Appears in *Kool Chips Workshop, 33rd International Symposium on Microarchitecture*, Monterey, CA, December 2000

Highly-Associative Caches for Low-Power Processors

Michael Zhang and Krste Asanović
MIT Laboratory for Computer Science, Cambridge, MA 02139
{rzhang|krste}@lcs.mit.edu

1 Introduction

Since caches consume a significant fraction of total processor energy, e.g., 43% for StrongARM-1 [8], many studies have investigated energy-efficient cache designs [1, 5, 12, 13, 14, 15, 18]. However, none of these design studies have considered using content-addressable-memory (CAM) tags in highly-associative caches. This is particularly surprising given that the leading commercial low-power processors over the last decade have all employed CAM-tag caches. For example, the ARM3 with 4 KBytes of 64-way set-associative CAM-tag cache was released in 1989 [9] and the new Intel XScale processor employs 64-way set-associative CAM tags. Publications which describe processors with CAM-tag caches [8, 9, 11, 16] include some discussion of the reasons for choosing CAM tags but do not include detailed quantitative arguments in favor of these designs.

In this paper, we restate the advantages of CAM-tag caches based on a new cache energy model extracted from circuit designs for aggressive low-power cache designs in a 0.25 μm 2.5 V CMOS process. We show that CAM-tag caches have comparable access latency, but give lower hit energy and higher hit rates than RAM-tag set-associative caches at the expense of approximately 10% area overhead. Although direct-mapped caches have lower hit energies and faster access, they suffer higher miss rates which result in much larger total memory access energy as well as reduced performance. Our results demonstrate that CAM-tag caches are fast and energy-efficient, and are well-suited for both high-performance and low-power designs.

The rest of this paper is structured as follows. Section 2 describes the circuit design and layout for our cache designs. Section 3 gives evaluation results for some benchmarks taken from SPECint95 and MediaBench. Section 4 compares our design and energy model with the popular CACTI cache model [17, 19] and illustrates the limitations of CACTI energy model when applied to low-power designs, then Section 5 summarizes the paper.

2 Cache Design and Energy Models

This section describes the designs of our low-power caches and their accompanying energy models. We first describe the RAM arrays that are common to all designs, then describe RAM and CAM tag designs.

2.1 RAM Array Design

The RAM arrays used in our designs employ conventional six-transistor SRAM cells with differential reads and writes. The area of the RAM cell is $4.32\,\mu\text{m}\times4.56\,\mu\text{m}=19.7\,\mu\text{m}^2$. For RAM reads, a self-timed circuit is used to pulse word-lines to limit bitline swing to only about 15% of V_{dd} and bitline isolating latching sense-amplifiers are used to provide a full rail result [2, 10]. To further reduce bitline energy, we divide wordlines every 32 columns and have a single sense-amp per bit column, so that we only discharge the bitlines for the 32-bit word we are accessing [2]. The additional sense-amps do not increase energy because they are conditionally enabled. The per-bit sense-amps and output drivers add about 3% area overhead to the array compared to a scheme with column muxes. Writes are performed differentially with full rail voltage swings.

The main disadvantage of using a sense-amp on each bit-column is that it increases the amount of multiplexing required between the sense-amps and the CPU. We employ pulsed low-swing differential I/O bus drivers to connect word output drivers to the CPU which limits the performance and energy impact of multiplexing additional sense-amps. We also segment the cache-to-CPU bus to minimize the bus wire capacitance that is driven, improving both speed and energy consumption.

The cache storage is divided into smaller sub-banks, balancing energy savings with area overhead. By combining sub-banking with word-line segmentation, we reduce the active RAM array for each word access to just 32 columns and 32 to 64 rows.

2.2 RAM Tag Designs

Figure 1 shows the organization of a traditional n-way set-associative cache. Each cache is divided into n partitions, each with its own tag array. RAM tags are often stored in smaller RAM arrays to allow faster access than data RAM. Once the address is decoded, tags and data from all ways are read out concurrently. Tags are compared in parallel to select the correct way and to generate the hit signal. This approach, however, is not particularly energy efficient since the bits read out from the $n-1$ incorrect ways are discarded. A more energy-conscious two-phased cache [6] first checks tags, then only reads data out from the correct way, practically doubling access latency for a primary cache. Two-phased accesses are more appropriate for secondary caches [7] where an extra cycle of latency to check tags has less overall performance impact.

Direct-mapped caches ($n = 1$) can be considerably faster than set-associative caches because they do not need to wait for the tag compare result before forwarding the data value to the CPU. Also, they have lower access energy because only one set of tag and data is accessed.

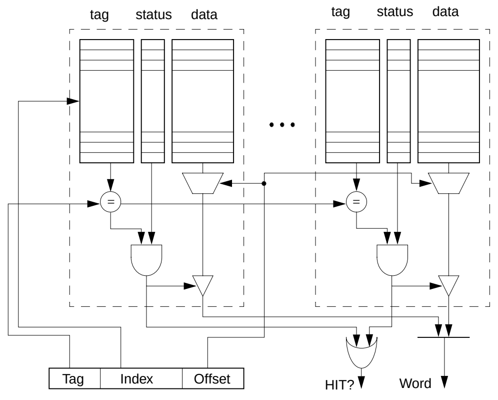

Figure 1: Organization of a set-associative RAM-tag cache.

2.3 CAM Tag Designs

Figure 2 shows the overall organization of one sub-bank of a CAM-tag cache [10]. Each cache line in the sub-bank has a local tag that compares its contents with the broadcast search tag bits. Each CAM cell is a standard ten-transistor design laid out to be exactly twice the RAM cell area at 4.32 μm×9.12 μm=39.4 μm^2 as shown in Figure 3. The cell contains a SRAM cell and a dynamic XOR gate used for comparison. The match line is precharged high and conditionally discharged on a mismatch. All match lines are OR-ed to generate the hit signal.

The search bitlines, match lines, and buffers that drive control signals across the tag array are the main consumers

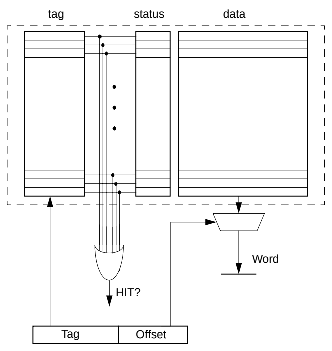

Figure 2: Organization of a highly-associative CAM-tag cache.

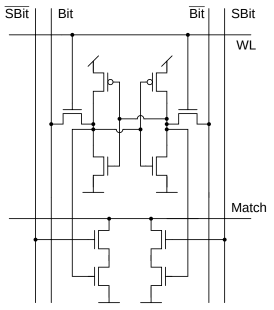

Figure 3: CAM cell circuitry.

of energy in the CAM-tag cache. To reduce the capacitance switched during a search operation, we separate the search bitlines from the write bitlines. To reduce the energy dissipated on the match lines, they are only precharged to $V_{dd} - V_{tn}$ through n-type precharge transistors and single-ended sense-amps are used to regenerate a full-rail match signal. As shown in Figure 4, we also break the entire row of tags into two equal partitions such that the worst-case delay of the match line capacitance discharge can be halved [17].

As with RAM-tag designs, we break up the cache into sub-banks using low order address bits and only activate a search within the enabled sub-bank. We can further reduce the energy of a CAM-tag design by only enabling a smaller number of rows within a sub-bank, effectively reducing the associativity. For example, the StrongARM design [10] has 64 CAM rows (128 RAM rows) in each cache sub-bank but only enables one 32-way subset on each access.

Figure 5 shows the layout of a complete 1 KB 32-way set-associative cache sub-bank. This consumes around 10% greater area than a 1 KB RAM-tag sub-bank. CAM tags have the property of providing very high associativity within a single sub-bank without having to partition the RAM array. There would be significant area overhead for sense-amps and muxes if we were to try to implement a small and highly-associative RAM-tag cache sub-bank.

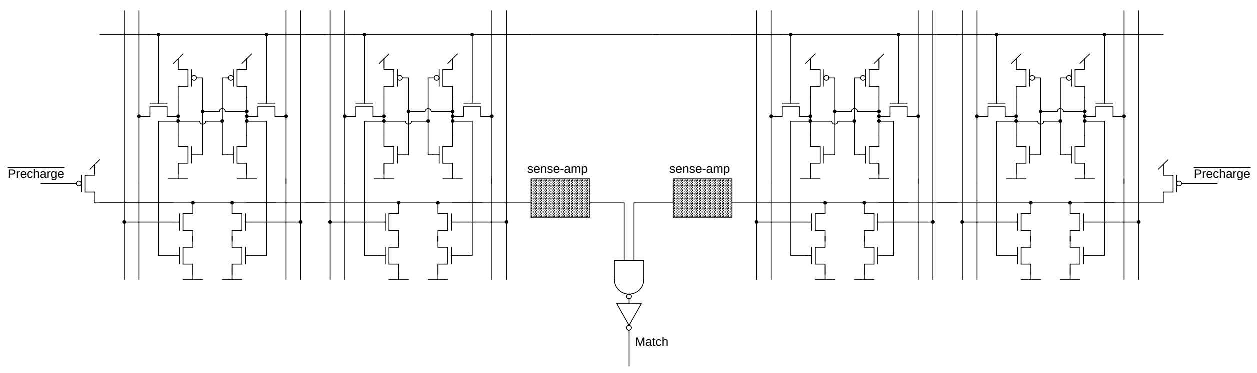

Figure 4: Split CAM row operation.

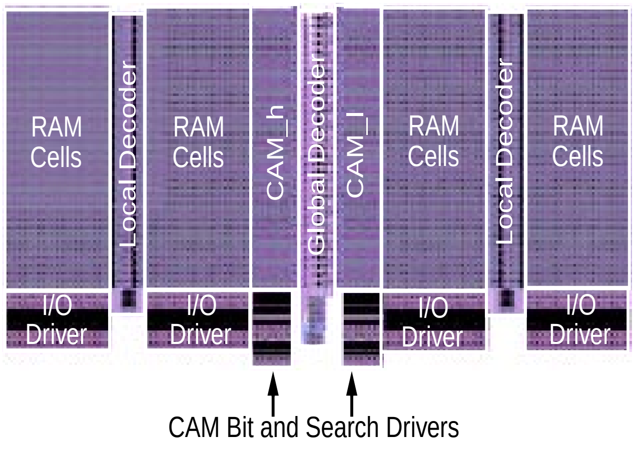

Figure 5: Layout of a 1 KB 32-way set-associative CAM-tag cache sub-bank holding 32 lines with 32 bytes each.

Another advantage of CAM-tag designs is that they simplify the handling of cache stores. In a CAM-tag design, the data RAM word lines are only enabled on a cache hit and so stores can happen in a single cycle. A conventional RAM-tag design has to split the store access across two cycles: the first cycle checks the tag, and the second cycle writes the data storage on a hit. To allow full speed writes, RAM-tag designs often include a write buffer ahead of the primary cache to avoid stalling on stores, adding additional complexity and energy overhead. We ignore this additional delay and energy cost in the comparison below.

2.4 Energy Modeling

To obtain accurate energy models, we performed complete circuit design and layout of both RAM and CAM arrays of various sizes in a 0.25 μm 2.5 V TSMC CMOS process. A 2-dimensional circuit extractor was used to obtain a SPICE netlist including detailed circuit parasitic capacitances. We used HSPICE to simulate the extracted netlist and from these simulations we extracted effective capacitance for our energy models. These simulations capture the effects of inter-wire coupling capacitance and short-circuit currents. Our simplified models predict RAM and CAM array energy very accurately (within 2%) since these pieces are extremely regular, and predict within 10% error for the entire cache energy including decoders and I/O.

2.5 Delay Comparison

The critical paths for RAM and CAM-tag caches are shown in Figure 6. From the figure, we observe that the delays of the CAM-tag and RAM-tag caches are very similar — they share many of the same components. The CAM-tag critical path includes buffers to drive the search bitlines, the dynamic match comparators, the data RAM array, and then the tristate I/O drivers connecting back to the CPU. The RAM-tag critical path has the address index bits decoder, tag RAM array, the dynamic match comparator, match signal buffering to drive the data mux control, and then a tristate mux connecting back to the CPU. The tristate bus that connects one 32-bit sub-bank column back to the CPU 32-bit load datapath has the same fan-in in all configurations. There are two main differences between the two critical paths. First, the RAM block has a faster address decoder to set up the data word lines whereas the CAM block performs a slower tag match operation. Second, the RAM block has to compare the output of the RAM tags, then buffer up the match signal to enable 32 tristate bus drivers, whereas for the CAM block all the tristate enables are setup using low-order address bits before the RAM data appears. We assume that the tag RAM array has half the number of rows as the data RAM, and hence part of the tag compare and tristate enable setup overlaps with the slower data RAM access. From HSPICE simulations, we have found that the delay to valid data of RAM-tag and CAM-tag cache designs with same sub-bank size are within 1%

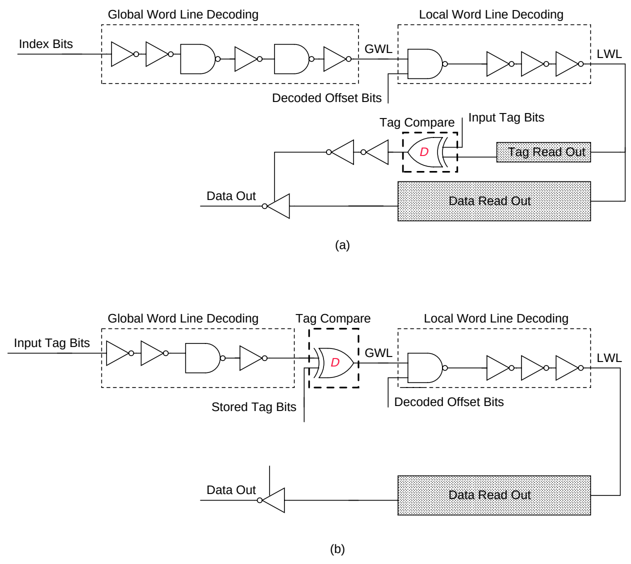

Figure 6: (a) Critical path of RAM-tag cache. (b) Critical path of CAM-tag caches.

of each other (2.92 ns and 2.89 ns respectively). A direct-mapped scheme is considerably faster at around 2.82 ns access time. CAM tags generate the hit signal faster than RAM tags, which can be an advantage for controlling machine pipelines.

3 Results

In this paper, we reduce the design space to a small set of candidate primary data cache designs. We compare 8 KB and 16 KB capacities with 32 byte lines, which are typical sizes for embedded processor primary caches. The optimal size of data RAM sub-banks, balancing lower energy with increased area, is found to be around 1–2 KB. Thus, we fixed the data capacity of each sub-bank for all configurations to be 1 KB organized as 64 rows with 128 columns, where the 128 columns are divided into four 32-bit word groups, i.e., a cache line fits on two rows of a sub-bank. For lower energy and faster access, the RAM tags are held in separate smaller tag RAM arrays with 32 rows. The double-height CAM tag bits are built into the middle of the data RAM arrays stretching over the two RAM rows of each cache line. The RAM tag caches treat each sub-bank as a way, and access n sub-banks together to form an n-way set-associative cache. Within each way, the word-line segmentation ensures that only the required word is fetched from the data array. The CAM tag caches access only one sub-bank at a time, with a maximum of 32-way set-associativity, when all rows are activated on search.

We simulated a few SPECint95 and Mediabench benchmarks to obtain representative data cache miss rate information for a 16 KB cache and a 8 KB cache. The results are shown in Figure 7. We note a wide variation in miss rate, with `LZW` and `pegwit` having much higher miss rates dominated by capacity misses, while `m88ksim` and `perl` have almost no misses in small caches once associativity is used to remove conflict misses. We see that direct-mapped caches perform poorly compared with associative designs averaging around 2–3% higher miss rates, and that for most benchmarks, larger associativities give some small benefits. As expected, we also see that increased associativity is more important for the smaller cache.

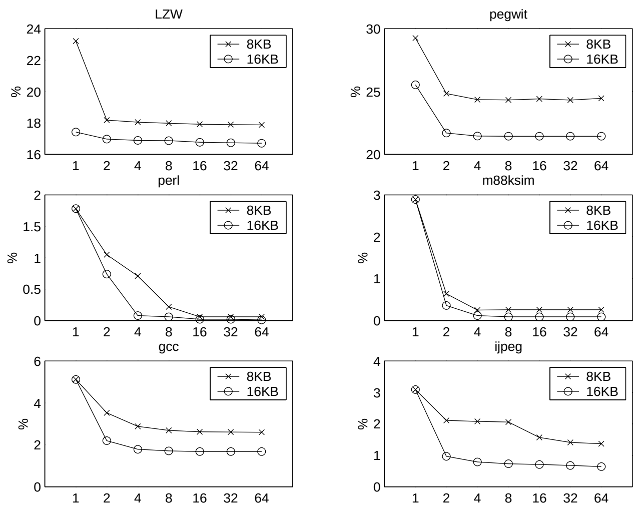

Figure 7: Miss rate versus associativity for 8 KB and 16 KB data caches.

Figure 8 plots the energy for each access that hits in an 8 KB cache for each of the benchmarks and for a range of RAM and CAM associativities. The variation in energy across benchmarks is due to differences in the mixture of reads and writes. We observe that as we vary CAM associativity from 8-way to 32-way, the hit energy is comparable to the 2-way RAM design but significantly lower than the 4-way RAM tag. The more highly associative RAM tag designs waste considerable energy fetching multiple data and tag items that are usually discarded. The direct-mapped cache has significantly lower hit energy than any of the associative designs, but this metric ignores the energy and performance cost of cache misses.

To show the effects that misses have on memory access energy consumption, we plot in Figures 9 and 10 the total energy per access for `perl` and `pegwit` respectively, which represent two extremes in terms of miss rate. Rather than pick a single design of a memory hierarchy, we show how total access energy varies as a function of miss energy where miss energy is expressed as a multiple of the

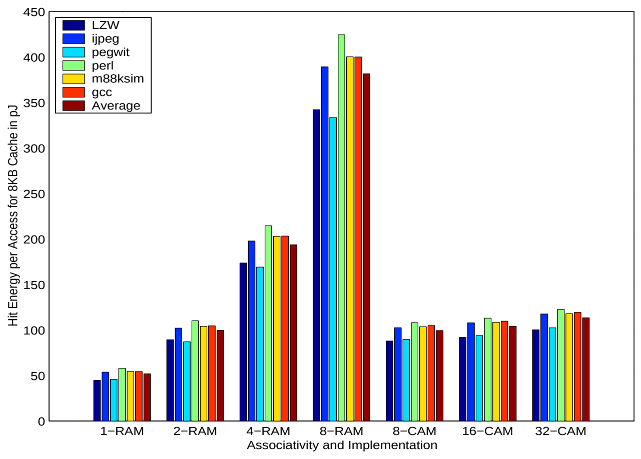

Figure 8: Hit energy per access for 8 KB cache for various cache associativities.

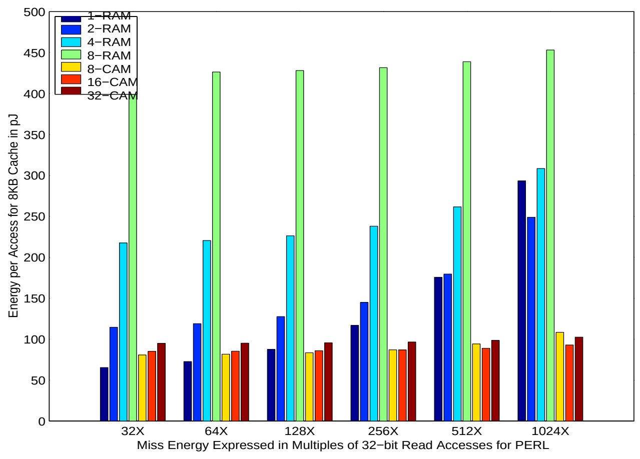

Figure 9: Total energy per access for `perl` in an 8 KB data cache for varying miss energy cost.

energy required to read a single 32-bit data word from primary cache. At minimum, a cache miss will require that 8 words are read from second level memory and written into primary cache. This will generally consume considerably greater than 32$\times$ single word read energy. In practice, miss energies could easily be in the range of 128–1024$\times$ single word read energy once victim writebacks are accounted for and if external DRAM accesses are involved.

These large miss energy penalties amplify small differences in miss rate. For `perl`, we see that the CAM tag cache can reduce effective access energy by greater than a factor of 2 compared with direct-mapped caches for high miss energies. Compared with the direct mapped cache, it has greater hit energy but much lower total miss energy due to considerably fewer misses. Compared with the more associative RAM-tag designs, it has lower hit energy and slightly lower total miss energy due to fewer misses. The `pegwit` benchmark is dominated by miss energy across all configurations, but the slightly lower miss rate of the CAM tag solutions gives them comparable or lower total energy across all realistic miss energies.

The CAM tag variants also provide better performance by having fewer cache miss stalls. This combined effect of higher performance and lower effective access energy explains the popularity of CAM tag caches in low-power processors. We also note that we have not optimized the CAM tag circuitry as heavily as the RAM tag circuitry, which includes pulsed low-swing techniques. We believe there is considerable room for further reductions in CAM hit energy through circuit techniques.

4 Related Work

The CACTI model [17, 19] has been used in many architectural cache studies and has recently been extended to include CAM-tag cache designs [17] and energy models. However, the base CACTI cache design is not particularly energy-efficient. We attempted to replicate our 8 KB CAM-tag cache configurations using the CACTI model scaled to our technology and power supply. We found that CACTI agreed to within 10% on our access time numbers, but over-estimated energy consumption by around a factor of ten for a single sub-bank. Our model gave RAM energy figures in close agreement with those from the experimental circuit measured in [3]. We will try to discuss some of the factors which might attribute to this difference.

Our design uses a low-power sensing scheme which limits bitline swing to only 15% of V_{dd} by using self-timed pulsed word lines and bitline-isolating latching sense amplifiers [4, 10]. In the CACTI model, however, reads use a sensing scheme with a statically-biased sense amplifier and large bitline voltage swings that will not scale to lower supply voltages. The CACTI model also assumes that multiple words are read for each access with a column mux before the sense-amps, whereas low-power designs need only discharge the individual word that is being accessed and employ a single sense amp per bit column [2].

The new CAM tag model in CACTI 2.0 do not use split the write bitlines and the search bitlines in CAM tags, and assume all match lines are charged to V_{dd}. It also assumes that CAM tags will always be used in fully-associative configurations, where all tags in the cache are active, whereas existing commercial designs employ sub-banking to activate only 8–64 ways in one cache sub-bank. When we tried to configure CACTI 2.0 to model the entire CAM-tag

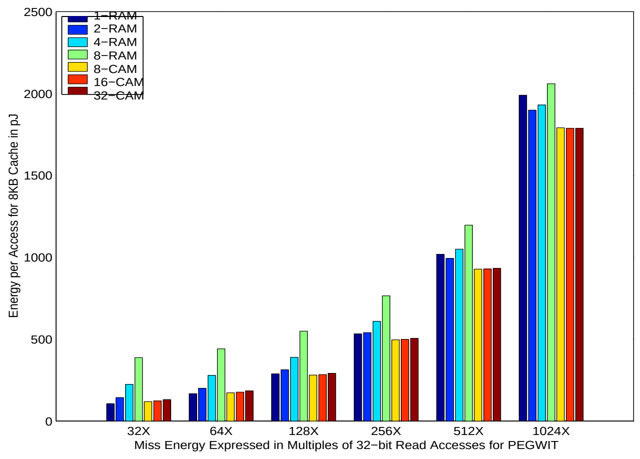

Figure 10: Total energy per access for `pegwit` in an 8 KB cache for varying miss energy cost.

cache, we obtained a factor of thirty difference due to this effect.

5 Summary

In this paper, we have presented results modeling RAM-tag and CAM-tag data cache energy dissipation through HSPICE simulation of extracted layout. Using our energy model, we have shown how CAM-tag caches can provide lower total memory access energy by reducing the hit energy cost of the high associativities required to avoid costly misses. We also show that there is no significant performance overhead associated with CAM designs, although there is around a 10% area overhead.

6 Acknowledgments

Thanks to members of the MIT SCALE group for feedback and comments on earlier drafts of this paper. This work was partly funded by DARPA award F30602-00-2-0562.

References

[1] G. Albera and I. Bahar. Power and performance tradeoffs using various cache configurations. In *Power Driven Microarchitecture Workshop at ISCA98*, Barcelona, Spain, June 1998.

[2] B. Amrutur and M. Horowitz. Techniques to reduce power in fast wide memories. In *ISLPED*, pages 92–93, October 1994.

[3] B. Amrutur and M. Horowitz. A replica technique for wordline and sense control in low-power SRAMs. *IEEE JSSC*, 33(8):1208–1219, August 1998.

[4] B. Amrutur and M. Horowitz. Speed and power scaling of SRAMs. *IEEE JSSC*, 35(2):175–185, Feburary 2000.

[5] N. Bellas, I. Hajj, and C. Polychronopoulos. Using dynamic cache management techniques to reduce energy in a high-performance processor. In *ISLPED*, pages 64–69, August 1999.

[6] A. Hasegawa *et al.* SH3: High code density, low power. *IEEE Micro*, 15(6):11–19, December 1995.

[7] B. J. Benschneider *et al.* A 300-MHz 64-b quad-issue CMOS RISC microprocessor. *IEEE JSSC*, 30(11):1203–1214, November 1995.

[8] J. Montanaro *et al.* A 160-MHz, 32-b, 0.5-W CMOS RISC microprocessor. *IEEE JSSC*, 31(11):1703–1714, November 1996.

[9] S. Furber *et al.* ARM3 - 32b RISC processor with 4kbyte on-chip cache. In G. Musgrave and U. Lauther, editors, *Proc. IFIP TC 10/WG 10.5 Int. Conf. on VLSI*, pages 35–44. Elsevier (North Holland), 1989.

[10] S. Santhanam *et al.* A low-cost, 300-MHz, RISC CPU with attached media processor. *IEEE JSSC*, 33(11):1829–1838, November 1998.

[11] S. Furber, J. Garside, and S. Temple. Power saving features in Amulet2e. In *Power Driven Microarchitecture Workshop at ISCA98*, Barcelona, Spain, June 1998.

[12] K. Ghose and M. B. Kamble. Reducing power in superscalar processor caches using subbanking, multiple line buffers and bit-line segmentation. In *ISLPED*, pages 70–75, August 1999.

[13] K. Inoue, T. Ishihara, and K. Murakami. Way-predicting set-associative cache for high performance and low energy consumption. In *ISLPED*, pages 273–275, August 1999.

[14] J. Kin, M. Gupta, and W. Mangione-Smith. The Filter Cache: An energy efficient memory structure. In *Micro-30*, December 1997.

[15] U. Ko, P. Balsara, and A. Nanda. Energy optimization of multilevel cache architectures for RISC and CISC processors. *IEEE Trans. VLSI Systems*, 6(2):299–308, June 1998.

[16] T. Pering, T. Burd, and R. Broderson. Dynamic voltage scaling and the design of a low-power microprocessor system. In *Power Driven Microarchitecture Workshop at ISCA98*, Barcelona, Spain, June 1998.

[17] G. Reinman and N. Jouppi. An integrated cache and timing model. http://research.compaq.com/wrl/people/jouppi/cacti2.pdf, 1999.

[18] C.-L. Su and A. Despain. Cache design trade-offs for power and performance optimization: A case study. In *ISLPED*, pages 63–68, October 1995.

[19] S. Wilton and N. P. Jouppi. An enhanced access and cycle time model for on-chip caches. Technical Report 93/5, Compaq Western Research Lab, July 1994.