Low-Power and Conflict Tolerant Cache Design

Mathias Spjuth, Martin Karlsson and Erik Hagersten
Uppsala University
Information Technology
Department of Computer Systems
P.O. Box 325, SE-751 05 Uppsala, Sweden
{mspjuth, martink, eh}@it.uu.se

ABSTRACT

The common approach to reduce cache conflicts is to increase the associativity. From a dynamic cache power perspective this associativity comes at a high cost. In this paper we present miss ratio performance and dynamic power estimates for a skewed cache and also for a new organization proposed, the elbow cache. The elbow cache extends the skewed cache organization with a relocation strategy for conflicting blocks. We show that these skewed designs significantly reduces the conflict problems while consuming up to 56% less dynamic power than a comparably performing 8-way set-associative cache.

1. INTRODUCTION

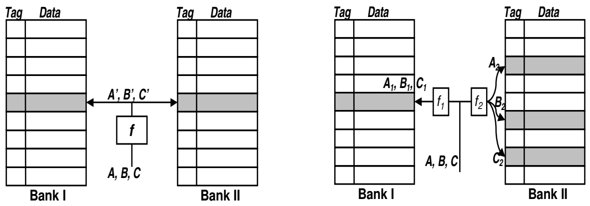
The emerging trends towards chip multiprocessors and *Symmetric Multi-Threading* (SMT) make shared on-chip caches increasingly common. With multiple threads sharing a cache the likelihood of destructive sharing increases [9, 19], which will increase the importance of conflict-tolerant caches. Skewed 2-way associative caches have been shown to perform comparably with a 4-way set-associative cache [24]. In this paper we introduce a new skewed cache design with a replacement strategy based on victim relocation and show that this design outperforms a conventional skewed cache. This scheme is called the *elbow cache.*

Higher associativity usually comes with an increased cost in dynamic power consumption since, in a conventional n-way set-associative cache, n different locations are probed in the cache for each access. A 2-way associative skewed cache on the other hand only probes two locations in each access, thereby consuming less energy than the similarly performing 4-way set-associative cache. In this paper we quantify the dynamic power consumption of a skewed cache and compare it to both set-associative caches and the proposed elbow cache organizations.

The contribution of this paper is threefold:

- We introduce a new cache scheme, the elbow cache, that allows selected victims in a skewed cache to be relocated.
- We present a power consumption evaluation that show that the skewed and the elbow caches are viable low power alternatives to a highly associative cache.
- We present various implementation optimizations that could be applied to skewed caches such as timestamp-based replacement, pass-transistor based skewing functions and interleaved way-banks.

Technical report 2004-024
Department of Information Technology
Uppsala University



(a) 2-way set-associative cache
(b) Skewed cache

Figure 1: Describing a skewed cache organization.

2. THE SKEWED ASSOCIATIVE CACHE

The skewed associative cache was first introduced by Seznec et. al. [24, 26]. A skewed cache is conceptually divided into multiple sub-banks each indexed by a different hash function. For a skewed 2-way associative cache, the idea is that cache blocks that map to the same location in one of the banks are likely to map to different locations in the other. This is illustrated in Figure 1 with cache blocks A, B and C conflicting in the set-associative case but not in the skewed case. Mapping conflicts of this type, are effectively reduced by a skewed cache. The effectiveness is of course highly dependent on the skewing functions. Thorough discussions on skewing functions are provided in [7, 17, 24, 25, 26]. We have throughout this paper used the XOR-based skewing functions proposed by Bodin and Seznec [7], where two subsets, A_1 and A_2, of the address bits are XOR'ed together as described below. Let σ be the one bit rotational shift[1] on an n-bit number and $\oplus$ represents a bitwise XOR. The skewing functions are then:

$$f_1(A) = A_1 \oplus A_2 \quad (1)$$
$$f_2(A) = \sigma(A_1) \oplus A_2. \quad (2)$$

[1] The least significant bit shifted becomes the most significant bit.

3. TIMESTAMP-BASED REPLACEMENT

One of the challenges with skewed caches is the replacement algorithm. Since there are no fixed sets, any combination of victim pairs, one block from each bank, is possible. This makes it difficult to implement an exact ordering-based replacement algorithm like LRU [5]. Instead, approximative algorithms like the *not recently used, enhanced* (NRUE) has been proposed [7]. Another way of providing an ordering between cache blocks is to store a timestamp for each block. At replacement, the block with the least recent timestamp is chosen for eviction. Timestamp-based replacement algorithms have been found to outperform NRUE-replacement for skewed caches [28] at the cost of extra area to hold the timestamps.

In this paper we use 5-bit timestamps based on *Cache Allocation Ticks* (CAT), first introduced in [15], as replacement metric for the skewed caches. The scheme uses a k-bit[2] global counter which is incremented each time a new cache block is allocated in the cache. To limit the area overhead of the timestamp array, the timestamps are composed of only a few of the most significant bits of the CAT-counter value. A sensitivity study has shown that 5 bits are sufficient [28].

For every cache hit, the most significant bits of the current CAT-counter become the new timestamp for the cache block that was accessed. Note that the new timestamp simply overwrites the old so no read-modify-write operation is needed. At the time of a replacement, the timestamps of the cache blocks are retrieved and the CAT-distance d for each block is calculated as shown in Figure 2. T_{curr} is the current value of the global CAT-counter and n is the number of bits in the timestamp. The cache block with highest distance is chosen for replacement. CAT-based timestamps give a

$$d = \begin{cases} T_{curr} - T_{st} & T_{curr} \geq T_{st} \\ T_{curr} + 2^n - T_{st} & T_{curr} < T_{st} \end{cases}$$

Figure 2: Calculating the timestamp distance.

coarse recency ordering that is still fine-grained enough to distinguish between recent and old data. The advantage of using cache allocations instead of cache references or cycles is that the resolution is adjusted according to the miss ratio of the cache.

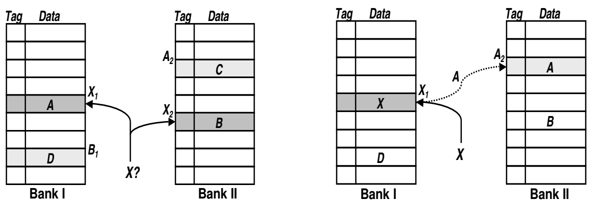
4. THE ELBOW CACHE

The use of timestamps as replacement metric enables a global, temporal ordering of all blocks in the cache. This ordering property is exploited by the scheme proposed in this paper, the elbow cache.

4.1 Victim Relocation

An elbow cache [28] extends a skewed organization by carefully selecting its victim and, in the case of a conflict, move the conflicting cache block to its alternate location. In a sense, the new data item "uses its elbows" to make space for conflicting data instead of evicting it.

In the common case, a cache hit, the elbow cache works just as a CAT-based skewed cache and updates the timestamp



(a) Elbow cache miss (b) Elbow relocation

Summary of elbow cache actions:

Hit:

- **The data and the tag of the two possible locations of X is read.**
- **The hit is detected**
- **The current time is written to the timestamp associated with X.**

Miss:

- **The data and the tag of the two possible locations of X is read.**
- **X is detected as a cache miss.**
- **The alternate locations of A and B, C and D are computed and the timestamps of A, B, C and D are retrieved.**
- **The oldest cache block C is chosen as victim.**
- **If the victim, C, resides in a alternate location not accessible to X, another block, A, that is accessible to X, is moved there and X is allocated where A used to be.**

Figure 3: Describing an elbow cache replacement.

for the accessed block. In the case of a cache miss there are four possible victims: The two *primary* locations, hashed to by the address that generated the miss, and also the two alternate locations of the data currently in the primary locations. Figure 4.1a illustrates this, where blocks A and B are in the primary locations (i.e. where the new data item X can be placed) and C and D are the alternate locations of A and B respectively. Which cache block to select for eviction is determined by retrieving and comparing the four timestamps associated with the blocks. Among the four blocks, the one with the oldest timestamp is selected as the victim. If the victim resides in one of the alternate locations, data from the primary location is moved there, overwriting the victim. The primary location is thereby freed for new data to be filled into the cache (Figure 4.1b). Note that relocation may only occur in the case of a cache miss and only if one of the alternate locations (C or D) is chosen as victim. Also note that this relocation is off the critical path of a read access. A more detailed discussion on the proposed implementation is provided in Section 8.

4.2 Reducing the number of relocations

A relocation is costly from a power perspective, since a full cache block must be read and written. By imposing restrictions on when to relocate, the power consumption can be reduced. For all the simulation results reported in this paper the following restriction has been applied: First of all,

[2]The maximum counter value is four times the number of blocks in the cache ($k = log_2(blocks) + 2$). This is usually large enough to avoid *aliasing* of the replacement metric.

at most one relocation per four misses is allowed.[3] This prevents extensive power consumption during high miss ratios [20]. Secondly, we make use of the temporal information provided by our timestamps and only consider blocks with a short distance for relocation. Any other block would be "old" and therefore less likely to be reused in the near future. In our simulations, a relocation can occur only if the selected primary victim has a distance of 3 or less. We find that this significantly reduces the number of relocations without having any substantial impact on hit-ratio performance.

5. METHODOLOGY

The focus of this paper is comparing power consumption for different cache organizations. We have therefore limited our study to cache performance and do not present execution time approximations. Our performance results are reported in terms of miss ratio reduction compared with a conventional 2-way set-associative cache of the same size. Estimations of the dynamic power and access time of the elbow organization was obtained by modifying [20] the Cacti 3.2 model [27]. We have throughout this paper assumed a process technology of 100nm and simulated caches with a single read/write port.

We used the Simics full system simulator [21] to simulate our different workloads. The Simics setup simulated a SPARC-V9 system running an unmodified Solaris 9 operating system. Our evaluation is based on the SPECint 2000 and SPLASH-2 benchmark suites. The input data sets for SPLASH was taken from [23] while SPEC was run with the reference dataset.

Our simulated system is a four-way multi-threaded processor with a single data cache shared by all threads. We ran the SPLASH benchmarks with four threads to simulate a single application with several threads sharing the same address space and sharing the same cache. The SPEC benchmarks were used to simulate a machine with several unrelated applications running at once, also sharing the same cache. To create multi-cpu benchmarks from the SPEC-suite we mixed four different sub-benchmarks from the suite into a single benchmark. The benchmark mix is shown in Table 1, Appendix A. The SPLASH benchmarks were studied only in the parallel region of the code, and the caches were all warmed up before the measurements started. For the SPEC runs we fast-forwarded 10 Billion instructions into the benchmark before warming up the caches for 500 Million instructions. The measurements was then obtained for the following 2 Billion instructions.

The skewed and elbow caches are compared to a number of conventional set-associative caches with LRU replacement and various degrees of associativity. We present results for a 32KB data cache with 64-byte blocks. Results for 16KB and 64KB caches are reported in Appendix B.

6. CACHE PERFORMANCE RESULTS

In this section we present miss ratio performance results for a 2-way skewed cache, a 2-way elbow cache as well as 4 and 8-way set-associative caches. We present our performance evaluation in the terms of miss ratio reduction compared to a 2-way set-associative cache, where miss ratio reduction (MRR) is computed as MRR_X=1-$\frac{Missratio_X}{Missratio_{2-way}}$. Table 2, Appendix A shows the miss ratios for the 2-way set-associative cache.

[3]We use a sliding window that allows for up to 16 replacements during the last 64 misses.

6.1 CAT-Timestamp Metric Performance

Our first results are a comparison between different replacement metrics for skewed caches. We simulate a simple uniprocessor system and run the SPECint benchmarks. We test a perfect LRU replacement, a NRUE replacement and replacement based on CAT-timestamps (TS). The results are shown, as miss ratio reductions compared to a two-way set-associative cache, in Figure 6, were we also included a four-way set-associative cache for comparison. As claimed earlier, the results show that timestamp replacement is better than NRUE-replacement.

6.2 Shared Cache Performance

The results of the SPLASH simulations are shown in Figure 4. When the set-associativity is increased from 2-way to 4-way, we observe a substantial miss ratio reduction. The reduction is very limited beyond 4-ways for most benchmarks. We do note however, that both the skewed and the elbow cache perform well in comparison to the other two caches.

The SPEC benchmarks results are shown in Figure 5. Unlike the SPLASH benchmarks, the improvements over 2-way set-associativity are quite small, around 10—20%. On the other hand, the 2-way set-associative cache have significantly higher miss ratio to begin with. Like in the SPLASH case above, 4-way set-associativity captures most of the conflicting data. Benchmarks 1, 3 and 7 (BMK_01, BMK_03 and BMK_07) are exceptions however, and show a much higher reduction for the 8-way, skewed and elbow caches.[4] This is indicating a high amount of conflict misses and that conflict reduction beyond 4-way set-associativity is worthwhile. The results also show that both skewed and our proposed elbow cache indeed fulfills these requirements.

7. POWER ESTIMATES

The power consumption of current architectures consists almost entirely of dynamic power. However as process technology advances the static power consumption is projected to make up an increasingly large portion of the total power.

In this section we present dynamic power estimations for the skewed cache, the elbow cache and the 2, 4 and 8-way set-associative caches. The static power is expected to be the same for all of these designs. The proposed architectures may very well be combined with leakage power reduction techniques [10, 16, 33]

7.1 Dynamic Power

As process technology scale to smaller feature sizes, the power consumed by the cache's sense-amplifiers makes up an increasingly large fraction of the total power consumed by a cache read. This can be observed by running Cacti with different technologies. Since the number of words read out in parallel is a factor in how many bit-lines and sense-amplifiers that needs to be activated during an access, associativity comes at an increasing cost in power.

[4]The higher miss ratios of the 4-way cache for BMK_01 and BMK_03 might seem un-intuitive but results from non-optimal replacement (LRU).

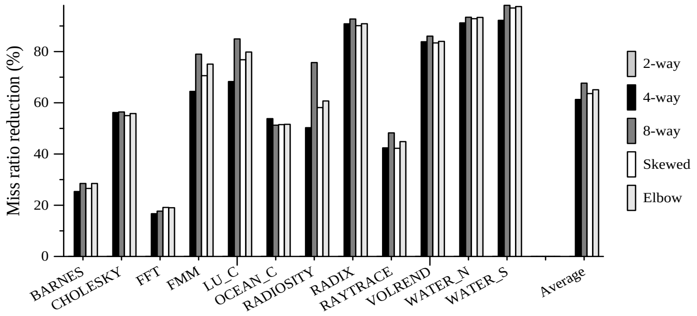


Figure 4: Miss ratio reduction for one multi-treaded application sharing one data cache (SPLASH), compared with a 2-way set-associative cache (miss ratios listed in Appendix A).

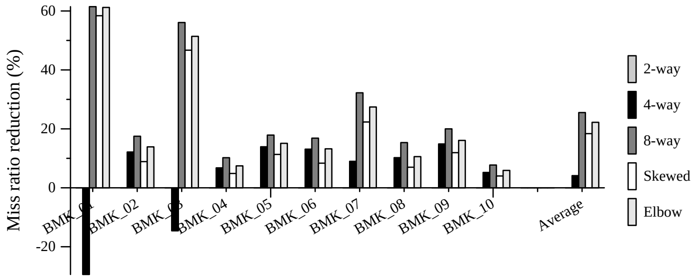


Figure 5: Miss ratio reduction for four single-threaded applications sharing one data cache (SPECint), compared with a 2-way set-associative cache (miss ratios listed in Appendix A).

A common approach to reduce the access time and power consumption of a cache is to divide the SRAM-cell array into sub-arrays. Depending on the optimal aspect-ratio and sub-array size, bit-line-wise and word-line-wise divisions are applied. Hence an 8-way associative cache is not necessarily twice as wide as an 4-way associative cache. In our evaluation, the sub-array divisions used for each particular organization was given by Cacti's optimization function. This function takes power as well as access time, area and aspect ratio into account.

The dynamic power consumed by a cache can be computed by equation (3) below, where P_{Ld} denotes the power consumed by loads and P_{St} represents the power associated with stores:

$$P_{Dynamic} = P_{Ld} + P_{St} \quad (3)$$

$$P_{LdConv} = P_{Hit} + Ld_{MissFrac} * P_{Fill} \quad (4)$$

$$P_{LdElbow} = P_{LdConv} + Ld_{MissFrac} * P_{Relocation} * freq_{Reloc} \quad (5)$$

As can be seen in Figure 7, the per lookup power for a cache hit, P_{Hit}, in the skewed/elbow cache[5] is 17% higher than for a 2-way set-associative cache, but only 75% and 44% of the power consumption of a 4-way and 8-way cache respectively.

To simplify our study we have opted to exclude store power consumption.[6] The P_{Fill} term is the additional power consumed while reading a new block from the L2 cache and filling it into the L1 cache on a cache miss. The total dynamic load power for a conventional (or skewed cache) can be expressed as (4). More details of our implementation assumptions are presented in Section 8.

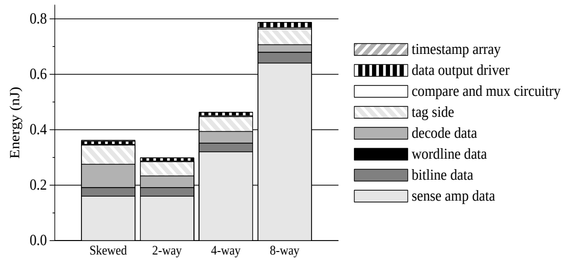


Figure 7: Dynamic power breakdown per hit for a 32KB cache.

7.2 Elbow Relocation Power

Due to the energy associated with a relocation in the elbow cache, the cache power consumption is also dependent on how often relocations occur. This, in turn, depend on the miss ratio. Hence, in the case of zero percent miss ratio and therefore no relocations, the elbow cache will consume the same amount of power as a skewed cache. For non-zero miss rates the power consumption will increase linearly with the miss rate and relocation frequency. Formula (5) shows the average load power for the elbow cache. $freq_{Reloc}$ is the probability of a relocation whenever a miss occurs.

7.3 L2 Load Power

It is also important to consider the power cost of accesses to the next level in the memory system. To account for this we have included the power consumption of a 64-byte block load from an 8-way 1MB level two (L2) cache, with serial tag and data array lookups, in the fill power term (P_{Fill}). The traffic between a large L2 cache and the lower levels of

[5]The power estimates are identical if both caches use timestamp-based replacement.

[6]Since the tag and data is looked up sequentially in the case of a write, the write power is approximately the same for all organizations.

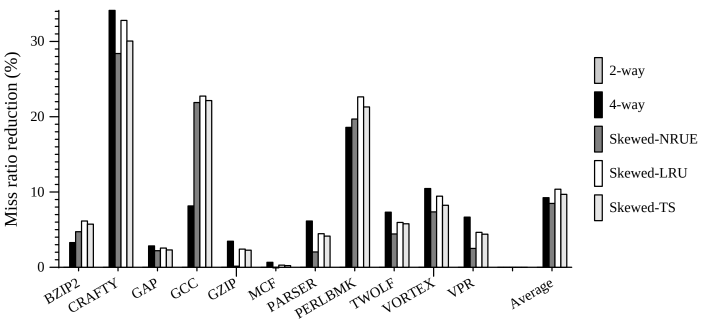
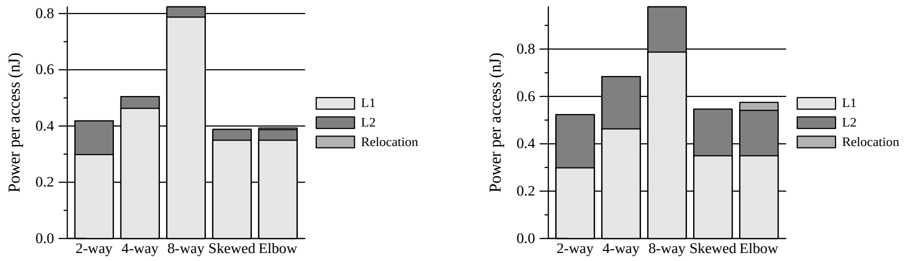


Figure 6: Miss ratio reduction for 32K uniprocessor data caches (SPECint), compared with a 2-way set-associative cache.



(a) Average power breakdown for SPLASH.

(b) Average power breakdown for SPEC.

Figure 8: Load power consumption estimates for SPEC benchmarks and different cache architectures.

the memory hierarchy should be similar for all the different L1 cache systems. The power consumption of other parts of the memory hierarchy, like the bus and memory, will therefore not affect our evaluation and is excluded.

Figures 8a and 8b show the average power-per-access cost of the SPLASH and SPEC simulations broken down into level one and level two costs. The power consumption for relocations in the elbow cache is also included. As can be seen in the graphs, the total power consumption is highly dependent on the miss ratio. For example, the total cache system power consumption for SPLASH is higher for the 2-way associative cache than for the skewed and elbow caches, although it is lower from an L1 power perspective.

8. SKEWED AND ELBOW IMPLEMENTATION

This section describes the assumed implementation of the skewed/elbow cache on which our modified Cacti-model was based.

8.1 Issues with Skewed Caches

There are a few issues surrounding skewed caches that one must bear in mind. Since the elbow cache is an extension of the skewed cache concept it too will inherit these issues.

The additional critical-path delay introduced by the skewing functions is one of the drawbacks. We will discuss how this problem can be addressed in the following section. Another drawback, primarily affecting L1 caches, is that an additional number of physical address bits are required by the skewing functions to calculate the index. This makes virtually-indexed caches infeasible for realistic cache sizes.

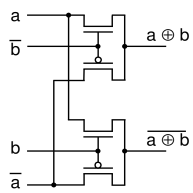


Figure 9: Pass-transistor based skewing functions.

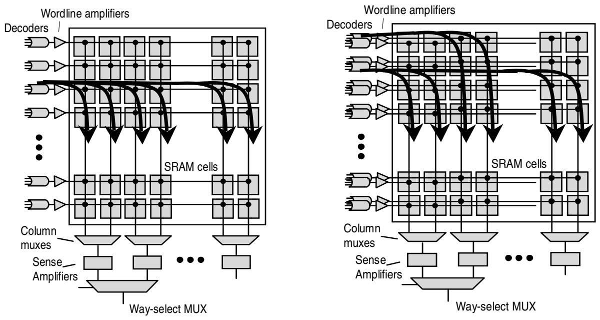
8.2 Additional Structures

Our skewed cache implementation requires some additional structures compared to a set-associative cache. First a separate timestamp structure must be provided to hold the timestamp of each cache block in the cache. Assuming a 64-byte cache block size and 5-bit timestamps this gives an extra area cost of approximately 2% of the data array. The timestamps are separated per bank so that the first bank has one timestamp array and the second has another.

Secondly, the skewing functions must be added. In all of our experiments we have used the hash-functions (1) and (2) described earlier in Section 2, which are both XOR-based. To reduce the skewing delay addition to the critical path we have opted to use a pass-transistor layout [31] for the XOR-functions (Figure 9). If bits b and b-bar in the address are available before a and a-bar, this design can be used to reduce the skew-delay to a fraction of that of a logical gate. If the bit b is taken from the part of the address that does not need to be translated via the TLB, this criteria is fulfilled. This solution has limited the number of bits that is available to the skewing function. If we assume a page size of 8KB,[7] an address A can be written as $A = \{a_N, ..., a_0, b_{12}, ..., b_0\}$ were bits b_n belong to the physical part and a_n to the virtual part. Bits $\{b_5, ..., b_0\}$ are used to index the word in each block. For our 32KB, 2-way elbow cache that has 256 blocks in each bank, bits $\{a_0, b_{12}, ..., b_6\}$ would be XOR'ed with bits $\{a_9, ..., a_1\}$ to produce an ideal skewing function.[8] If we instead only apply the XOR-function to the sets $\{b_{12}, ..., b_6\}$ and $\{a_8, ..., a_1\}$, and let a_0 fill in the missing most significant bit, we get a more restricted skew, but one were the pass-transistor layout is possible. All performance results reported in this paper use this type of skewing function, since the performance of skewed caches is not very sensitive to this type of restrictions [30]. In our case, the pass-transistor design yielded an access time overhead for the skewing function of less than 0.7%. This is however dependant on the width of the pass-transistors.

8.3 Physical Layout

In the physical layout of our skewed cache, the two logically separate way-banks are interleaved into one physical bank, so that bit-line n from the first bank is placed close to bit-line n from the second bank, similar to the layout that of a 2-way set-associative array (Figure 10). By doing so, we get a better aspect-ratio on the array and simplify the design of the way-select logic. This also helps in implementing the relocation functionality in the elbow cache, since the bit-lines of any two bit-cell pairs affected by the move will be physically adjacent. To be able to look up both logical banks concurrently, two decoders are required. Each decoder is connected to half of the cells in the array. We assume that the additional word-lines can be routed in a different metal layer making extra word-line spacing unnecessary. It is still possible to make sub-array divisions with this arrangement, but when splitting an array bit-line-wise, restrictions must be made on the skewing functions so that an address is always mapped into the same sub-array by both way-functions. In the particular layout for our 32KB elbow cache however, these restrictions are less rigid than the ones already posed by our solution to reduce the skewing delay, and do therefore not affect performance.



(a) 2-way set-associative cache. (b) Skewed cache.

Figure 10: The physical implementation.

[7]Including page coloring.

[8]The inverse address bits also needed by the address decoder are obtained in the same way.

9. ACCESS TIME

The access time of a cache is very important for system performance, since it may affect the processor cycle time [11, 12]. By changing the sub-array division of the cache, speed can be traded for power and vice versa. We have chosen a configuration that enables our skewed/elbow cache to have a shorter access time, including skewing delay, than that of the 4-way or 8-way set-associative caches. Table 3, Appendix A, lists the access time and power consumption results, together with the sub-array configurations used.[9] Our Cacti results yield a skewed cache with an access time that is approximately the same as the 4-way set-associative cache and 12% lower than the 8-way. Note that this proposal is focused on timing-critical caches where tag and data are accessed in parallel. If the latency of sequential tag and data lookup is tolerable, such a design is of course far better from a dynamic power perspective. This may however result in up to 60% higher access time [32].

10. RELATED WORK

Some of the earlier work that has addressed the issue of reducing cache conflicts [12] is presented here. Jouppi [14] presented the *victim cache*, primarily for use in direct-mapped caches. Topham and Gonzales [29] studied the use of hash-functions for indexing a cache. Agarwal and Pudar [2] suggested *column-associativity* for improving direct-mapped caches. Seznec and Bodin [26] pioneered the work on *skewed-associative* caches. Further work on skewed caches is presented in some of their later papers [7, 24, 25]. The elbow cache organization was first evaluated from a pure performance centric view in [28] that also showed, that by extending the elbow cache idea to n relocation steps, a further reduction in miss ratio can be achieved. Karlsson and Hagersten [15] showed the usefulness of CAT-timestamps as block survival time metric in the RASCAL-cache.

Recently many proposals has been made to reduce the dynamic power consumption in various memory structures on-chip [1, 4, 6, 8, 13, 18, 34]. To overcome the latency penalty of a sequential access, Powell et. al. proposes *way-prediction* [22], to predict which way to access. This method has the drawback of multiple hit times, a correctly predicted hit and a mispredicted hit, and a significant prediction table size.

Another power saving approach is to shutdown unnecessary parts of the cache during periods of low utilization. Albonesi proposed the idea of Selective cache ways [3] where unneeded ways in a cache are shutdown.

11. CONCLUSION

Skewed caches are often very effective in reducing conflicts. The multi-threaded nature of emerging processors and the shared caches that follow, are likely to increase the need for such highly conflict tolerant designs. In this paper we revisit the skewed caches and evaluate them in the context of power consumption. A timestamp-based replacement metric is proposed. We also introduce a new cache organization called the elbow cache. It extends a skewed cache by relocating conflicting cache blocks to their alternate locations.

[9]Since only half as many bit cells are connected to each decoder in the skewed cache, the word line delay of the SRAM-array is slightly shorter than for a similar set-associative cache, compensating for the skewing delay.

We present a dynamic power evaluation for both the skewed and elbow caches and compare them to traditional set-associative caches. Since the skewed and elbow caches only look up two locations on a cache hit, their per access power consumption is significantly lower than that of an 8-way set-associative cache. When also taking L2 power into account, we find that the skewed and elbow cache configurations consume less total power than both 4-way and 8-way set-associative cache configurations.

From our miss ratio evaluation we conclude that the timestamp based skewed cache shows surprisingly good performance compared with set-associative caches. The elbow cache further reduces the miss ratio at the cost of more complexity. Considering the constrained power budgets of today's processors, a skewed or an elbow cache should make an interesting alternative to those more traditional designs.

12. ACKNOWLEDGMENTS

We would like to thank Mark Hill for contributing to the initial discussion leading up to the idea of relocation. We would also like to thank Magnus Ekman for help with Cacti. This work is funded by the PAMP research program, supported by the Swedish Foundation for Strategic Research.

APPENDIX

A. TABLES

	SPECint mix		SPECint mix
BMK_01	BZIP2_GFX GAP GZIP_LOG PERLBMK_M	BMK_06	GCC_166 GZIP_LOG PERLBMK_P VORTEX_BE2
BMK_02	VORTEX_BE1 BZIP2_SRC GCC_200 GZIP_RND	BMK_07	CRAFTY GCC_EXPR0 GZIP_SRC PERLBMK_SM2
BMK_03	PERLBMK_SM1 VORTEX_BE3 EON_COOK GCC_INTEGR	BMK_08	VPR_PLAC EON_KAJI GCC_SCILAB PARSER
BMK_04	MCF PERLBMK_SM3 VPR_ROUT EON_RUSH	BMK_09	PERLBMK_SM4 BZIP2_PGR GCC_200 GZIP_SRC
BMK_05	GZIP_GFX PERLBMK_D TWOLF BZIP2_PGR	BMK_10	GAP MCF TWOLF CRAFTY

Table 1: The mix of four SPEC sub-benchmarks used to create the four-cpu multiprocessor benchmarks.

B. DATA FOR 16K AND 64K CACHES

Figures 11–13 show the same results as presented earlier in this paper, but for a 16K L1-cache, and Figures 14–16 show the same results for a 64K L1-cache.

SPECint mix	16KB	32KB	64KB
BMK_01	18.93%	4.91%	1.72%
BMK_02	9.70%	6.74%	4.81%
BMK_03	21.46%	5.70%	2.12%
BMK_04	24.01%	18.99%	15.46%
BMK_05	13.05%	10.09%	8.00%
BMK_06	12.82%	8.95%	5.83%
BMK_07	30.17%	9.26%	4.90%
BMK_08	14.70%	10.44%	7.64%
BMK_09	13.69%	9.60%	6.20%
BMK_10	24.97%	20.79%	17.84%
Average	18.35%	10.55%	7.45%
SPLASH-2	16KB	32KB	64KB
BARNES	3.13%	2.49%	1.28%
CHOLESKY	7.46%	5.75%	1.72%
FFT	8.66%	6.62%	3.81%
FMM	3.32%	2.21%	0.75%
LU_C	4.60%	2.96%	0.78%
OCEAN_C	6.09%	5.58%	2.41%
RADIOSITY	5.99%	3.94%	1.51%
RADIX	17.12%	15.40%	0.65%
RAYTRACE	7.28%	5.06%	3.10%
VOLREND	—	6.95%	0.91%
WATER_N	—	4.97%	1.02%
WATER_S	6.49%	5.57%	0.85%
Average	7.01%	5.62%	1.57%

Table 2: Miss ratios for the 2-way set-associative reference cache.

C. REFERENCES

[1] J. Abella and A. Gonzalez. Power Efficient Data Cache Designs. In *Proceedings of the 21st International Conference in Computer Design*, 2003.

[2] A. Agarwal and S. D. Pudar. Column-Associative Caches: A Technique for Reducing the Miss Rate of Direct-Mapped Caches. In *Proceedings of the 20th International Symposium on Computer Architecture*, pages 179–190, May 1993.

[3] David H. Albonesi. Selective cache ways: On-demand cache resource allocation. In *International Symposium on Microarchitecture*, pages 248–, 1999.

[4] B. Bannon and T. N. Vijaykumar. Reactive-Associative Caches. In *Proceedings of the International Conference on Parallel Architectures and Compilation Techniques*, 2001.

[5] L. A. Belady. A study of replacement algorithms for a virtual storage computer. *IBM Systems Journal*, 5:78–101, 1966.

[6] N. Bellas, I. Hajj, and C. Polychronopoulos. Using dynamic management techniques to reduce energy in high-performance processors. In *Proceedings of the 1999 International Symposium on Low Power Electronics and Design (ISLPED)*, 1999.

[7] F. Bodin and A. Seznec. Skewed associativity improves program performance and enhances predictability. In *IEEE Transactions on Computers*, May 1997.

[8] Magnus Ekman, Fredrik Dahlgren, and Per Stenström. TLB and Snoop Energy-Reduction using Virtual Caches in Low-Power Chip-Multiprocessors. In *Proceedings of the 2002 International Symposium on Low Power Electronics and Design (ISLPED)*, 2002.

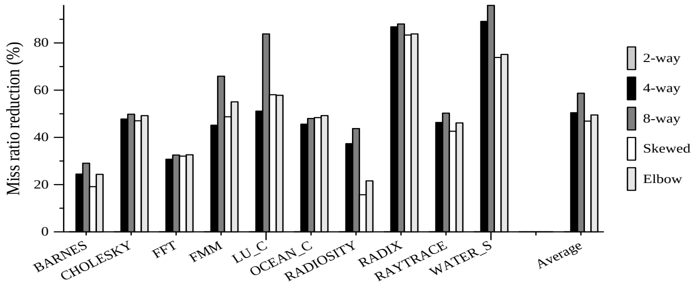


Figure 11: Miss ratio reduction for one multi-treaded application (SPLASH) sharing one 16K data cache with different configurations, compared with a 2-way set-associative cache.

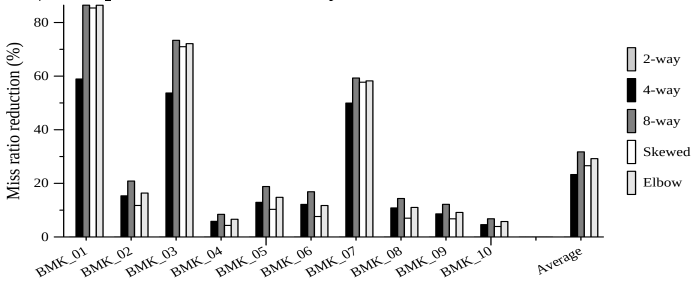


Figure 12: Miss ratio reduction for one multi-treaded application (SPECint) sharing one 16K data cache with different configurations, compared with a 2-way set-associative cache.

Arch.	Access Time [ns]	Power [nJ]	Sub-array configuration Ndwl-Ndbl-Nspd–Ntwl-Ntbl-Ntspd
16KB			
2-way	0.681	0.269	4-1-1–1-2-1
4-way	0.709	0.428	8-1-1–1-2-1
8-way	0.806	0.773	16-1-1–1-4-1
Skewed	0.688	0.294	4-1-1–1-2-1
32KB			
2-way	0.744	0.298	4-2-1–1-2-2
4-way	0.741	0.463	8-1-1–1-2-1
8-way	0.830	0.787	8-1-1–1-2-1
Skewed	0.740	0.349	4-2-1–1-2-2
64KB			
2-way	0.830	0.349	4-2-1–1-4-2
4-way	0.860	0.518	8-1-1–1-2-1
8-way	0.867	0.839	8-1-1–1-2-1
Skewed	0.819	0.406	2-2-1–1-2-4

Table 3: Access time, hit power consumption and configuration for 100nm process caches.

[9] Andrew Erlichson, Basem A. Nayfeh, Jaswinder Pal Singh, and Kunle Olukotun. The Benefits of Clustering in Shared Address Space Multiprocessors: An Applications-Driven Investigation. In *Supercomputing*, 1995.

[10] K. Flautner, N. Kim, S. Martin, D. Blaauw, and T. Mudge. Drowsy Caches: Simple Techniques for Reducing Leakage Power. In *Proceedings of the 29th Annual International Symposium on Computer Architecture (ISCA'02)*, 2002.

[11] R. Heald, K. Shin, V. Reddy, I.-F. Kao, M. Khan, W. L. Lynch, G. Lauterbach, and J. Petolino. 64kB Sum-Addressed-Memory Cache with 1.6ns Cycle and 2.6ns Latency. *IEEE Journal of Solid-State Circuits 33*, page 16821689, 1998.

[12] M.D. Hill. *Aspects of Cache Memory and Instruction Buffer Performance.* PhD thesis, University of California, Berkeley, 1987.

[13] J. Jalminger and P. Stenström. Improvements of Energy-Efficiency in Off-Chip Caches by Selective Prefetching. *Microprocessors and Microsystems*, 2001.

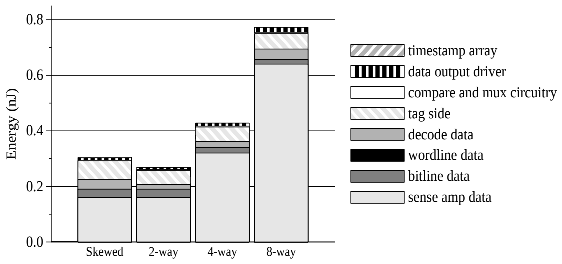
[14] N. P. Jouppi. Improving Direct-Mapped Cache Performance by the addition of a Small Fully-Associative Cache and Prefetch Buffers. In *Proceedings of the 17th International Symposium on Computer Architecture*, June 1990.

[15] M. Karlsson and E. Hagersten. Timestamp-based Selective Cache Allocation. In *Proceedings of the Workshop on Memory Performance Issues*, June 2001. held in conjunction with the 28th International Symposium on Computer Architecture (ISCA28).

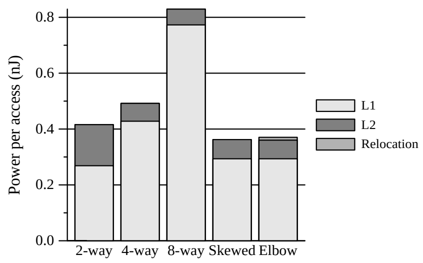
[16] Stefanos Kaxiras, Zhigang Hu, and Margaret Martonosi. Cache decay: Exploiting generational behavior to reduce cache leakage power. In *Proceedings of the 28th International Symposium on Computer Architecture*, pages 240–251, 2001.

[17] Mazen Kharbutli, Keith Irwin, Yan Solihin, and Jaejin Lee. Using Prime Numbers for Cache Indexing to Eliminate Conflict Misses. In *Proceedings of the 10th International Symposium on High-Performance Computer Architecture*, pages 288–299, 2004.

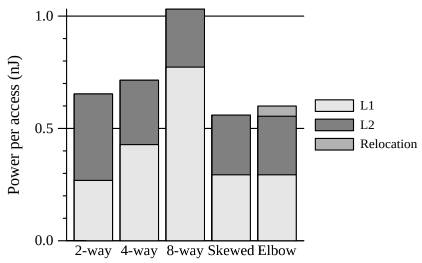
[18] J. Kin, M. Gupta, and W. H. Mangione-Smith. The filter cache:An energy efficient memory structure. In *Proceedings of the 30th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-30)*, 1997.



(a) L1 hit power breakdown.



(b) Average power breakdown for SPLASH.



(c) Average power breakdown for SPEC.

Figure 13: Load power consumption estimates for SPEC benchmarks and different 16K cache architectures.

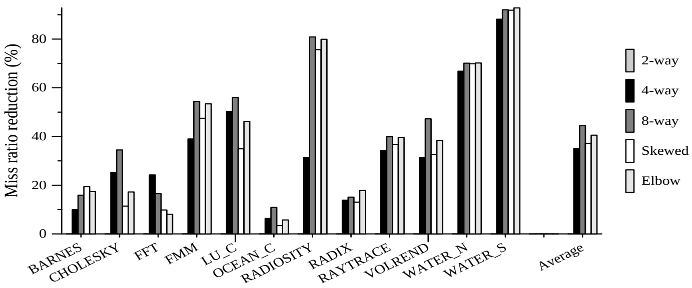


Figure 14: Miss ratio reduction for one multi-treaded application (SPLASH) sharing one 64K data cache with different configurations, compared with a 2-way set-associative cache.

[19] Jack L. Lo, Luiz Andre Barroso, Susan J. Eggers, Kourosh Gharachorloo, Henry M. Levy, and Sujay S. Parekh. An Analysis of Database Workload Performance on Simultaneous Multithreaded Processors. In *Proceedings of the 25th Annual International Symposium on Computer Architecture (ISCA'98)*, pages 39–50, 1998.

[20] M. Karlsson M. Spjuth and E. Hagersten. The Elbow Cache: A Power-Efficient Alternative to Highly Associative Caches. Technical Report No 2003-046, Department of Information Technology, Uppsala University, September 2003.

[21] P. S. Magnusson, M. Christensson, D. Forsgren J. Eskilson, G. Hållberg, J. Högberg, A. Moestedt F. Larsson, and B. Werner. Simics: A Full System Simulation Platform. *IEEE Computer*, February 2002.

[22] M. D. Powell, A. Agarval, T. N. Vijaykumar, Babak Falsafi, and Kaushik Roy. Reducing Set-Associative Cache Energy via Way-Prediction and Selective Direct-Mapping. In *Proceedings of the 34th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-34)*, 2001.

[23] S. Woo, M. Ohara, E. Toorie, J.P. Singh, and A. Gupta. The SPLASH-2 Programs: Characterization and Methodological Considerations. In *Proceedings of the 22th International Symposium on Computer Architecture*, pages 24–36, June 1995.

[24] A. Seznec. A case for two-way skewed associative caches. In *Proceedings of the 20th International Symposium on Computer Architecture*, pages 169–178, May 1993.

[25] A. Seznec. A new case for skewed-associativity. Internal Publication No 1114, IRISA-INRIA, July 1997.

[26] A. Seznec and F. Bodin. Skewed-associative caches. In *Proceedings of PARLE '93, Munich*, pages 305–316, June 1993.

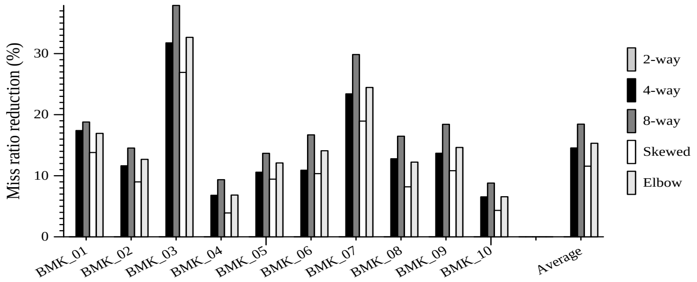
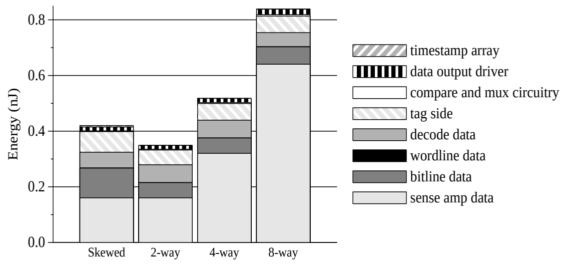
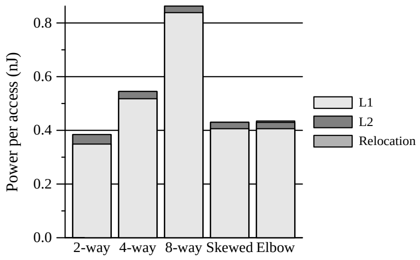


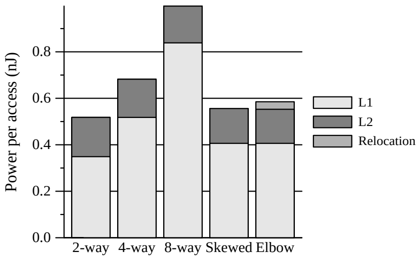
Figure 15: Miss ratio reduction for one multi-treaded application (SPECint) sharing one 64K data cache with different configurations, compared with a 2-way set-associative cache.



(a) L1 hit power breakdown.



(b) Average power breakdown for SPLASH.



(c) Average power breakdown for SPEC.

Figure 16: Load power consumption estimates for SPEC benchmarks and different 64K cache architectures.

[27] P. Shivakumar and N. Jouppi. CACTI 3.0 An integrated Cache Timing, Power and Area Model. Technical Report 2001/2, DEC Western Research Lab, 2001.

[28] M. Spjuth. Refinement and Evaluation of the Elbow Cache. Master's thesis, School of Engineering, Uppsala University, Sweden, April 2002.

[29] Nigel P. Topham and Antonio Gonzalez. Randomized Cache Placement for Eliminating Conflicts. *IEEE Transactions on Computers*, 48(2):185–192, 1999.

[30] Hans Vandierendonck and Koen De Bosschere. Trade-offs for Skewed-Associative Caches. In *Parallel Computing (PARCO)*, September 2003.

[31] N. H. E. Weste and K. Eshraghian. *Principles of CMOS VLSI Design*. Addison-Wesley, second edition, 1993.

[32] S. Wilton and N. Jouppi. An enhanced access and cycle time model for on-chip caches, 1994.

[33] Se-Hyun Yang, Michael D. Powell, Babak Falsafi, Kaushik Roy, and T. N. Vijaykumar. An Integrated Circuit/Architecture Approach to Reducing Leakage in Deep-Submicron High-Performance I-Caches. In *International Symposium on High-Performance Computer Architecture (HPCA)*, 2001.

[34] C. Zhang, F. Vahid, and W. Najjar. A highly configurable cache architecture for embedded systems. In *Proceedings of the 30th Annual International Symposium on Computer Architecture (ISCA'03)*, 2003.