

## Architectures for large-capacity CAMs

Kenneth J. Schultz\*, P. Glenn Gulak

*Department of Electrical and Computer Engineering, University of Toronto, Toronto, Ont, Canada, M5S 1A4*

Received 26 August 1994

---

### Abstract

Content addressable memories (CAMs) have significantly lower capacities than RAMs. Following a summary of large-capacity CAM applications and a brief tutorial look at CAM operation, this paper reviews the sources of this capacity disadvantage: comparator area overhead and difficulty implementing two-dimensional decoding. Past attempts at achieving higher CAM density and capacity are reviewed, and advantages and disadvantages of each are discussed qualitatively. Architectures are divided into the broad classes of serial and fully parallel. The former include bit-serial, Orthogonal-RAM-based, insertion-memory, word-serial, multiport, vector-centered, pattern-addressable memory, and systolic associative memory. The latter include standard architectures, post-encoding, and pre-classification. A taxonomy, providing the first structured comparison of existing techniques, is presented. Thereafter, four architectures (two serial and two fully parallel) are quantitatively analyzed, in terms of delay, area, and power, and the cost-performance measures  $\text{area} \times \text{delay}$  and  $\text{power} \times \text{delay}$ . The fully-parallel architectures, despite their high cost, produce superior cost-performance results.

*Keywords:* CAM; Content addressable memory; Associative memory; Large-capacity; Density; Memory architecture; Pre-classified CAM; Taxonomy

---

### List of symbols

$B$	number of blocks in the memory
$\beta$	width of the orthogonal port word in bits
$C$	number of classes for a pre-classified CAM
$E$	number of standard-architecture strips in a post-encoded CAM
$\varepsilon$	number of entries per class
$k_A$	ratio of one-bit comparator area to one-bit storage area
$k_P$	ratio of power per storage bit to power per comparator bit
$K$	number of comparators
$m$	number of match lines per physical row
$n$	width of CAM word in bits

\*Corresponding author.

$p$	number of memory ports
$r$	number of target RAM decode bits
$S_b$	number of serial cycles to search all bits
$S_w$	number of serial cycles to search all words
$w$	number of words
$\Omega$	number of words along a match line
$x$	number of row address bits
$X$	row decode ratio
$y$	number of column address bits
$Y$	column decode ratio

## 1. Introduction

In the vast majority of today's semiconductor memories, data is accessed based on its location. There are, however, many applications – ranging from artificial neural networks to LAN bridges – for which the data's location is irrelevant, and access based on data content is more natural. Memories operating on this access principle are known as content addressable memories (CAMs) or associative memories [1, 2] and have been studied for over 30 years [3]. Despite the introduction of integrated circuit CAMs as early as 1969 [4], these memories have yet to gain a significant market share.

Market acceptance has been hampered due to the large disparity between CAM chip capacity and that of RAMs. This disparity is caused by a number of factors. The added circuit complexity, in the form of comparators, that is required to implement content addressability dictates that CAM density lags RAM density. At the same time, inherent architectural barriers restrict total chip capacity. Also, the relatively small CAM market provides chip manufacturers little return on investment for applying state-of-the-art fabrication technology. And without adequate density at competitive prices, the market will not grow. A catch-22 ensues, from which the only escape is to develop architectural or circuit techniques that lead to significant density and capacity improvements at an affordable cost.

Numerous architectures have been proposed for large-capacity CAMs. While none have single-handedly provided an escape from the catch-22 described above, many show promise in certain applications. We believe that a structured view of the developments in this area can lead to increased use of CAMs, and hopefully to further fundamental architectural improvements. Hence, this paper provides such a taxonomic and quantitative review. Section 2 comprises necessary background material on CAMs. Various large-capacity CAM realizations are then reviewed: first, serial architectures in Section 3, followed by fully parallel architectures in Section 4. A taxonomy of these implementation schemes – the first such taxonomy in the literature – is presented in Section 5, followed by a quantitative cost-performance analysis in Section 6. Conclusions follow in Section 7.

## 2. CAM background

### 2.1. Applications

There are many applications requiring multi-megabit memory capacity, for which content addressable memory access is the most natural choice. Because CAMs of this size are unavailable,

these applications are: (1) limited to unrealistically and impractically small sizes, (2) implemented in a compromising way using RAMs, or (3) simply unrealizable.

Megabit-capacity associative memories are required for databases (where a CAM-based implementation could realize an intelligent hardware database requiring minimal micro-processor control), main memory for Prolog or LISP computers, and artificial neural networks.

Megabit look-up tables could make hardware dictionary processors commercially feasible. They could increase resolution and system throughput for motion detection and image compression, for such varied applications as target-tracking and HDTV broadcast. They would be useful in translation look-aside buffers (TLBs), especially as virtual address spaces increase to 64 bits and beyond. Lastly, there are numerous applications in data communications and telecommunications, including LAN bridges and address translation for ATM switches.

## 2.2. CAM operation

Basic CAM operation can be understood with the aid of Fig. 1. Data stored in the array's core cells is searched by applying a reference word (the data you hope to find) on the *bit lines*, which run vertically. The *match lines*, running horizontally, will be pulled low by any mismatched bit to which they are connected. Thus, in rows where the reference data matches the stored data exactly, the match line will remain high. The encoder will select a single row in the case of multiple matches, and will output a *hit* signal along with the binary address of the selected row. That row can then be accessed for subsequent reads and writes. If desired, certain bits can be excluded from the search criteria by employing a *mask*. There are many variations on this architecture, and these will be explained in the following sections of this paper.

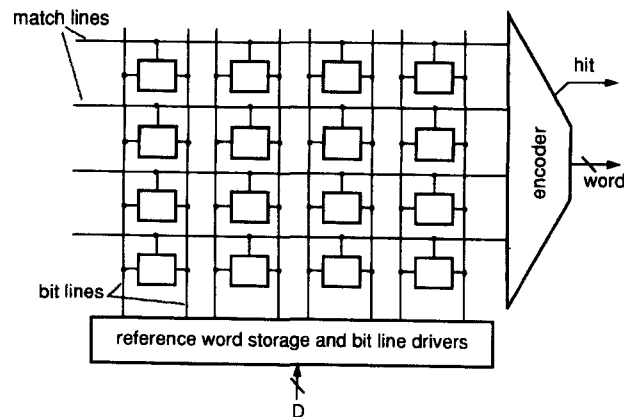


Fig. 1. CAM architecture: “D” is the input data, used both for loading the contents of the CAM, and for the search reference word; “word” is the encoded address of the matched word; and “hit” is a binary signal indicating whether a match was found.

### 2.3. Density and capacity problems

Each core cell in Fig. 1 requires functionality other than simply the storage of a bit of data – it must be able to compare that stored bit with the data on the bit lines. This function requires a one-bit comparator, which is usually implemented as an XNOR gate. This logic gate may be implemented using either 3 or 4 transistors, depending on the core cell design. In terms of actual area, the overhead is considerable. The arrangement of transistors used for bit storage (for static RAM, usually 4 or 6 in number) is hand-crafted for maximum density, often involving close cooperation between circuit designers and fabrication process engineers. Extra transistors are definitely unwelcome here. Their presence could require re-engineering of the fabrication process, but it seldom does, due to prohibitive cost. More likely, the transistors will be retro-fitted into an existing layout, resulting in an area overhead of 60–100%. Certainly, if one could get away with less than one XNOR gate per storage bit, density would improve. This is the basic intent of most of the architectures described in this paper.

However, even with zero-area comparators, architectural difficulties still limit CAM capacity. To see why, let us review memory decoding. We conceptually think of a memory as a  $w \times n$  entity ( $w$  is the number of words, and  $n$  is the number of bits per word). In a naive approach, a  $1\text{M} \times 4$  memory with a  $5\ \mu\text{m} \times 5\ \mu\text{m}$  core cell would have dimensions of  $5\ \text{m} \times 20\ \mu\text{m}$  (this aspect ratio is clearly unacceptable). In order to avoid this problem, words are decoded in two dimensions. Row ( $X$ ) decoding selects among the rows of the memory array, and column ( $Y$ ) decoding selects among the columns (i.e. among the multiple words in the selected row). Using the example above, we may use  $x = 11$  row address bits (or a row decode ratio of  $X = 2^x = 2048$ ) and  $y = 9$  column address bits (or a column decode ratio of  $Y = 2^y = 512$ , such that  $X \times Y = w$ ) to achieve a more reasonable dimension of  $1\ \text{cm} \times 1\ \text{cm}$ . In a RAM,  $X$ -decoding results in a single *word line* being asserted; all cells along the selected word line are thereby connected to their associated bit lines, and the appropriate bit lines are selected by the  $Y$ -decode circuitry.

The main barrier to large-capacity CAM development has been the architectural difficulty in implementing column decoding. As shown in Fig. 1, a match line, rather than a word line, connects all cells in a single physical row, and all cells are normally checked as part of a single search. If all cells in a row belong to the same word ( $y = 0$ ,  $Y = 1$ ) one is checking a word stored in memory versus a reference word, as desired. However, under this constraint, one cannot achieve high capacity with practical dimensions (witness the above 5 m long memory). If  $y \neq 0$ , either all words in given row must match for the match line to remain high (this is not the operation we want), or only one word at a time, in each row, is enabled to pull down the match line, requiring  $Y \neq 1$  sequential operations to complete the search.

We see, therefore, that serial operation is a natural way of implementing a CAM with  $Y > 1$ . It is also a way of sharing comparators between multiple bits of stored data, thus achieving higher density. Unfortunately, many applications cannot afford the delay associated with serial operation; for these applications, *fully parallel* architectures are required (a search operation can be performed in a single clock cycle using these architectures). We describe both serial and fully parallel architectures in the following sections.

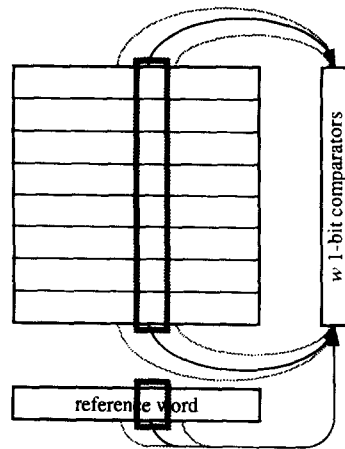


Fig. 2. Conceptual view of a bit-serial CAM.

### 3. Serial architectures

#### 3.1. Bit-serial

Usually  $n \ll w$ ; thus, a bit-serial word-parallel search, in which all bits must be searched sequentially, requires far fewer clock cycles than a bit-parallel word-serial search, in which all words must be searched sequentially. A *bit-serial* CAM is shown conceptually in Fig. 2. A one-bit comparator is associated with each word, along with a single bit of state storage that indicates whether, so far, the word is a match or a mismatch. A “column” of data, comprising a single bit from each word, is read out in parallel; the check, in all rows, is against the same reference bit.

The bit-serial hardware is basically a RAM rotated by  $90^\circ$ ; in fact, appropriately configured RAMs can be used to economically implement this memory [1, pp. 146–150]. Density is essentially equivalent to RAM density, except for the comparators, state flip-flops and buses at the periphery. Because RAMs are typically designed for  $n \ll w$ , and we are reversing the roles of  $w$  and  $n$ , it may prove difficult to construct a CAM from RAMs in this way, whether using commodity RAMs in a board-level design, or semi-custom RAMs in a VLSI design. Reads and writes must also be performed bit-serially, necessitating parallel-serial shift registers and multiple-cycle operations.

One of the largest CAMs described to date [5] has a capacity of 4 Mb, and uses the bit-serial approach. The memory is continually maintained in sorted order by contents; all writes are insertions at the appropriate location, and searches are simplified because there is only one place where a given data word could reside, if it was present. A simplified view of the peripheral circuitry associated with each row is shown in Fig. 3. An insertion operation proceeds by alternating reads and writes, one of each per bit, beginning with the most significant bit and proceeding downward to the least significant bit. As soon as the reference data and the read data differ, the comparator makes a decision and holds that decision for the remainder of the insertion operation; as long as all bits from the read bit-stream match the bits from the reference bit-stream, it is safe to write back the

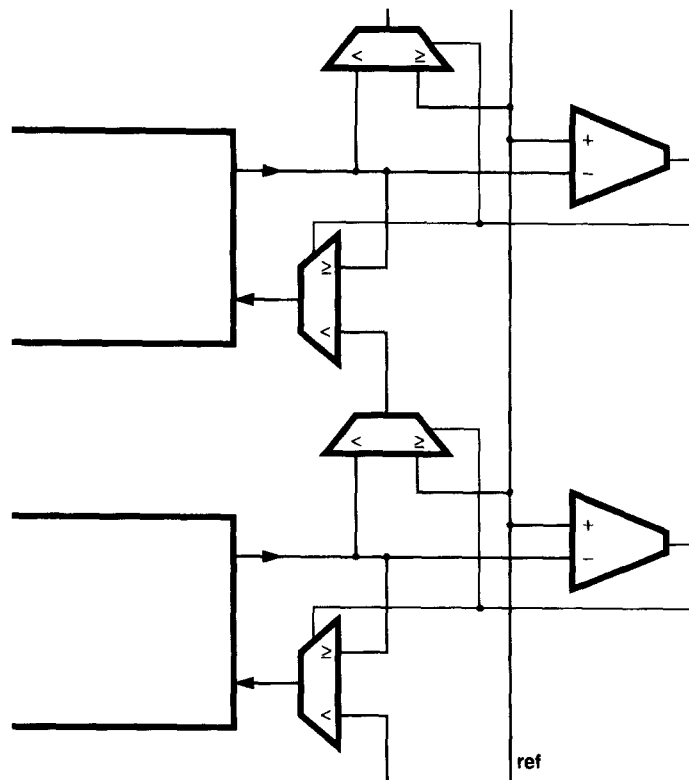


Fig. 3. Peripheral circuitry for an insertion-sorting bit-serial CAM. Each row requires a comparator and two multiplexers. Two rows are shown to demonstrate their interaction.

read bits (the comparator output is “=”). A search operation proceeds similarly, with the write-back disabled. The result is a remarkably fast sorting speed of  $w \times 2.6 \mu\text{s}$  (linear with  $w$ ), but an extremely slow search speed of  $2.6 \mu\text{s}$  (only 385 kHz). As well, power dissipation is extremely high due to the large number of reads occurring in parallel (64 k), and this may be the reason why, to the best of our knowledge, no report exists of a successful implementation of this chip.

An alternative to the “rotated RAM” is the *orthogonal RAM*, or *ORAM* [6]. It has two access ports perpendicular to each other, and may be used to perform normal bit-parallel reads and writes in addition to bit-serial searches. Density is poor due to the complexity of the core cell.

All of the architectures described so far in this section require  $n$  clock cycles to perform a search, one per I/O bit. While this may introduce too much latency for some applications, it allows general-purpose (and powerful) bit-serial processing on the bit-streams; this is known as *content addressable processing* or *associative processing* [1, 7]. Simply put, the comparator is augmented with an opcode-controlled ALU, and a large number of possible operations can be performed on search responders.

An alternative architecture, functionally similar to a bit-serial approach and architecturally similar to an ORAM, but requiring only  $n/\beta$  clock cycles for a search, could be based on the buffer output circuitry described in [8], as shown in Fig. 4 for  $\beta = 4$ . Unless three or four levels of metal

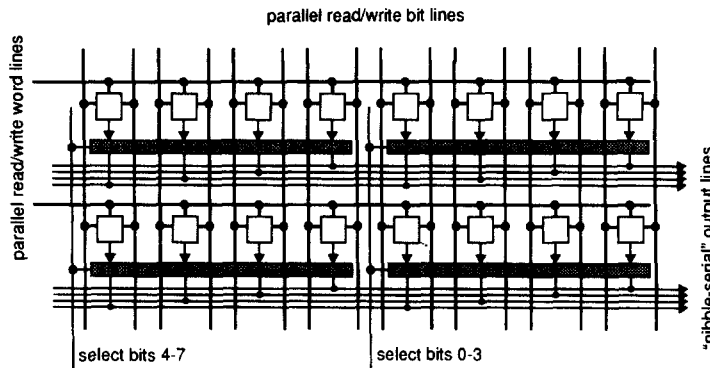


Fig. 4. A memory core for “nibble-serial” searches. Words are written into the memory in a single clock cycle, and may be read similarly. Searches are performed by checking  $\beta = 4$  bits at a time, after reading them from the memory via an orthogonal port with  $\beta \times w$  output lines.

are available, the memory core would have poor density due to the area required to accommodate the multiple wires per row destined for the “nibble-serial” output. The significance of this architecture (hereafter “modified ORAM”) will be explained in Section 5.

The above architectures time-share comparator gates between multiple bits for improved density at the expense of throughput. The extent to which the decoding-related capacity problem is solved varies, depending on the architecture. The “rotated RAM” implementations usually include two-dimensional decoding. The insertion memory can be divided into blocks, similar to the memories that will be presented in Section 3.3. The ORAM and modified ORAM do not effectively address the capacity issue.

### 3.2. Word-serial

A word-serial CAM is one in which the words are read one by one (in their full bit-width), for comparison to the reference word, as shown in Fig. 5. This is obviously just a RAM with an external  $n$ -bit comparator. Therefore, it is simple to construct, achieves density equivalent to a standard RAM, and can employ two-dimensional decoding for RAM-like capacities; unfortunately,  $w$  clock cycles are required to perform the search operation.

Of course, if one is willing to sacrifice density, one may read more than one word in each clock cycle. In fact, the ORAM of Section 3.1 is a *multiport* memory that allows multiple simultaneous accesses. Multiport memories need not have orthogonal ports, however. For example, Silburt et al. [9] describe a memory architecture and circuit design that can be used to implement one-, two- or four-port memories ( $p = 1, p = 2, p = 4$ ), with  $p$  parallel accesses (to full words, not word fragments or orthogonal words). The more ports, the larger and more complex the core cell. To implement a CAM, an  $n$ -bit comparator can be provided in conjunction with each port, resulting in a search time of  $w/p$  cycles. A conceptual view of this CAM is shown in Fig. 6. While throughput is improved with this architecture, and two-dimensional decoding can still be employed, density is not notably better than for an XNOR-per-bit CAM (in fact, it will certainly be poorer for  $p > 2$ ).

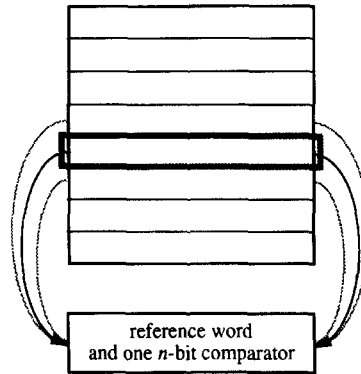


Fig. 5. Conceptual view of a word-serial CAM.

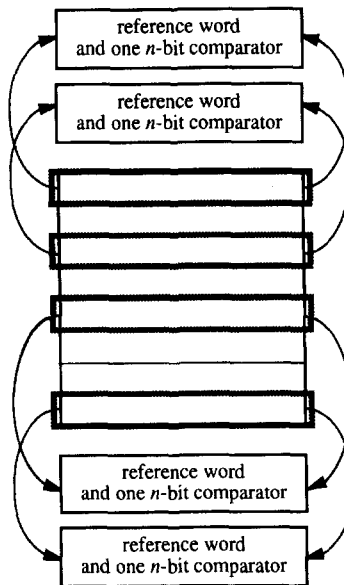


Fig. 6. Conceptual view of a multiport RAM used in a word-serial CAM with search time of  $w/p$  (here,  $p = 4$ ).

### 3.3. Variations on word-serial

In the general-purpose multiport memory above, each word can be accessed by each port. This flexibility is not actually required to achieve a faster CAM search. Instead, we can use single-port core cells, but divide the entire memory array into  $B$  blocks. As shown in Fig. 7, each block has its own  $n$ -bit comparator. We can thus perform a search in  $w/B$  clock cycles, with memory core density equivalent to a standard RAM. There is a density disadvantage due to the replication of peripheral circuitry, but some control circuits and global I/O circuitry can still be shared between blocks, so the area is considerably smaller than that of  $B$  completely separate memories. The division into

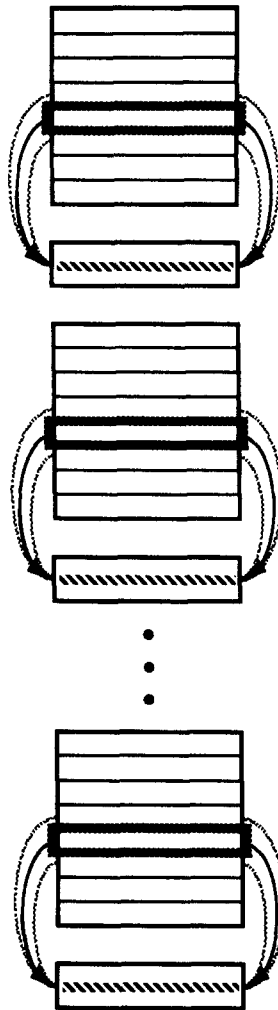


Fig. 7. Searching multiple words in parallel using multiple blocks and single-port core cells. The cross-hatched lines represent the match lines.

multiple blocks (*block decoding*) provides us with the second dimension of decoding we need to achieve large densities. The blocks themselves can be arranged in two-dimensional arrays, as well, to allow increased flexibility in aspect ratios.

Different versions of this architecture can be distinguished by the manner in which data are distributed among the blocks. If each block is used to store a vector of words [10], we may perform vector-matching by cycling through the blocks in parallel, checking the first element in each vector versus the first reference element, then the second, and so on. This type of CAM, used for code-book matching, requires that the match line remain high through an entire cycle of  $w/B$  searches for a hit.

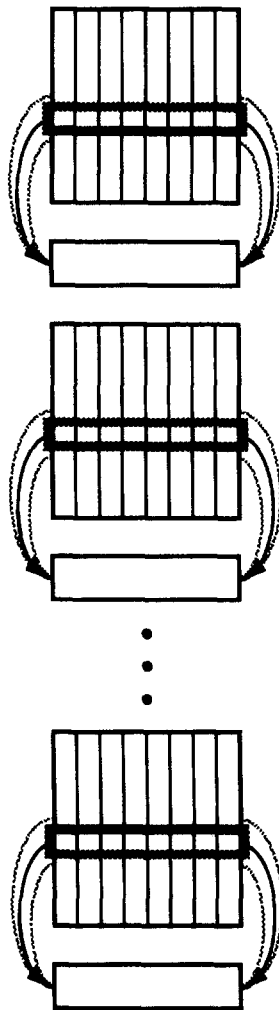


Fig. 8. The same physical architecture, with bits aligned to enable bit-serial searches and processing. There are no match lines.

Alternatively, we may organize the memory such that the  $i$ th word of each block belongs to the  $i$ th *page*. We search pages serially by stepping through the blocks in parallel. This type of CAM is called a *pattern addressable memory* or *PAM* [11]. The match lines are oriented as above, but we do not require matches across successive search cycles for a hit. If we know in advance that the data we are seeking could only be located in some subset of the pages, these are all that have to be searched (the search can be “pruned”), and the number of cycles required for a search is further decreased.

If we access the multiple-block memory as in Fig. 7, but arrange the words as shown in Fig. 8, we again have bit-serial searches. This memory requires no match lines, since all processing associated with a single word is carried out with a one-bit comparator and state information. This architecture was employed by Lipovski [12] in his *systolic associative memory* (hereafter “*SAM*”). When the

memory core is implemented using DRAM, much higher density and capacity can be achieved, and the multi-cycle search operation can be coordinated with the refresh operation. The memory described in [12] has a capacity of 4 Mb, and can be implemented with only slight layout modifications to commodity DRAMs. This architecture is very promising where: (1) extremely large data bases must be searched, (2) bit-serial processing is a desirable feature, and (3) search times of over 1  $\mu$ s can be tolerated.

As shown in Fig. 1, CAMs have an encoder to resolve among multiple matches and derive the binary address of the selected matched word. All match lines are used as inputs to this encoder. While this hardware is not required by the SAM (because operations are performed on all match responders in parallel, and no address output is generated), it is needed to implement general-purpose CAM functionality. In the architectures presented in this section, match lines do not naturally converge on a single location, and must be routed across the chip, introducing delay and area overhead.

A number of concepts from this section prove useful in architectures for large-capacity, fully parallel CAMs. These include: (1) the PAM with a pruned search (carried to its logical extreme), (2) the independence of adjacent comparators in the SAM, and (3) the need for physically converging match lines.

### 3.4. Word-serial bit-serial

For the sake of completeness, let us briefly consider architectures with serial operation in both dimensions. Two of these are worth noting. If the operation is totally serialized, such that it requires  $w \times n$  clock cycles, only a single comparator is required, and off-the-shelf memories with a by-1 organization may be employed. This architecture may be suitable in the case of extremely slow reference bit streams, but it is generally not very useful.

Alternatively, the search may step through the memory address space one *pixel block* at a time, where a pixel block is an entity associated with an image, and could be  $16 \times 16$  bits in dimension, for example. This type of CAM could be useful in performing motion detection for compression algorithms used in video (MPEG) transmission. The problem of shifting the image through the comparator array is a complicated one, and to the best of our knowledge, no such memory has been implemented.

## 4. Fully-parallel architectures

While the architectures described in Section 3 have many uses, none are applicable to the most throughput-intensive or delay-critical applications. For these, the result is needed in a single clock cycle.

### 4.1. Standard architecture

The *standard fully parallel architecture* was described in Section 2.2 and illustrated in Fig. 1. A one-bit comparator is associated with each core cell, and each physical row must be comprised of a single word. Both density and capacity are thus limited with this high-speed architecture.

#### 4.2. Post-encoding

One may use an aggressive process technology and essentially implement a large number of small CAMs (with narrow aspect ratios,  $y = 0$ ) on a single chip. This was the approach taken by Yamagata et al. in [13], where 288 kb are comprised of  $m = 32$  arrays of  $256 \times 36$  bits of CAM. With reference to the 5 m-long memory of Section 2.3, this is simply breaking the long narrow strip into  $m$  much shorter strips, and packing them together to occupy an approximately square aspect ratio. This results in  $m$  match lines spanning a single physical row of the memory. We call this approach *post-encoding*, because additional circuitry is required to combine and encode match results between the separate arrays, making them behave as a single memory. This is necessary because all match lines do not converge to a location where encoding can be performed in one step. Although the column decode problem is addressed by the architecture and process technology, density (for equivalent technologies) is actually worse than that of a standard fully parallel CAM, due to the area overhead of the match line routing and post-encoding circuits. As well, power dissipation is slightly higher than for a standard implementation.

This architecture is likely the best alternative where: (1) DRAM fabrication technology is available, (2) single-cycle searches are required, and (3) density less than half that of a RAM is tolerable.

#### 4.3. Pre-classification

Alternatively, one may perform *pre-classification* prior to the fully parallel operation to assign the data to one of  $C$  classes and thereby determine which of the  $C$  words sharing a comparator will have access to it. This method, first described by Motomura et al. [14], is logically equivalent to a PAM search pruned down to a single page. High density is achieved because the comparator is time-shared. Unlike any of the memories described in Section 3.3, match lines traverse the entire physical length of the memory. Class-controlled access determines which words along a match line are enabled to pull it down.

Fig. 9 shows a block diagram of a *pre-classified CAM*. The core of the CAM is composed of  $n$  separate RAM blocks, one for each I/O bit. Each block has  $C$  columns; based on the class chosen by pre-classification circuitry, the same column number will be accessed simultaneously in each of the blocks. As in the SAM, adjacent comparators are not associated with the same word, and words are aligned perpendicularly to the *class lines* (these are physically – not logically – analogous to word lines). Unlike the SAM, all bits of a word are searched in parallel. Unlike any memory described thus far, the bits of a word are distributed across multiple blocks. This arrangement provides two advantages: (1) the match lines can converge at the encoder at one end of the memory core, and (2) all core cells associated with a given bit are grouped together, so that input/output and search-masking circuitry need not be replicated. Because of the significant capacitive load on the long match lines, pull-down circuits may require powerful BiCMOS drivers [15].

During a search, the reference data are asserted onto the *reference bit lines* (running vertically). Data from the selected class are read onto the *subarray bit lines* (running horizontally) and are compared to the reference data in the shaded boxes. Each of the rows has access to an XNOR gate and match line pull-down circuit. The match lines (running horizontally toward the right) are

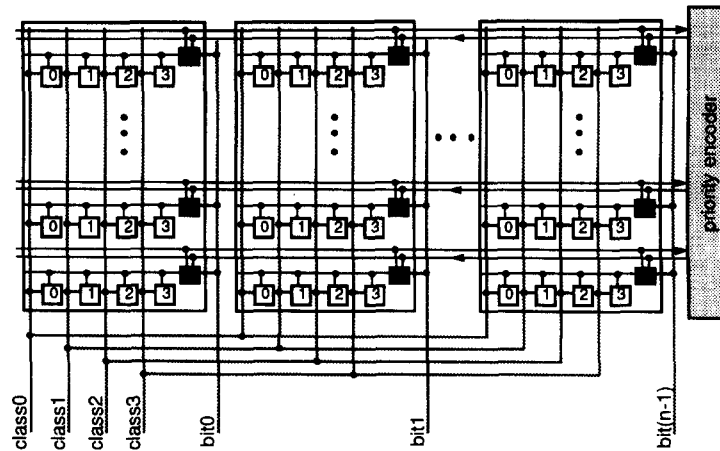


Fig. 9. Pre-classified CAM block diagram.

pulled low by any unmatched bit in their row. Match lines are processed in the priority encoder, where a single match is selected in the event of multiple matches.

During a read, a single *responder select line* (running horizontally to the left) will be asserted, based on the previous search result. In each block, the accessed class will be read onto the subarray bit lines. In the row corresponding to the asserted responder select line, the sensed data will be connected, through the shaded box, to the vertical reference bit lines, for reading out of the memory.

During a write, a single column is selected in each subarray by a class line. In the row selected by the responder select line, write data will be asserted from the reference bit lines to the subarray bit lines, and from there written into the selected cell.

With the match lines running in the same dimension as in the standard architecture, the extra dimension of decoding that is provided by the class lines is logically equivalent to column decoding; the analysis of Section 5 will treat pre-classification in this light.

Pre-classification puts constraints on the location in the CAM in which a given word may be placed – a  $w$ -word CAM is normally fully associative or “ $w$ -way set associative” (for a quantitative explanation of associativity, see [16]); a pre-classified CAM is  $w/C$ -way set associative. As a consequence, a class of the CAM may become full before the CAM as a whole is full. Methods of successfully dealing with this situation are presented in [17].

#### 4.4. Combining post-encoding and pre-classification

If we assign the variable  $\Omega$  to the number of words sharing a match line, then  $\Omega = C$  for a pre-classified CAM. We have  $\Omega \times n$  cells in the  $y$  dimension, and  $\epsilon$  cells in the  $x$  dimension (where  $\epsilon$  is the number of entries per class). In some applications, we may wish to have  $\epsilon \geq \Omega \times n$ , and a reasonable aspect ratio is unattainable. In this case, we may use a combination of post-encoding and pre-classification, whereby  $m$  strips of  $\Omega \times n$  by  $\epsilon/m$  cells are placed side-by-side (yielding  $m \times \Omega \times n$  columns), with post-encoding required to combine the search results of the  $m$  strips.

Density is slightly poorer than a pure pre-classified CAM due to the added post-encoding circuitry, but is still good because the comparators are shared within each strip.

Another aspect ratio problem,  $\varepsilon \ll \Omega \times n$ , is solved to some extent by integrating  $2^r - 1$  RAM words with each CAM word [17], with access controlled by  $r$  additional address bits ANDed with the responder select lines, yielding  $\Omega \times n$  columns by  $2^r \times \varepsilon$  rows. As described in [17], the added functionality and layout efficiency provided by this integration are also beneficial.

## 5. A taxonomy

In order to get a better understanding of the variables and relationships discussed above, one may construct a taxonomy of CAM architectures, as shown in Fig. 10. This analysis, an extension of the work in [17], is the first structured comparison of existing techniques, to the best of our knowledge.

The root node is simply denoted “CAM”. The variable  $S_b$  denotes the number of serial steps needed to search all bits;  $S_b = 1$  means all bits are searched in parallel. The variable  $S_w$  denotes the number of serial steps needed to search all words;  $S_w = 1$  means all words are searched in parallel.

The leaves of the  $S_b \neq 1, S_w \neq 1$  branch include the low-performance fully serial architecture, requiring only 1 comparator gate ( $K = 1$ ), and the pixel block architecture, requiring  $K = i \times j$  comparators. The total number of clock cycles for a search can be found by multiplying  $S_b \times S_w$ .

Bit-serial CAMs (with  $S_b = n$ ) constitute one leaf of  $S_b \neq 1, S_w = 1$ , and the other leaf represents architectures in which multiple bits are searched at once, from all words in parallel. The modified ORAM of Fig. 4 functions in this way, and the orthogonal word-width  $\beta$  can be varied in a density-versus-speed trade-off.

Various leaves of  $S_b = 1, S_w \neq 1$  could include strictly word-serial CAMs (implemented simply with a RAM and one external  $n$ -bit comparator, for  $S_w = w$ ), or architectures with  $K = n \times p$  comparators and a  $p$ -port memory to compare  $p$  words in parallel ( $S_w = w/p$ ). The block architectures, with  $S_w = w/B$ , also belong to this branch – the vector-centered CAM, the PAM, and the SAM.

The fourth branch from the root node represents fully parallel architectures, with  $S_w = S_b = 1$ . At this point we employ two variables introduced in the previous section:  $m$ , the number of match lines spanning a physical memory row, and  $\Omega$ , the number of words attached to each match line. Recall from the discussion of Section 2.3 that  $Y$  is the number of words in a physical row, and  $Y \neq 1$  is required to realize large-capacity CAMs without serialization. The actual physical factors that lead to this variable can be expressed as  $Y = m \times \Omega$ . The standard fully parallel architecture may be denoted by  $Y = m = \Omega = 1$ .

We have also described a number of cases with  $Y \neq 1$ . If the memory is divided into a number of strips  $Y = m = E$ , all of which are searched in parallel, we have post-encoding. As mentioned earlier, such memories are useful when a density disadvantage is tolerable in light of the high throughput rate. Note, as well, that this architecture does not suffer from an associativity decrease, as does the pre-classified CAM. Such an architecture can be viewed as a means of attacking the CAM capacity problem with as much hardware as possible. This is exactly the approach advocated (for digital circuits in general) by Chandrakasan et al. [18] for achieving minimal power dissipation – trade-off chip area versus clock speed by increasing processing parallelism and decreasing power

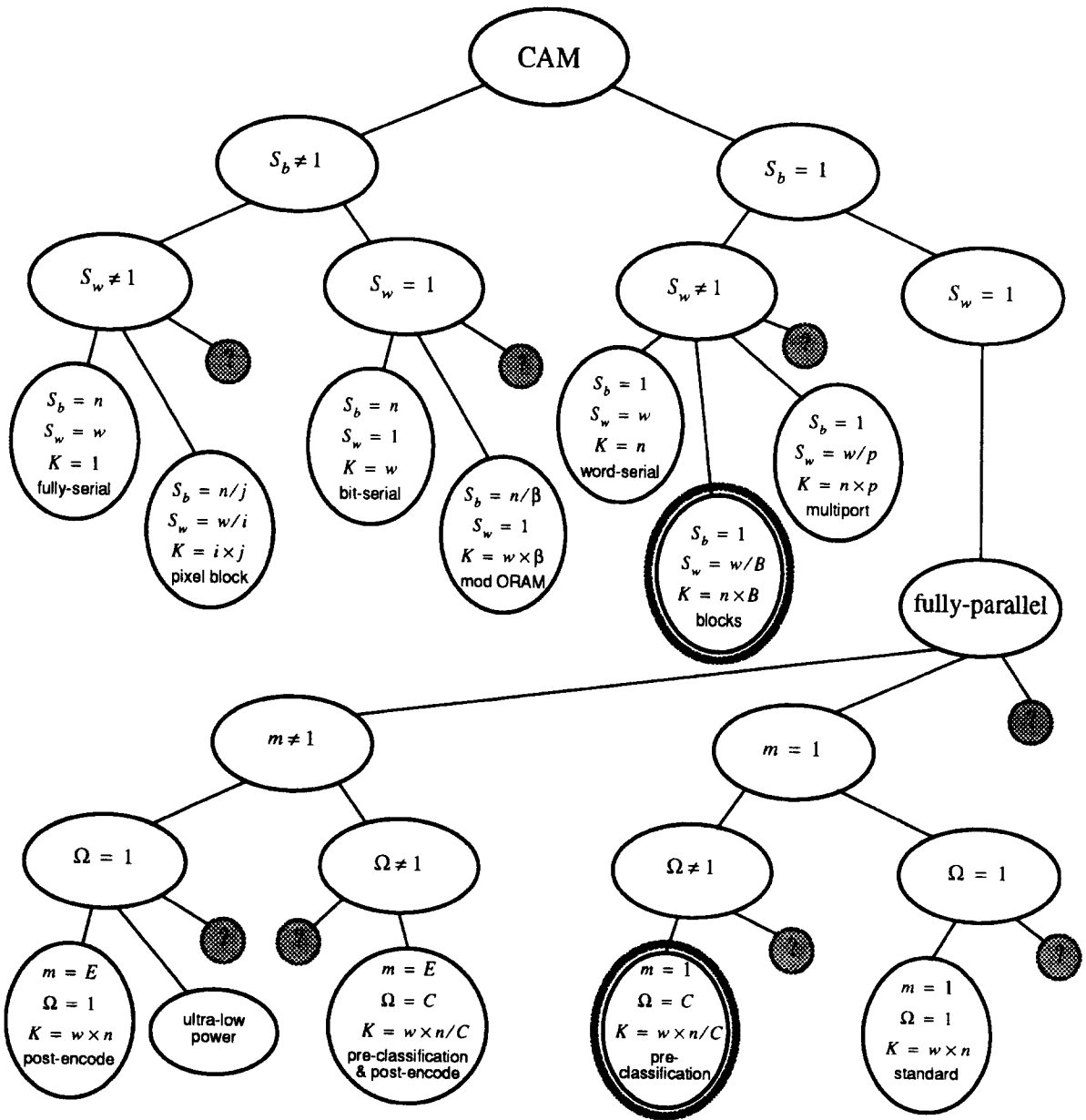


Fig. 10. Taxonomy of CAM architectures. Variables are defined at the beginning of the paper and explained in the text. The nodes outlined in bold, though not directly connected here, are somewhat similar, as described in Section 4.3 of this paper.

supply voltage. The following example illustrates the point: A post-encoded CAM, occupying 200 mm<sup>2</sup> of chip area, with a 1 V power supply, performing a single-cycle search at 1 MHz, will always dissipate less power than a 50 mm<sup>2</sup> bit-serial CAM (of the same capacity) at 5 V requiring

100 clock cycles at 100 MHz to complete the same search in the same amount of time. This is a very useful notion for the design of future battery-operated personal communication devices, many of which may need CAM functionality.

Alternatively, one may perform pre-classification, with  $Y = \Omega = C$  classes. This architecture requires a factor of  $C$  fewer comparators, but the associativity is decreased by the same factor, and memory capacity inefficiencies may result. Where aspect ratio concerns dictate, post-encoding can be performed, allowing the combination of  $m$  blocks of pre-classified CAM, each with  $\Omega \times n \times \varepsilon/m$  bits and  $n \times \varepsilon/m$  comparators, where  $\varepsilon = w/C$ .

By formulating a taxonomy, one may discover new branches to be explored. A few of the avenues for future research are denoted by question marks in Fig. 10. The most fruitful of these is likely the relatively unconstrained branch attached to the “fully parallel” node – perhaps some completely new variables can be introduced here to give further insight.

## 6. Analysis of cost and performance

In Section 5, the delay of each architecture was quantified (in clock cycles), along with a measure of complexity (the comparator count). It would be useful to quantify standard cost measures such as power and area. While exact data are impossible to obtain without real circuit implementations, preferably in the same process technology, illuminating technology-independent trends can be derived from the information we have at our disposal.

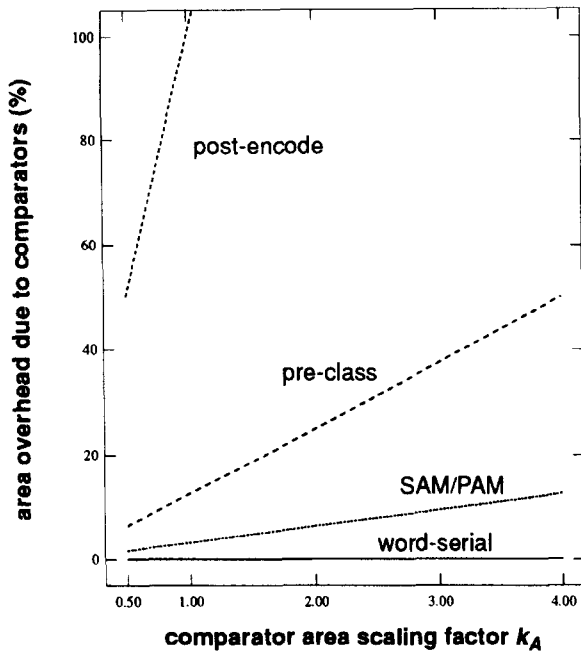
Area can be approximately determined as the sum of storage area and comparator area. Uncertainty in this calculation is due to the unknown ratio of the area of a single one-bit comparator to the area of one bit of storage. If we introduce a scaling factor  $k_A$  to represent this ratio,<sup>1</sup> we can determine the area for each architecture as a function of  $k_A$ . Given this area relationship, we can then find a trend for area-delay product, since delay is also known. This product is a good indicator of the relative merit of different architectures, since it is basically a cost-performance ratio.

To a first-order, power dissipation is the sum of power due to comparators plus power due to bit line transitions. The former is proportional to the number of comparator bits, and the latter is proportional to the total number of memory bits. If only a fraction of the memory was active in each cycle, bit line power would be proportional to the total number of bits in active portions of the memory; all architectures under consideration here maintain a completely activated memory space during all searches. The only unknown is the ratio between power per storage bit and power per comparator. Again, we introduce a scaling factor  $k_P$  to represent this ratio,<sup>2</sup> and then determine power as a function of  $k_P$ . From this, we may derive a trend for the power-delay product.

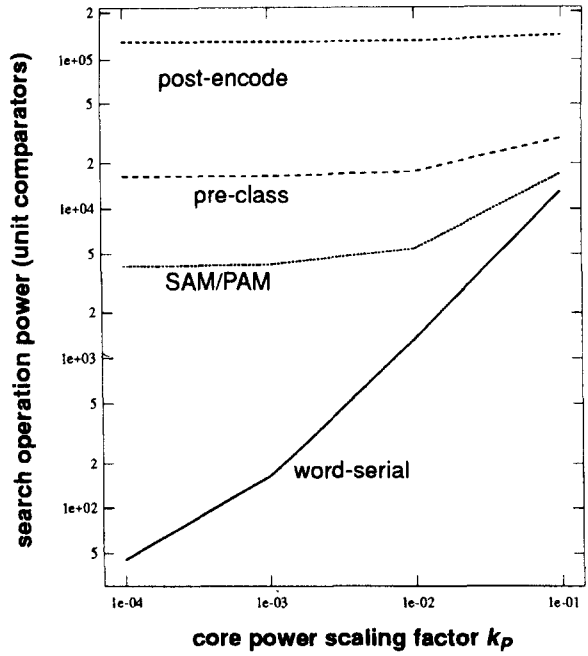
We need not analyze every one of the architectures presented in this paper. First, note that the standard fully parallel architecture is not capable of achieving truly large capacities. Similarly, unless the bit-serial architecture is divided into blocks, large capacities are not attainable. The most logical block-decoded bit-serial architecture to select is the SAM.

<sup>1</sup> We let  $k_A$  vary from 0.5 to 4.

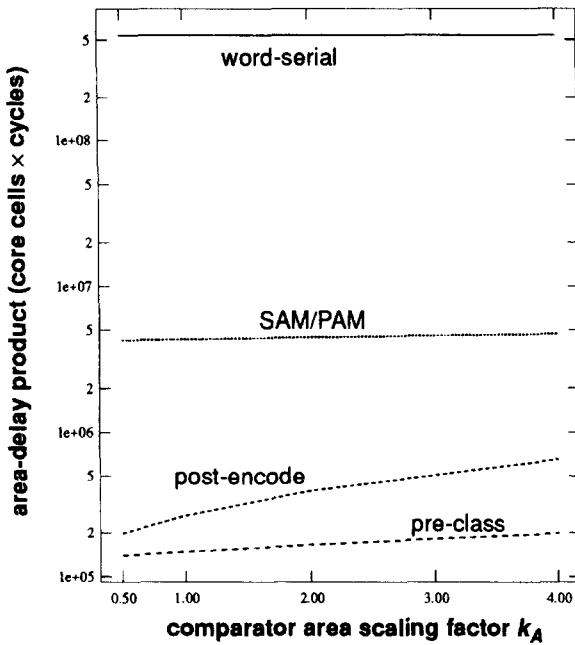
<sup>2</sup> We let  $k_P$  vary from  $10^{-4}$  to  $10^{-1}$ .



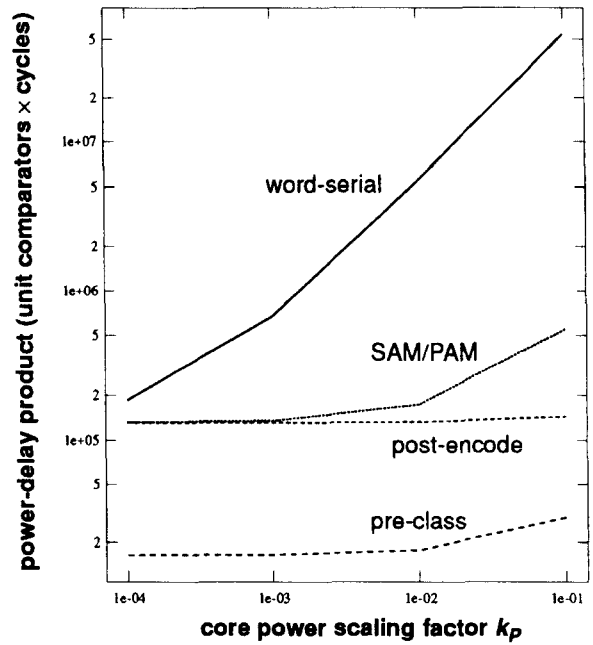
(a)



(b)

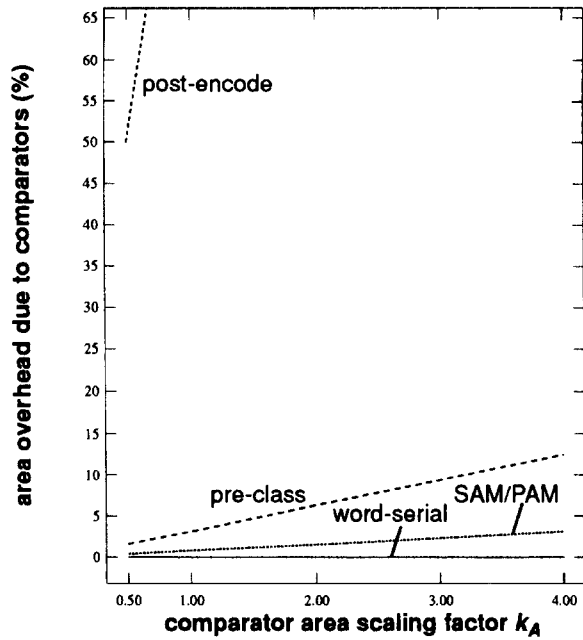


(c)

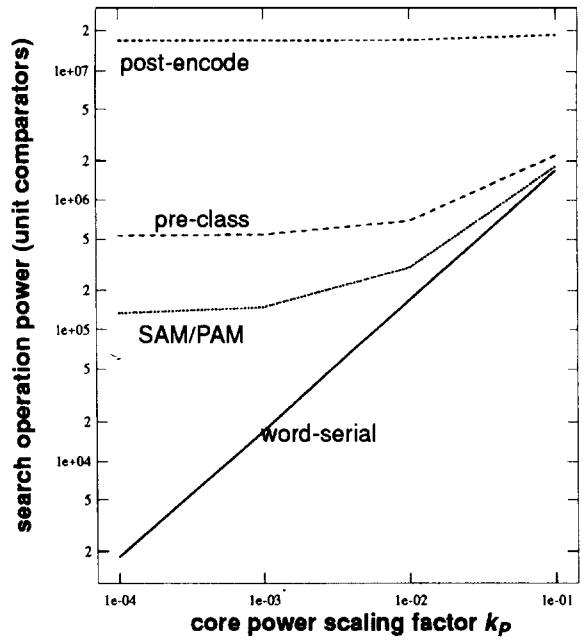


(d)

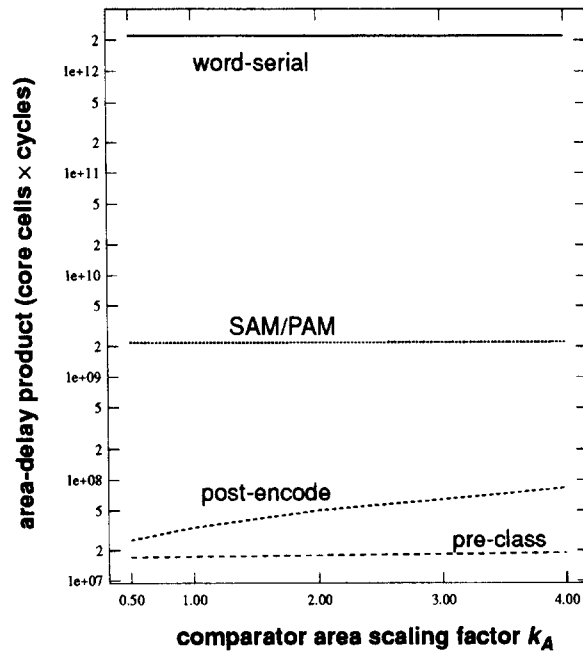
Fig. 11. Cost and performance trends for four CAM architectures for a 4 K × 32 configuration.



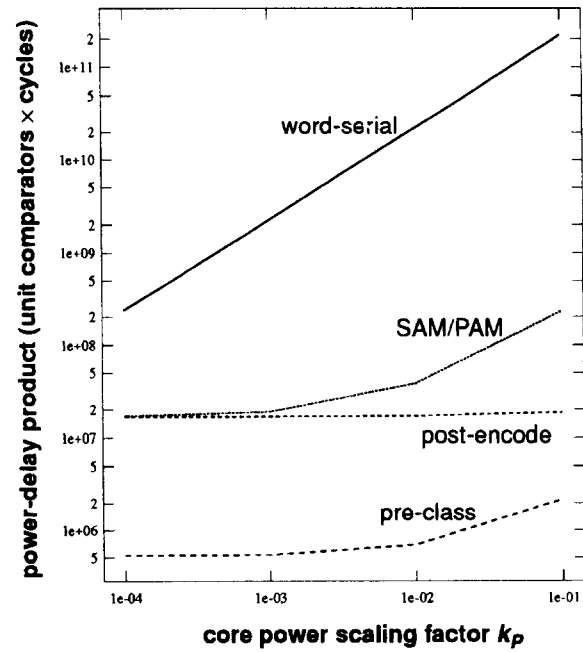
(a)



(b)



(c)



(d)

Fig. 12. Cost and performance trends for four CAM architectures for a 128 K  $\times$  128 configuration.

For the purposes of our simple analysis, the cost and performance of the PAM can be shown to be identical to those of the SAM, as follows. A SAM requires  $w$  comparators and  $n$  cycles to complete a search. In dividing a PAM into blocks, a logical assumption would be to insist that each block be square, or  $n \times n$ . Hence, the number of blocks is  $w/n$ , and the total number of comparators is  $n$  per block times  $w/n$  blocks, or  $w$ . The page size is  $n$ , which is also the cycle count for a search.

While comparator counts are simply determined as  $w \times n$  for the post-encoded architecture, we also have to make some assumptions in the case of the pre-classified CAM. Here, we insist that the over-all aspect ratio of the memory be approximately square (as close as possible with power-of-two decode ratios). The total number of rows is set to  $\varepsilon$ , and the total number of classes to  $C = w/\varepsilon$ . Therefore, the number of comparators can be found as  $K = (w \times n)/C = n \times \varepsilon$ .

Trend lines are plotted for the four remaining “interesting” architectures for two different memory capacities. Fig. 11 shows results for a 4 K  $\times$  32 CAM; this capacity approximates the largest commercially available CAMs today. Fig. 12 shows results for a 128 K  $\times$  128 memory; the 16 Mb total capacity approximates today’s largest commodity RAMs, although the word is considerably wider (this is usually the case for CAMs). Area is measured in terms of core cells, with the area of comparator equal to  $k_A$  times that of a core cell. Power is measured in terms of “unit comparators”, with the power associated with a storage bit equal to  $k_P$  of a unit comparator.

The cost of the post-encoded CAM, in terms of area and power, is considerably higher than that of the other architectures; this architecture is not a good choice if chip area and power dissipation are the most important design parameters. However, due to the single-cycle operation, area-delay and power-delay results for the post-encoded architecture are very good. The same general comments can be made regarding the pre-classified CAM, except that costs are not as high, leading to the best cost-performance ratios among the four architectures.

This analysis does not prove that the pre-classified architecture is always preferable (keep in mind that this architecture could suffer from poor capacity usage and efficiency due to the decrease in associativity<sup>3</sup>). Different cost and performance parameters will be more important in different design situations. Conclusions will vary along with  $k_A$  and  $k_P$ . The analysis presented here does, however, highlight some useful general trends.

## 7. Conclusions

Numerous applications would benefit from the availability of CAMs with RAM-like capacities. Such large CAMs have not been realizable, due to circuit density disadvantages, architectural hurdles, and market constraints. Architectural innovations are required to overcome these challenges. Several memory architectures have been proposed in this vein, and these are reviewed in this paper.

Serial CAMs can be implemented with only a few alterations to existing RAM designs, and achieve the highest density. They are suitable for many applications, especially when used in conjunction with bit-serial processing. However, the throughput degradation caused by multi-cycle

<sup>3</sup> Counters could be employed to manage overflows into adjacent classes, leading to more efficient memory usage. For classes that have overflowed, multiple cycles would be required for all searches, and for some writes. Different strategies for dealing with overflows are discussed in [17].

searches is often not tolerable. Various fully parallel CAMs are capable of performing searches in a single clock cycle.

By classifying the architectures into a taxonomy, one is able to visualize design trade-offs, select the best architecture for a set of design goals, and explore new techniques in a systematic manner. In addition to the taxonomic study, a technology-independent cost-performance analysis was carried out, demonstrating the benefits of the fully parallel architectures. Post-encoded CAMs are especially well-suited to advanced fabrication processes or low-supply-voltage environments. The pre-classified CAM provides the best cost-performance results, if memory data can readily be partitioned into classes and a decrease in associativity can be tolerated.

## Acknowledgements

This work has been financially supported by Bell-Northern Research, the Natural Sciences and Engineering Research Council of Canada, the North American Life Assurance Company, the Canadian Advanced Technology Association, and by a Walter Sumner Memorial Fellowship.

## References

- [1] T. Kohonen, *Content Addressable Memories*, (Springer, Berlin, 2nd ed., 1987).
- [2] L. Chisvin and J.R. Duckworth, Content-addressable and associative memory: alternatives to the ubiquitous RAM, *IEEE Comput.* **22** (1989) 51–64.
- [3] A.G. Hanlon, Content-addressable and associative memory systems: a survey, *IEEE Trans. Electronic Comput.* **C-15** (1966) 509–521.
- [4] R.F. Herlein and A.V. Thompson, An integrated associative memory element, *ISSCC Dig. Tech. Papers* (1969) 42–43.
- [5] I. Okabayashi, H. Kotani and H. Kadota, A proposed structure of a 4 Mbit content-addressable and sorting memory, *Proc. Symp. VLSI Circuits* (1990) 109–110.
- [6] A. Kokubu, M. Kuroda and T. Furuya, Orthogonal memory – a step toward realization of large capacity associative memory, in: E. Horbst, ed., *VLSI 85* (North-Holland, Amsterdam, 1986) 165–174.
- [7] C.C. Foster, *Content Addressable Parallel Processors* (Van Nostrand, New York, 1976).
- [8] K.J. Schultz and P.G. Gulak, CAM-based single-chip shared buffer ATM switch, *Proc. Int. Commun. Conf.* (1994) 1190–1195.
- [9] A.L. Silburt, R.S. Phillips, G.F.R. Gibson, S.W. Wood, A.G. Bluschke, J.S. Fujimoto, S.P. Kornachuk, B. Nadeau-Dostie, R.K. Verma and P.M.J. Diedrich, A 180 MHz 0.8  $\mu\text{m}$  BiCMOS modular memory family of DRAM and multiport SRAM, *IEEE J. Solid-State Circuits* **SC-28** (1993) 222–232.
- [10] S. Panchanathan and M. Goldberg, Vector-centered CAM architecture for image coding using vector quantization, *Proc. SPIE 1199, Visual Communications and Image Processing IV* (1989) pp. 1084–1094.
- [11] I.N. Robinson, Pattern-addressable memory, *IEEE Micro* **12** (1992) 20–30.
- [12] G.J. Lipovski, A four megabit dynamic systolic associative memory chip, *J. VLSI Signal Process.* **4** (1992) 37–51.
- [13] T. Yamagata, M. Mihara, T. Hamamoto, Y. Murai, T. Kobayashi, M. Yamada and H. Ozaki, A 288 kbit fully parallel content addressable memory using stacked capacitor cell structure, *IEEE J. Solid-State Circuits* **SC-27** (1992) 1927–1933.
- [14] M. Motomura, J. Toyoura, K. Hirata, H. Ooka, H. Yamada and T. Enomoto, A 1.2 million transistor, 33 MHz, 20 b dictionary search processor (DISP) ULSI with a 160 kb CAM, *IEEE J. Solid-State Circuits* **SC-25** (1990) 1158–1165.

- [15] K.J. Schultz and P.G. Gulak, Fully-parallel multi-megabit integrated CAM/RAM design, *Records of the IEEE Int. Workshop on Memory Technology, Design and Testing* (1994) pp. 46–51.
- [16] J.L. Hennessy and D.A. Patterson, *Computer Architecture: A Quantitative Approach* (Morgan Kaufmann, San Mateo, CA, 1990) p. 409.
- [17] K.J. Schultz and P.G. Gulak, Architecture for multi-megabit integrated CAM/RAM, *Proc. Canadian Conf. on VLSI* (1993) pp. 6A1–6A7.
- [18] A.P. Chandrakasan, S. Sheng and R.W. Broderson, Low-power CMOS digital design, *IEEE J. Solid-State Circuits SC-27* (1992) 473–484.



**Kenneth J. Schultz** was born in Winnipeg, Manitoba, Canada in 1965. He received the B.Sc. degree (Gold Medalist) from the University of Manitoba in 1987, and the M.A.Sc. degree from the University of Toronto in 1989, both in Electrical Engineering. From 1989 to 1991 he was employed in the Memory Development Group at Bell-Northern Research, Ottawa, Ontario, Canada. Since 1991, he has been on a leave of absence from BNR, pursuing doctoral studies at the University of Toronto, where his research is focused on CAM-based circuits for ATM switching. Mr. Schultz has held a Natural Sciences and Engineering Research Council of Canada “1967 Scholarship”, and is the 1994–95 recipient of the IEEE Solid-State Circuits Council Pre-doctoral Fellowship. He was co-recipient of the Best Paper Award at the Canadian Conference on VLSI in both 1988 and 1993, and is a Student Member of the IEEE and a licenced Professional Engineer.



**P. Glenn Gulak, Ph.D.**, is an associate professor in the Department of Electrical and Computer Engineering at the University of Toronto. His research interests are in the areas of circuits, algorithms and VLSI architectures for digital communications and signal processing applications. He has received four teaching awards for undergraduate courses taught in both the Department of Computer Science and the Department of Electrical and Computer Engineering at the University of Toronto. Dr. Gulak received his Ph.D. from the University of Manitoba while holding a Natural Sciences and Engineering Research Council of Canada Postgraduate Scholarship. From 1985 to 1988 he was a research associate in the Information Systems Laboratory and the Computer Systems Laboratory at Stanford University. He has served on the ISSCC Signal Processing Technical Subcommittee from 1990 to 1994 and currently serves as Program Secretary for ISSCC.