HALF WEIGHT BLOCK CODES FOR OPTICAL COMMUNICATIONS

Ernest E. Bergmann, Andrew M. Odlyzko, and Suresh H. Sangani

This paper discusses the possibilities of using certain constant-weight block codes in optical data links and in optical ring networks. The codes that appear to be particulary attractive have exactly as many 1s as 0s in each code word, so there is no penalty for ac coupling. Codes can be simple, such as the Manchester code. Or they may be more complicated and support control channel signaling, error detection, and forward error correction. This paper bridges the gap that sometimes exists between the mathematically "ideal" block codes and the typical requirements of particular applications. It presents a variety of constant-weight codes of increasing length, including a code of length 16 that is being implemented for a very high-speed, ultra-reliable, ring data network.

Ernest E. Bergmann *is a member of technical staff in the Lightwave Data Links and Interfaces Department at AT&T Bell Laboratories in Allentown, Pennsylvania.* ***Andrew M. Odlyzko*** *is head of the Mathematical Studies Department at AT&T Bell Laboratories in Murray Hill, New Jersey.* ***Suresh H. Sangani*** *is a member of technical staff in the IMS (Interprocessor Message Switch) Development Department at AT&T Bell Laboratories in Columbus, Ohio. Mr. Bergmann, who joined AT&T in 1984, works on fiber-optic local area networks. He has an A.B. from Columbia University and an M.A. and Ph.D. from Princeton University, all in physics. Mr. Odlyzko joined AT&T in 1975. He does research in complexity, probability, cryptography, combinatorics, error-correcting codes, and analysis. He has a B.S. and M.S. from*

(continued on page 93)

Background

The evolution of data communications through optical fibers now permits the use of inexpensive light emitting diodes (LEDs) and PIN (p-type intrinsic n-type) detectors, instead of expensive lasers and complex photodetectors. (Examples are AT&T's ODL™-50 and ODL-200 optical data link products.) But this evolution is coupled to the need to explore new, cost-effective, coding methods that can be used at high data rates.

The coding methods must provide reliable detection capability with, as options, out-of-channel signaling, error detection, and forward error correction. We expect that the most suitable codes will continue to change because speed, cost, and reliability tradeoffs keep changing.

Block codes, which encode each group of m bits of data as an n-bit block, provide potential advantages:

- no dc component in detection if we use 50-percent weighting
- a guaranteed minimum number of transitions between 0s and 1s for clock recovery
- only short runs of consecutive 1s and 0s
- higher data-format efficiency

- error detection and forward error correction.

There appears to be a natural progression of block codes with weights that equal half the block length. The simplest one—the well-known Manchester code (Table I)—is of length two. As the block length increases, so does the electronic complexity for coding and decoding, but we can achieve more services such as out-of-channel signaling, error control, and higher coding efficiency.

This paper is intended to bridge the gap between mathematically ideal block codes and the requirements that are typical of particular applications. After we discuss some requirements that appear at the physical layer (part of the network protocol), we shall present a variety of constant-weight codes of increasing length.

Modulation Requirements

This study was motivated by the desire to promote effective usage of AT&T's ODL-200 optical data link[1] that uses an LED transmitter and a PIN receiver and operates at 200 Mb/s. However, many of the principles should apply to similar current and future optical links that operate at other rates.

The most convenient way to operate an LED transmitter is with on-off keying. Varying the frequency or phase of the electromagnetic radiation (the infrared light) for an LED transmitter does not appear practical.

Similarly, the ODL-200 receiver acts as an incoherent square-law detector (a PIN diode and linear amplifier). The analog signal at the decision circuit is a voltage that is almost proportional to the received light, averaged over 1 to 2 ns, plus a constant dc offset.

Because the ODL-200 receiver does not preserve a dc level, we need a 50-percent duty cycle (the fraction of time that the light is on compared to the total elapsed time). Whenever the modulation causes the duty cycle to differ from 50 percent, we expect that receiver designs will work at a disadvantage in terms of the bit error rate (BER) at a given peak received power.

Requirements List. Here is our list of requirements for coding.

Table I. Coding Table for Manchester Code

Data	Code word
0	10
1	01

Code weight. If dc coupling or restoration is absent, extending the low-frequency response provides only a partial solution. Performance requirements usually are defined for worst-case situations.

For block codes, a worst case usually consists of the indefinite repetition of the code word that has either the least number of 1s or the most 1s. Such cases produce the greatest shift in the baseline. Therefore, we chose to consider block codes where all the code words have 50-percent weight, i.e., each code word has as many 0s as 1s.

High efficiency. Because our interest is to maintain a high data rate, we want a reasonably high coding efficiency (the fraction formed by the data rate compared to the clock rate). For block codes, we can use the ratio of the data length to the block length, m/n, to identify this efficiency. The ODL-200 receiver supports clock rates up to 220 MHz.

A variety of block codes seem able to support data rates from 80 to 200 Mb/s if NRZ (nonreturn to zero) on-off modulation methods are used to encode each code word onto the fiber. Namely, for each bit value, we assume that the transmitter is either on or off for the full duration of one clock cycle. We also assume that a 0 corresponds to no light, whereas a 1 corresponds to light.

NRZ coding. We want to use NRZ coding; a common variant of NRZ is NRZI (nonreturn to zero inverted), which we do not favor.

Both coding methods are similar in appearance; i.e., during each clock cycle, the light level is constant (either on or off). However, NRZI decoding identifies a 0 as no change in level from one clock interval to the next and interprets a change as a 1.

NRZI is most attractive in balanced wire-pair transmission, because demodulation is equally successful regardless of the received signal's polarity. Because polarity confusions are not a problem for on-off optical transmissions, polarity insensitivity is not a particular advantage here.

However, an error in detecting the wrong level during one clock interval would produce a single raw bit error with NRZ but a pair of bit errors for NRZI decoding. This error multiplication for many block codes is considered an undesirable effect for most applications.

Error detection and control. Error detection and forward error correction are desirable capabilities.

Here, we explore more complex codes with possibly lower format efficiencies, because we can support higher clock rates and want better error detection and control. With the rapid progress of technology, today's complex, high performance circuits may be tomorrow's conventional approach.

Control codes. Besides providing "transparent" data transport, we would like to provide control codes for functions such as idle, frame, etc.

We also try to address the possible disadvantages that some authors[2] have listed for using block codes:

- complexity of the encoder and decoder circuits
- increased latency in the decoder
- high internal-clock frequencies in the circuitry
- length of consecutive coded bits without transitions.

Clock Recovery

For NRZ encoding, the recommended way to recover the clock at high bit rates (greater than 125 Mb/s) is to use a SAW (surface acoustic wave) filter in the clock recovery circuit.[3,4]

Such filters, which are typically constructed of quartz, consist of two sets of interdigitated electrodes. One set generates a surface acoustic wave, and the other set receives the wave and reconverts it into an electrical signal.

The electrodes typically contain hundreds of fingers (spaced to match the sound velocity divided by the bit rate). Therefore, the transient response of SAW traversal filters is a nearly sinusoidal wave with a smooth, yet time-limited envelope. This received envelope peaks several hundred cycles after the input transient and is several hundred cycles broad.

In operation, the output from the SAW filter is the convolution of the stream of impulses (each a transition in the received data stream) and of the transient response for one impulse. Generally, this output will be a sine wave whose amplitude depends on the number of transitions that were received during the several-hundred-cycle interval of the transient response's temporal width. Under the usual operating conditions, the number of transitions in several hundred cycles will seldom vary more than 5 to 10 percent.

If we assume worst-case scenarios, however, we can suppose that the symbols (code words) with the fewest transitions and with the most transitions have been used. The ratio of these numbers of transitions will show the maximum range in amplitude of the SAW filter's electrical output. Of course, additional variation could occur if there is noticeable jitter (timing variations) on the incoming data stream.

We want to contrast our concerns with those about multiplexed communications circuits that use NRZ without block codes,[2] where we can reasonably assume that the patterns of 0s and 1s follow Gaussian statistics.

In summary, for clock recovery, it is important to consider the number of transitions that each symbol might contain in the block codes under consideration.

Framing Recovery

Recovery of clock is not enough to start accepting data. For data communications that use block codes, we still need to establish frame synchronization, i.e., to determine the boundaries between blocks.

The typical *raw* BER of 10^{-9} suggests that we will rarely detect errors in decoding blocks; finding clusters of blocks with errors nearly always means that the framing is misaligned. Either a hardware decision circuit or more

Table II. Codes of Length 4, Weight 2

Code word
0011
0101
0110
1001
1010
1100

flexible, software control at higher levels can be used to decide when to start hunting to realign the frame.

Because we expect very high signaling rates, the most practical technique is to drop one pulse from the counter that is used to convert the bit-clocking signal into frame clocking and, thus, stretch the frame clock cycle one extra bit interval. In this way, a new frame alignment is tried. If it doesn't work, we can slip another bit interval (and so on) until frame alignment appears successful.

There are two main concerns with this procedure. First, we do not want to assume frame misalignment erroneously because of isolated bit errors. Second, we want to reestablish alignment quickly.

A third possible concern is to provide a way to communicate the current status (framing established, hunting) to higher levels. This concern is addressed automatically if the higher levels of software handle search decisions.

Bylanski and Ingram[5] describe algorithms for the hunt decision process. One method is to use an up-down counter that is initially set to zero. Each time a block is decoded and an error is detected, the counter is incremented by x. Every time a block is decoded without detectable errors, the count is decremented by y. When the count exceeds some value z, alignment is considered lost and hunting is started or continued.

We also consider scenarios where *idle* characters precede messages or where the frequent gaps between messages are filled in with idle characters. For these, we should choose an idle character that is not only rich in transitions (for bit clock recovery) but whose cyclic permutations never provide an acceptable code word except when properly aligned. If the BER is quite low, such as less than 10^{-9}, then the probability of a double-bit error is so low that we can assume frame misalignment caused it. Thus, we can hunt every time a double-bit (or worse) error is seen.

Constant-Weight Block Codes

Now let us consider some specific constant-weight block codes. We will start with block length 2 and progress to greater lengths, complexity, and features.

Manchester Code. The simplest example of a 50-percent weight block code appears to be the Manchester or *biphase level* code[2] (Table I). Often, it is not thought of as a block code because of its manifest simplicity.

As Table I clearly shows, there are only two ways that we can have a single 1 in a coded block of two bits. Each block has at least one transition, but there will be additional transitions whenever the data stream does not alternate bit values. It is relatively easy to acquire and maintain clocking, and there is no dc component.

Because there are no spare symbols for signaling, one usually uses a *code violation* that consists of an absence of transitions at midcode. This code violation does have a dc component and abnormally few transitions.

Length 4, Weight 2. Table II lists all code words of weight 2. With six symbols, we can use four for coding two bits of data and still have two *spare* symbols.

Thus the coding format efficiency is one half just as it is for the simpler Manchester code, but now we can use the spare pair of symbols for control.

Length 6, Weight 3. Table III lists all code words of weight 3. We can use 16 of the 20 symbols for coding four bits of data and still have four spare symbols.

We could use the spare symbols for control or elect not to use them at all. For example, we could remove the words 000111, 001110, 011100, and 111000 that have fewer than average intraword transitions.

Table III. Codes of Length 6, Weight 3

Code word
000111
001011*
001101
001110
010011
010101
010110*
011001
011010
011100
100011
100101*
100110
101001
101010
101100
110001
110010*
110100
111000

NOTE: Using only those marked with an asterisk (*) provides a code set of Hamming distance 4.

Because we can encode four data bits for six symbol bits, we have a two-thirds coding format efficiency. Recently, Proteon Inc. and the University of Wisconsin[6] jointly developed an 80-Mb/s fiber optic ring. They use this type of block coding because of its higher coding efficiency and balanced 0 and 1 content.

For all block codes with 50-percent weight, we can detect single-bit errors by noting any discrepancies in the weight.

If we use only the four words marked with an asterisk (*) in Table III, we have a code with Hamming distance 4. *Distance 4* means that at least four bit inversions are needed to convert any word in the code set into another word, so we can correct any single-bit error and detect all two-bit errors. Thus, with single-bit error correction, we can use a length 2 to length 6 block code.

Incidentally, only two code symbols would have been possible with a distance of 4 for a block length of 4. If we use forward error correction with a block length of 4, we could encode only one bit of data per block.

Length 8, Weight 4. Rather than list all the ways to have four 1s in a symbol of length 8, we point out that there are 70 ways to do it: $(8 \cdot 7 \cdot 6 \cdot 5)/(4 \cdot 3 \cdot 2 \cdot 1) = 8!/(4! \cdot 4!)$. Because $70 = 64 + 6$, we can use this set of words to encode six bits of data with six spare symbols.

If we use single-bit, forward error correction with a block length of 8, we are limited to a symbol set of only 14 elements.[7] Thus, we have only three bits of data and six spare symbols.

Lengths 10, 12, 14, and 16. Because we have been so detailed with the shorter lengths, here we merely summarize the results. Table IV gives the maximum number of code words in a half-weight code of a given length.

We don't know the maximum number of words in single- and double-error correcting codes of constant weight for longer block lengths. Therefore, we used the lower bounds that Graham and Sloane[7] presented.

Length 16 with Error Correction

A linear block code of length 16, known as the *extended Hamming code*,[8] can be used to encode 11 bits of data.

This is not a constant-weight coding scheme. But we will show how to use it as the basis for encoding more than nine bits of data with 50-percent weight and transition enrichment to aid in clock recovery. Also, we show by example how to improve the effective BER significantly, so it is acceptable in ultra-reliable applications.

Extended Hamming Code. One way to create the extended Hamming code of length 16 from 11 bits of data is to use a generator matrix, c_{ij}, where $0 \leqslant i \leqslant 10$ and $0 \leqslant j \leqslant 15$. Table V shows such a matrix; we added two extra columns, p and q, that will be discussed shortly.

Table IV. Maximum Code Words for Lengths 10, 12, 14, and 16

Length	No error correction	Single error	Double error
10	$252 = 2^7 + 124$	$36 = 2^5 + 4$	$6 = 2^2 + 2$
12	$924 = 2^9 + 412$	$132 = 2^7 + 4$	$6 = 2^2 + 2$
14	$3432 = 2^{11} + 1384$	$\geqslant 316 = 2^8 + 60$	$\geqslant 42 = 2^5 + 9$
16	$12870 = 2^{13} + 4678$	$\geqslant 1164 = 2^{10} + 140$	$\geqslant 120 = 2^6 + 56$

Table V. Matrix for Extended Hamming Code

p	q	Generator matrix
0	3	1001100000000001
1	5	1010100000000010
2	6	1011000000000100
3	7	0011100000001000
4	9	1100100000010000
5	10	1101000000100000
6	11	0101100001000000
7	12	1110000010000000
8	13	0110100100000000
9	14	0111001000000000
10	15	1111110000000000

In the matrix (Table V), number the columns from right to left and the rows from top to bottom; for example, $c_{ii} = 1$, for $0 \leqslant i \leqslant 10$. If our 11 bits of data are described by $e_0, e_1, \ldots, e_{10}$ and consist of 1s and 0s, then the corresponding code word $b_0, b_1, \ldots, b_{15}$ is given by

$$b_j = \sum_i e_i c_{ij}$$

where the subscript j takes on the 16 different values 0 through 15.

The summation over all values of i (0 through 10) is taken modulo 2. That is, an even number of 1s added together will yield 0, and an odd number of 1s yield 1.

The appealing aspects of this encoding is that $b_i = e_i$, for all i in the range 0 through 10. We may think of the code word's remaining bits $b_{11}, \ldots, b_{15}$ as a variety of even-parity check bits. Specifically, we can compute the following five parity checks:

$$\sum_j b_j = 0 \qquad [0]$$

$$b_{11} + b_{10} + b_8 + b_6 + b_4 + b_3 + b_1 + b_0 = 0 \qquad [1]$$

$$b_{12} + b_{10} + b_9 + b_6 + b_5 + b_3 + b_2 + b_0 = 0 \qquad [2]$$

$$b_{13} + b_{10} + b_9 + b_8 + b_7 + b_3 + b_2 + b_1 = 0 \qquad [4]$$

$$b_{14} + b_{10} + b_9 + b_8 + b_7 + b_6 + b_5 + b_4 = 0 \qquad [8]$$

If any of these parity checks fail at the receiver, we have one or more transmission errors.

Correction Strategy. A correction strategy might take the following form.

If the parity check above labeled [0] fails, we have an odd number of bit errors. (Let's hope for only one!) Otherwise, we have either no errors (great!) or an even number of errors that we cannot fix with certainty.

Assume we have one bit error. First, we examine all the parity tests and add the values (the 0, 1, 2, 4, or 8 in the square brackets) for all the tests that fail. Then, we look up the resulting sum in column q, Table V. If we find it, then the corresponding number in column p of the table specifies the position of the data bit that needs to be corrected. If we don't find the sum in column q, then the error was in the check bits.

We constructed column p to list the possible data positions. In column q, we listed—in ascending order—the numbers whose odd parity, 5-bit binary representation is at least a distance of 2 from 00000.

Here is an example. Suppose that we have the code word 1110100010001011. Also suppose that, for some reason (a cosmic ray hit, perhaps), one bit—say b_1—is converted so that the received code word is 1110100010001001.

Of course, we can see that we have an error, because the weight is now 7 instead of 8; i.e., test [0] fails. We see that parity tests [1] and [4] fail as well.

Adding the values in the square brackets produces the value 5. When we examine column q (Table V), we find that 5 corresponds to the 1 position (from column p) and conclude that the error is in code bit b_1. So, we flip this bit back to 0 and restore the code word to its original value, in spite of the single-bit error that occurred.

Constant-Weight Scheme. For 50-percent weight, we cannot use all (2048) code words in the extended Hamming code described above. If we take just the code words of weight 8, we still have 870 symbols and can still apply single-bit error correction as described above. We have enough symbols for nine bits of data (512) with many spare symbols that could be used for control (idle, start, stop, separator, etc.).

The decoding function done purely by look-up doesn't look attractive. It would require 16 bits of address range, which is not easy to obtain at high throughput. If direct look-up is acceptable for block codes of length 16, then there is a code set of constant weight that supports 10 bits of data.[9]

For a particular application that needs nine bits of data, we elected to try simplifying the hardware requirements at the expense of discarding some spare symbols. To *thin out* the symbol set, we exclude one symbol from each pair of symbols with duplicate rightmost 10 bits, as well as symbols that have relatively few (less than seven) internal transitions. If we look at the symbol's representation as a sequence of 1s and 0s, an internal transition means: any location where a 0 follows a 1 or a 1 follows a 0.

Even after this thinning out, we still have 524 code words, which lets us transmit nine bits of data and leaves 12 spare symbols for control.

In decoding, we take the received block of 16 bits and go through the error correcting procedure described above. If this procedure succeeds, i.e., $b_0, b_1, \ldots, b_{15}$ is the code word that was transmitted, then $b_0, b_1, \ldots, b_9$ is enough to identify it as one of the 524 selected code words. Thus, we can use a ROM (read-only memory) with 10-bit addressing to regenerate the original data or control symbol.

The reduction in the number of symbols (from 870 down to 524) results from two requirements that we imposed.

- Because we ignore bit b_{10}, we cannot use both of the symbols that might be available if we examined it. Where both symbols had weight 8, we chose to retain the one with more internal transitions (to provide the best possible clock-recovery signal). Even with this stipulation, we have 690 symbols.
- Because we still had more spare symbols than appeared necessary for control, we removed symbols with six or less internal transitions. That left us 524 usable symbols with seven or more transitions. The average number of internal transitions for a random selection of symbols from this final list is 8.85.

Exactly 12 code words have 13 internal transitions, the maximum number of transitions encountered. We have arranged the correspondences between data and code words so that these 12 code words are identified with the 12 spares.

In particular, the spares 1001010101011010 and 0110101010100101 would be excellent idle characters. Not only do we have 14 transitions per block when either is used repetitively, but there are no nontrivial cyclic permutations of these two characters that are valid code words.

BER Improvement

To conclude our treatment of this block code, we show how forward error correction can lower the effective bit-error rate.

Suppose that the physical data link has a raw BER of 10^{-9}. Further suppose that the error rate is symmetric (the accidental conversion of a 1 to a 0 is the same as the conversion of a 0 to a 1), and the errors are uncorrelated.

The probability that a particular word of length 16 has a single-bit error is $1.6 \cdot 10^{-8}$. These errors are correctable and, therefore, do not cause any data loss. However, the chance of a double error in a particular block

Table VI. Matrix for Extended Golay Code

Generator matrix
110000000000011011100010
101000000000001101110001
100100000000010110111000
100010000000001011011100
100001000000000101101110
100000100000000010110111
100000010000010001011011
100000001000011000101101
100000000100011100010110
100000000010001110001011
100000000001000111000101
000000000000111111111111

is $16 \cdot (15/2!) \cdot 10^{-18}$. Because double errors can be detected but not corrected, the probability that any 9-bit data symbol is lost is $1.2 \cdot 10^{-16}$.

On the other hand, we would need a raw BER of about $1.3 \cdot 10^{-17}$ to achieve the same low loss of data characters without forward error correction.

We should emphasize that the effective error rate depends, in detail, on definitions and the degree of correlation between error pairs. We can ascertain this behavior only from extensive experience in field operation of such data links; measuring such low error rates on a short time scale in a typical laboratory environment seems impractical.

Because this coding method is being implemented in a high-speed, ultra-high-reliability data ring, we expect to be able to confirm the exceptionally low BER in the field.

Length 24 with Multiple Error Correction

The extended Golay code[8] is an important code of length 24 and minimum distance of 8 (three errors correctable, four detectable).

Large-scale-integration chips that operate at up to 1.5 Mhz currently exist for this code.[10] However, the code's complexity requires three printed-circuit boards of discrete catalog components. One board is for encoding, and two are for decoding with error detection and correction for operation at speeds of 140 Mb/s.

The extended Golay code is a linear code; Table VI presents a suitable generator matrix.

Interestingly, the weights of the code elements are limited to:

Weight	0	8	12	16	24
Number of words	1	759	2576	759	1

Thus, we can apply our trick of limiting our use to those code words of only one weight. Because 2576 code words have weight 12, we can encode 11 bits of data and still have 528 spare symbols. However, so many symbols are involved that we have not investigated the distribution of internal transitions.

The complexity of decoding this code (with current technology) may make it impractical for most high-speed optical communications. But we conclude that this code could encode 11 bits of data per block with almost 50-percent encoding efficiency and correct up to three errors in any one block.

Summary

We have reviewed and detailed some half-weight block codes that provide freedom from dc-baseline wander in ac-coupled optical data links. Additional features of the more complex codes are control signaling, error correction and detection, and enrichment of transitions to simplify clock recovery. We have explored questions of decoding, as well as encoding.

Practical forward-error correction is possible in codes as long as 16 and 24 bits. In particular, the 16-bit block code we described will be the basis for an ultra-reliable, high-speed data ring.

Acknowledgments

It is our pleasure to acknowledge helpful discussions with J. E. Morris, D. C. Farley, and N. J. A. Sloane.

References

1. J. E. Morris and R. D. Brooks, "Long Wavelength Compact Data Link Modules for 40 to 200 Mbit/s Applications," *Conference on Optical Fiber Communications*, February 1985, pp. 38-39.
2. D. J. Morris, *Pulse Code Formats For Fiber Optical Data Communication, Basic Principles and Applications*, Marcel Dekker, New York, 1983, pp. 106, 158.
3. R. L. Rosenberg, D. G. Gross, P. R. Trischitta, D. A. Fishman, and C.B. Armitage, "Optical Fibered Repeatered Transmission Systems utilizing SAW Filters," *IEEE Transactions on Sonics and Ultrasonics*, Vol. U-30, May 1986, pp. 119-129.
4. R. L. Rosenberg, C. Chamas, and D. A. Fishman, "Timing Recovery in SAW Travesal Filters in the Regenerators of Undersea Long-Haul Fiber Transmission Systems," *IEEE Journal of Lightwave Technology*, Vol. LT-12, December 1984, pp. 917-925.
5. P. Bylanski and D. G. W. Ingram, *Digital Transmission Systems*, Second Edition, IEE Telecommunications Series 4, Peter Peregrinus Ltd., Southgate House, Stevenage, Herts., England, 1980.
6. "80-Mbit/s fiber-optic ring boosts detection and correction," *Electronic Design*, Vol. 33, No. 7, March 21, 1985, p. 36.
7. R. L. Graham and N. J. A. Sloane, "Lower Bounds for Constant Weight Codes," *IEEE Transactions on Information Theory*, Vol. T-16, No. 1, January 1980, pp. 37-43.
8. F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error Correcting Codes*, Vol. 16, North-Holland Mathematical Series, North-Holland Publishing Company, New York, 1977.
9. A. E. Brouwer, "A Few More Constant Weight Codes," *IEEE Transactions on Information Theory*, Vol. IT-26, 1980, p. 366.
10. George D. Dolan, "Performance Test Report for 24-12-03 Error Correcting System," Space Research Technology, Inc. (17317 El Camino Real, Houston, TX 77058, 713-480-3086).

Biographies (continued)

the California Institute of Technology and a Ph.D. from Massachusetts Institute of Technology, all in mathematics. Mr. Sangani, who joined AT&T in 1978, works on the ring-based, packet switching data network that will use the block code described in this paper. He has a B.S. from Bombay University, an M.S. from Texas Agricultural and Industrial University, and a Ph.D. from Texas Technical Institute, all in electrical engineering.

(Manuscript received February 10, 1986)