An ASIC CAM Design for Associative Set Processors

Nelson Correa, Antonio García, María Cristina Duarte, and Fernel Gonzalez*
Department of Electrical Engineering
Universidad de los Andes
Apartado Aéreo 4976
Bogotá, D.E., Colombia

bitnet : NCORREA at ANDESCOL
ANGARCIA at ANDESCOL

Abstract

One of the key requirements in systems for symbolic computation is the fast and efficient execution of set operations, such as addition, removal, or test for membership of an element in a given set S. Content-addressable memory (CAM) offers the potential for massive fine-grained parallelism in the implementation of these operations, yielding a potential speedup of O(| S|). This paper describes an ASIC design of a 32x32 CMOS static CAM for use in an associative set processor.

1. Introduction

Application-specific integrated circuits (ASIC) are finding increased application in the design and implementation of special purpose hardware for symbolic computation. One of the key requirements in these systems is the fast and efficient execution of set operations, such as addition, removal, or test for membership of an element in a given set S. The applications include logic programming [9], functional programming [7], natural language processing [4], and syntactic pattern recognition [2], as well as the more traditional fields of relational databases and address translation in virtual memory systems.

Content-addressable memory (CAM) offers the potential for massive fine-grained parallelism in the implementation of these operations, yielding a potential speedup of O(| S|). While a conventional von Neumann computer accesses main memory serially to read instructions and data for operations on a central processor, an *associative* or CAM-based computer architecture moves some of the processing functions of the central processor to the memory, thereby achieving a more efficient utilization of the silicon resources in the computer. It is estimated in [5] that 88 percent of the silicon area in a conventional von Neumann computer is spent on its main memory system, which is by and large inactive since it is accessed serially. Moving some of the processor functions to memory also helps in reducing bus contention problems and memory bandwidth requirements, which are some of the major obstacles to parallel computation.

The main hindrance to the wide-spread use of CAM in ASIC is its high cost per bit, compared to that of RAM. While the dynamic RAM cell requires one transistor and a fully static one takes six, the smallest dynamic CAM design uses five transistors [11], and static designs range anywhere from nine to twenty-four transistors, depending on their capabilities [10]. Thus, a requirement for CAM applications in ASIC designs is the availability of space-efficient and fast dynamic and static CAM basic cells. A 20K-bit CAM design is presented in [8], and [1, 3] represent recent commercial announcements.

This paper describes the ASIC design of a 32x32 prototype static CAM for use in an associative set processor. The basic cell and the entire CAM memory were layed out using 2 µm design rules for a CMOS process. The paper is organized as follows: Section 2 gives a functional description of the circuit; section 3 discusses the basic CAM cell circuit and operation; section 4 presents SPICE simulation results for the basic cell; and section 5 outlines the design of the associative set processor. Finally, a conclusion and note on chip fabrication results are given.

---

* M. C. Duarte is now with the Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA 02139.

91TH0379-8/91/0000-P18-3.1$01.00

2. Functional description

A content-addressable memory or CAM is a memory in which any stored data item can be retrieved on the basis of a partial description of its contents [6]. The partial description is established by a *Data Input* word oi data to be compared and a *Mask Register* , which indicates the data fields (bits) relevant for the comparison. The data stored can be the Boolean values "0" and "1," and in some CAM cells and additional "don't care" state which is useful for some logical operations. The search is done in parallel across all words in the memory array and across all bits in each word. The result of the search operation is a match vector containing a "1" at the position of each word satisfying the search pattern. The vector may be stored in a *Match Register* for subsequent processing, and since the number of words satisfying the search pattern may be more than one, a *Multiple Response Resolver* is used to select one of the matching words for readout.

The memory that has been designed consists of seven major function blocks: cell array, address decoder, data I/O, mask register, match register, priority encoder, and control section. The block diagram is shown in Fig. 1.

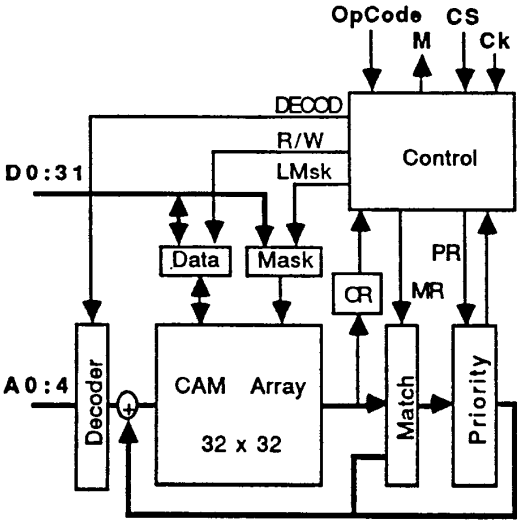

Fig. 1 Block Diagram of the CAM chip

The cell array consists of 32 words of 32 bits each, with capacity for bit storage (values "0" and "1"). The memory circuit implements seven functions selected by a 3-bit opcode, as follows:

Clear : Reset to "0" all memory cells, the control, and the Match and Mask registers.

Read/ Write : These are the conventional RAM-mode read and write.

Load Mask : Writes the 32-bit input word into the Mask Register.

Match : Compare de 32-bit input word with the contents of all words in parallel, according to the contents of the Mask Register.

Read Match : Read out the data word corresponding to the lowest "1" in the Match register and reset this position of the register to "0."

Write Set : Compares the 32 bit input word to all words and writes it into the word at the supplied address if not present.

3. Cell design and operation

There exist two basic alternatives for the design of a CAM cell: On one alternative, charge is stored on the gates of two MOS transistors; the other uses a cross-coupled inverter pair to store the data. The first alternative is attractive for the high density it offers and the possibility of storing a "don't-care" value, but suffers from a more complicated and less reliable design, lower noise margins, and the need for a periodic refresh operation. The basic CAM cell chosen was a ten-transistor static cell shown in Fig. 2, consisting of a six-transistor static RAM cell and a four-transistor XNOR comparator, which is used to pull down a precharged Match line in case the search data do not coincide with the stored data.

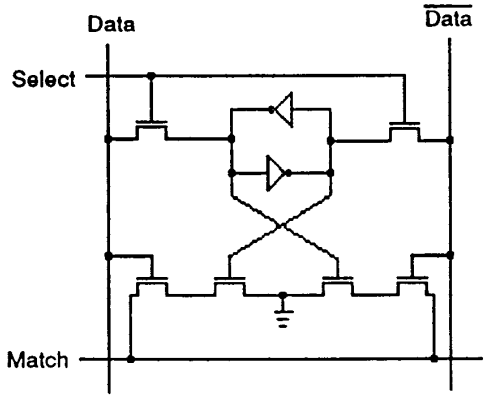

Fig. 2 Basic CAM cell

The Read and Write operations are as in a RAM cell. Writing is accomplished by raising the Select line and asserting the desired logic value on the Data and Data' lines; this forces the value into the cross-coupled latch. The Select line should be returned to zero at the end of the write operation. Reading is done by selecting the cell and then sensing the resulting value on the Data and Data' lines.

For the Match operation, the cell must be deselected and the Match line precharged while both Data ands Data' lines are at "0". The Data and Data' lines are then driven to the value to be compared, depending on the input word and the contents of the mask register:

Input	Mask	Data	Data'
0	0	0	1
1	0	1	0
X	1	0	0

We note that the four transistor XNOR comparator can be implemented with three, resulting in some performance advantages [8, 10], but that, curiously, the four transistor version can be layed out in less area.

The layout of the basic cell was done in an area of 50x54 μm^2 using 2 μm design rules for an N-well CMOS process with two metal layers. The area of the 32x32 cell array was 1.52x1.6 mm^2. This area is less than expected and was achieved using mirror images of the basic cell for adjacent bit and word cells, so that adjacent contacts and transistor drains could be shared between the cells.

4. Simulation results

After circuit extraction, the CAM cell was simulated with SPICE using a capacitive loading on the Data and Data' lines to account for the loading effect of 32 CAM words, and on the MATCH line to account for the loading effect of the 32 bits on a word. The simulation results show that cycle times in the order of 20 ns are attainable for the Read and Write operations. These times are for the CAM array alone, and do not take into account the delays of the read-write circuitry and the I/O pads.

The Match operation takes place in two phases. During the first, the Match line is precharged to a high level, and during the second evaluation takes place. Each phase takes 20 ns, so that the entire Match operation is done in 40 ns.

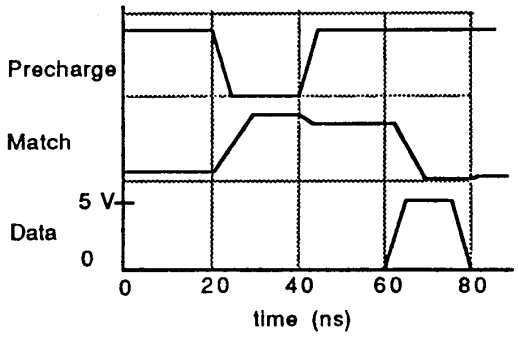

Fig. 3 SPICE simulation of a MATCH operation for a stored "0"

The Write Set operation is useful to implement the set-union operation and requires three cycles. During the first two a Match is performed and then, on the third, depending on the result of the match, a Write is executed. The total time is 60 ns.

5. Associative set processor

The associative set processor consists of a host CPU, a RAM module for the storage of program and data, and a 2k-word CAM module for asociative storage, as shown in Fig. 4. The architecture is not intended to be general purpose, but rather specialized for the fast and efficient execution of associative algorithms commonly and repetitively used in symbolic computation. These include, for example, unification in logic programming and pattern matching applications [9], and parsing algorithms in natural language processing [4].

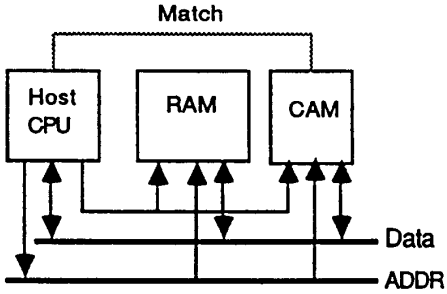

Fig. 4 Associative set processor

We note that while current CAM size and cost make it impractical for realistic relational database applications, they are ideal for symbolic computation, since the sets generated are continuously being searched and their size and that of their elements is modest. In the application considered in [4], for example, the elements generated (states for a context-free grammar and given input

string) can conveniently be encoded in 32 bits, and the size of the state sets generated typically reaches a few thousand elements.

Due to the complexity of each component in the associative processor, the first prototype will be a multichip system, with CAM alone accounting for 16 chips (assuming 128-word chips). However, we anticipate that the design will be integrated as higher density and cell-area efficiency becomes available for ASIC CAMs.

Conclusion

This paper has presented an ASIC CAM design for use in associative set processors. The paper has highlighted the importance of CAM in systems for symbolic computation, and pointed out the need for fast and area-efficient CAMs for use in the ASIC implementation of these systems.

The entire memory was layed out on a 4x4 mm^2 chip according to design rules for the 2 µm two-metal CMOS process of ES2 (European Silicon Structures), and fabricated with the cooperation of the CYTED-D program of the Spanish government for the celebration of the 500 years of the discovery of America. The first prototypes will available in May and be used for the construction of a hardware prototype of the associative set processor.

Acknowledgements

The authors would like to thank Nikos B. Troullinos of Syracuse University, Tere Oses of the Centro Nacional de Microelectrónica in Barcelona, and Alain Gauthier, head of the Electrical Engineering Department at Universidad de los Andes for their cooperation and support of this project.

References

[1] American Micro Devices. 1990. "Am99C10A Content Addressable Memory." Sunnyvale, CA.

[2] Chiang, Y. T. y K. S. Fu. 1984. "Parallel Parsing Algorithms and VLSI Implementations for Syntactic Pattern Recognition." IEEE *Transactions on Pattern Analysis and Machine Intelligence* , Vol. 6, No. 3, p. 302-314.

[3] Coherent Research, Inc. 1990. "CRC32256 Content Addressable Memory." Syracuse, New York.

[4] Correa, Nelson. 1991. "An Extension of Earley's Algorithm for S-Attributed Grammars." *Proceedings of the Fifth European Conference of the Association for Computational Linguistics* ; Berlin, April 9-11.

[5] Hillis, Dennis. 1985. *The Connection Machine* . MIT Press, Cambridge, MA.

[6] Kohonen, T. 1987. *Content-Addressable Memories* , 2nd Ed. Springer-Verlag, Berlin.

[7] McGehearthy, P. y E. Krall. 1986. "Potentials for Parallel Execution of Common LISP Programs." *Proceedings of the 1986 Int'l Conference on Parallel Processing*, I.E.E.E. Computer Society; p. 696-702.

[8] Ogura, Takeshi, J. Yamada, S.-I. Yamada, and M.-A. Tan-no. 1989. "A 20-kbit Associative Memory LSI for Artificial Intelligence Machines." IEEE *Journal of Solid State Circuits* , Vol. 24, No. 4.

[9] Oldfield, John, C. Stormon, y M. Brule. 1987. "The Application of VLSI Content-Addressable Memories to the Acceleration of Logic Programming Systems." CASE Center TR-8704, Syracuse University, Syracuse, New York.

[10] Troullinos, Nikos B. and C. D. Stormon. 1988. "Design Issues in Static Content-Addressable Memories." CASE Center technical report, Syracuse University, Syracuse, New York.

[11] Wade, Jon, and C. G. Sodini. 1987. "Dynamic Cross-coupled Bitline Content Addressable Memory Cell for High Density Arrays." IEEE *Journal of Solid State Circuits* , Vol. SC-22.