Fully Parallel Integrated CAM/RAM Using Preclassification to Enable Large Capacities

Kenneth J. Schultz, *Member, IEEE*, and P. Glenn Gulak, *Member, IEEE*

Abstract— **Many applications would benefit from the availability of large-capacity content addressable memories (CAM's). However, while RAM's, EEPROM's, and other memory types achieve ever-increasing per-chip bit counts, CAM's show little promise of following suit, due primarily to an inherent difficulty in implementing two-dimensional decoding. The serialized operation of most proposed solutions is not acceptable in speed-sensitive environments. In response to the resulting need, this paper describes a fully-parallel (single-clock-cycle) CAM architecture that uses the concept of "preclassification" to realize a second dimension of decoding without compromising throughput. As is typically the case, each CAM entry is used as an index to additional data in a RAM. To achieve improved system integration, the preclassified CAM is merged into the same physical array as its target RAM, and both use the same core cells. Architecture and operation of the resulting novel memory are described, as are two critical-path circuits: the match-line pull-down and the multiple match resolver. The memory circuits, designed in 0.8 μm BiCMOS technology, may be employed in chips as large as 1 Mb, and simulations confirm 37 MHz operation for this capacity. To experimentally verify the feasibility of the architectural and circuit design, an 8 kb test chip was fabricated and found to be fully functional at clock speeds up to 59 MHz, with a power dissipation of 260 mW at 50 MHz.**

I. Introduction

In content addressable memories (CAM's), data areaccessed based on their content, rather than their physical location or address [1], [2]. This access method provides increased flexibility and efficiency in look-up operations. Hence, a CAM is an ideal solution for our intended application: the address translation task required in asynchronous transfer mode (ATM) broadband networks [3]. As such networks expand, higher throughput requirements will be accompanied by a need to store more entries. Unfortunately, existing CAM's—both off-the-shelf components and full-custom architectures—are not capable of searching megabits of data at rates approaching 100 MHz.

In fact, while RAM's, EEPROM's, and other memory types achieve ever-increasing per-chip bit counts, CAM's show little promise of following suit. This is partially attributable to the obviously lower density of the CAM core relative to that of a RAM, since CAM's require comparison circuitry together with bit storage. However, the low capacity is primarily due to an inherent difficulty in implementing two-dimensional decoding in a CAM. Most of the proposed solutions to this problem require serialized operation (see [4], for example), which is not acceptable in speed-sensitive applications, such as ATM address translation. Other solutions take advantage of advanced fabrication techniques like DRAM technology [5]. We believe that the best solution is to develop architectural and circuit innovations that lead to a significant capacity improvement at an affordable cost. Such a solution could lead not only to application-specific CAM's, but also toward RAM-like capacities in commodity CAM's. In this vein, this paper describes a fully-parallel (single-clock-cycle) CAM architecture that uses the concept of "preclassification" to achieve a second dimension of decoding without compromising throughput.

There are many applications that could benefit from economical large-capacity CAM's. Uses for large-capacity associative memories include "smart" databases, memory for Prolog computers, and artificial neural networks. Megabit look-up tables could make hardware dictionary processors commercially feasible, or they could increase resolution and system throughput for motion detection and image compression. They would be useful in Translation Look-aside Buffers (TLB's), especially as virtual address spaces increase to 64 bits and beyond. And there are numerous applications in telecommunications aside from ATM translation tables, including LAN bridges and time switches.

Following a brief survey of existing large-capacity CAM's in Section II, a novel architecture is proposed in Section III. This architecture allows CAM's to achieve RAM-like density and capacity and enables the efficient integration of CAM and its target RAM into a physically contiguous and homogeneous array. The architecture is of little practical use, however, without circuits that actually allow the realization of megabit-capacity chips. In Section IV, we describe the clocking and pipelining strategy, as well as design details for two critical-path circuits: the comparator and match line pull-down circuit, and the multiple match resolver. Simulation results, presented in Section V, demonstrate the effectiveness of the architecture and circuits in a 1 Mb configuration. Section VI describes the 8 kb test chip used to verify these techniques, as well as experimental results. Conclusions follow in Section VII. A 0.8 μm BiCMOS ASIC process [6] is employed for all simulations, as well as for test chip implementation.

Manuscript received January 19, 1995; revised August 8, 1995. This work was supported by Bell-Northern Research and the Natural Sciences and Engineering Research Council of Canada, and by scholarships from the North American Life Assurance Company, the Walter Sumner Memorial Foundation, the Canadian Advanced Technology Association, and the IEEE Solid-State Circuits Council.

K. J. Schultz was with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, Ontario, M5S 1A4 Canada. He is now with Nortel North America, Ottawa Ontario K1Y 4H7 Canada.

P. G. Gulak is with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, Ontario, M5S 1A4 Canada.

Publisher Item Identifier S 0018-9200(96)03405-1.

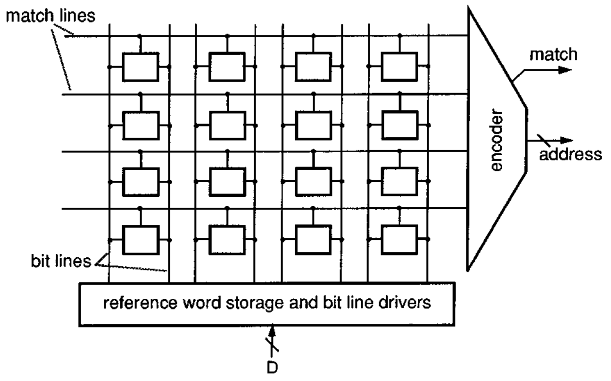


Fig. 1. Fully-parallel CAM architecture: "D" is the input data, used both for loading the contents of the CAM, and for the search reference word; "address" is the encoded location of the matched word; and "match" is a binary signal indicating whether a match was found.

II. Large-Capacity CAM Architectures

In this section, we review existing large-capacity CAM architectures. The description is condensed from the more detailed treatment found in [7]. For the following discussion, we define n as the number of bits per word, w as the number of words, and B as the number of core cell blocks.

A. Standard CAM Architecture

The main barrier to large-capacity CAM development has been the architectural difficulty in implementing column decoding in a *fully-parallel* CAM. Fully-parallel (or "word-parallel bit-parallel") CAM's are those in which all stored words, in all bit positions, are checked against the reference word in a single clock cycle. In a fully-parallel CAM, a match line connects all cells in a single physical row (see Fig. 1). The reference word is asserted on the bit lines, and if there is a mismatch in a cell, the cell will pull down its match line. If all cells in a row match their respective bit lines, the match line remains high, and a match has been found. If all cells in a row belong to the same word, one is checking a stored word versus a reference word, as desired. However, under this "no-column-decode" constraint, one cannot achieve large capacity with practical dimensions. If there are multiple words along each physical row, either all words must match for the match line to remain high (the wrong operation), or only one word at a time is enabled to pull down the match line (requiring serialized operation).

B. Serial Architectures

Usually $n \ll w$, and thus, a bit-serial word-parallel search requires far fewer clock cycles than a bit-parallel word-serial search. In a bit-serial CAM, 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. If we divide the bit-serial memory into blocks and arrange them to achieve a relatively square aspect ratio, large-capacity implementations are possible. This architecture, employed by Lipovski [4] in his systolic associative memory, is not suited to general-purpose CAM applications that require arbitration among multiple matches, because match lines do not naturally converge on a single location. Reads and writes are performed bit-serially, necessitating parallel-serial shift registers and multiple-cycle operations.

A word-serial CAM search can be performed in w/B clock cycles if we simultaneously read all B blocks of a standard w-word block-decoded RAM, provided each of the blocks has its own n-bit comparator. Here, CAM core density is equivalent to that of a standard RAM. Different versions of this architecture can be distinguished by the manner in which data is distributed among the blocks. Each block could store a vector of words [8], and the search could proceed "vector-parallel element-serial." Alternatively, the ith word of each block may belong to the ith "page" [9], and the search proceeds "element-parallel page-serial."

C. Post-Encoding

To achieve a fully-parallel search, one may use an aggressive process technology and essentially implement a large number of narrow CAM's (each without column decoding) on a single chip. Yamagata *et al.* [5] use this approach to implement a 288 kb CAM, made up of 32 arrays of 256×36 bits each. We refer to this approach as post-encoding, because additional circuitry is required to combine and encode match results between the 32 separate arrays.

D. Preclassification

Alternatively, one may perform "preclassification" in conjunction with a fully parallel operation. An n-bit comparator is shared between C words, and these words each belong to a different "class." If the class of the reference data is known before the search takes place, we need only compare one of every C stored words (those of the appropriate class) to that reference, and these are the words that use the shared comparators during that search. This method, based on a physical structure similar to that in [9] above, was first described by Motomura *et al.* [10], but that reference neither explains the trade-offs involved in the architectural design nor places the design in perspective as a means of solving the decoding problem. The second dimension of decoding is realized by the division into classes. In addition, high density is achieved because the comparator is shared, and every cell need not contain match circuitry. But preclassification puts constraints on the location in the CAM at which a given word may be placed—a w-word CAM is normally fully-associative or "w-way set associative,"[1] whereas a preclassified 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.

III. Preclassified CAM

The remainder of this paper uses the word of Motomura *et al.* [10] as a starting point and extends it in two significant ways. First, we consider a number of implementation issues and thus define a complete design space, of which [10] represents a single point. Second, we integrate RAM together

[1] See [11, p. 409] for a quantitative discussion of associativity.

with the preclassified CAM into a single contiguous physical structure, resulting in a novel, high-density, high-speed, stand-alone associative memory.

A. Architecture and Basic Operation

Fig. 2 shows a block diagram of a preclassified CAM. The core of the CAM is physically divided into n separate RAM sub-arrays, one for each I/O bit. It is logically divided into C classes, with each class distributed among the blocks, occupying one row of each. The core cells labeled "0" in the figure all belong to class 0, and likewise for cells labeled "1," "2," and "3." Each block obviously has C rows; based on the class of the data (which is determined by preclassification circuitry), the same row number will be accessed simultaneously in each of the subarrays, by the assertion of the appropriate class lines. Class lines are physically analogous to word lines in a RAM, but in this case the cells they span do not belong to the same word. Rather, words are oriented vertically, such that the least-significant bit of the first class-0 word is the top-left cell in the top block, the second bit of the same word is in the same column at the top of the second block, and so on, down to the most significant bit in the top-left cell of the bottom block. Additional class-0 words are in other columns, and the remaining words in the first column belong to other classes. Two dimensions of decoding are realized, though with the RAM-like decoding functions reversed: cells sharing a row belong to different words, with their selection controlled by the priority encoder as described below, and multiple words share a column, with their selection controlled by the assertion of class lines.

During a search, the reference data is asserted onto the reference bit lines (running horizontally). A class line is asserted, and data from the selected class are read onto the subarray bit lines (running vertically) and compared to the reference data in the shaded boxes. It follows that all of the bits of a given word are being compared to the reference data in the shaded boxes of a given column. A match line (running vertically downward to the priority encoder) is pulled low by any unmatched bit in its column. Match lines are processed by the priority encoder, which selects a single match in the event of multiple matches. The encoder outputs a hit/miss signal and the encoded address of the selected matched word.

During a read, a single *responder select line* (running vertically upward) is asserted by the priority encoder, based on the previous search result. In each subarray, the accessed class will be read onto the subarray bit lines. In the column corresponding to the asserted responder select line, the sensed data will be connected, through the shaded box, to the horizontal bit lines, for reading out of the memory.

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

The number of physical rows in the memory is equal to $n \times C$. The number of physical columns corresponds to the number of entries per class w/C. Note that these two dimensions have to be within, at most, a factor of four to

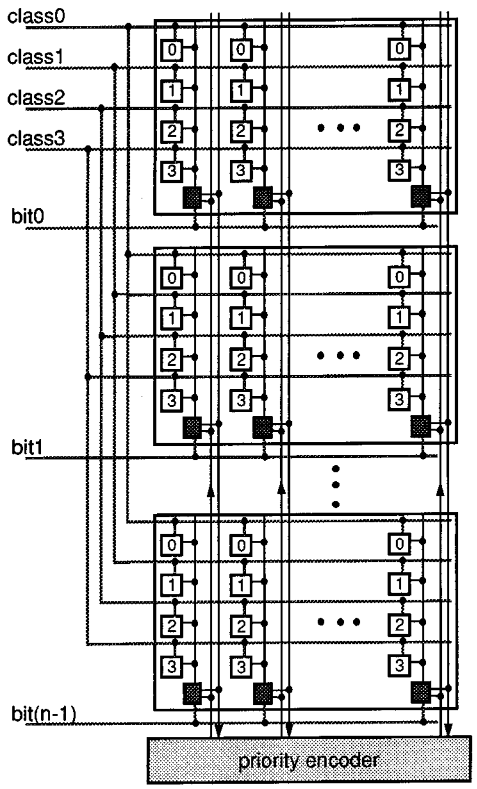


Fig. 2. Preclassified CAM block diagram.

achieve an implementable aspect ratio. It is thus not possible to arbitrarily choose the three quantities n, w, C; rather, only certain combinations make sense for a given application. For example, for $n = 32$ and $w = 32$ K, we are constrained to $16 \leq C \leq 64$.

B. How to Preclassify

We have simply stated that data is preclassified into one of C classes prior to CAM operation. Aside from the obvious observation that the preclassification should be based on data contents, it is not clear how it is implemented. In fact, there are a number of possibilities. The simplest technique is to use an extracted bit field from the data to directly address a class. For example, we may use bits $\langle 7{:}0 \rangle$ of a 32-bit word to select one of 256 classes. To minimize data clustering in a subset of the classes, the bits chosen should have high "information content" or entropy (this is more likely in LSB's than MSB's).

More flexibility is attainable by employing indirect addressing. If the extracted bit field is used to address a small preclassification RAM (PRAM), which in turn stores classes, the contents of this PRAM may be rewritten occasionally to respond to class traffic patterns to evenly distribute load among the classes. This is the technique we have chosen to implement in our test chip—bits $\langle 5{:}0 \rangle$ of the data word are used to address a 64×4 PRAM, and the 4-bit PRAM output word selects one of 16 classes. The most flexible indirect approach is to use a preclassification CAM (or "Control Code CAM" [10]) in place of the PRAM, so that both the class access method and the class loading may be altered. We do not believe the additional flexibility provided by this latter method justifies the additional area and complexity required.

C. Class Overflow Handling

Due to the decreased associativity of a preclassified CAM, individual classes may be full well before the CAM, as a whole, is filled to capacity. If the match encoding circuitry determines that a class is full (no entries match a search for an invalid word), there are a number of ways to deal with the situation:

1) Simply live with it; treat the CAM as C separate units, only one of which may be accessed at a time. This is the simplest solution in terms of control hardware overhead, but the least efficient in terms of memory usage.
2) Allow a class to overflow into adjacent classes. To implement this feature, overflow-handling circuitry is required. For example, a 2-bit counter for each class could keep track of up to three consecutive overflowed classes. Once a class has filled, the counter associated with that class is incremented, and subsequent write requests to that class are directed to an adjacent class, as defined by the counter. Independent of the count, write operations (aside from the ones that result in counter incrementation) all require two clock cycles. Searches require a variable number of clock cycles—if a search to the target class results in a miss, the number of additional classes denoted by the counter also must be searched before a true miss has occurred. This approach provides efficient memory usage at the cost of increased control hardware complexity, but results in nondeterministic throughput, which may not be acceptable for real-time applications.
3) To provide deterministic throughput, allow a single overflow (at most), and always take two cycles for a search—the second will usually be a “no-op,” but will occasionally be required.

We believe option 2) to be appropriate for most applications, and it is used for our circuit design and test chip implementation.

Because we allow overflows into adjacent classes, we introduce a difficult control problem—managing the overflowed data. If classes i and $i-1$ are both full, writes destined for both will end up in class $i+1$. If this situation persists, it may become as likely to find class i or $i-1$ data in class $i+1$ as in their “default” classes. As old class i entries become purged, spaces may be filled with new class $i-1$ data. Unless some means of “garbage collection” is employed, data may become randomly distributed among allowed overflow classes. Such a maintenance task would necessitate intermittent interruptions to move overflowed class i entries back into newly freed class i space from classes $i+1, i+2$ and $i+3$. Optimal solutions to this problem would depend on the application and traffic. No garbage collection is included in the present implementation, but we do provide the capability to clear individual counters during operation.

D. Addressability Options

CAM’s must support reads and writes in addition to searches. There are several means of achieving this:

1) Reads and writes are performed only on search responders, as in [12]. A single responder select line is asserted based on the match result, and the combination of responder select line and class line uniquely select a word. No address decoders are required, and testing is simplified, as is reconfiguring around faults. The disadvantage is that at least one bit of each CAM word must be asynchronously resetable to initialize all entries as invalid on power-up. This reset feature is difficult to implement in a preclassified CAM, since the RAM cells associated with the valid bit must be more complicated than other RAM cells.
2) The read–write port is location addressable. If it is necessary that external control circuitry select the read and write locations, this option must be employed. Initialization does not require special cells, since an external controller can now access individual cells at will, for initializing or any other purpose. Since match results must be encoded, fed back into address decoders, and decoded again, significant throughput degradation occurs on responder-directed operations.
3) Use scheme 1), without resetable cells but with an address decoder that can over-ride the match circuitry when required. Associative throughput is maximized, and external control circuitry may intervene on power-up, or whenever necessary. This option is selected for implementation.

E. CAM/RAM Integration

In virtually all CAM applications, one is interested in more information than simply “is this data there?” The result of the search is typically used to point at more data, which is to be read or written. This additional data is usually stored in a RAM. Since we have implemented our CAM with RAM core cells, it is a straightforward extension of the preclassified CAM to integrate RAM into the same physically contiguous structure.

Fig. 3 is adapted from Fig. 2 to show the additional circuitry required to integrate RAM into the preclassified CAM. The “CAM” core cells from Fig. 2 are shown [cj0,0,0]shaded in Fig. 3. For every column of CAM cells, 2^R-1 [nbcolumns of RAM cells are inserted (in this case, $R=2$). Since CAM cells are physically implemented with RAM cells, the subarray core is still a physically contiguous array of identical cells. The diamonds in the figure represent sense/write access devices that are enabled by the adjacent vertical signal, the *decoded responder select line (DRSL).* The *DRSL* is derived by ANDing the responder select line with R additional address bits, $r0$ and $r1$ in this example. In each subarray, a unique cell is selected by each combination of class and *DRSL*. While a single *DRSL* line is required for every column, a match line is only required for each column of CAM cells, and one XNOR (comparator) gate is required per CAM column per sub-array. The XNOR gate is represented as the shaded portion of the diamond.

A search operation is identical to that of the CAM alone, with the priority encoder selecting a responder for subsequent operations. The subsequent read or write may be performed

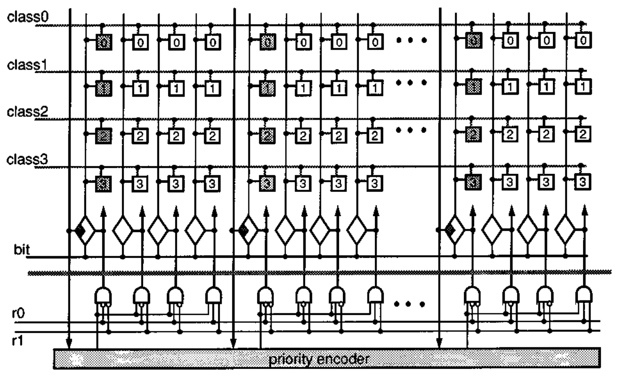


Fig. 3. Integrated CAM/RAM architecture. The circuitry above the horizontal gray line represents a single sub-array; circuits below are shared by all sub-arrays.

on any of the 2^R words belonging to that responder (its *page*), including the CAM word itself, which corresponds to $r = 0$. A series of reads and writes may be carried out on the selected page by changing the r decode lines before performing a new search.

If we introduce the variable $P = 2^R$ to represent the page size in words, the memory has $n \times C$ physical rows and $P \times w/C$ columns, where w is the number of CAM words and the number of pages. The total number of words in the memory is $P \times w$. The designer now has another degree of freedom in choosing configuration parameters to achieve a reasonable aspect ratio.

Using this novel architecture, area overhead due to content addressable circuitry asymptotically approaches zero, and the integrated CAM/RAM should achieve nearly the same chip capacity and operating frequency as state-of-the-art SRAM's.

If the desired CAM/RAM capacity cannot be realized on a single IC, the architecture is easily partitioned across multiple chips for implementation on an MCM (multi-chip module). The PRAM can be placed on a separate chip, and the class (its output) then selects among CAM/RAM chips, as well as among classes on a chip. This saves power because only a single CAM/RAM chip is active during each cycle. For example, a 16 Mb CAM/RAM with $w = 16$ K$, n = 64, P = 4$, and $C = 128$ can be implemented using four CAM/RAM chips, each with $w = 4$ K and $C = 32$. Each chip has $n \times C = 2$ K rows and $P \times w/C = 2$ K columns.

By integrating CAM and RAM together, one can obviously achieve higher densities. In a board-level design, this leads to lower part counts. One could argue that the new memory type could not compete on price with mass-produced commodity RAM's. This is largely offset by the potential price savings on the CAM part—this CAM/RAM may be so much more economical to fabricate than existing CAM's that you get your RAM for free. The integration provides speed advantages, as well. When a CAM and RAM are implemented as separate chips, the matched address must be encoded, driven off the CAM chip, onto the RAM chip, and decoded, leading to significant delay that is completely avoided in the integrated structure.

In custom or semicustom silicon, the new architecture leads to decreased area requirements. In terms of speed and density, it compares well with the traditional approach of butting CAM match line to RAM word line [13]. This traditional approach, of course, suffers from the same capacity limitations as the standard CAM (see Section II-A).

Another notable advantage is the flexibility of the design. It is a simple matter to vary the CAM:RAM ratio from[2] 1:3 to 1:15 or more by inserting or removing XNOR gates and match lines, and this flexibility could be embodied in a module generator. In a reconfigurable implementation, status bits could be altered on-the-fly to enable or disable XNOR's and thus alter the CAM:RAM ratio in real-time. The XNOR gates could even be replaced by other logic functions, perhaps opcode-controlled arbitrary-function look-up tables, as in FPGA's.

IV. Circuit Design

This section presents the most important circuit design features for the integrated CAM/RAM. We first look at the control circuitry and pipeline, followed by a description of the two most important critical-path circuits: the XNOR and match line pull-down, and the multiple-match resolver. Simulations confirm that these circuits are applicable to megabit-capacity memories.

A. Control Circuitry and Pipelined Operation

Since the CAM/RAM contents are partitioned into classes, and *a priori* knowledge of precise data distribution among classes is impossible to obtain, some classes will fill before the memory, as a whole, is full. For improved memory efficiency, the control circuitry manages overflows into adjacent classes, providing 2-bit count storage for each class and allowing up to three adja[chcent overflows. Once a class has filled, subsequent write requests to that class are directed to an adjacent class, as defined by the counter. If a search to the target class results in a miss, the number of additional classes denoted by the counter also must be searched before a true miss has occurred.

This leads to efficient memory usage, but results in rather complex timing requirements, as well as nondeterministic throughput. All operations (search, read, and write) are essentially instructions executed by the control circuitry, and they require a number of clock cycles that varies depending on the count. Operation is timed by two clocks, as shown in Fig. 4(a). The following constraints must be satisfied:

1) For maximum speed, the PRAM and CAM/RAM memory operations should occur in parallel.
2) The control circuitry must take the class that is read from the PRAM in cycle j and add the appropriate counter value before applying it to the CAM/RAM in cycle $j+1$.
3) If a class has a nonzero count, multiple clock cycles may be required to perform a search. CAM/RAM search results from cycle j, together with the counter value,

[2]Ratios of 1:0 or 1:1 are achievable by the architecture, but they present severe layout constraints, because the narrow pitch does not allow sufficient room for vertical signal routing or for the large bipolar transistor used for match line pull-down. 1:3 to 1:15 or more by inserting or removing XNOR gates and match lines, and this flexibility could be embodied in a module generator. In a reconfigurable implementation, status bits could be altered on-the-fly to enable or disable XNOR's and thus alter the CAM:RAM ratio in real-time. The XNOR gates could even be replaced by other logic functions, perhaps opcode-controlled arbitrary-function look-up tables, as in FPGA's.

determine whether another search will be performed in cycle $j+1$. Control circuits thus must process cycle j search results near the end of cycle j.

The constraints can be simultaneously satisfied by pipelined operation, requiring the pipeline registers shown in Fig. 4(b). Data associated with a search must be applied to the PRAM one cycle before it is applied to the CAM/RAM. Thus, a `ck0` falling-edge register holds the PRAM output, ensuring a stable input for the count processing circuitry. Registers clocked by the same edge are also included for addresses and data passing directly to the CAM/RAM, so that they may be applied at the same time as the PRAM inputs (during `ck0` low). A `ck0` rising-edge register holds the output of the count processing (the class input to the CAM/RAM). Because of the pipeline, the CAM/RAM will be completing a previous search, read or write at the same time that the PRAM is reading a new class for a new search, and after new CAM/RAM data has been applied to the pipeline input. This new data does not interfere with the data being searched for or written in the CAM/RAM. Fig. 4(a) shows a read followed by a search to demonstrate the concurrence of old and new operations.

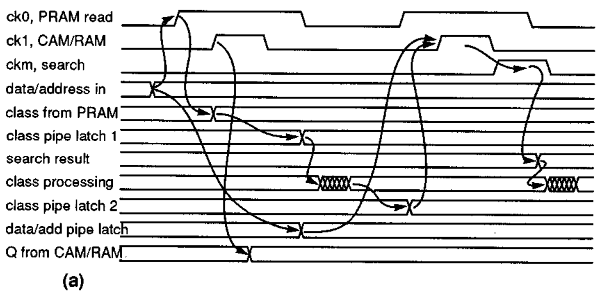
This scheme allows maximum throughput. For a search operation that steps through o class overflows, followed by i responder-directed reads or writes, $o+i+1$ cycles are required. For some sequences of operations, however, "peak" throughput is not sustainable, because the controlling processor must wait for search results before applying the next set of inputs to the memory.

B. XNOR and Match Line Pull-Down

During a read operation, data is passed through the column selection circuits in the diamonds of Fig. 3, toward the memory periphery, where the sensing can be performed by a single sense amplifier per memory I/O bit. Each CAM column in a block, however, needs its own dedicated sense amplifier for search operations. In both cases, the differential latch sense amplifier of [14] is used. The CAM sense amplifier outputs (`sao` and [`saon`) drive a dynamic differential XNOR gate, which is shown in Fig. 5 along with the BiCMOS match line pull-down circuit.

A full-rail match line signal transition is desirable, since it leads to faster priority encoder operation and a simpler design. The match line pull-down circuit should have small load sensitivity, and low delay for loads typical in a megabit configuration: 1024 rows and $n = 32$. For typical sub-half-micron multi-megabit SRAM technologies, such results are achievable using MOS devices. In the 0.8 μm technology available to us, a 5X NPN device (20×0.8 μm emitter) provides the required performance,[3] as shown in Fig. 6. The

[3] In fact, BiCMOS buffers are employed extensively in the memory periphery for such functions as clock, address, and class line drivers. The design is based on the modular SRAM described in [14]., as shown in Fig. 6. The worst-case scenario was simulated–a single-bit mismatch, requiring a single circuit to pull down the match line. The right-most curve, corresponding to a load of 1024 memory rows plus 32 pull-downs (31 of them inactive), confirms operation of this circuit for megabit capacities. The load sensitivity is 240 ps/pF. For this load, match line power dissipation is 8.0 mW per line at 40 MHz, or 1.0 W for a typical configuration. A 160 μm PMOS device is used for match line precharge.



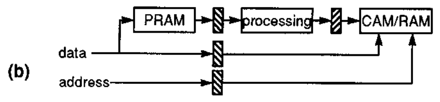


Fig. 4. (a) Pipelined operation, read followed by a new search. (b) Location of pipeline registers.

worst-case scenario was simulated—a single-bit mismatch, requiring a single circuit to pull down the match line. The right-most curve, corresponding to a load of 1024 memory rows plus 32 pull-downs (31 of them inactive), confirms operation of this circuit for megabit capacities. The load sensitivity is 240 ps/pF. For this load, match line power dissipation is 8.0 mW per line at 40 MHz, or 1.0 W for a typical configuration. A 160 μm PMOS device is used for match line precharge.

The horizontal "bit line" in Fig. 3 is actually a conceptual representation of three differential signals: read data, write data, and reference data. The reference data signals (`ref/refn`) are applied to the XNOR gates as diffusion inputs (this is shown in Fig. 5). In the case of a mismatch ($\texttt{ref} \neq \texttt{sao}$), one of these inputs—driven by a 5X push-pull BiCMOS buffer (not shown)—is required to supply current to the pull-down circuit through two series PMOS transistors. In the worst case, all XNOR gates connected to the `ref/refn` signals (numbering up to 128 in a typical megabit configuration) have a mismatch of the same polarity. To limit the dynamic load on these nodes, two extra stages of buffering are used in the pull-down gate, as shown in the box of Fig. 5. The effect of this extra buffering is demonstrated in Fig. 7, where the `ref/refn` signals drive 128 XNOR gates. In Fig. 7(a), only a single data mismatch occurs, leading to minimal dynamic loading; inclusion of the inverters results in slightly slower operation. However, as shown in Fig. 7(b), match line pulldown is not successfully achieved without the extra stages in the case of 128 mismatches of the same polarity.

Area requirements for the bit line precharge and *DRSL*-column access circuitry (the diamonds in Fig. 3) are equivalent to 4.4 memory rows per I/O bit. The CAM sense amplifier, XNOR, and match line pull-down occupy 4.1 rows when straddling four columns. Fewer rows are required for larger page sizes, since the layout can spread in the column dimension. The minimum allowable page size is four words (corresponding to $R = 2$).

C. Multiple-Match Resolution

The multiple-match resolver (MMR) is the critical-path portion of the priority encoder. It selects from among multiple

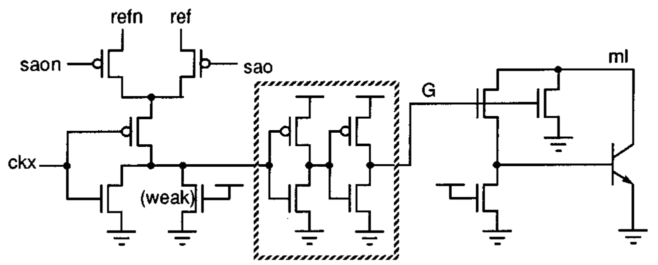


Fig. 5. Shared XNOR gate and BiCMOS match line pull-down. The extra buffering stages, enclosed by the dashed box, are explained in the text.

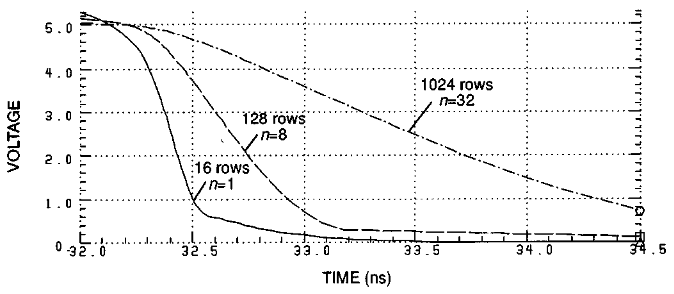


Fig. 6. Simulated match line pull-down waveforms for various loads.

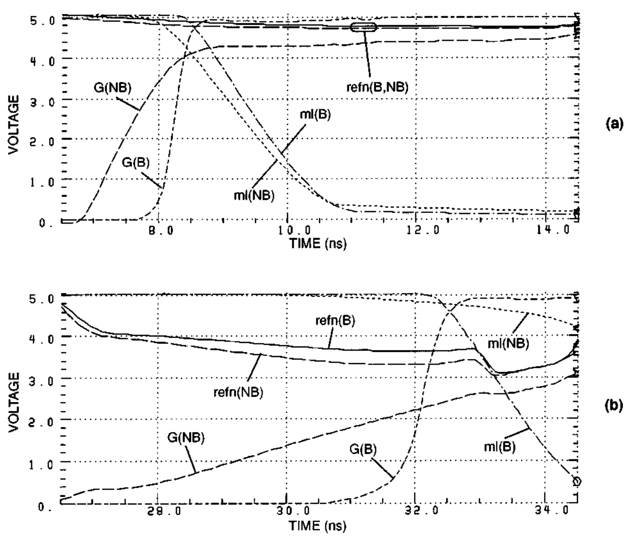


Fig. 7. Simulated XNOR performance, both with (B) and without (NB) the extra buffer stages outlined in Fig. 5: (a) data mismatch in 1 of 128 gates, (b) data mismatch in all 128 gates. Node names are the same as those used in Fig. 5.

matches, giving priority to matches based on their physical location. A large variety of MMR designs have been proposed in the literature, but most of these are unsuitable to the large fan-ins that must be accommodated in megabit-capacity CAM's (we choose 128 inputs for our benchmark requirement). The delay of the MMR must increase sublinearly with fan-in, so any technique relying on bit-wise signal propagation or long pass-transistor networks is unacceptable. We employ a hierarchical method similar to a carry-look-ahead adder. A 16-input MMR—an illustrative example, which corresponds to our test chip implementation—is shown in Fig. 8. Fig. 8(a) demonstrates the hierarchical connection of four four-input sections.

The lowest-level gate is shown in Fig. 8(b), with four inputs. The circuit efficiently shares transistors between outputs and is derived from the NOR logic in [5]. We use NAND logic instead, with inverted match line inputs, resulting in an NMOS series path. This provides higher speed and allows up to eight inputs to this lowest-level gate. Single-stage inverter "sense amplifiers" are used on the full-rail match lines for simplicity and high speed.

Our benchmark 128-input MMR uses an eight-input lowest-level gate. The second stage, like the one shown in Fig. 8(a), connects four gates, in this case accommodating 32 inputs. A third level of hierarchy is then employed, analogously combining four 32-input sections to give a total fan-in of 128. The critical path of such a 128-input MMR consists of only six gate delays.

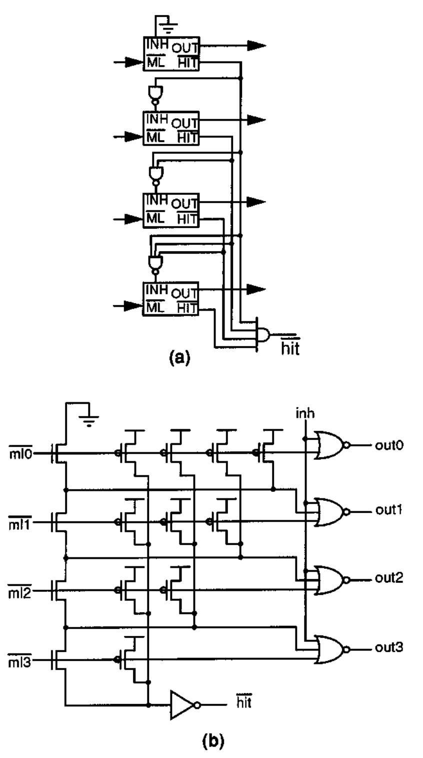


Fig. 8. Multiple-match resolver: (a) four 4-bit sections hierarchically connected and (b) 4-bit section.

ROM-like circuitry is used to derive the address output of the priority encoder from the MMR outputs.

V. Megabit Simulation Results

As described in Section IV, the memory design is scalable up to megabit capacities. It has been successfully simulated with a netlist corresponding to a 1 Mb configuration. The configuration has 32 K words $\times$ 32 bits and is comprised of $C = 32$ classes of 128 entries each, and a page size of eight words $(R = 3)$. We propose a layout wherein I/O circuitry, class decoders, and `ref/refn` buffers are located in a central spine, with 512 columns on either side. Maximum loading conditions roughly correspond to those analyzed in Section IV-B of this paper—the match lines span 1024 rows and 32 pull-down circuits, but the `ref/refn` drivers are loaded by only 64 XNOR gates. The 128-input MMR has three levels of hierarchy: an eight-input lowest level, and two higher levels with four inputs each, as described in Section IV-C.

Waveforms in Fig. 9 show a successful search followed by a read at 37 MHz. This verifies the critical path from the match lines, through the MMR and class processing circuitry, and back to the CAM/RAM class line decoders. This speed compares favorably to the previously-implemented large-capacity CAM's that we have referenced: 33 MHz for 160 kb [10], 6.7 MHz for 288 kb [5], and 8.3 MHz clock and 16 kHz search for 4 Mb [4]. If implemented, this memory would be the largest-capacity fully-parallel associative memory ever reported; it would not be the largest CAM, since only 128 kb are CAM storage.

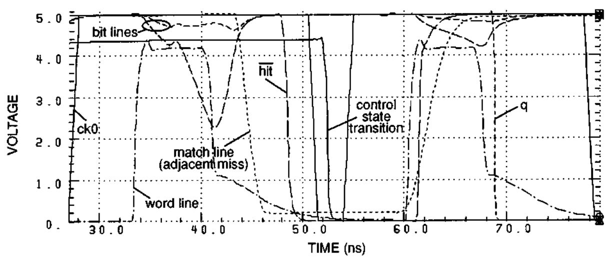


Fig. 9. Simulation of a 1 Mb CAM/RAM at 37 MHz, illustrating a successful search followed by a read.

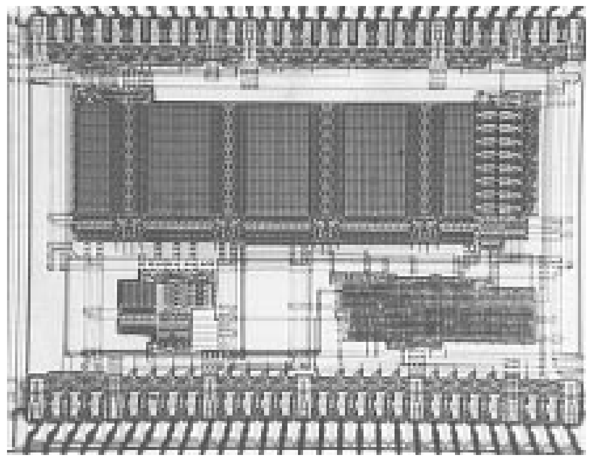


Fig. 10. 8 kb CAM/RAM test chip die photo.

We believe that the same circuit design, if implemented in state-of-the-art SRAM fabrication technology, would result in multimegabit capacities and similar speeds. A BiCMOS technology would likely not be required.

VI. Test Chip Implementation

An 8 kb test chip was implemented in $0.8\ \mu\mathrm{m}$ BiCMOS technology [6] to verify the circuit design. A die photo is shown in Fig. 10. The CAM/RAM occupies the top portion of the chip. The 64 $\times$ 4 PRAM is at the bottom left, and the random-logic control circuitry is at the bottom right.

A. Chip Design

The 8 kb CAM/RAM has a 1 K word $\times$ 8 bit configuration. There are $C = 16$ classes with 16 entries each, and a page size of four words $(R = 2)$. The resulting physical dimensions are 128 rows $\times$ 64 columns (3.1 mm $\times$ 1.1 mm including peripheral circuitry). Note that the "bottom" of the CAM/RAM is at the right of the photo. Fig. 11 clarifies the location of the various functions in the CAM/RAM layout. The SRAM core cell size is $131.3\ \mu\mathrm{m}^2$ $(13.4\ \mu\mathrm{m} \times 9.8\ \mu\mathrm{m})$. Operation is in one of two modes: in "mission mode," reads and writes are performed on search responders, while in "over-ride mode," access is specified by an externally supplied address. Over-ride mode is used to initialize the contents of the CAM/RAM to an all-invalid state at the beginning of each test.

The preclassification task is performed by a 64 $\times$ 4 PRAM. The six LSB's of the input data are used to read the PRAM and thus select one of 16 classes. Counters and control circuitry

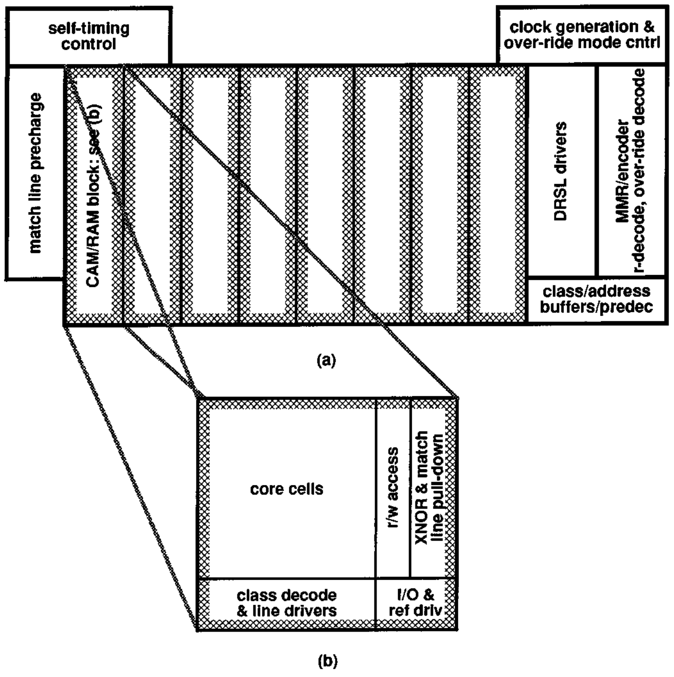

self-timing control
clock generation & over-ride mode cntrl
match line precharge
CAM/RAM block: see (b)
DRSL drivers
MMR/encoder r-decode, over-ride decode
class/address buffers/predec
(a)
core cells
r/w access
XNOR & match line pull-down
class decode & line drivers
I/O & ref driv
(b)

Fig. 11. 8 kb CAM/RAM layout floorplan: (a) The entire memory, as shown at the top of Fig. 10. (b) Details of a single block.

keep track of overflows into a maximum of three adjacent classes. The control circuitry, comprising approximately 1 K equivalent gates, was implemented using standard cells.

The chip contains 85 423 transistors, and the die size is 3 mm $\times$ 4 mm, including pads and frame.

Since the RAM arrays are the densest and most structurally complicated parts of the chip, we expect the majority of fabrication faults to occur there. Standard RAM test algorithms can be used to test the CAM/RAM, because the address and data inputs are directly controllable from chip pins (using over-ride mode), and the data outputs are directly observable. The desired class must be loaded through a register, and thus the entire CAM/RAM can be tested as a series of 16 64 $\times$ 8 RAM's (one per class). This is a valuable test mechanism, even for production versions of chips using this architecture. The PRAM can also be controlled and observed from the chip pins. It is advisable to screen the memories in over-ride mode to pare down the number of dice before commencing functional testing.

This test strategy can also be used to implement yield enhancement through graceful degradation. If an extra bit of CAM storage is added per page (the *faulty* bit), it can be used together with over-ride mode testing to detect and identify faulty pages. If any faults are detected in a given page, that page's *faulty* bit is set. That CAM entry will then fail to match any write or search requests, and the entire page in question is effectively ignored. This results in a decrease in usable chip capacity, rather than a bad part.

The CAM/RAM data output (`q`) and the priority encoder address output (`aout`) are chip outputs, along with the `hit` signal and the `full` and `done` status signals from the control circuitry. These signals indicate the following when asserted:

1) `hit`: the search operation of the present cycle has resulted in a match.
2) `done`: a (possibly)-multicycle search has completed.
3) `full`: a (possibly)-multicycle search has completed unsuccessfully.

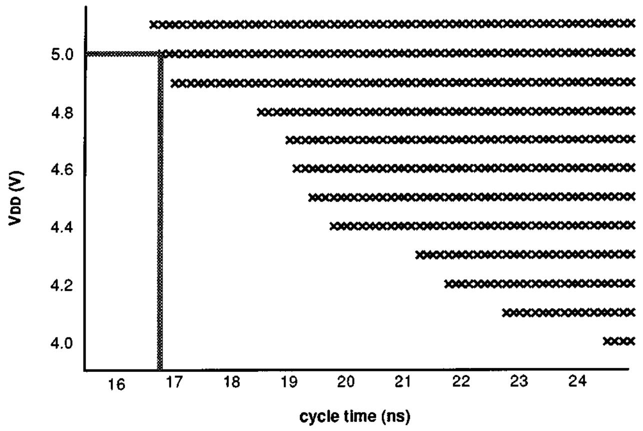


Fig. 12. Shmoo plot of cycle time versus power supply voltage.

Multiplexed output pins enable observation of many internal buses, including the control state, the latched input data, the PRAM output, and the class input to the CAM/RAM.

B. Test Results

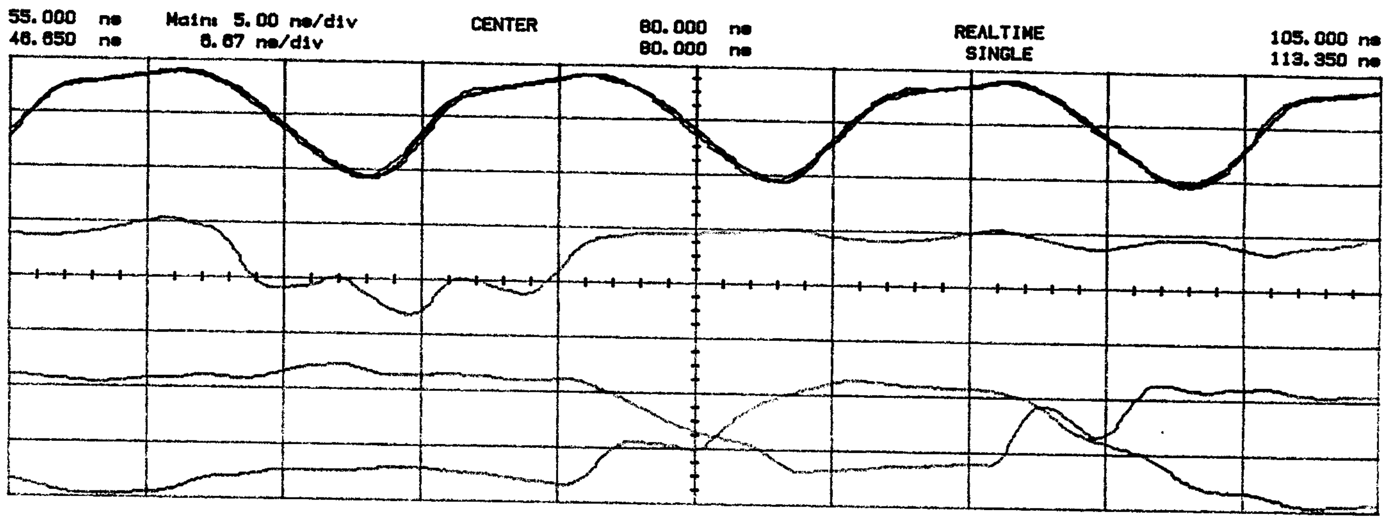
The chip was tested and found fully functional after microsurgery to correct a minor layout error in top-level metal. A shmoo plot of cycle time versus supply voltage is shown in Fig. 12. Maximum clock frequency is 59.7 MHz at room temperature and $V_{DD} = 5$ V, corresponding to a cycle time of 16.75 ns.[4] As the maximum speed of our tester was 60 MHz, we were not able to characterize the maximum operating frequencies of the two memories in RAM-mode at $V_{DD} = 5$ V. However, we could determine that the PRAM functioned at 60 MHz down to $V_{DD} = 3.5$ V, and the CAM/RAM (in RAM-mode) functioned at 60 MHz down to $V_{DD} = 4.6$ V.

Fig. 13 shows an oscilloscope trace at 50 MHz. Three clock cycles of a successful search/read operation are shown. The hit signal, asserted low after the search cycle, leads to two read cycles, corresponding to $r = 0$ and $r = 1$, respectively.

Dynamic power dissipation was found to be 5.1 mW/MHz. Total average power dissipation is 260 mW at 50 MHz. Chip characteristics are summarized in Table I.

The 59 MHz clock frequency is somewhat slower than the 167 MHz of the butted CAM/RAM described in [13] and numerous other high-speed CAM's, but it is considerably faster than typical commodity parts (the 64 kb "LANCAM" described in [15], for example, has a top speed of 11 MHz).

[4]This is peak throughput. Using our test bench, status signals were not observable at the tester until a short time into the following clock cycle, at this high clock rate. Thus, it is not clear whether a controlling processor in close physical proximity to the CAM/RAM would receive status information soon enough to initiate another operation without inserting "no-op" cycles (these are in addition to those required due to the pipeline as explained in Section IV-A). Since we were able to anticipate correct output vectors, this phenomenon did not prevent us from achieving true peak performance in the test environment.



```
          Sensitivity   Offset       Probe   Coupling   Impedance
Channel 1   2.50  V/div  -4.50000  V   10:1      dc        1M ohm
Channel 2   2.50  V/div   8.60000  V   10:1      dc        1M ohm

  Trigger Mode: Edge
On the Negative Edge of Channel1
Trigger Level(s)
  Channel1 = 2.50000  V (noise reject OFF)
HoldOff = 40.000  ns
```

Fig. 13. Oscilloscope trace showing a successful search followed by two reads at 50 MHz. The vertical scale is 2.5 V/div and the horizontal scale is 6.67 ns/div. The signals, from the top down, are `ck0`, `hitn` and `q` (two bits shown).

TABLE I
8 kb CAM/RAM Chip Characteristics

CAM/RAM organization	$1K \times 8$
PRAM organization	64×4
Total RAM bits	8,448
Process	0.8 µm BiCMOS
Core cell size	13.4 µm × 9.8 µm (131 µm^2)
Chip size	3 mm × 4 mm (12 mm^2)
Total number of transistors	85,423
Maximum clock frequency	59.7 MHz @ VDD = 5 V, 25 °C
Power dissipation	260 mW @ 50 MHz, 5 V

VII. Conclusions

Architectural and circuit innovations are required to implement CAM's with RAM-like densities and capacities. We have selected preclassification as the best means of implementing high-density, high-speed, high-capacity CAM's. This technique allows two-dimensional decoding and comparator sharing, without throughput degradation. Among numerous design options, we chose to implement: 1) a preclassification RAM (PRAM) to perform the preclassification task, 2) counter-based class overflow handling for improved memory efficiency, and 3) associative (responder-directed) read/write access with location-addressable over-ride capability.

The preclassified CAM core is composed of RAM cells. Thus, the target RAM may be integrated together with the CAM in a homogeneous physical array, resulting in high-speed associative read/write operations and single-chip megabit capacity. Two critical-path circuits, the dynamic XNOR match line pull-down and the multiple-match resolver, were developed for this architecture. A "RAM-mode" test feature can be used in conjunction with known RAM test algorithms to provide graceful degradation in the case of faulty RAM core cells or periphery.

The novel architecture and associated circuit design result in 37 MHz performance for a 1 Mb configuration (simulated) and 59 MHz performance for an 8 kb test chip (measured) in 0.8 μm BiCMOS technology. We project similar speeds and larger capacities for state-of-the-art SRAM fabrication technology.

Acknowledgment

The authors gratefully acknowledge design and fabrication support provided by the Telecom Microelectronics Centre of Northern Telecom Ltd. Thanks, specifically, to D. Somppi, R. Gibson, and S. Komachuk for their assistance and comments. Invaluable help in testing the chip was provided by R. Gibbins, J. Pristupa, M. Phan, and R. Collins.

References

[1] L. Chisvin and J. R. Duckworth, "Content-addressable and associative memory: alternatives to the ubiquitous RAM," *IEEE Computer*, vol. 22, pp. 51–64, July 1989.
[2] T. Kohonen, *Content Addressable Memories*, 2nd ed. Berlin: Springer Verlag, 1987.
[3] K. J. Schultz, "CAM-based circuits for ATM switching networks," Ph.D. dissertation, University of Toronto, 1996.
[4] G. J. Lipovski, "A four megabit dynamic systolic associative memory chip," *J. VLSI Signal Processing*, vol. 4, no. 1, pp. 37–51, Feb. 1992.
[5] 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*, vol. 27, pp. 1927–1933, Dec. 1992.

[6] R. Hadaway, P. Kempf, P. Schvan, M. Rowlandson, V. Ho, J. Kolk, B. Tait, D. Sutherland, G. Jolly, and I. Emesh, "A sub-micron BiCMOS technology for telecommunications," in *Proc. European Solid-State Device Research Conf.*, 1991, pp. 513–516.

[7] K. J. Schultz and P. G. Gulak, "Architectures for large-capacity CAM's," *INTEGRATION: the VLSI Journal*, vol. 18, pp. 151–172, June 1995.

[8] S. Panchanathan and M. Goldberg, "Vector-centered CAM architecture for image coding using vector quantization," *Visual Communications and Image Processing IV*, SPIE vol. 1199, pp. 1084–1094, 1989.

[9] I. N. Robinson, "Pattern-addressable memory," *IEEE Micro*, vol. 12, pp. 20–30, June 1992.

[10] 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*, vol. 25, pp. 1158–1165, Oct. 1990.

[11] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*. San Meteo, CA: Morgan Kaufmann, 1990.

[12] A. J. McAuley and C. J. Cotton, "A self-testing and reconfigurable CAM," *IEEE J. Solid-State Circuits*, vol. 26, pp. 257–261, Mar. 1991.

[13] R. A. Heald and J. C. Holst, "6ns Cycle 256kb cache memory and memory management unit," *ISSCC Dig. Tech. Papers*, pp. 88–89, 1993.

[14] 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-μm BiCMOS modular memory family of DRAM and multiport SRAM," *IEEE J. Solid-State Circuits*, vol. 28, pp. 222–232, Mar. 1993.

[15] MUSIC Semiconductor, *MU9C1480 LANCAM Preliminary Specification*, May 1993.

Kenneth J. Schultz (S'85–M'96) 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 (BNR), 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 was 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. He is a licensed Professional Engineer.

P. Glenn Gulak (S'82–M'83) received the Ph.D. degree 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 was on a research leave at Telecom Paris in the 1994–95 academic year. He is a 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 served on the ISSCC Signal Processing Technical Subcommittee from 1990 to 1994 and currently serves as Program Secretary for ISSCC.