A Self-Testing Reconfigurable CAM

Anthony J. McAuley and Charles J. Cotton

Abstract—**This paper describes a 2-μm CMOS reconfigurable content addressable memory (RCAM) organized as 256 words by 64 b, that can be written, searched, or read with 100-ns cycle time. Designed for general high-speed table look-up applications, the RCAM has self-test and reconfiguration features to simplify testing and improve yield. In test mode the RCAM tests the whole memory array in under 300 clock cycles, and automatically reconfigures itself to correct any hard errors it finds, greatly enhancing the chip yield. As an example, we describe the application of the RCAM to address translation. Compared to RAM-based address translation, using the RCAM increases speed and reduces complexity. Also, the RCAM's storage reclamation feature simplifies system storage management by allowing old entries that have not been accessed recently to be freed for reuse.**

I. Introduction

CONTENT addressable memory (CAM) technology has wide application to high-performance data communication. While access times are typically not critical, the applications require storage and management of hundreds or thousands of records, and often require high reliability. This paper describes an experimental reconfigurable CAM (RCAM) designed to meet these requirements in a single 2-μm CMOS chip.

A CAM has three main functions. The write function stores data in unused word locations. The match function searches through the stored words in parallel for a match against an input word, indicating whether it has found a match. The read function returns the data associated with the matched word; if there is more than one matched word (indicated by the F_d line remaining low after reading the first word), the words can be read out sequentially one per clock cycle. There are two general methods of achieving these three functions: using an addressable CAM and an addressless CAM. The differences between the two architectures are reflected in their write and read functions.

Functionally, an addressable CAM [1]–[5] is a RAM with a comparator tied to each stored word. An external controller decides in which address to store the data; a matched word returns this address as the associated information, which often requires translation (e.g., through a RAM) to obtain the required data. An addressless CAM has no address encoder or decoder. The CAM's own control logic decides where to store the data (the system has no indication of which word was used) and the return information is part of the data word supplied by the system when the word was written. In general, the addressless CAM has more flexibility in what data to associate with each word.

Manuscript received August 3, 1990; revised October 30, 1990.
The authors are with Bellcore, Morristown, NJ 07962-1910.
IEEE Log Number 9041569.

There is an interesting reliability difference between the two forms of CAM. Addressable CAM's, like RAM's, need a "full" address space (with no holes) to provide useful system function, so switching out bad words is difficult. In contrast, the addressless CAM does not need a "full" address space, since it controls where words are placed in the memory array. Therefore, the addressless CAM can mark words as bad without having to replace them. The corrected addressless CAM will shrink the number of words available by the number of bad words (see Section IV), but the memory has no holes.

For our data communication applications we chose the addressless-CAM architecture for three main advantages: 1) the need for flexibility in choosing the associated data, 2) the need for simplified memory management, and 3) the need for reliability. Section II describes a typical application, bringing out the need for the first two characteristics. Section III highlights the advantages of using self-test and reconfiguration in CAM implementations. Section IV describes a custom chip implementation. To store the large quantities of data required by our application, Section V illustrates how multiple RCAM's can be cascaded. Finally, Section VI describes the experimental results.

II. Application to a High-Speed Packet Switch

Accessing data by its contents, instead of by its address, has many potential applications [1], [6], [7]. The RCAM can be used as a general CAM, but is designed primarily for address translation in a high-speed packet switching network.

In a high-speed packet switch [8], data arrive in packets, with each packet having an address in the header. The main function of the switch is to route the packet onto the correct output port based on the extracted address field. Assuming the address-to-port translation has already been done, the switch must translate the packet's destination address of 24–60 b into an output port number of 40–4 b (an address-port pair has up to 64 b for the RCAM). The task for the CAM (which has up to 256 active connections) is to allow simple, fast access to the address-port mapping table.

If the address space is relatively small (less than 24–32 b) it is possible to use the address field as the address into a large RAM. This meets our speed requirements; but, assuming current dynamic RAM technology, the memory takes up considerable board space, requiring complex refresh logic and significant power. Use of a hashing table is possible, but is complex and does not guarantee unique mapping and the search time is not constant. However, using CAM technology meets all of our requirements. The destination address and

port address are written into a 64-b CAM word and a search mask register is set so that searches are only made over the address field; subsequently, following a successful match, a read yields the port number of the matching address.

For example, if the mask register enables only the bottom 48 b during a search, writing a 48-b address into the lower 48 b of the RCAM allows the top 16 b to be used to store a port address. After receiving a packet, the extracted address is put onto the lower 48 b (with the top 16 b being don't care); if any word matches, the port address can be read out on the top 16 b of data (with the bottom 48 b duplicating the address).

An important advantage of using the addressless-CAM architecture in the RCAM for the address-port translation is that it simplifies free-space management. In some packet switching applications (e.g., a MAC layer bridge), the RCAM serves as a high-speed address cache. One would like to keep only actively used addresses in the cache and free inactive addresses for reuse. The RCAM is able to manage the memory without increasing the system complexity using the *flush* operation. A flush identifies those RCAM words which have not been accessed (i.e., written, read, or searched for) since the last flush operation. RCAM words thus identified are marked as "old." A second flush will mark any "old" locations which still have not been accessed as free for reuse.

III. Self-Test and Reconfiguration

Two important aspects of chip design are reducing test cost and increasing yield. This is particularly true for large, dense designs such as the RCAM, which has over a quarter million transistors. The RCAM addresses these important issues through its self-test logic (which only requires the tester to provide a clock input and look at the F_o output) and its automatic reconfiguration logic (which can be done at any time, without any specialized laser equipment).

The RCAM's self-test and reconfiguration features exploit the nature of the CAM, particularly the addressless architecture. Self-test is easier because a CAM has built-in matching logic, so separate pattern-matching logic is not necessary. Not only can all the RCAM words test themselves in parallel in one clock cycle, but the RCAM has a parallel write operation, which allows all words to be written in one clock cycle (see chip description). Reconfiguration is easier because each word is addressed by its contents, instead of a one-to-one mapping with an external address (e.g., a RAM); so, instead of a bad word causing a hole, bad words can be bypassed. The only external effect of reconfigured words is a small reduction in CAM capacity.

The RCAM cell uses a static six-transistor RAM cell, so it is relatively safe from soft, transient errors. If hard, permanent errors develop at any time, the RCAM is able to test itself in 260 clock cycles (less than 1 ms) and reconfigure to bypass any bad word locations it finds. Typically, the self-test and reconfiguration is run when power is first switched on.

IV. Chip Description

Fig. 1 shows a block diagram of the RCAM and Table I summarizes its characteristics. There are a total of 84 pins: 64 for a bidirectional data bus (D[63:0]), nine for power (V_{dd} and V_{ss}), three for multichip expansion and matching or full signals (F_i, F_o, and F_d), and eight for control. The RCAM's

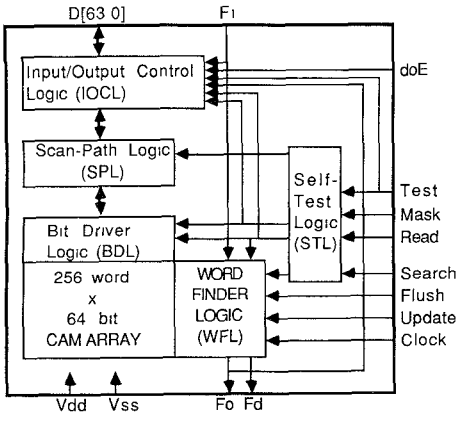


Fig. 1. RCAM functional block diagram.

TABLE I
RCAM Characteristics

Process Technology	2-μm CMOS double metal
Memory Structure	256 words×64 b
Die Size	8.888 mm×8.888 mm
Number of Devices	257 190
Number of Pins	84
Package (Cavity) Type	LCC (JDEC B)
Supply Voltage	5 V
I/O Interface	TTL (50 pF)
Power dissipation	850 mW (at 10 MHz, with 16 outputs changing)
Cycle Time	100 ns
CAM Cell Area	48 μm×42 μm
Complete Self-Test	260 cycles (at 2.5 MHz)
Fault Tolerance	Approximately 75% of die area

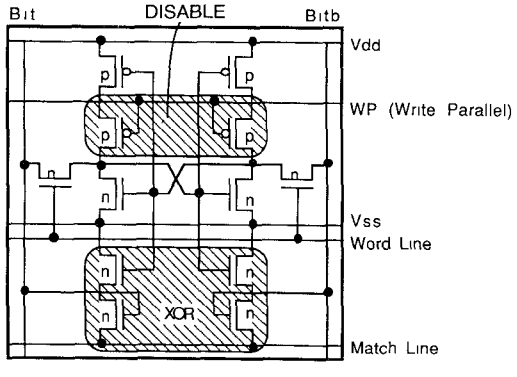


Fig. 2. Twelve-transistor CAM cell.

functions include: write, search, read, mask write, mask read, flush, and self-test.

A. CAM Array

The CAM array has 16K associative memory elements, each composed of 12 transistors as shown in Fig. 2. The static design is made up of three sections: the six-transistor static RAM memory, the four-transistor exclusive-OR logic (labeled XOR), and the two-transistor parallel write pull-up disable gate (labeled DISABLE). The design is similar to that described by Kodata *et al.* [5], with the addition of two transistors to disable the pull-up transistors during parallel write operations used for self-test. The two extra devices do

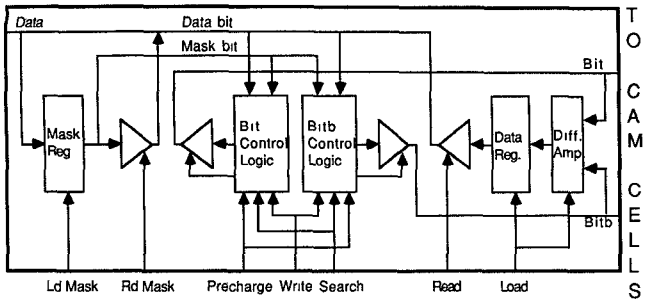


Fig. 3. Simplified block diagram of the bit driver logic cell.

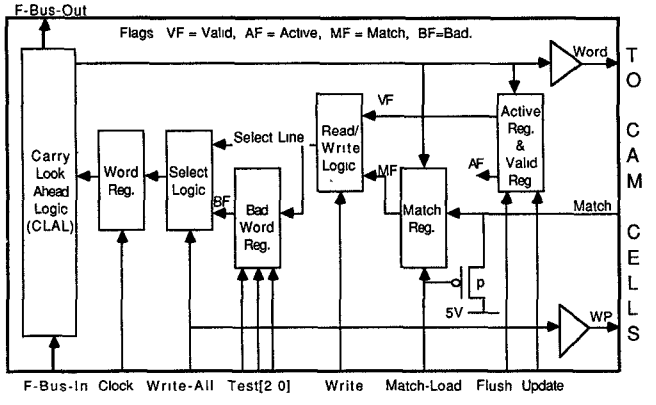


Fig. 4. Simplified block diagram of the word finder logic cell.

not increase the cell area, which is constrained by the word finding logic (WFL) and bit driver logic (BDL) pitches.

B. Bit Driver Logic

The BDL is similar to that used in a RAM, with the addition of a search operation and mask register. A reset mask register bit forces the bit and bit$b$ lines low simultaneously during a search, making the associated search field bit a don't care.

Fig. 3 shows one cell of the BDL. For a read, the tristate drivers precharge the bit lines when the clock is high (while data and control signals are being changed); the differential amplifier senses when the clock goes low and the data are latched on the rising edge of the clock. For a write or search, the bit/bit$b$ tristate drivers charge one line and discharge the other, and if the mask register is reset during a search, both lines are discharged.

C. Word Finder Logic

By contrast, the WFL is different from the address decoders found in RAM's and other CAM's [1]–[5]. The WFL is the heart of the special features of the RCAM; replacing the conventional address decoder with the WFL removes the one-to-one correspondence between address and location, simplifying the reconfiguration logic and improving the chip's yield.

One cell of the WFL, shown in Fig. 4, has four 1-b flags as described in Table II. AF and VF act as a low complexity least-recently-used storage management mechanism, simplifying system storage reclamation. A word sets AF and VF when its word line is active and the external update pin is set; a word resets AF when Flush is set, and resets VF when Flush is set with AF already low. Thus, the RCAM frees a word if it has not been touched between two flushes. During a search, a matched word has no CAM cells pulling the match line low, so the weak p transistor pulls the match line up (MF = 1); in contrast, an unmatched word has at least one CAM cell pulling the match line low (MF = 0). To read data, the WFL passes MF into the word register, allowing the carry lookahead logic (CLAL) to find the first matched word (MF = 1). To write data, the WFL passes VF into the word register, allowing the CLAL to find the first free word (VF = 0). Finally, BF is set if a word failed to pass the latest self-test cycle. Any word with BF set takes no part in searches (the word always appears to have MF = 0) or writes (the word always appears to have VF = 1).

TABLE II
FLAGS IN THE WORD FINDER LOGIC

Flag	Description	Flag is set if
AF	Active Flag	The word has been accessed, with Update set, since the last flush.
VF	Valid Flag	The word has been accessed, with Update set, since the last two flushes.
MF	Match Flag	A Valid word is matched with the masked data string.
BF	Bad Flag	The word has failed to pass the latest test cycle.

TABLE III
RCAM TEST SEQUENCE

Clock Period	CAM Array	Search Pattern	MF Should Be
0–64*	*XXX · · · XXXXX*	*XXX · · · XXXXX*	*X*
65	111 · · · 11111	*XXX · · · XXXXX*	*X*
66	111 · · · 11111	111 · · · 11111	1
67	111 · · · 11111	111 · · · 11110	0
68	111 · · · 11111	111 · · · 11101	0
69	111 · · · 11111	111 · · · 11011	0
· · ·			
128	111 · · · 11111	101 · · · 11111	0
129	111 · · · 11111	011 · · · 11111	0
130–194*	*XXX · · · XXXXX*	*XXX · · · XXXXX*	*X*
195	000 · · · 00000	*XXX · · · XXXXX*	*X*
196	000 · · · 00000	000 · · · 00000	1
197	000 · · · 00000	000 · · · 00001	0
198	000 · · · 00000	000 · · · 00010	0
199	000 · · · 00000	000 · · · 00100	0
· · ·			
258	000 · · · 00000	010 · · · 00000	0
259	000 · · · 00000	100 · · · 00000	0

*Scan path logic being loaded

D. Self-Test and Scan-Path Logic

The self-test logic (STL) is a finite state machine which generates and loads the test patterns, shown in Table III, into the scan-path logic (SPL), serially, one bit at a time. It has 12 internal state latches which keep track of the bit count (e.g., so it knows when it has finished loading a 64-b test vector) and the current state (e.g., it is loading the first vector, or some other state from Table III). The remaining circuitry is random logic and buffers. The STL must generate all the signals necessary to load the SPL and control the CAM. It controls the CAM by driving the internal control

signals; the external control pins (except the clock and test pins) are disabled at the pads when the test pin is active.

During self-test, the clock (at one-fourth maximum frequency, to allow parallel writing) controls the STL. It takes the first 64 clock cycles to put the first word, $\langle 111 \cdots 111 \rangle$, into the SPL. After the word is written into every word of the memory array on the 65th clock cycle, the same word is matched on the 66th clock cycle. Subsequently, while the data in the CAM remain unchanged (at $\langle 111 \cdots 111 \rangle$), a walking ZERO's test pattern (a word containing just one ZERO) is matched against every word. After the 130th clock cycle, the operation is repeated with the STL generating the inverted bits (see Table III).

The 260-step self-test sequence tests all simple combinations of stuck-open and stuck-closed faults in the array. When the Test line is first raised, all the BF's are cleared to mark all words as good (BF = 0); later, any word failing a test, by producing a match when no match was expected or no match when a match was expected, is marked as bad (BF = 1). For example, in the 66th cycle, when all words contain $\langle 111 \cdots 111 \rangle$, the RCAM matches the words with $\langle 111 \cdots 111 \rangle$ (the contents of the SPL). All words should match (MF = 1); so, the WFL marks as bad any word that fails to match (MF = 0). In the 67th cycle, the STL shifts a ZERO into the SPL and matches the new SPL contents, $\langle 111 \cdots 110 \rangle$, against all CAM words (still $\langle 111 \cdots 111 \rangle$). No word should match (MF = 1); those that do are marked as bad (BF = 1). The select logic disables words that were marked as bad; so, the RCAM has $256 - x$ usable words, where x is the number of bad word locations found during self-test. The coverage is approximately 75%, since a bad bit line or pad disables the chip.

E. Input / Output Control Logic

The input/output control logic ensures that only one chip writes onto the external data bus. To write onto the data bus Read and doE (data output Enable) must be active, Test must be inactive, and either Mask must be active (i.e., we are reading the mask data) or F_i must be active and F_d inactive (i.e., this chip is the next chip with matched data to be read).

V. Building Larger CAM Arrays

It is often necessary to form larger memories than can be built from one chip. For more than 256 words, the RCAM has extension logic built into the WFL. For larger multichip applications, all RCAM pins are tied together, except for the F-line pins (F_i, F_o, and F_d) which are chained. The F-line pins ensure that only one chip writes onto the data bus and only one chip stores a word.

F_i, F_o, and F_d are analogous to the carry-in, carry-propagate, and carry-generate signals of a CLA adder. If F_i is high, both F_o and F_d are set high if: the system is writing into the last free RCAM word, any word produces a match during a search, or there are more words to be read during a read. The difference between F_o and F_d is that F_o only goes high if F_i is high.

When combining RCAM's the F-line may be combined either serially or in parallel. Fig. 5 shows a 2K-word×64-b CAM system built using eight RCAM chips; the ripple carry F_i / F_o lines add 8 ns to the cycle delay per stage. The F_d signal, together with an external CLAL (PLA, PAL, or similar device) such as that shown in Fig. 6 improves the cycle time for large CAM's.

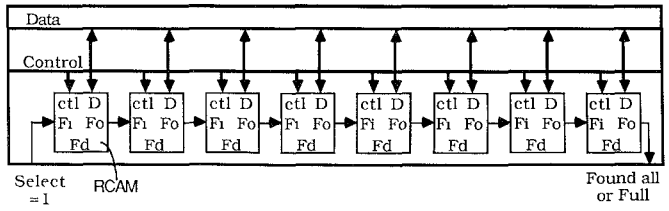


Fig. 5. A 2K-word CAM system (using eight RCAM chips), without carry lookahead.

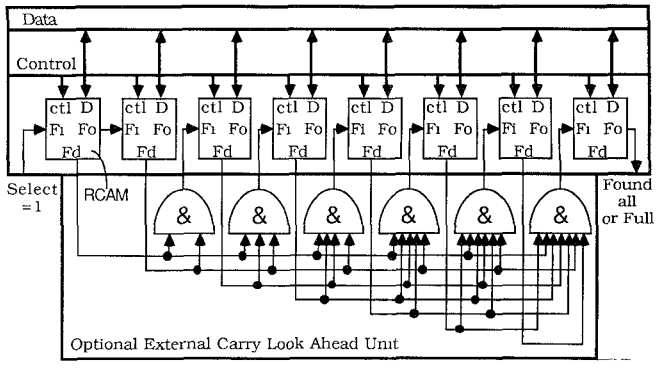


Fig. 6. A 2K-word CAM system (using eight RCAM chips), with carry lookahead.

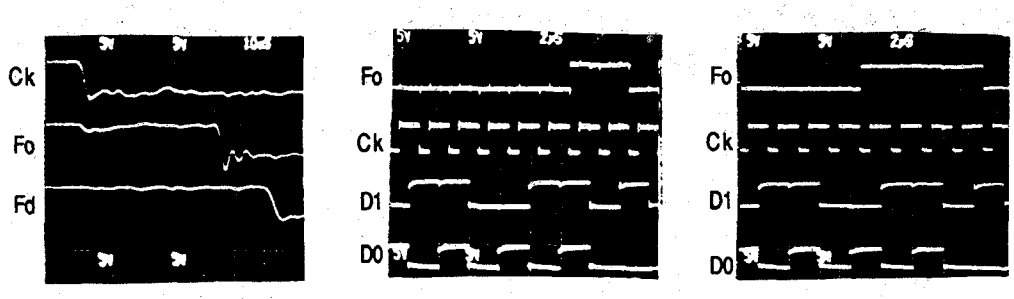
For very large memories it becomes desirable to provide some protection against soft errors. Though it is possible to use some expensive form of replication, such as triplication, it is not possible to use RAM error correction coding techniques in a CAM. Therefore, the RCAM avoids using dynamic logic, which is more prone to soft errors.

VI. Experimental Results and Chip Layout

Table I summarizes the test results. The figure for power was obtained while changing only 16 of the data lines (which is all that is required for our application). Fig. 7(a) shows a successful search forcing F_o and F_d high after the clock goes low. The worst-case find time, from the clock going low to F_d going high, was 50 ns.

One RCAM chip was successfully reconfigured. After a self-test cycle and two flush cycles the RCAM should have 256 free locations; therefore, writing to all 256 locations should force F_o high when the last word is being written (indicating that there are no more free locations). Fig. 7(b) shows a "good" chip forcing F_o high when all words have been written into, while Fig.7(c) shows a "faulty" chip forcing F_o high two clock cycles earlier. Because it fills up after 254 words have been written, we know the "faulty" chip, which was otherwise functionally perfect, has had two locations removed by self-test.

Fig. 8 shows a photograph of the layout. The CAM array is split into two vertical slices, with 128 words in each half. The self-test logic and scan-path registers take up less than 1% of the total chip area (in the middle close to the bottom). The RCAM's cells were placed and routed using a high-level language, allowing quick redesign for different technologies or functions.



(a) (b) (c)

Fig. 7. Typical RCAM timing diagrams: (a) search time, (b) "good" chip (full at 256), and (c) "faulty" chip (full at 254).

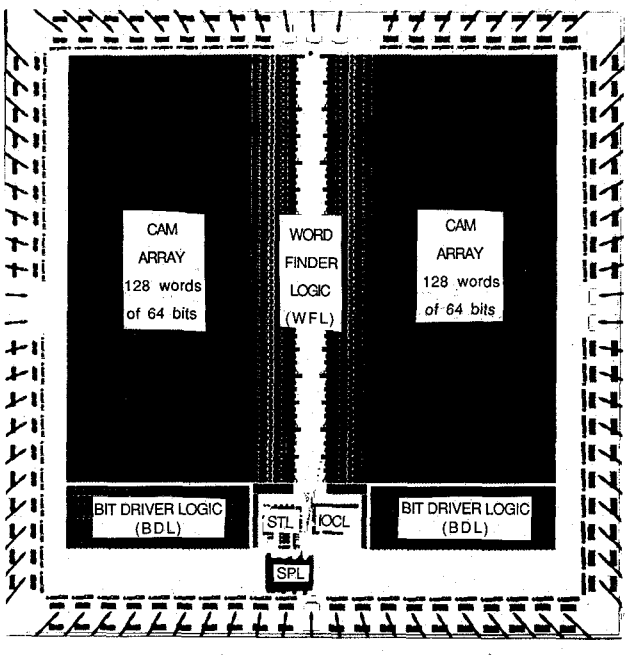


Fig. 8. Photograph of the packaged RCAM.

Acknowledgment

The authors wish to thank the anonymous reviewers for their helpful suggestions.

References

[1] L. Chisvin and R. Duckworth, "Content-addressable and associative memory: Alternatives to the ubiquitous RAM," *IEEE Computer Mag.*, pp. 51–64, July 1989.

[2] 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.

[3] S. Jones, I. Jalowiecki, S. Hedge, and R. Lea, "A 9-kbit associative memory for high-speed parallel processing applications," *IEEE J. Solid-State Circuits*, vol. 23, pp. 543–548, Apr. 1988.

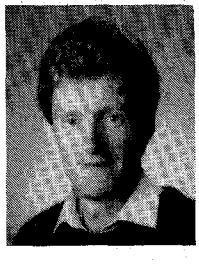
[4] T. Ogura, S.-I. Yamada, and T. Nikaido, "A 4-kbit associative memory LSI," *IEEE J. Solid-State Circuits*, vol. SC-20, pp. 1277–1281, Dec. 1985.

[5] H. Kodata, J. Miyake, Y. Nishimichi, H. Kudo, and K. Kagawa. "An 8Kb content-addressable and reentrant memory," in *ISSCC Dig. Tech. Papers*, Feb. 1985, pp. 42–43.

[6] J. G. Bonar and S. P. Levitan, "Real-time LISP using a content addressable memory," in *Proc. 10th Int. Conf. Parallel Processing*, Aug. 1981, pp. 112–117.

[7] H. Yamada *et al.*, "Real-time string search engine LSI for 800-Mbit/sec LANs," in *Proc. Custom Integrated Circuits Conf.*, May 1988, pp. 21.6.1–21.6.4.

[8] W. D. Sincoskie and C. J. Cotton, "Algorithms for adaptive routing in communication networks," *IEEE Network*, pp. 16–24, Jan. 1988.



Anthony J. McAuley was born in Liverpool, England. He received the B.S., and Ph.D. degrees from Hull University, England, in 1981 and 1985, respectively.

From 1985 to 1987 he was a Post-Doctoral Research Fellow at the California Institute of Technology, Pasadena. Since 1987 he has worked with the Computer Communications Research District in Bellcore, Morristown, NJ. His active research interests include packet communication, error control, self-timed systems, VLSI architecture, and computer arithmetic.

Charles J. Cotton received the B.S.M.E. degree from the University of Texas at Austin in 1975 and the Ph.D. (E.E.) degree from the University of Delaware, Newark, in 1984.

From 1975 to 1979 he was involved in communications and distributed control research at E. I. Dupont's Engineering Physics Laboratory. From 1984 to 1986 he was a Member of the Technical Staff at Bellcore in Morristown, NJ, engaged in network traffic monitoring and LAN interconnection and operation. Promoted to District Manager in 1987, he now heads the Computer Communication Research group, currently investigating multiservice transport protocols for broad-band packet networks.