$$\sum_{d|n} dM(w/(n/d),d) = \text{total number of } n\text{-tuples of weight } w. \quad (6)$$

However the right side can be determined. For each combination of w positions in an n-tuple, there are $(r-1)^w$ possible choices which produce the same weight w; there are $\binom{n}{w}$ (binomial coefficient) such combinations. Equation (6) becomes

$$\sum_{d|n} dM(w/(n/d),d) = \binom{n}{w}(r-1)^w. \quad (7)$$

The Mobius inversion formula [1] can be applied and we have

$$M(w,n) = \frac{1}{n}\sum_{d|(n,w)} \mu(d)\binom{n/d}{w/d}(r-1)^{w/d}. \quad (8)$$

The symbol (n,w) denotes the greatest common divisor of the integers n and w and $\mu(d)$ is the Mobius function [1].

We may combine (5) and (8) and incorporate the properties of the greatest common divisor to arrive at a single expression for $T_{n,r}(w)$.

$$T_{n,r}(w) = \sum_{x|(n,w)}\left\{\frac{x}{n}\sum_{y|[(n,w)/x]} \mu(y)(r-1)^{w/xy}\binom{n/xy}{w/xy}\right\}. \quad (9)$$

The weight distribution for binary CPCW up to length 14 is presented in Table II.[2]

REFERENCES

[1] M. Hall, Jr., *Combinatorial Theory*. Waltham, Mass.: Blaisdell, 1967, pp. 11–12.
[2] S. W. Golomb, *Shift Register Sequences*. San Francisco, Calif.: Holden-Day, 1967, pp. 118–122.
[3] I. N. Herstein, *Topics in Algebra*. Waltham, Mass.: Blaisdell, 1964, p. 65.

[2] This table was suggested and provided by a reviewer; we gratefully acknowledge his assistance.

Syndrome Decoding of Convolutional Codes

J. P. M. SCHALKWIJK AND A. J. VINCK

Abstract—**The classical Viterbi decoder recursively finds the trellis path (code word) closest to the received data. Given the received data, the syndrome decoder first forms a syndrome, instead. A recursive algorithm like Viterbi's is used to determine the noise sequence of minimum Hamming weight that can be a possible cause of this syndrome. Given the estimate of the noise sequence, one derives an estimate of the original data sequence. While the bit error probability of the syndrome decoder is no different from that of the classical Viterbi decoder, the syndrome decoder can be implemented using a read only memory (ROM), thus obtaining a considerable saving in hardware.**

I. INTRODUCTION

The principle of syndrome decoding of convolutional codes will be explained using the binary code generated by the encoder of Fig. 1. The additions in Fig. 1 are modulo-2, and all binary sequences $b_0,b_1,b_2,\cdots$ are represented as power series $b(\alpha) = b_0 + b_1\alpha + b_2\alpha^2 + \cdots$. The encoder has connection polynomials $C_1(\alpha) = 1 + \alpha^2$ and $C_2(\alpha) = 1 + \alpha + \alpha^2$. Hence, the encoder outputs are $C_1(\alpha)x(\alpha)$ and $C_2(\alpha)x(\alpha)$. The syndrome $z(\alpha)$ only depends on $n_1(\alpha)$ and $n_2(\alpha)$, i.e., not on the data sequence $x(\alpha)$, for

$$\begin{aligned} z(\alpha) &= C_2(\alpha)[C_1(\alpha)x(\alpha) + n_1(\alpha)] + C_1(\alpha)[C_2(\alpha)x(\alpha) + n_2(\alpha)] \\ &= C_2(\alpha)n_1(\alpha) + C_1(\alpha)n_2(\alpha). \end{aligned} \quad (1)$$

Having formed the syndrome $z(\alpha)$, the next section describes a recursive algorithm like Viterbi's [1] to determine from the syndrome $z(\alpha)$ the noise sequence pair $[\hat{n}_1(\alpha),\hat{n}_2(\alpha)]$ of minimum Hamming weight that can be a possible cause of this syndrome.

Given the estimate $[\hat{n}_1(\alpha),\hat{n}_2(\alpha)]$ of the noise sequence pair, one derives an estimate $\hat{x}(\alpha)$ of the original data sequence $x(\alpha)$ as follows. For a noncatastrophic code, $C_1(\alpha)$ and $C_2(\alpha)$ are relatively prime. Hence, by Euclid's algorithm [2] there exist polynomials $D_1(\alpha)$ and $D_2(\alpha)$ such that $D_1(\alpha)C_1(\alpha) + D_2(\alpha)C_2(\alpha) = 1$. For the example of Fig. 1, we have $D_1(\alpha) = 1 + \alpha$, $D_2(\alpha) = \alpha$. We receive the sequence pair

$$y_i(\alpha) = C_i(\alpha)x(\alpha) + n_i(\alpha), \quad i = 1,2, \quad (2)$$

and form the estimate

$$\hat{x}(\alpha) = D_1(\alpha)[y_1(\alpha) + \hat{n}_1(\alpha)] + D_2(\alpha)[y_2(\alpha) + \hat{n}_2(\alpha)]. \quad (3)$$

Note that if the noise sequence estimate $[\hat{n}_1(\alpha),\hat{n}_2(\alpha)]$ is correct we have

$$\begin{aligned} y_i(\alpha) + \hat{n}_i(\alpha) &= C_i(\alpha)x(\alpha) + n_i(\alpha) + \hat{n}_i(\alpha) \\ &= C_i(\alpha)x(\alpha), \quad i = 1,2, \end{aligned}$$

and, hence,

$$\hat{x}(\alpha) = D_1(\alpha)C_1(\alpha)x(\alpha) + D_2(\alpha)C_2(\alpha)x(\alpha) = x(\alpha).$$

Note that (3) for the estimate $\hat{x}(\alpha)$ of the data sequence $x(\alpha)$ can be rewritten as

$$\hat{x}(\alpha) = [D_1(\alpha)y_1(\alpha) + D_2(\alpha)y_2(\alpha)] + \omega(\alpha), \quad (4)$$

where

$$\omega(\alpha) = D_1(\alpha)\hat{n}_1(\alpha) + D_2(\alpha)\hat{n}_2(\alpha). \quad (5)$$

The term in square brackets in (4) can be computed directly from the received data using very simple circuitry. As there is no need to distinguish between pairs $[\hat{n}_1(\alpha),\hat{n}_2(\alpha)]$ and $[\hat{n}_1(\alpha),\hat{n}_2(\alpha)]'$ that lead to the same value for $\omega(\alpha)$ in (5), the algorithm to be discussed in the next section computes $\omega(\alpha)$ directly.

II. ALGORITHM

In Fig. 2 we have redrawn the syndrome former. As, according to (1), the syndrome $z(\alpha)$ only depends on the noise pair $[n_1(\alpha), n_2(\alpha)]$ all other binary sequences have been omitted from Fig. 2. For minimum distance decoding we are now presented with the following problem. Given the syndrome $z(\alpha)$, determine the noise pair $[\hat{n}_1(\alpha),\hat{n}_2(\alpha)]$ of minimum Hamming weight that can be a cause of this syndrome.

At first sight the state diagram of the syndrome former of Fig. 2 has $2^4 = 16$ states and, hence, is more complicated than the state diagram used to implement the classical Viterbi decoder [1] that has only $2^2 = 4$ states. However, a closer inspection of Fig. 2 reveals that the syndrome former has also $2^2 = 4$ states. In general, for an encoder with ν memory stages, the syndrome former has 2^ν states just like the state diagram used to implement the classical Viterbi decoder. This can be seen as follows. Writing

$$[n_1(\alpha),n_2(\alpha)] = [n_{10},n_{20}] + [n_{11},n_{21}]\alpha + [n_{12},n_{22}]\alpha^2 + \cdots, \quad (6)$$

we can replace the first coefficient pair $[n_{10},n_{20}]$ by its modulo-2 complement $[\bar{n}_{10},\bar{n}_{20}]$ without affecting z_0. However, the complementation of $[n_{10},n_{20}]$ may affect $z_1,z_2,\cdots,z_{\nu-1}$. Nonetheless, there are two complementary choices of $[n_{11},n_{21}]$ that give us the required value of z_1, etc. Hence, each of the $2^{2\nu}$ possible memory states in Fig. 2 is equivalent to $2^\nu - 1$ others as far as $z(\alpha)$ is concerned, leaving $2^{2\nu}/2^\nu = 2^\nu$ different states. Fig. 3 gives the state diagram of the syndrome former of Fig. 2. Solid transitions in Fig. 3 correspond to

Paper approved by the Associate Editor for European Contributions of the IEEE Communications Society for publication without oral presentation. Manuscript received November 12, 1974; revised February 8, 1975. This work was supported in part by the U. S. Navy under Contract N00123-75-C-0557.

The authors are with the Department of Electrical Engineering, University of Eindhoven, Eindhoven, The Netherlands.

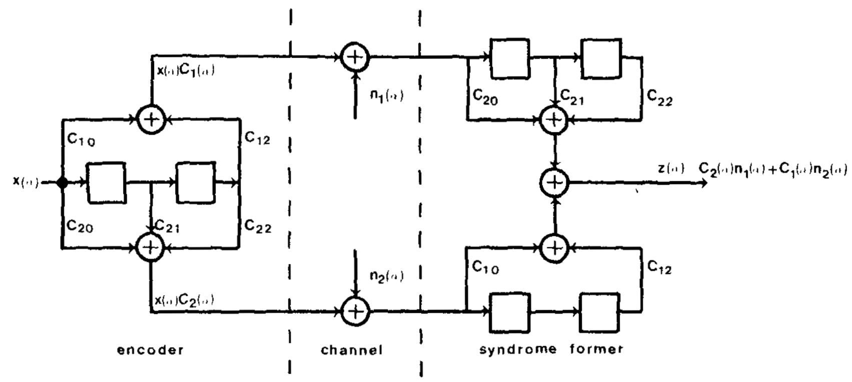

Fig. 1. Encoding and syndrome forming for a $R = \frac{1}{2}$ code.

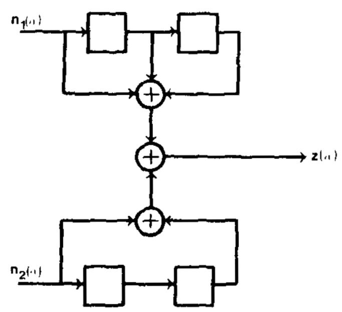

Fig. 2. Syndrome former.

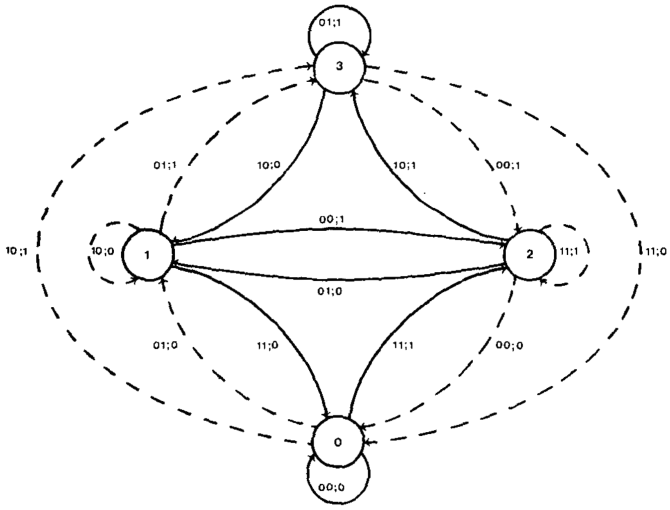

Fig. 3. State diagram of syndrome former.

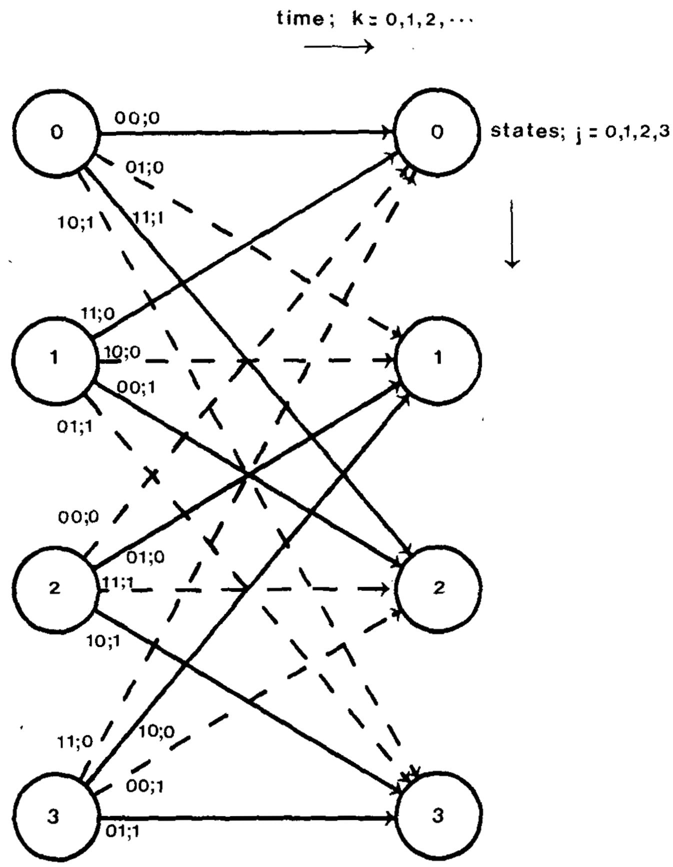

Fig. 4. kth section of the trellis diagram, $k = 0,1,2,\cdots$.

z_k 0 = and dashed transitions to $z_k = 1$, $k = 0,1,2,\cdots$. Next to each transition one finds the value of $\hat{n}_{1k},\hat{n}_{2k};\omega_k$, $k = 0,1,2,\cdots$. Fig. 4 gives the kth, $k = 0,1,2,\cdots$, section of the trellis diagram that corresponds to the state diagram of Fig. 3. The algorithm that determines $\omega(\alpha)$ according to (5) now operates as follows. With each state in Fig. 4 we associate a metric $M_j(k)$, $j = 0,1,2,3$, $k = 0,1,2,\cdots$, that equals the minimum Hamming weight of a path, $[\hat{n}_1(\alpha),\hat{n}_2(\alpha)]^{(j)}$, leading from state $j = 0$ at time $k = 0$ to that particular state. This path has a solid or dashed lth branch, $0 \leq l \leq k-1$, according to whether $z_l = 0$ or $z_l = 1$, respectively. The metric $M_j(k+1)$ at time $k+1$ can be determined recursively, i.e.,

$$M_0(k+1) = \bar{z}_k \min [M_0(k), M_1(k)+2] + z_k \min [M_2(k), M_3(k)+2] \quad (7a)$$

$$M_1(k+1) = \bar{z}_k \min [M_2(k)+1, M_3(k)+1] + z_k \min [M_0(k)+1, M_1(k)+1] \quad (7b)$$

$$M_2(k+1) = \bar{z}_k \min [M_0(k)+2, M_1(k)] + z_k \min [M_2(k)+2, M_3(k)] \quad (7c)$$

$$M_3(k+1) = \bar{z}_k \min [M_2(k)+1, M_3(k)+1] + z_k \min [M_0(k)+1, M_1(k)+1]. \quad (7d)$$

Given the value of z_k, i.e., $z_k = 0$ or $z_k = 1$, each $(k+1)$ state can be reached from two k states. For each of these two k states add to the metric the Hamming weight of the transition, i.e., of $[\hat{n}_{1k},\hat{n}_{2k}]$, to the particular $(k+1)$ state. The minimum of the two values thus obtained is $M_j(k+1)$. The transition associated with the minimum value is called the "survivor." In case of a tie, choose the survivor at random among the two candidates. The survivor for $(k+1)$ state $j = 0,1,2,3$ can be specified by the associated k state $j_j(k) = 0,1,2,3$. Going back from a $(k+1)$ state each time choosing the survivor we obtain the path $[\hat{n}_1(\alpha),\hat{n}_2(\alpha)]^{(j)}$, $j = 0,1,2,3$, of minimum Hamming weight leading to that particular $(k+1)$ state. The coefficients $\omega_{k-D+1}^{(j)},\omega_{k-D+2}^{(j)},\cdots,\omega_k^{(j)}$, associated with the path $[\hat{n}_1(\alpha),\hat{n}_2(\alpha)]^{(j)}$

TABLE I
METRIC TRANSITIONS

Row Number	Old Metrics	$z_k = 0$ Survivors	$z_k = 0$ New Metrics	$z_k = 1$ Survivors	$z_k = 1$ New Metrics
0	0000	0(2,3) 1 (2,3)	0101	2 (0,1)3(0,1)	0101
1	0101	0 2 1 2	0111	2 0 3 0	0111
2	0111	0(2,3) 1 (2,3)	0212	2 0 3 0	0000
3	0212	0 2 (0,1) 2	0222	2 0 3 0	0010
4	0222	0(2,3)(0,1)(2,3)	0323	2 0 3 0	1010
5	0010	0 3 1 3	0101	2 (0,1)3(0,1)	1101
6	0323	0 2 0 2	0323	2 0 3 0	1020
7	1010	0 3 1 3	1101	2 1 3 1	1101
8	1101	0 2 1 2	0000	2 (0,1)3(0,1)	0212
9	1020	0 3 1 3	1101	(2,3) 1 3 1	2101
10	2101	0 2 1 2	1000	2 1 3 1	0212
11	1000	0(2,3) 1 (2,3)	1101	2 1 3 1	0101

TABLE II
CONTENTS OF THE ROM

Old ROM Address	$z_k = 0$ Survivors	$z_k = 0$ New ROM Address	$z_k = 0$ Index j_m	$z_k = 1$ Survivors	$z_k = 1$ New ROM Address	$z_k = 1$ Index j_m
0	0(2,3) 1 (2,3)	1	(0,2)	2 (0,1)3(0,1)	1	(0,2)
1	0 2 1 2	2	0	2 0 3 0	2	0
2	0(2,3) 1 (2,3)	3	0	2 0 3 0	0	(0,1,2,3)
3	0 2 (0,1) 2	4	0	2 0 3 0	5	(0,1,3)
4	0(2,3)(0,1)(2,3)	6	0	2 0 3 0	7	(1,3)
5	0 3 1 3	1	(0,2)	2 (0,1)3(0,1)	8	2
6	0 2 0 2	6	0	2 0 3 0	9	(1,3)
7	0 3 1 3	8	2	2 1 3 1	8	2
8	0 2 1 2	0	(0,1,2,3)	2 (0,1)3(0,1)	3	0
9	0 3 1 3	8	2	(2,3) 1 3 1	10	2
10	0 2 1 2	11	(1,2,3)	2 1 3 1	3	0
11	0(2,3) 1 (2,3)	8	2	2 1 3 1	1	(0,2)

of minimum Hamming weight, are stored in the path register for the jth state, $j = 0,1,2,3$. If

$$M_{j_m}(k+1) = \min_j M_j(k+1), \tag{8}$$

we set

$$\omega_{k-D+1} = \omega_{k-D+1}^{(j_m)}. \tag{9}$$

If more than one j satisfies (8), we select j_m arbitrarily among the candidates. The longer the path register length D, the smaller the resulting bit error probability P_b. Increasing D beyond $5(\nu + 1)$ does not lead to an appreciable further decrease in P_b. We have done detailed calculations concerning the relationship between D and P_b, which will be published shortly. The next section is concerned with the practical implementation of the syndrome decoder.

III. IMPLEMENTATION

Using (7) we construct Table I. The first column just numbers the rows of the table. The second column lists all possible metric combinations $M_0(k),M_1(k),M_2(k),M_3(k)$ at time k. As only the differences between the metrics of a quadruple matter, we subtract from each member of a quadruple of metrics the minimum value of the quadruple, i.e., all quadruples of metrics in Table I have one or more zeros. Columns 3 and 4 apply to the case that $z_k = 0$ and columns 5 and 6 to the case that $z_k = 1$. Columns 3 and 5 list the survivors $j_0(k),j_1(k),j_2(k),j_3(k)$, and columns 4 and 6 the new metrics $M_0(k+1)$, $M_1(k+1),M_2(k+1),M_3(k+1)$ as given by (7). If there is a choice of survivors the candidates are placed within parentheses in the survivor columns.

Table I contains more information than is necessary for the actual implementation of the syndrome decoder. As explained in Section II knowledge of the successive survivors for each state, together with the index j_m of the minimum within each new quadruple of metrics, suffices to determine the key sequence $\omega(\alpha)$ of (5). Hence, we omit the quadruples of metrics from Table I and store the resulting Table II in a read only memory (ROM). The operation of the core part of the syndrome decoder can now be explained using the block diagram of Fig. 5. Assume that at time k the ROM address register AR contains (AR) = 7 and the ROM data register DR contains (DR) = (ROM,7). Let $z_k = 1$. Note, see Fig. 4, that $\omega_k^{(0)} = \omega_k^{(1)} = 0, \omega_k^{(2)} = \omega_k^{(3)} = 1$ independent of $k = 0,1,2,\cdots$, i.e., always fill the left-most stages of the four path registers, $PR_0[0:0],PR_1[0:0]$, $PR_2[0:0],PR_3[0:0]$, with 0011, respectively. Then according to row 7 and column 5 of Table II, or according to the contents (DR) of the DR, replace

$$PR_0[1:D-1] \leftarrow \text{CONTENTS } PR_2[1:D-1]$$

$$PR_1[1:D-1] \leftarrow \text{CONTENTS } PR_1[1:D-1]$$

$$PR_2[1:D-1] \leftarrow \text{CONTENTS } PR_3[1:D-1]$$

$$PR_3[1:D-1] \leftarrow \text{CONTENTS } PR_1[1:D-1].$$

The right-most digit, $PR_0[D-1:D-1],PR_1[D-1:D-1]$, $PR_2[D-1:D-1],PR_3[D-1:D-1]$, of all four path registers is fed to the selector, see Fig. 5, that determines ω_{k-D+1} according to (9) using the entry in row 7 and column 7, i.e., $j_m = 2$, of Table II which can also be found in the DR. To complete the kth cycle of the syndrome decoder, set (AR) = 8 and read DR ← (ROM,8).

The ROM decoder for the code of Fig. 1 has been realized in hardware using path registers of length $D = 11$. The solid line in Fig. 6 gives the measured bit error probability P_b as a function of the transi-

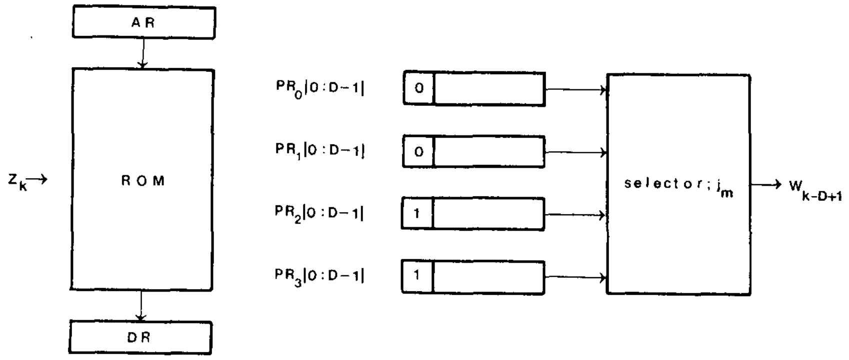

Fig. 5. Block diagram of the core of the syndrome decoder.

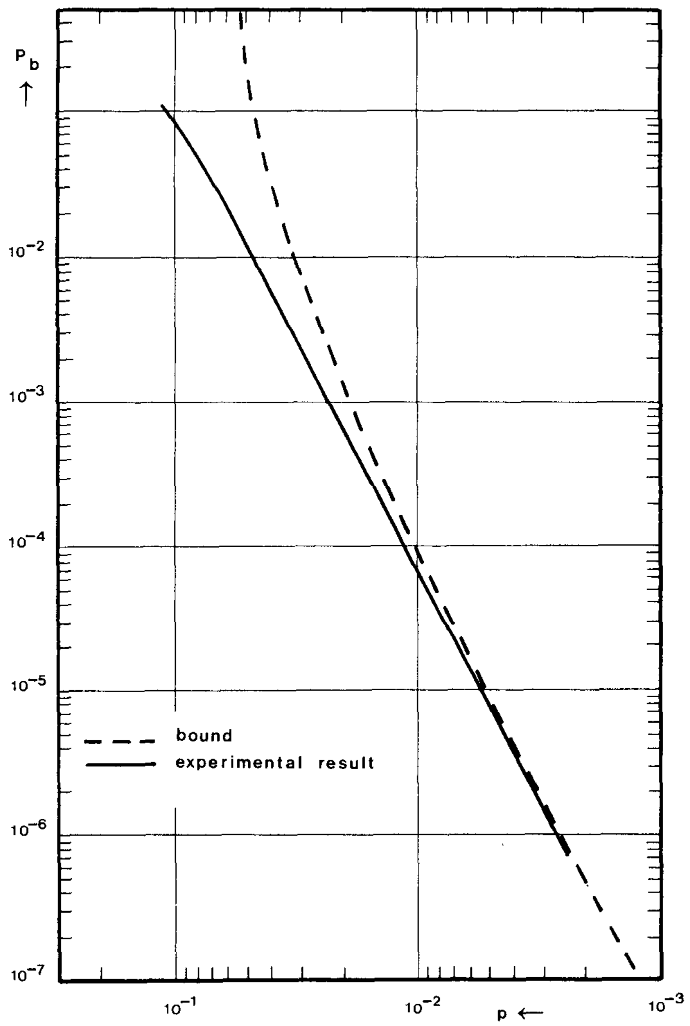

Fig. 6. Bit error rate P_b versus channel transition probability p.

tion probability p of the binary symmetric channel. The dashed curve is an upper bound [3] on the bit error probability P_b.

IV. CONCLUSIONS

This correspondence describes a syndrome decoder for convolutional codes. The recursive algorithm that forms the core part of the decoder can be implemented with a ROM.

As to a comparison with the classical Viterbi decoder, which has also been realized using a ROM, we would like to point out that the syndrome decoder has fewer metric combinations and can thus be implemented using a smaller ROM. Secondly, Table II provides us with some choices as to the survivors in columns 2 and 5. By making the appropriate nonrandom selection it is possible to save one path register (path register 1 or 3). We have not yet been able to give a general formula for the hardware saving of the syndrome decoder as compared to the classical Viterbi decoder. However, we have built decoders for $\nu = 2$ and $\nu = 4$ and observed a hardware saving in each case. For $\nu = 4$, for example, the classical Viterbi decoder uses 16 path registers, while it is possible to build a syndrome decoder with only 9 path registers. This particular $\nu = 4$ syndrome decoder uses only 1817 ROM addresses. A program has been developed that computes the contents of the ROM for an arbitrary rate-$\frac{1}{2}$ binary convolutional code. This program enables us to quickly design an extremely efficient minimum distance decoder.

ACKNOWLEDGMENT

The authors want to thank L. J. A. E. Rust for his help with the hardware realization.

REFERENCE

[1] A. J. Viterbi, "Convolutional codes and their performance in communication systems," *IEEE Trans. Commun. Technol.*, vol. COM-19, pp. 751–772, Oct. 1971.
[2] E. R. Berlekamp, *Algebraic Coding Theory.* New York: McGraw-Hill, 1968.
[3] L. van de Meeberg, "A tightened upper bound on the error probability of binary convolutional codes with Viterbi decoding," *IEEE Trans. Inform. Theory* (Corresp.), vol. IT-20, pp. 389–391, May 1974.