A Special-Purpose Content Addressable Memory Chip for Real-Time Image Processing

Yong-Chul Shin, *Student Member, IEEE,* Ramalingam Sridhar, *Member, IEEE,* Victor Demjanenko, *Member, IEEE,* Paul W. Palumbo, *Member, IEEE,* and Sargur N. Srihari, *Senior Member, IEEE*

Abstract—**An ASIC which implements a special-purpose content addressable memory that performs parallel search and multiple update (PSMU) operation is presented. This chip, referred to as MUCAM, can search 256 8-b-wide locations in parallel for target data and update all such locations with new data within 50 ns. MUCAM has been developed for image component labeling and merging operation in a connected component analyzer. It was fabricated using 0.9-μm CMOS technology.**

I. Introduction

In image processing, connected component analysis is an indispensable step towards finding a region of interest [1]. This involves locating connected components and identifying various information associated with them. A real-time address block location (RT-ABL) system has been developed at the Center of Excellence for Document Analysis and Recognition (CEDAR), at the State University of New York at Buffalo [2]. This system processes a stream of mail piece images and extracts the destination address block. This has been achieved using a pipelined architecture, with a set of image processing tools implemented using off-the-shelf and custom prototype hardware boards that use the Maxbus technology of Datacube Inc. [3], and block analysis tools implemented in software on a multiprocessor platform.

The connected component analyzer [2] identifies components that are eight-way connected. The eight-way connectivity analysis is a grouping of pixels which are adjacent in a diagonal direction as well as in a vertical or horizontal direction. The analysis is performed by a unique single-pass algorithm implemented in hardware. To achieve the throughput required of the RT-ABL system, this task must be performed in real time. The analysis involves identification of the connected components, labeling each with a unique index number in sequence, and the extraction of the coordinates of the minimum enclosing box and the total number of foreground pixels. At any point, many components that were assigned different indices earlier may have to be merged into one, due to a newfound connectivity. Thus, an arbitrary number of labels may have to be updated to a new label in real time. This process can cause a significant bottleneck without a special type of memory that can handle parallel search and multiple update operations. This requires a register array that implements parallel search capability to locate target data and update all these locations with a new index number in a single clock cycle.

Conventionally, a parallel search has been implemented using a content addressable memory (CAM) [4], [5]. A number of different CAM architectures have been developed for a variety of uses, such as computer graphics and image processing [6], [7], numerical and nonnumerical computations [8]–[10], logical search operations [11], artificial intelligence applications [12], and symbolic computation using set operations [13]. A general-purpose CAM architecture has also been developed [14]. Advanced Micro Devices has two CAM chips available in the market [15], [16]. Also recently, a high-density CAM has been implemented for a dictionary search processor [17]. In some of the aforementioned implementations, multiple update operation can be performed by first executing a parallel search operation, which sets match registers, and then updating the matched locations. None of them, however, implements a parallel search and multiple update (PSMU) in a single clock cycle, which is necessary for the real-time connected component analysis.

This paper first describes the implementation of a register array as an ASIC, called a multiple-update content addressable memory (MUCAM). MUCAM is a 4560-μm $\times$ 7120-μm chip, packaged in a 96-pin PGA and fabricated using full-custom 0.9-μm CMOS technology. It has the capability to read, write, or perform parallel search and multiple update for up to 256 8-b locations in less than 50 ns. The chip has a 256 $\times$ 8-b register array and a 256 $\times$ 8-b comparator array. It has been laid out, fully simulated, fabricated, and tested. VLSI implementation details, performance, and the test result of the chip are presented in the paper. MUCAM can also be used in many other applications, which require a fast updatable register array. A modified memory cell based version of the MUCAM has also been designed and is presented.

Manuscript received October 7, 1991; revised January 14, 1992. This work was supported in part by the Office of Advanced Technology of the United States Postal Service under USPS Contract 104230-88D-2267.

Y.-C. Shin, R. Sridhar, and V. Demjanenko are with the Department of Electrical and Computer Engineering and the Center of Excellence for Document Analysis and Recognition, State University of New York, Buffalo, NY 14260.

P. W. Palumbo and S. N. Srihari are with the Department of Computer Science and the Center of Excellence for Document Analysis and Recognition, State University of New York, Buffalo, NY 14260.

IEEE Log Number 9107223.

II. Connected Component Labeling

In image processing applications, one of the basic concepts used for identifying and grouping components of interest is called a connected component analysis [1]. The method used in the real-time address block location system requires identification of eight-way connected foreground pixels in a binary image that are connected together forming a single component. Many existing algorithms for this operation require two passes and are not quite suitable for real-time low-latency applications. It was necessary to find a single-pass algorithm that could be implemented to fit in a pipelined architecture, allowing a stream of pixels to be processed in less than 100 ns/pixel. The method used is explained in this section.

In the real-time one-pass implementation of the connected component analyzer, it is not trivial to change the labels of all the previous pixels when they all have to be merged into a single label. The MUCAM is used as an *equivalence table* for this purpose. Using this table, one can easily update an arbitrary number of labels and retrieve the status for each component.

Fig. 1 can be used to illustrate the functionality of the MUCAM chip. The input image is scanned from left to right, and from top to bottom. As the image is scanned, at (3, 3), a foreground pixel is encountered, which will be assigned a new component index number, 1. The coordinates (minimum x, y and maximum x, y) and the size of the component (initially 1) are registered to keep track of its locational and physical information. Next, an index number, 2, is assigned to the pixel (5, 3), because of the discontinuity at (4, 3).

For the pixel that is adjacent to another pixel with a component index number already assigned, the same index number is reused, according to the eight-way connectivity between the current pixel and the previous one or those in the previous scan line. Therefore, the same index number, 2, is used for the foreground pixels from (6, 3) to (17, 3). At the same time, locational and physical information of component 2 needs to be updated as the component grows. In the next scan line, component index numbers for pixel (3, 4), (5, 4), and (17, 4) are selected from the component index numbers of previous scan line, based on the connectivity. Similarly, new index numbers 3, 4, 5, and 6 are assigned to the pixels at (7, 5), (9, 5), (13, 7), and (11, 8), respectively.

At (12, 10), the component index numbers 6 and 5 are merged together, because of the newly found connectivity. When two indices are merged together, the smaller index is preserved, and the larger index is discarded and released for future use. Hence, information associated with the component index 6 needs to be reflected on that of component index 5. Also, the merge of two components is stored in an equivalence table, so that the equivalence of the two can be retrieved for future reference. Whenever a pixel with index 6 is encountered later, locational and physical information is updated only for the component index 5 (because of the recent merge of 6 with 5). Hence, only minimal effort is required to keep track of component information. For instance, the pixel (11, 11) is assigned an index of 5, even though it is adjacent to a pixel with index 6 at (11, 10).

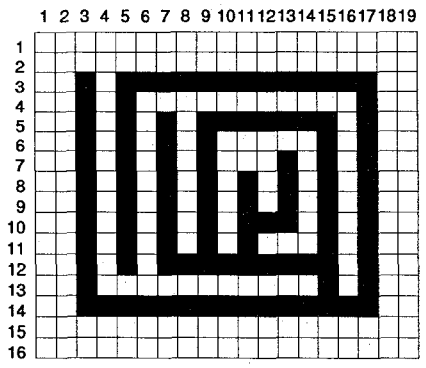


Fig. 1. Example binary image.

At (8, 12), component index 4 is merged with component index 3. Another merge occurs at (10, 12) between component indices 3 and 5. This time, the merge operation affects both component indices 5 and 6, since 6 already contains component index 5 as a result of a previous merge. These relationships are stored in the equivalence table. So far, the component indices 4, 5, and 6 are all equivalent to the component index 3.

At (14, 14), component index 3 is merged into index 1. Now, in the equivalence table, component indices 3–6 are to be set as equivalent to index 1. Finally, at (16, 14), component index 2 is merged into index 1, leading to the detection of labeling of a single connected component, and the equivalence table shows that component indices 2–6 are equivalent to component index 1.

III. MUCAM Operations

To meet the requirements specified in Section II, the MUCAM needs the following operations:

- parallel search of 256 8-b memory cells, and multiple updates within one clock cycle (PSMU operation);
- WRITE operation in one clock cycle;
- READ operation in one clock cycle;
- clash indications.

where a clock cycle is 50 ns.

A. PSMU Operation

In this unique operation, MUCAM searches all 256 cells and finds multiple locations that have compare data. If a match is found, then the cells containing the compare data will be updated with new input data. This operation needs to be performed in a single clock cycle. Two buses are used for this operation, a compare data bus and an input data bus.

This operation can be explained using C program constructs as shown below. *MUCAM[x]* is the content of MUCAM at address x.

```
for (address = 0; address < = 255; address+ +) {
   if (MUCAM[address] = = target_data) {
      MUCAM[address] = new_data;
   }
}
```

For various applications, the first and/or the last location may contain special value which may not change with the PSMU operation. In our application, MUCAM is used to store an equivalence table, where the nth location contains the equivalence of the label n. In other words, the content j at the location i means that the label i is equivalent to the label j. For example, label 255 could be used as a background label, and hence, the last location may never change its value. For this purpose, the updates of the first and the last location are controlled by two external signals, *FCNT* and *LCNT*, respectively.

B. WRITE Operation

This is a conventional WRITE operation. It uses a dedicated write address bus and an input data bus. This operation performs a single WRITE per clock cycle. An optional vector WRITE implementation that allows parallel data I/O [9] can be added if required.

C. READ Operation

This is a conventional READ operation. The READ operation is independent of the WRITE or the PSMU operations. Again, two buses are associated with this operation, an output data bus and a read address bus. With separate input and output data buses along with separate read and write address buses, cells can be selectively written or read like a dual-port random access memory.

D. Auxiliary Functions

There are two auxiliary functions implemented in MUCAM. One is a match output similar to that of a conventional CAM. If there is at least one location that has the same data as the compare data, the match signal is asserted. The other is a critical condition indication which is used for potential data clash. There are two such clash signals. One of them (*RAWACL*) is asserted when the read address and the write address are identical. This indicates that the output data may not be correct because of the possible writing to the same location. The other clash signal is asserted when the compare data and the output data are identical (*DCDOCL*). If the output data are the same as the compare data, then the output data may be in transition due to a PSMU operation.

IV. MUCAM Architecture

MUCAM consists of 256 identical cells, address decoders, input/output buffers, and multiplexers. Fig. 2 shows the functional block diagram of MUCAM. *WA* is the write address bus, *RA* is the read address bus, *DI* is the input data bus, and *DC* is the compare data bus. *WE*, *UE*, *CE*, and *OE* are write enable, update enable, chip enable, and output enable, respectively. *M* is a match output and it is also available as a latched output, *MQ*, with a separate clock *MC*. *RAWACL* and *DCDOCL* are clash indicator output signals. *DO* is the output bus. All the address and data buses are 8-b wide.

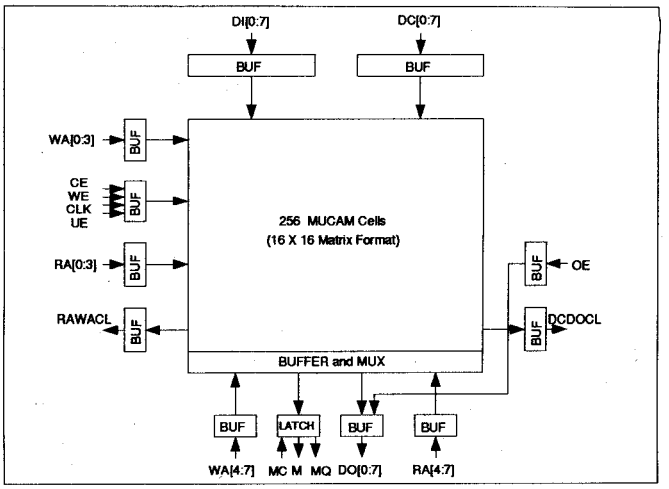


Fig. 2. Functional block diagram of MUCAM.

Two hundred and fifty six cells were laid out as a 16 × 16 matrix. Two successive columns (each column containing 16 cells) are mirror images of each other, thus sharing one local bus. Fast two-dimensional decoding is used for the *WA* and the *RA*. Clock signal and various enable signals are implemented as feedthrough lines.

Each cell contains a decoding logic, an 8-b register, an 8-b comparator, an 8-b tri-state buffer, and a clocking circuitry as shown in the schematic diagram of a cell in Fig. 3. The operations of the cell are summarized in Table I. *CWADX* and *CWADY* are outputs of the write address x and y decoders, respectively. Similarly, *CRADX* and *CRADY* are read address decoder outputs. *CWEN* and *CUEN* are clocked write enable and update enable signals. The output of the comparator is cascaded using the two-input OR gate through a column to generate a match output signal. The *FLCNT* line is represented as *FCNT* and *LCNT* at the first cell and the last cell, respectively.

V. Simulation and Test Results

Fig. 4 shows the timing diagram obtained from simulation. In the figure, during the first eight clock cycles, the locations from 0 to 7 are initialized with 55 (hex). During the next clock cycle, parallel search and multiple update operation is performed. Here, all the locations containing 55 (hex) are updated with 03 (hex). During the next eight clock cycles, the contents of MUCAM are read. The rest of the timing diagram shows similar results for different data. The simulation verified the correctness of the functionality and the timing.

The chip has been designed, fabricated, and tested. Bit-error test, stuck-at-fault test, and power failure test were performed. Timing tests were carried out based on the connected component analyzer timing specification using an active-low clock with 50-ns cycle time and 50% duty

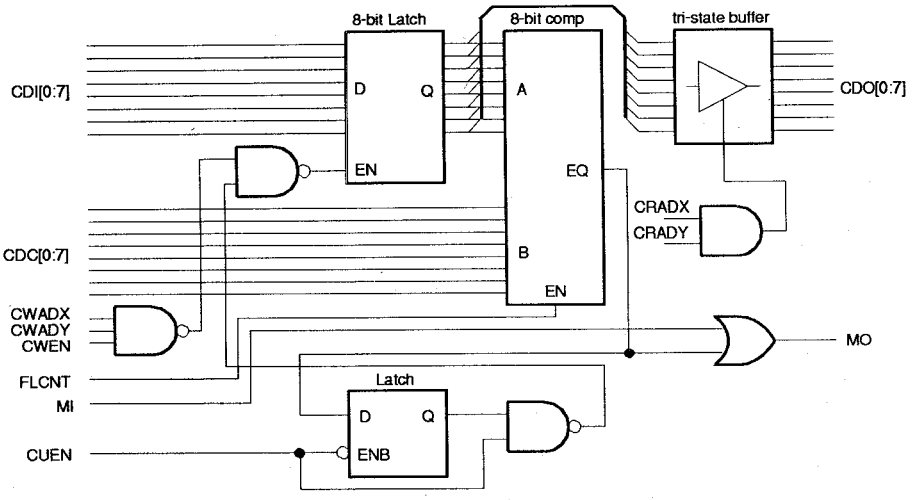


Fig. 3. Schematic diagram of a cell.

TABLE I
WRITE AND PSMU OPERATIONS OF A CELL

Operations	*CDI*	*CDC*	*CUEN*	*CWEN*	*CWADX*, *CWADY*	*CRADX*, *CRADY*	*EQ* of 8-b Comp	8-b Latch
WRITE	input data	X	0	⊓	11	XX	X	input data
PSMU	input data	compare data	⊓	0	XX	XX	0	no change
	input data	compare data	⊓	0	XX	XX	1	input data

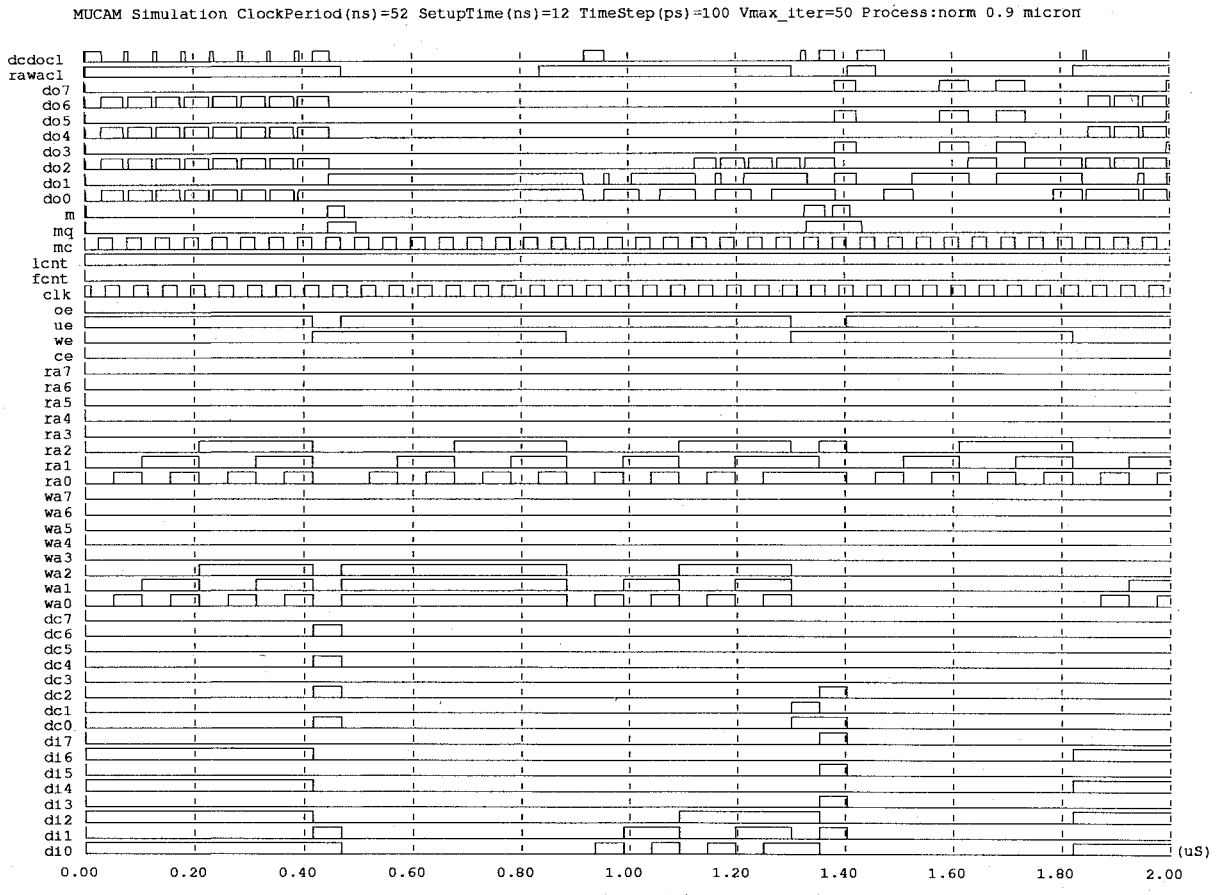


Fig. 4. Simulation result of first version.

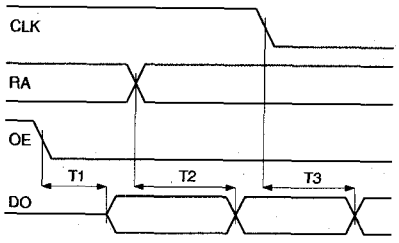


Fig. 5. Chip test timing diagram.

TABLE II
TIMING INFORMATION OF THE MUCAM

Symbol	Description	Timing
$T1$	Output enable time from *OE* to *DO* high or low level	8 ns
$T2$	Read access time from *RA* to *DO*	22 ns
$T3$	Propagation delay from *CLK* to *DO*	27 ns

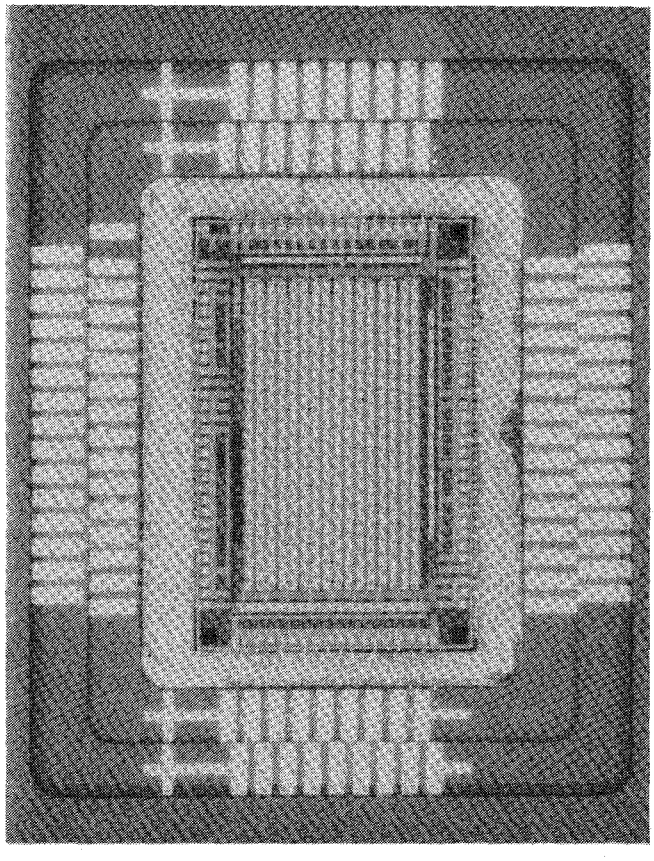


Fig. 6. Chip microphotograph.

cycle. Three types of timing numbers were measured (Fig. 5) at room temperature and nominal power supply voltage. Timing information is summarized in Table II. These timings satisfy the design specification of the connected component analyzer. The chip was found to be functioning correctly at a clock cycle up to 26 ns.

A microphotograph of the chip is shown in Fig. 6. The connected component analyzer based on this chip has also been developed.

VI. MEMORY CELL BASED MUCAM ARCHITECTURE

Another version of the MUCAM chip has been designed using memory cells. A basic CAM cell [13], [18], [19] has been modified to include the parallel search and multiple update operation. Fig. 7 shows a 1-b MUCAM cell and the update control circuitry. Transistors $M1$–$M6$ form a static RAM cell. Transistors $M9$–$M11$ form a comparator which compares the data stored in the RAM cell and the compare data, *DC*. Transistors $M7$ and $M8$ provide a path for the new input data during the PSMU operation. Transistors $M12$–$M16$ implement the update control circuitry.

A. READ and WRITE Operations of the 1-b MUCAM Cell

The READ and the WRITE operations of the cell are similar to those of a conventional static RAM cell. *DEC* is the output of a decoder. With *DEC* high, the input data on *DI* and *DIB* are passed through $M5$ and $M6$, respectively, and stored in the latch. During the read operation, *DI* and *DIB* are precharged, then *DEC* goes high. The signal change at *DI* and *DIB* is detected by a sense amplifier and presented to the output.

B. PSMU Operation of the 1-b MUCAM Cell

During the PSMU operation, the signals *EP*, *EV*, and UPDATE will be produced as shown in Fig. 8. Because of the precharging used, the PSMU operation is dynamic. During the time period $P1$, the transistor $M12$ is on, and the MATCH line will be precharged.

The compare operation starts with the compare data at *DC* and *DCB*, MATCH in precharged condition, and *EP*, *EV*, and UPDATE low. Then, both the *EP* and *EV* go high. With the transistion of *EP* and *EV* lines, the transistor $M12$ is turned off, and the transistor $M16$ is turned on.

If the stored data and the compare data are different, one of the transistors, either $M9$ or $M10$, is on, and only high signal value will be passed through the transistor which is on, regardless of the data stored in the RAM cell. In this case, transistor $M11$ will be on, and the charge stored in the MATCH line will be discharged through $M11$ and $M16$. Hence, the MATCH line will be low and transistor $M14$ will be off. Then the UPDATE line goes high (shown as $P5$ in Fig. 8). With the change of the UPDATE line, precharging of the *MU* line will stop and transistor $M15$ will be turned on. This change, however, does not produce any further change on line *MU* since transistor $M14$ is off. Therefore, the data stored in the RAM cell are intact.

If the stored data and the compare data are the same, then one of the transistors, either $M9$ or $M10$ is on, and only low signal value will be passed through the transistor which is on, regardless of the data stored in the RAM cell. In this case, transistor $M11$ will be off, and the precharged MATCH line will stay high. During the time period $P5$, both transistors $M14$ and $M15$ will be on, and

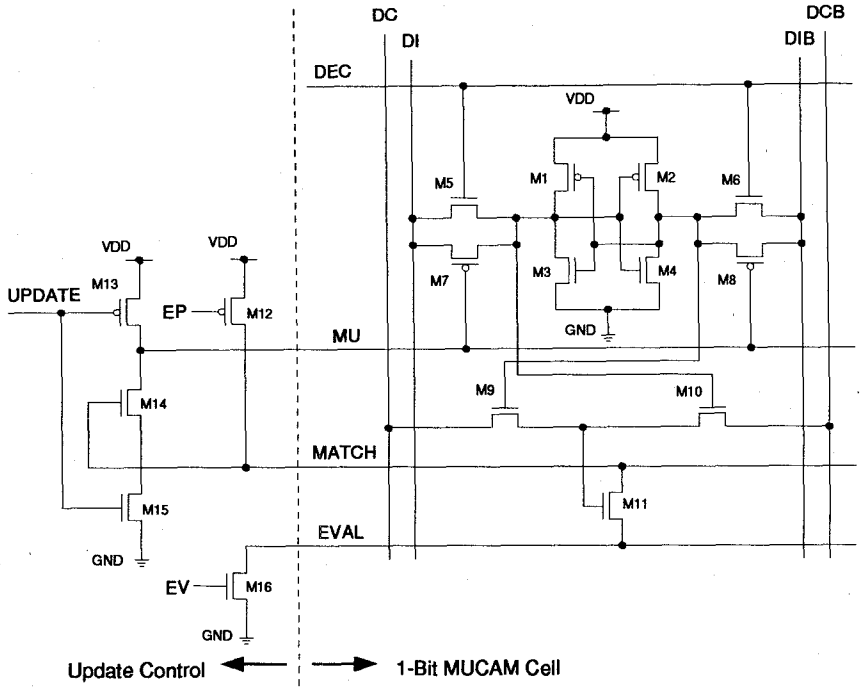


Fig. 7. One-bit MUCAM cell and update control circuitry.

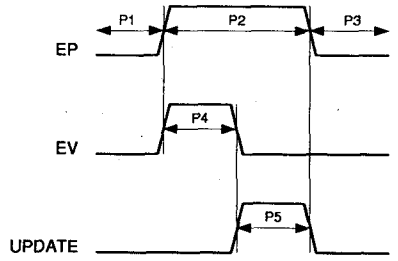
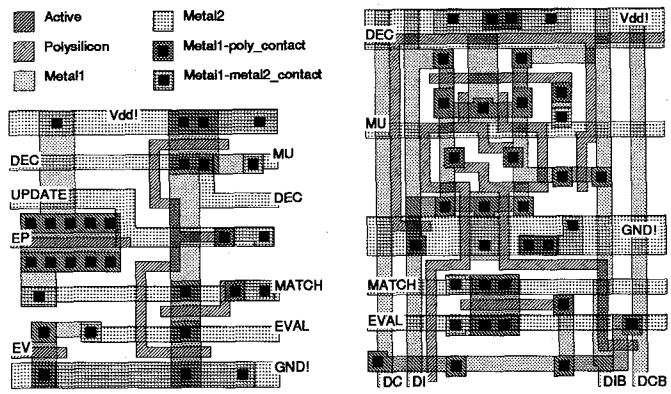


Fig. 8. Timing diagram of update control signals *EP*, *EV*, and UPDATE.



(a) (b)

Fig. 9. Layout of (a) update control circuitry and (b) 1-b MUCAM cell.

the charge stored in the *MU* line is discharged. This will force transistors *M*7 and *M*8 to turn on, and new data will be written into the RAM cell. It may be noted that this version of the MUCAM is not dual-ported.

C. N-bit MUCAM Cell

Fig. 9 shows the layout of the 1-b MUCAM cell. The 1-b MUCAM cell is cascadable to form an n-bit MUCAM cell as shown in Fig. 10. Update control circuitry is common to the n-bit word.

During the PSMU operation, the charge stored during the precharge cycle on the MATCH line will remain only if all the $DC[0:n]$ lines match with the data stored in the RAM cells. Hence, the *MU* line will go low and cause an update in the memory cells. If at least one bit is different, the charge stored in the MATCH line will be discharged and hence no further transition occurs on the *MU* line. Therefore, the entire word remains intact.

D. Simulation Results

Fig. 11 shows the simulation result of 2 × 2 cells with 50-ns cycle time using SPICE with Orbit processing parameters [20]. $DO0$ and $DO1$ are the output lines of the sense amplifier. During the first two cycles, the input data $DI[0:1] = 01$ and 10 are written into the first and second locations, respectively. During the next two cycles, each location is read in sequence. Next, the PSMU operation is illustrated. It searches for locations containing 10 and updates the content of matched locations with 01. Hence, the data at the second location are updated. The content of each location is read during the next two cycles. A second PSMU operation searches for locations containing 01 and updates the content of matched locations with 10. With this PSMU operation, the contents of both locations are updated with the new data. Each location is read during the next two cycles. The simulation result shows similar performance as the first version of the chip.

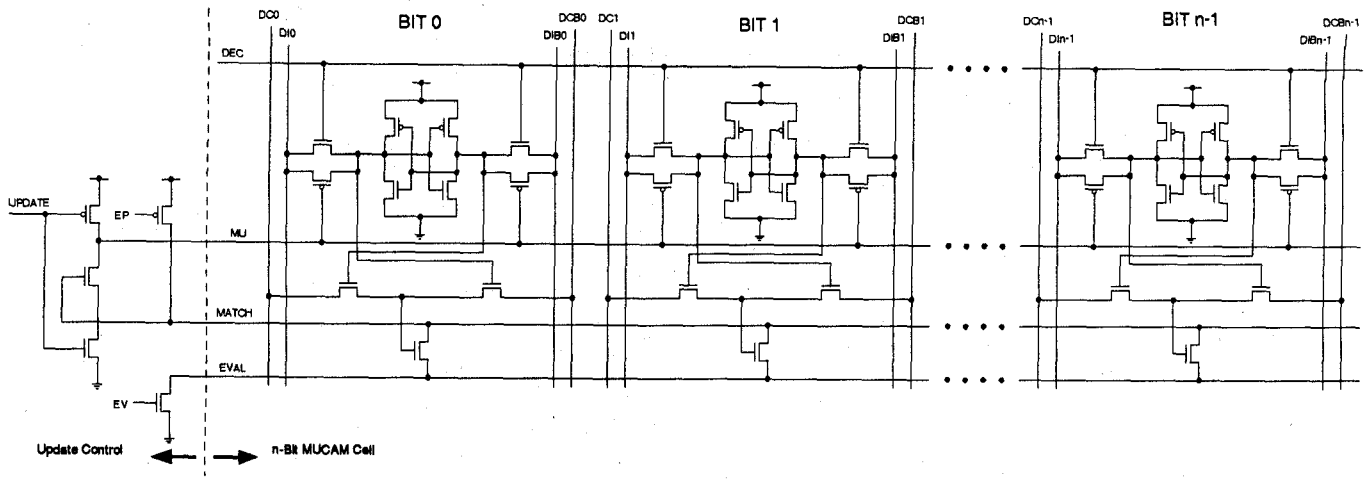


Fig. 10. Circuit diagram of n-bit MUCAM cell.

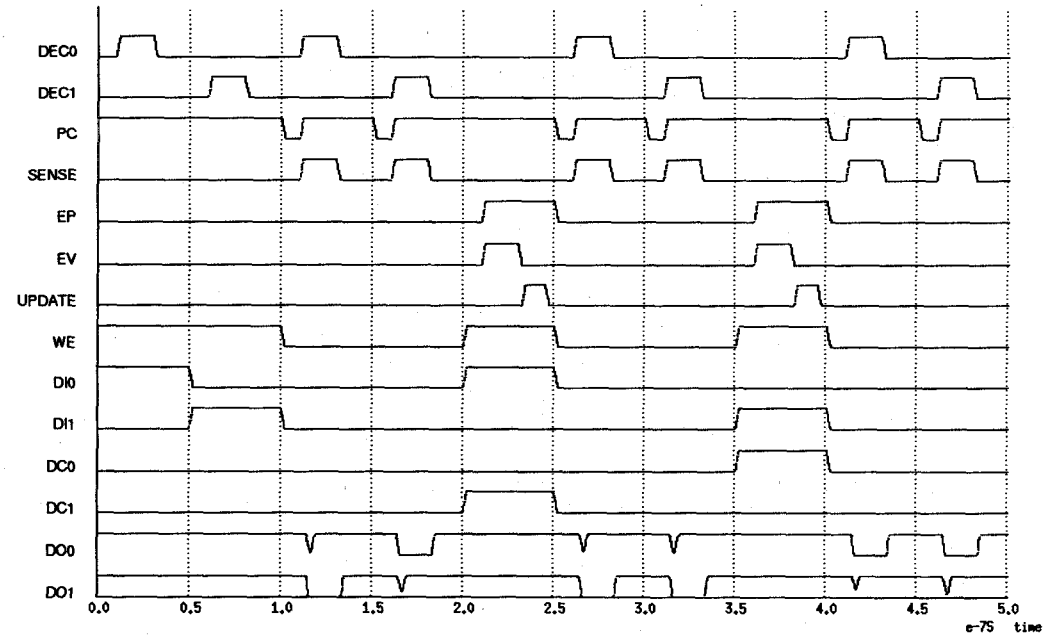


Fig. 11. Simulation result of second version.

E. Comparison of the Two Versions

The size of the first-version 8-b MUCAM cell with 0.9-μm feature size is 128 μm $\times$ 324 μm. In the second version with 2.0-μm process technology, the sizes of the 1-b MUCAM cell and the update control circuitry are 61 μm $\times$ 76 μm and 56 μm $\times$ 95 μm, respectively. Clearly, the second version is more area efficient and economical. A prototype (64 $\times$ 8 cells in a single column) of the second version of the MUCAM has been designed, laid out, simulated, and is fabricated using 2-μm CMOS technology through MOSIS.

VII. Conclusion

A high-speed special-purpose register array referred to as MUCAM was developed. It is shown that the unique feature of the parallel search and multiple update of MUCAM enables the one-pass implementation of a real-time connected component analyzer. Details of the design and test were presented. The chip is used as a key component of a real-time address block location system. MUCAM can be used in many other applications where high-speed parallel search and multiple update are required, such as database searches. Currently, the size of MUCAM is 256 $\times$ 8 bit. For other applications, a larger array may be needed, and memory cell based design can be easily modified to suit such requirements.

References

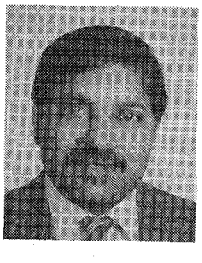
[1] A. Rosenfeld and A. C. Kak, *Digital Picture Processing.* New York: Academic, 1976.

[2] P. W. Palumbo, J. Soh, S. N. Srihari, V. Demjanenko, and R. Sridhar, "Real-time address block location using pipelining and multi-

processing," in *Proc. 1990 4th Advanced Technology Conf.*, Nov. 1990, pp. 73–87.
[3] *Maxbus Specifications*, Datacube Corp., Peabody, MA, 1987.
[4] C. Y. Lee and M. C. Paull, "A content addressable distributed logic memory with applications to information retrieval," *Proc. IEEE*, pp. 924–932, June 1963.
[5] L. Chisvin and R. J. Duckworth, "Content-addressable and associative memory: Alternatives to the ubiquitous RAM," *Computer*, pp. 51–64, July 1989.
[6] J. V. Oldfield, R. D. Williams, N. E. Wiseman, and M. R. Brule, "Content-addressable memories for quadtree-based images," CASE Center, Syracuse Univ., Syracuse, NY, Tech. Rep., 1988.
[7] C. C. Weems *et al.*, "The image understanding architecture," *Int. J. Comput. Vision*, vol. 2, pp. 251–282, 1989.
[8] C. C. Foster, *Content Addressable Parallel Processors.* New York: Van Nostrand Reinhold, 1976.
[9] R. M. Lea, "ASP: A cost-effective parallel microcomputer," *IEEE Micro*, pp. 10–29, Oct. 1988.
[10] B. D. Alleyne, D. A. Kramer, and I. Scherson, "A bit-parallel, word-parallel, massively parallel associative processor for scientific computing," in *Proc. 3rd Symp. Frontiers Massively Parallel Computation*, 1990, pp. 176–185.
[11] J. P. Wade and C. G. Sodini, "A ternary content addressable search engine," *IEEE J. Solid-State Circuits*, vol. 24, pp. 1003–1013, Aug. 1989.
[12] T. Ogura, J. Yamada, S.-I. Yamada, and M.-A. Tan-No, "A 20-kbit associative memory LSI for artificial intelligence machines," *IEEE J. Solid-State Circuits*, vol. 24, pp. 1014–1020, Aug. 1989.
[13] N. Correa, A. Garcia, M. C. Duarte, and F. Gonzalez, "An ASIC CAM design for associative set processors," in *Proc. 4th Annual IEEE Int. ASIC Conf. Exhibit*, Sept. 1991, pp. P18-3.1–P18-3.4.
[14] S. J. Adams, M. J. Irwin, and R. M. Owens, "A parallel, general purpose CAM architecture," in *Proc. 4th MIT Conf. Advanced Research VLSI*, 1986, pp. 51–71.
[15] *Am99C10: 256 X 48 Content Addressable Memory*, Advanced Micro Devices, Sunnyvale, CA, 1990.
[16] *Am95C85: Content Addressable Data Manager*, Advanced Micro Devices, Sunnyvale, CA, 1990.
[17] M. Motomura *et al.*, "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.
[18] N. Weste and K. Eshraghian, *Principles of CMOS VLSI Design, A Systems Perspective.* Reading, MA: Addison Wesley, 1985.
[19] J. P. Wade and C. G. Sodini, "Dynamic cross-coupled bit-line content addressable memory cell for high-density array," *IEEE J. Solid-State Circuits*, vol. SC-22, pp. 119–121, Feb. 1987.
[20] MOSIS, *Orbit Electrical Parameter Set.*

Yong-Chul Shin (S'85) received the B.S. and M.S. degrees in electronics engineering from Hanyang University, Seoul, Korea and the M.S. degree in electrical and computer engineering from the State University of New York at Buffalo in 1984, 1986, and 1989, respectively. He is currently a doctoral candidate in electrical and computer engineering at the State University of New York at Buffalo. His research interests are in the areas of associative processors, and neural networks, VLSI design, and computer architecture.

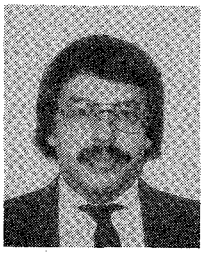
Mr. Shin is a member of the IEEE Computer Society.



Ramalingam Sridhar (S'82–M'87) received the B.E. degree in electrical engineering from the University of Madras, India, in 1980 and the M.S. and Ph.D. degrees in electrical and computer engineering from Washington State University, Pullman, in 1983 and 1987, respectively.

He is now an Assistant Professor of Electrical and Computer Engineering at the State University of New York at Buffalo. His research interests include language directed computer architecture, asynchronous processor design, neural networks, real-time computer architecture, special-purpose processor architectures, and VLSI design. He has been actively involved in the development of direct execution processors and real-time postal address recognition systems.

Dr. Sridhar is a member of the IEEE Computer Society and the Association for Computing Machinery.



Victor Demjanenko (S'82–M'83) received the B.S., M.S., and Ph.D. degrees in electrical engineering from the State University of New York at Buffalo in 1980, 1981, and 1983, respectively.

He is currently an Assistant Professor of Electrical and Computer Engineering at the State University of New York at Buffalo. He is also the President of Tree Technologies Corporation of Buffalo, NY, which specializes in product development and serves as consultant to other organizations. His present research interests include real-time image processing, custom computer architectures, network operating systems, and microprocessor-based control and diagnostic systems. He has been actively involved in the development of real-time postal address reading machines and in the integration of vibrational signal acquisition and diagnostic signal processing.

Paul W. Palumbo (S'85–M'87) received the B.A. and M.S. degrees from the State University of New York at Buffalo in 1983 and 1985, respectively.

He is the Project Manager for the CEDAR Real-Time Address Block Location project at the State University of New York at Buffalo and has worked there on postal research since 1984. His research interests are in the areas of image processing, segmentation, and parallel processing.

Sargur N. Srihari (S'74–M'75–SM'84) received the B.Sc. degree in physics and mathematics from Bangalore University in 1967, the B.Eng. degree in electrical communication engineering from the Indian Institute of Science, Bangalore, in 1970, and the M.S. and Ph.D. degrees in computer and information science from Ohio State University, Columbus, in 1971 and 1976, respectively.

He is the Director of the United States Postal Service Center of Excellence for Document Analysis and Recognition. The center's work is in developing methodologies, algorithms, and software and hardware for reading machines, with a focus on problems relevant to the postal domain. The center has over 70 individuals in its program, including graduate students, undergraduate students, full-time research scientists, and technical and administrative staff. He has been an Assistant, Associate, and full Professor at the State University of New York at Buffalo since 1978. He was Acting Chairman of the Computer Science Department during 1987–1988. He presently holds the Pattern Recognition Professorship of Computer Science. He is a coauthor of over 125 papers, two U.S. patents, and is the author of an IEEE tutorial on Computer Recognition and Error Correction.

Dr. Srihari is a member of the American Association for Artificial Intelligence, the Pattern Recognition Society, and the Association for Computing Machinery. He received a New York State/United University Professions Excellence Award for 1991.