''
David W. Wall.
WRL Research Report 93/6, November 1993.

''Fluoroelastomer Pressure Pad Design for Microelectronic Applications.''
.
WRL Research Report 93/7, November 1993.

WRL Technical Notes

''TCP/IP PrintServer: Print Server Protocol.''
Brian K. Reid and Christopher A. Kent.
WRL Technical Note TN-4, September 1988.

''TCP/IP PrintServer: Server Architecture and Implementation.''
Christopher A. Kent.
WRL Technical Note TN-7, November 1988.

''Smart Code, Stupid Memory: A Fast X Server for a Dumb Color Frame Buffer.''
Joel McCormack.
WRL Technical Note TN-9, September 1989.

''Why Aren't Operating Systems Getting Faster As Fast As Hardware?''
John Ousterhout.
WRL Technical Note TN-11, October 1989.

''Mostly-Copying Garbage Collection Picks Up Generations and C++.''
Joel F. Bartlett.
WRL Technical Note TN-12, October 1989.

''The Effect of Context Switches on Cache Performance.''
Jeffrey C. Mogul and Anita Borg.
WRL Technical Note TN-16, December 1990.

''MTOOL: A Method For Detecting Memory Bottlenecks.''
Aaron Goldberg and John Hennessy.
WRL Technical Note TN-17, December 1990.

''Predicting Program Behavior Using Real or Estimated Profiles.''
David W. Wall.
WRL Technical Note TN-18, December 1990.

''Cache Replacement with Dynamic Exclusion''
Scott McFarling.
WRL Technical Note TN-22, November 1991.

''Boiling Binary Mixtures at Subatmospheric Pressures''
Wade R. McGillis, John S. Fitch, William R. Hamburgen, Van P. Carey.
WRL Technical Note TN-23, January 1992.

''A Comparison of Acoustic and Infrared Inspection Techniques for Die Attach''
John S. Fitch.
WRL Technical Note TN-24, January 1992.

''TurboChannel Versatec Adapter''
David Boggs.
WRL Technical Note TN-26, January 1992.

''A Recovery Protocol For Spritely NFS''
Jeffrey C. Mogul.
WRL Technical Note TN-27, April 1992.

''Electrical Evaluation Of The BIPS-0 Package''
Patrick D. Boyle.
WRL Technical Note TN-29, July 1992.

''Transparent Controls for Interactive Graphics''
Joel F. Bartlett.
WRL Technical Note TN-30, July 1992.

''Design Tools for BIPS-0''
Jeremy Dion & Louis Monier.
WRL Technical Note TN-32, December 1992.
''Link-Time Optimization of Address Calculation on
a 64-Bit Architecture''
Amitabh Srivastava and David W. Wall.
WRL Technical Note TN-35, June 1993.

''Combining Branch Predictors''
Scott McFarling.
WRL Technical Note TN-36, June 1993.

''Boolean Matching for Full-Custom ECL Gates''
Robert N. Mayo and Herve Touati.
WRL Technical Note TN-37, June 1993.