An Adaptive Serial-Parallel CAM Architecture for Low-Power Cache Blocks

Aristides Efthymiou
efthym@cs.man.ac.uk

Jim D. Garside
jdg@cs.man.ac.uk

Department of Computer Science,
University of Manchester,
Oxford Road,
Manchester M13 9PL,
UK

ABSTRACT

There is an on-going debate about which consumes less energy: a RAM-tagged associative cache with an intelligent order of accessing its tags and ways (e.g. way prediction), or a CAM-tagged high associativity cache. If a CAM search can consume less than twice the energy of reading a tag RAM, it would probably be the preferred option for low-power applications. Based on memory traces — which usually cause tag mismatch within the lower four bits — a new serial CAM organisation is proposed which consumes just 45% more than a single tag RAM read and is only 25% slower than the conventional, parallel CAM. Furthermore, it can optionally be operated as a parallel CAM, at no speed penalty, and still reduce energy consumption.

Categories and Subject Descriptors

B.3.2 [**Memory Structures**]: Design Styles—*Associative memories*; B.3.2 [**Memory Structures**]: Design Styles—*Cache memories*; B.7.1 [**Types and Design Styles**]: VLSI

General Terms

Design, Performance

Keywords

CAM, cache design, VLSI, low power, low energy, asynchronous circuits

1. INTRODUCTION

The cache accounts for a considerable fraction of the energy consumption in an embedded system. For example, it is responsible for over 40% in StrongARM [1]. At the same time it is usually performance-critical, so implementations targeting low energy consumption cannot neglect the access time.

This work follows the idea of many other implementations [2][1] in assuming that the cache is made up out of many small blocks, trading silicon area for speed and low power consumption. Specifically a memory block size of 1 KByte and a line size of 32 bytes (8 32-bit words) are used here. These SRAM blocks are used for the data storage and are coupled with smaller structures for the tags. The choice of the block and line sizes means that each block contains 32 cache lines and, similarly, the tag structures consist of 32 tags.

Often in embedded applications the silicon area available for the cache is not large enough for a direct-mapped organisation to achieve the required high hit rates. This is where the question of using a RAM- or CAM-based tag structure must be addressed. Closely related to this decision is the degree of associativity required. Generally, the higher the associativity the better the hit rate, but the design complexity increases with associativity and the hit rate obeys the law of diminishing returns. As a rule of thumb, an associativity of four is usually adequate, although a higher associativity — at no extra cost — would still be preferred.

A RAM-tagged cache can be organised as a conventional n-way associative cache, where a number, n, of the (direct-mapped) blocks are accessed concurrently; within each block a part of the address is used to select a line directly. This method is quite wasteful of energy because at least $n-1$ tag and data reads will be wasted for each cache access. To conserve energy, phased caches [3] first access the tags and then only the data memory where there was a hit; however this doubles the cache latency. Pseudo-associative caches [4] and way-predicting caches [5] take a different approach. These access one block (or 'way') first and only try other blocks if they miss at the first. This results in a cache with variable hit (and sometimes miss) times. There is considerable range for variation here too: the method used to select the first block can be dynamic (e.g. PSAC [6]), or static. If the first access misses, the rest of the blocks can be accessed concurrently or sequentially [7]. Furthermore, in all the above cases, the tags can be read before the data to save even more energy at the expense of performance, as in the original phased cache.

A block-structured, CAM-based cache uses some of the

ISLPED'02, August 12-14, 2002, Monterey, California, USA.

low order bits of the address to select one of the blocks, within which a fully associative approach is used. In this case the associativity depends on the memory block and cache line sizes. With the example chosen here this would make a 32-way associative cache. In this implementation the CAM *must* be searched before the data can be accessed, so this is similar to the phased cache.

The following section continues the above description and sets a working example of a cache that is used throughout the rest of this paper. Section 3 compares the energy and speed of reading a tag RAM and searching a tag CAM, both extracted from layout using Spice simulation. This shows that a standard CAM search requires five times more energy than reading a RAM, which suggests that RAM-based caches should be more energy efficient. However section 4 analyses the behaviour on tag checking by applications; this hints at a more energy efficient CAM design, presented in section 5. The proposed CAM has two modes of operation depending on the energy/speed trade-off required. Finally, section 6 concludes the paper.

2. CACHE ORGANISATION OPTIONS

An example cache size of 16 KBytes is used here for demonstration purposes. This is broken into sixteen blocks of 1KB each. Virtual addressing is assumed to be used for the caches. This is more energy efficient as it does not require an address translation for every memory access.

Set associative

For a conventional 4-way set associative implementation, four of the sixteen blocks would be accessed in parallel. Two bits of the address would be used to select a set of 4 blocks in addition to those selecting the cache line and word in the line, leaving a tag size of 20 bits.

The pseudo-associative implementation can be described as an amalgamation of the direct-mapped and the parallel set-associative. Initially the address is mapped to a block and line and the corresponding tag is checked. If this fails a different mapping is applied and another access and comparison are made. The number of times this is repeated determines the (pseudo) associativity. The mappings can be dynamic, based on some 'hint' and a 'steering table', which will keep the average number of accesses low [6]. Although the tag size for these caches is the same as in a direct-mapped cache, additional information is required to keep the cache consistent (e.g. the rehash bit in [6]). This information is different for each variation of the pseudo-associative concept. Thus, the analysis here is based on the assumption that the pseudo-associative tag size is as big as that of the conventional way-associative cache and no other information is stored.

Fully associative

The fully-associative (within each block) approach uses some low order bits of the address to select one of the blocks, each of which is fully-associative internally. For the 16 KByte, 32-byte line size example the tag size would be 23 bits. The tag structure would require CAM implementation for fast parallel searching. Because the blocks contain 32 cache lines the cache would be 32-way set-associative, a 'side-effect' of the block size and the cache line size.

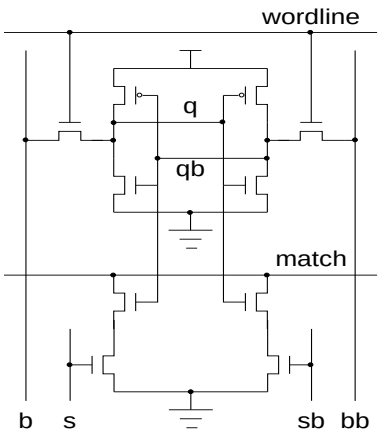

Figure 1: CAM cell

Comparison

Assuming an energy conserving policy of only accessing a data memory block when there is a hit, the differences in the above organisations will depend on the average number of tag accesses for the (pseudo) associative case, the energy cost of a RAM read for tags and the energy cost of a CAM search. The way-predicting cache [5] using a most recently used algorithm for selecting which block to access first, has a prediction rate of about 90%. In case of parallel checking all the other ways, the average number of tag accesses will be $0.9 \cdot 1 + 0.1 \cdot 4 = 1.3$. Considering the extra overhead of the prediction for the way-predicting caches, we can assume that if the energy for a CAM search is no more than twice that of a RAM read, a fully associative implementation would be preferable for low-power microprocessors. Moreover, the fully-associative implementation will achieve a somewhat better hit rate, which should save some (high energy consuming) accesses to higher level memories.

3. COMPARISON OF RAM/CAM

For this comparison a set of RAM, CAM and sense amplifier cells were designed and laid out in a $0.18\mu m$, 1.8V, dual V_t technology and simulated with Spectre, a Spice variant. Although the implementations could be further optimised, they were designed by the same person and a similar amount of time was spent on each of them, so this comparison should be fair.

The RAM was based on the low-power SRAM design of Amrutur and Horowitz [8]. It uses cross-coupled inverter-style sense amplifiers, enabled by a delay-matched dummy column, and pulsed wordlines to limit the voltage drop on the bit lines to about 15% of the supply voltage. The RAM block has 32 rows of 20 bits each (plus a dummy line for delay matching).

For the CAM design (fig. 1 shows the cell), separate search and bit lines were used to minimise the capacitance on the former, which are more frequently used. No power is consumed within the CAM cell when a search operation is performed, but the search lines must be pulled down while the match lines are being precharged, otherwise it is possible to short-circuit the supply and ground terminals through the match line. Thus, for every search, one of the search lines

will always have to be raised; this is responsible for almost half the CAM's energy budget.

To compare the two circuits, a series of ten reads and ten searches were performed on the RAM and CAM blocks respectively. As neither the value nor the position of the row being read affects the energy consumption of the SRAM, the Spectre circuit description was simplified having only one cell active and dummy cells to capture the effect of wire loading for the columns and rows. A number of probes were used to measure the supply currents which are then multiplied appropriately for the columns and the rows that would normally be activated. The CAM was modelled similarly, with two cells in a row active and dummy cells for the rest of the row and column. At most one row will match each time and that row consumes no energy, since it will not discharge its match line, but all the other rows do. All ten searches were made not to match in the simulation and the current consumed by the row is multiplied by the number of rows less one, to capture the most usual case of a cache hit.

The RAM-based tag block has an access time of 0.5ns and an average energy of 2.3pJ per read access, in typical silicon and operating conditions. These figures include the energy consumed by the drivers of the precharge and sense amplifier trigger signals, but not the address decoder consumption, which — for a 32 row memory — is not expected to contribute significantly. The final wordline driver consumption is included however.

The energy consumed by the CAM is 12pJ per search at a minimum access time of 0.8ns. This energy consumption is over five times that of the RAM, although the result is slightly biased in favour of the RAM because the energy consumed by the decoder and that of the comparator needed to test the tags are not included. The CAM block is also larger than the RAM, the cell being 25% bigger and the block width is 23 bits for each of the 32 rows (3 bits more).

Based on these results a (pseudo) associative cache with a low average number of sub-block accesses per memory request is better than the fully associative implementation. The following sections present an alternative CAM architecture that reverses this situation.

4. APPLICATION BEHAVIOUR IN CACHE TAG MATCHING

Much of the energy consumed in a CAM is due to the frequent precharging and discharging of all but one of the match lines for each access. How many bits are actually different in each comparison and is there any way of knowing their positions? An analysis of the memory traces of various SPECInt95 benchmarks was undertaken to gather information to answer this question.

The applications were compiled for an ARM processor, with all the speed optimisations enabled, using the ARM Developer's toolkit compiler and debugger/simulator version 2.51. The simulator was set to emulate the StrongARM implementation which is the closest to Un. of Manchester's asynchronous implementation of ARM, AMULET3 [9], where the cache will be used in the future.

For each benchmark two cases where analysed:

- A unified 16K cache.
- Two dedicated instruction and data caches, each 16K in size.

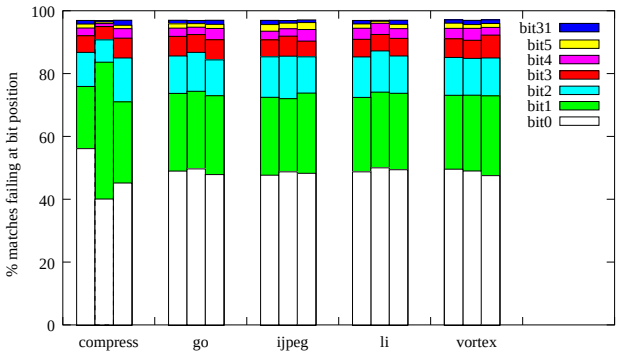

Figure 2: Ratio of tag checks ending at each bit position.

The cache line size is 32 bytes in all cases. The caches use a write-back, no write-allocate policy with round-robin line replacement.

Memory traces from the simulations were used to measure how many tag tests could stop at each bit position if they started from the least significant bit. The results, presented in figure 2, show that over 90% of the mismatches are determined within the four least significant bits of the tags. This figure is consistent for all the benchmarks and cache types: Unified, Instruction and Data, from left to right in the graph.

For these results virtual addressing was used for the caches. If the addresses were translated to physical addresses, the results may have been different, but for low-power operation, virtually addressed caches are preferred because they avoid address translation for most memory accesses.

5. PROPOSED CAM ARCHITECTURE

Based on the results of the tag matching behaviour of the applications, an adaptive serial-parallel CAM (SPCAM) organisation was designed to take advantage of the infrequently changing MSBs of the tags. When designing the circuits the following principles, which are the same as those in Hsiao *et al.* [10], were followed:

- Minimise the transitions on the match lines.
- Use separate search and bit lines.
- Do not force the search lines to '0' or '1' while charging the match lines.
- Minimise the use of timing signals.

The proposed CAM can operate either in parallel or in serial mode. In serial mode the four least significant bits of each row are checked serially and, if they all match, the remaining bits are checked in parallel. In parallel mode the four LSBs are checked serially again, but the remaining bits are tested concurrently. The results of the two matches are ANDed together in both cases to give the final result.

A row of the circuit is shown in figure 3. Note that there are different types of CAM cells for the serial and parallel parts. The parallel CAM cell is the same as that in figure 1, with the ground connection of the two pull down

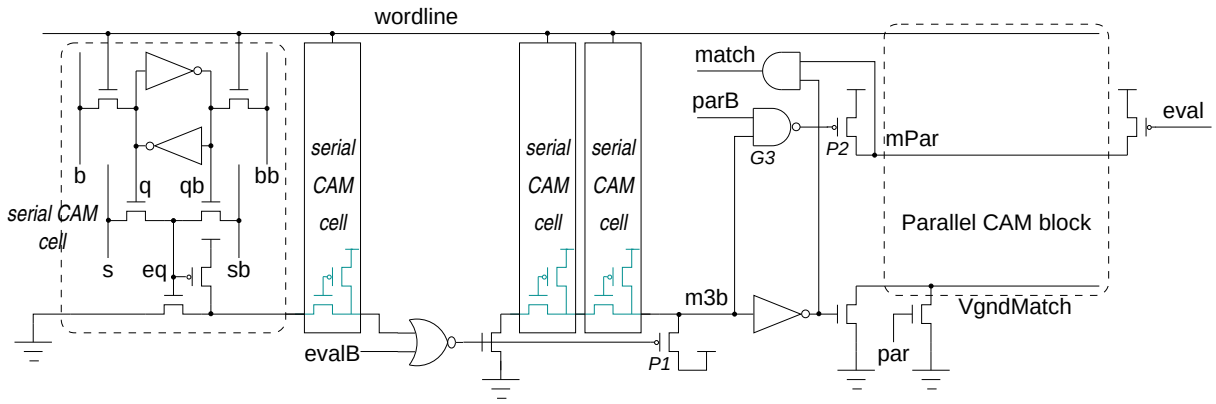

Figure 3: A row of the proposed CAM.

NMOS chains testing the equivalence, replaced with *VgndMatch*. This signal runs lengthwise and is connected to all the parallel CAM cells in the row. The four serial bits are broken down into two sets of two bits to limit the number of transistors in series to three.

The circuits used for the match operation in the serial part look very similar to the Manchester carry chain used in adders. A match propagates as a zero from the least significant bit to the most significant. A cell that matches opens its NMOS transistor, propagating the result from its less significant neighbour to its more significant neighbour. If it doesn't match it breaks the chain and generates a one to pass on. If the first cell of a set doesn't match but the second does then a one is propagated to the right through an NMOS transistor. This is allowed to propagate only through one device — another reason for separating the four bits into two sets. The gates at the output of these chains are designed so that their input threshold is lowered (with widened NMOS transistors) to compensate for the V_t voltage drop in this case. When the cells in the second set match but those in the first don't, transistor *P1* is used to pull up the serial part match signal *m3b* signifying a 'no match'. If the four LSBs all match, the virtual ground *VgndMatch* is pulled down allowing the rest of the row to evaluate the parallel match line; otherwise the match line of the parallel part is precharged, via transistor *P2* although, *mPar* would not have been discharged in most cases.

In serial operation the search lines are not pre-discharged; this saves energy as transitions only occur when the search data actually change. This means that the parallel CAM cells will be evaluating all the time, even when the match line is being precharged. If some of these cells don't match, there will be a path from the parallel match signal *mPar* to the virtual ground, which is left undriven. In this case either the precharge time should be made longer so that the extra capacitance will be charged, or larger precharge transistors should be used that are able to charge the increased capacitance in the same time. Obviously neither of these options is desirable. Instead we opted not to change the precharge transistor sizes or the time. This could leave *mPar* not fully charged but it will not affect the operation. If the LSBs match, *VgndMatch* will be pulled down and, as some of the parallel part tags don't match, *mPar* will also be discharged. If the LSBs don't match, the final match line will stay low, driven from *m3b* and the charging will continue, so eventually it will reach the appropriate level. Simulation indicates that the time between two accesses is sufficient to precharge *mPar* fully when this form of charge sharing happens.

An interesting situation can occur when the following operations happen consecutively: a search is started and in some row the LSBs all match but the MSBs do not. This will leave *mPar* discharged. A subsequent search is initiated which matches the whole row. This would normally fail because *mPar* was not precharged in the interim. For this reason *evalB* — the inverse of the parallel block evaluation signal *eval* — is connected to the NOR gate within the LSB match logic; this forces a 'no match' for the LSBs, imposing a precharge of the parallel match line (*mPar*) for every search. This implementation was preferred to combining the *evalB* signal with *m3b* in a logic gate to drive *VgndMatch* and *mPar* because it has less capacitive load on *evalB*.

When the CAM is in parallel mode the virtual ground line, *VgndMatch*, of the parallel part of the CAM is always connected to the ground. In addition the NAND gate *G3* isolates the precharging of the parallel part's match line, *mPar*, from the match signal of the serial part, *m3b*. A separate signal *eval* is then used for precharging. This signal is gated off in serial mode so that its driver does not consume power.

Figure 4 shows the most important signals in the SPCAM for five consecutive operations. The potential charge sharing problem mentioned earlier could occur in the third search, where *VgndMatch* is being charged up together with *mPar*. For the precharge transistor sizes chosen, the problem does not appear here. In the same operation *m3b* is shown not to reach Vdd; this is the case when it is being pulled high through an NMOS transistor, because the last cell in the serial part matched but the one before didn't, as explained above. As shown in the waveforms, the circuit correctly reports a 'no match'.

5.1 Results

SPCAM was simulated using the same simulator and technology as the previous designs. The results are summarised in table 1. The distribution of mismatches to bit positions follow the findings of the previous section. In serial mode the energy consumption is only 45% more than the RAM (which does not include the decoder and comparator); this

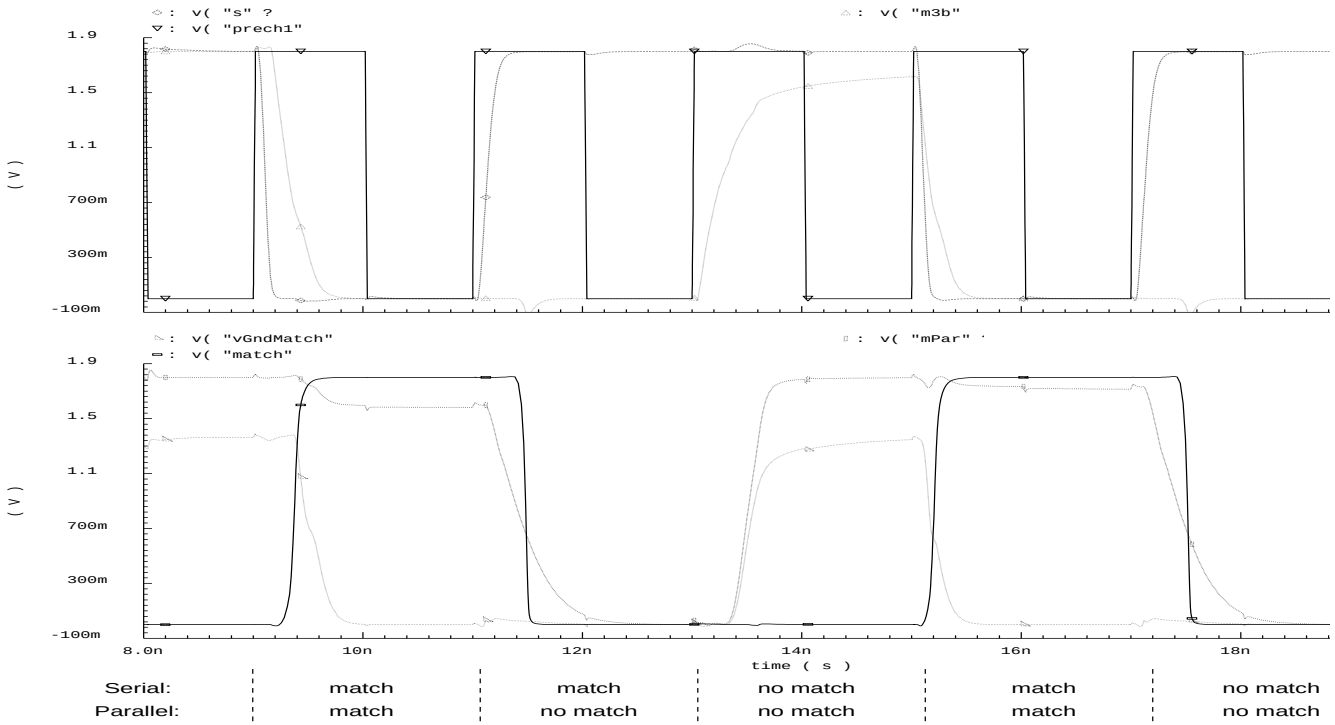

Figure 4: Waveforms of SPCAM in serial mode.

is almost a quarter of the standard low-power CAM energy consumption. The cycle time is twice that of the RAM, but only 25% slower than the original CAM. Thus SPCAM is much more energy efficient than the conventional CAM. In parallel mode, the energy consumption is 3.5 times that of the RAM, still 33% better than the conventional parallel CAM, while the performance is the same as the latter.

With a CAM search energy consumption so close to that of a single RAM read, the fully associative cache organisation becomes a much lower energy choice than any (pseudo) associative, way predicting cache. Caches with conventional CAMs are reported to have a similar access time to caches with RAM tags [11], thus the effect of the decoder and the comparator must slow down the RAM-based designs to a similar speed to the CAM-based ones. With the results presented here conventional CAMs are only 20% faster than the proposed SPCAM in serial operation. Thus the performance of a cache using the SPCAM should not be significantly slower than a cache built with RAM tags.

The proposed CAM is able to switch from serial to parallel mode, trading energy for speed. In a common synchronous processor this is hard to exploit, unless the cache access is made to take two cycles in serial mode and one cycle in parallel. As this design is intended for an asynchronous processor, the variation in speed can be accommodated more easily.

Table 1: Comparison of tag implementations

	Energy per search/access	Cycle time
CAM (32x23)	12.0 pJ	0.8ns
RAM (32x20)	2.3 pJ	0.5ns
Serial SPCAM	3.3 pJ	1.0ns
Parallel SPCAM	8.0 pJ	0.8ns

5.2 Related work

A recent CAM design by Hsiao, et. al. [10], claims to be the lowest power CAM yet reported. It evaluates the match lines serially (NAND-type) and does not require discharging of the search lines while the match line is precharged. However precharging and evaluating the match line segments requires more 'clocking' power than the design proposed here. They report a 45.5fJ/bit/search at 12ns cycle time in a .35μm, 3.3V technology. Converting their energy per bit per search result to our technology suggests about 11fJ/bit/search, which is over twice that of the SPCAM in serial mode.

Zhang and Asanovic [11] argue that CAM-based caches are preferable for low-power processors. They describe a CAM design with separate bit and search lines and they precharge the match lines through NMOS transistors to reduce voltage swings. As a speed enhancement they split each match line into two segments which, in view of the analysis here, would also save energy as the most significant part will be discharged infrequently. For further speed improvement they employ single-ended sense amplifiers on both segments of the match lines. The internals of these are not described but they are quite likely to consume significant power. The energy consumed in the tags is not directly compared in that paper but they showed that the total energy consumption of a cache, with the same configuration as the working example here, is similar to a 2-way associative conventional

(RAM-based) cache. Their CAM-based cache has an almost identical performance to a conventional RAM-based cache, which is not phased, i.e. all tags and data are read in parallel in all ways.

Huang et al. [7] compared a number of different pseudo-associative policies. In their results systems with *Fallback regular*, *Fallback phased* and *Predict phased* policies have very similar energy consumption and delays. Unfortunately they do not compare these results to CAM-based caches.

Burd [2] presented a CAM which consumes twice the energy of an equivalently sized RAM. This is quite surprising because it is a conventional parallel CAM with shared bit lines and search lines. The difference is that the bit lines are pulled up when the match lines are precharged, but one of them still has to be pulled down for each search. With this modification the bit lines are more heavily loaded, with two transistor gates and a drain for each cell. Burd argues that caches using this CAM consume the same energy as a conventional 2-way set associative cache built with RAM tags, so the CAM-based design is preferable since a higher associativity is needed for his design.

6. CONCLUSIONS

A new serial-parallel CAM (SPCAM) design has been proposed which consumes about a quarter of the energy of a conventional low-power CAM, when used as a cache tag store/comparator. It exploits the address patterns commonly found in application programs, where testing the four LSBs of the tag is sufficient to determine over 90% of the tag mismatches; the proposed CAM checks those bits first and evaluates the remainder of the tag only if they match. In addition the search lines do not have to be forced to '0' or '1' while precharging the match line, which accounts for almost half of the energy of a conventional CAM. The proposed CAM is also adaptive, i.e. it can be configured to work serially as described above or it can operate as a parallel CAM with less energy benefit than in serial mode, but at the same speed as a conventional CAM.

SPCAM's energy consumption is comparable to that of reading a RAM of similar capacity. Thus using this CAM for the tag parts of a sub-blocked high-associative cache, would make this cache more energy efficient compared to way-predicting (pseudo) associative caches.

7. ACKNOWLEDGEMENTS

Thanks are due to Jeff Pepper and Will Toms for their help with taming the technology and some of the tools used. The authors would also like to thank the reviewers for helping to improve the paper with their useful comments. A. Efthymiou is funded by the Department of Computer Science, University of Manchester. This support is gratefully appreciated.

8. REFERENCES

[1] J. Montanaro, R. T. Witek, K. Anne, A. J. Black, E. M. Cooper, D. W. Dobberpuhl, P. M. Donahue, J. Eno, G. W. Hoeppner, D. Kruckemyer, T. H. Lee, P. C. M. Lin, L. Madden, D. Murray, M. H. Pearce, S. Santhanam, K. J. Snyder, R. Stephany, and S.C. Thierauf. A 160-MHz, 32-b, 0.5-W CMOS RISC microprocessor. *IEEE Journal of Solid-State Circuits*, 31(11):1703–1714, November 1996.

[2] T. Burd. *Energy-Efficient Processor System Design*. PhD thesis, Department of Electrical Engineering and Computer Sciences, 2001.

[3] A. Hasegawa, I. Kawasaki, K. Yamada, S. Yoshioka, S. Kawasaki, and P. Biswas. SH3: High code density, low power. *IEEE Micro*, 15(6):11–19, December 1995.

[4] J. Hennessy and D. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann, second edition, 1996.

[5] K. Inoue, T. Ishihara, and K. Murakami. Way-predicting set-associative cache for high performance and low energy consumption. In *Proc. International Symposium on Low Power Electronics and Design*, pages 273–275. ACM Press, August 1999.

[6] B. Calder, D. Grunwald, and J. Emer. Predictive sequential associative cache. In *Proc. 2nd International Symposium on High-Performance Computer Architecture*, pages 244–253. IEEE Computer Society Press, February 1996.

[7] M. Huang, J. Renau, S. Yoo, and J. Torrellas. L1 data cache decomposition for energy efficiency. In *Proc. International Symposium on Low Power Electronics and Design*, pages 10–15. ACM Press, August 2001.

[8] B. Amrutur and M. Horowitz. A replica technique for wordline and sense control in low-power SRAM's. *IEEE Journal of Solid-State Circuits*, SC-33(8):1208–1218, August 1998.

[9] J. Garside, S. Furber, and S. Chung. AMULET3 revealed. In *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 51–59, April 1999.

[10] I. Hsiao, D. Wang, and C. Jen. Power modeling and low-power design of content addressable memories. In *Proc. International Symposium on Circuits and Systems*, pages 926–929. IEEE Computer Society Press, May 2001.

[11] M. Zhang and K. Asanovic. Highly-associative caches for low-power processors. In *Proc. Kool Chips Workshop, 33rd Intl. Symposium on Microarchitecture*, December 2000.