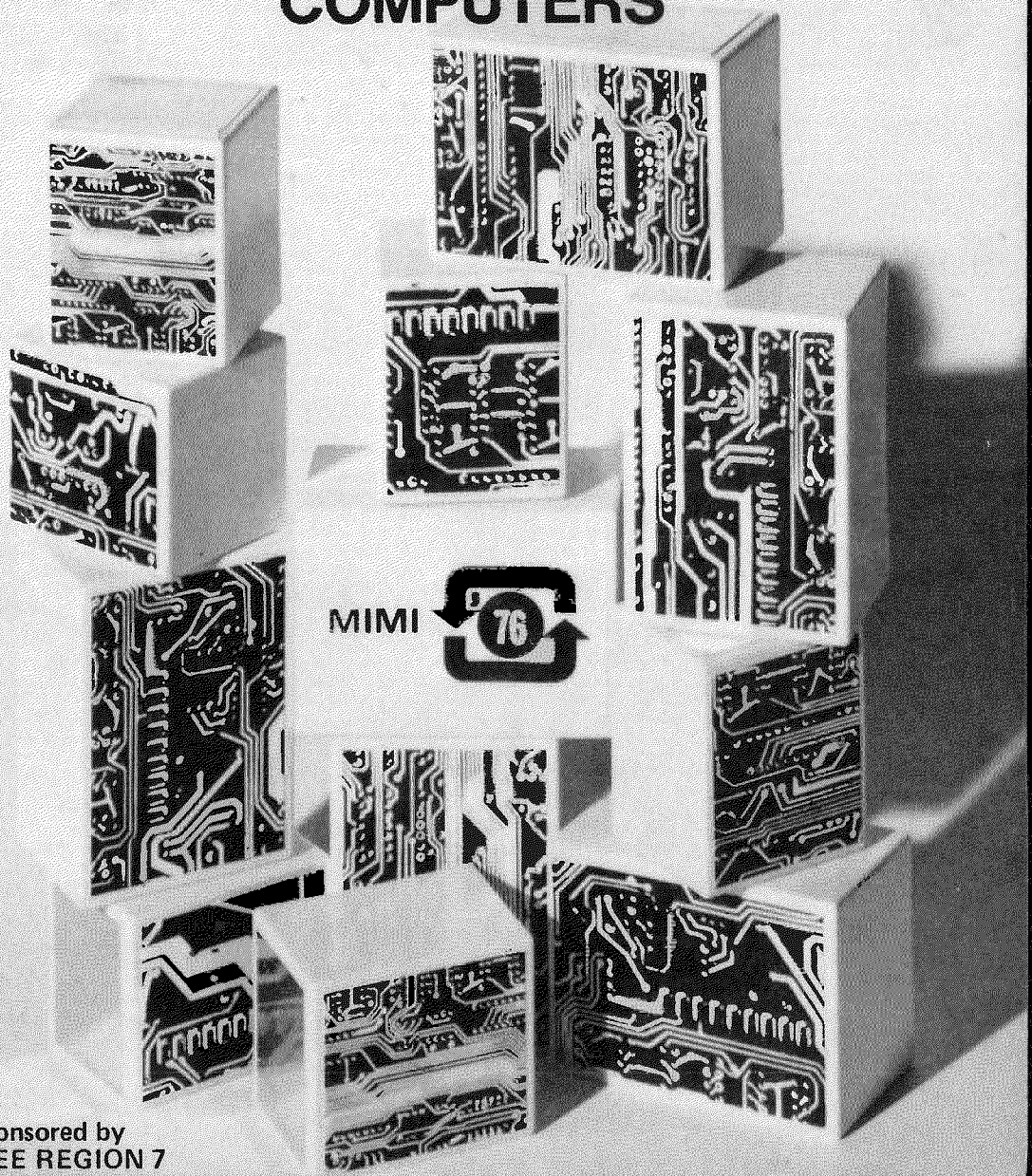
EXHIBIT L

MIMI 76

Proceedings of the International Symposium on

MINI and MICRO COMPUTERS

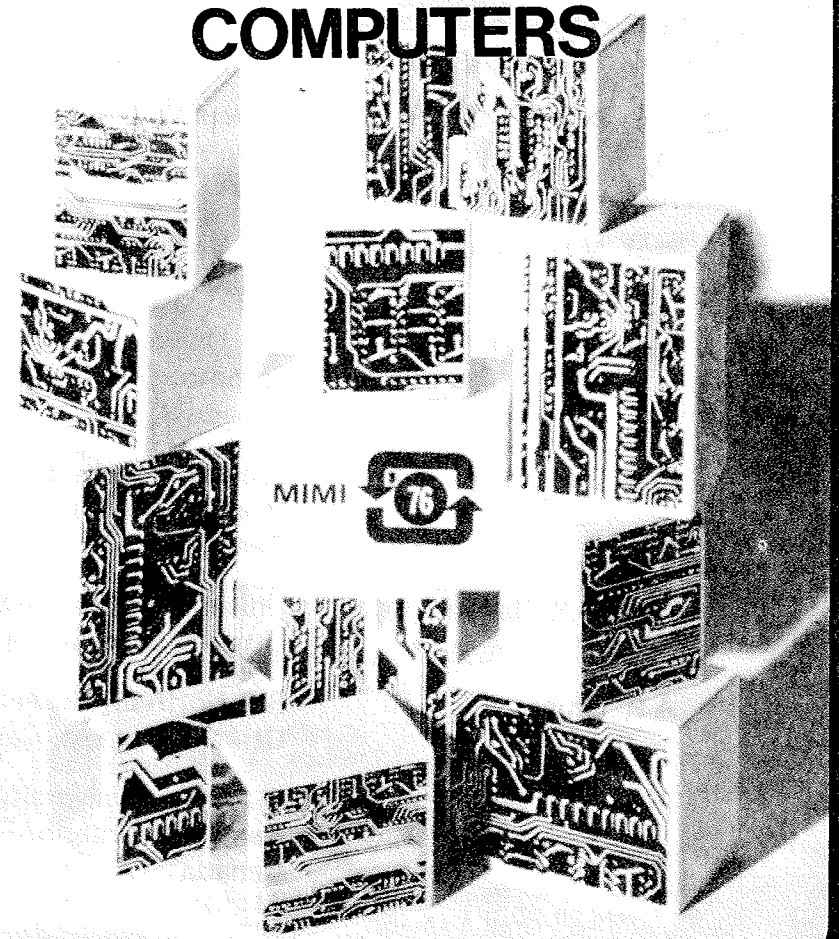


Sponsored by
IEEE REGION 7
IEEE COMPUTER SOCIETY

IEEE Catalog No. 76CH1180-9 C

MIMI 76

Proceedings of the International Symposium on MINI and MICRO COMPUTERS



Sponsored by
IEEE REGION 7
IEEE COMPUTER SOCIETY

TORONTO NOVEMBER 8-11, 1976
Editor: M.H. Hamza

IEEE Catalog No. 76CH1180-9 C

Copies available from:

IEEE Computer Society
5855 Naples Plaza, Suite 301
Long Beach, California 90803

IEEE Service Center
445 Hoes Lane
Piscataway, New Jersey 08854

Program Committee

Prof. Hoo-min D. Toong	M.I.T. Cambridge Mass., U.S.A.
Prof. I. Lee	University of California Berkeley, California U.S.A.
Prof. M.H. Hamza	The University of Calgary Calgary, Alberta, Canada
Prof. S.G. Zaky	The University of Toronto Toronto, Ontario, Canada
Prof. J.L. Houle	Ecole Polytechnique Montreal, Quebec, Canada
Prof. D.K. Banerji	University of Ottawa Ottawa, Ontario, Canada
A.R. Omar	Bell Northern Research Ottawa, Ontario, Canada
Prof. J.D. Wright	McMaster University Hamilton, Ontario, Canada

Manufactured in the U.S.A.

HARDWARE AND SOFTWARE IMPLEMENTATION OF THE VITERBI DECODING ALGORITHM FOR CONVOLUTIONAL CODES

JEAN CONAN
Département de Génie Electrique
Ecole Polytechnique de Montréal
Montréal, Québec, Canada

ROLANDO OLIVER
Groupe des Communications Informatiques
Bell Canada
Montréal, Québec, Canada

ABSTRACT

This paper presents practical methods to implement the Viterbi decoding algorithm for convolutional codes in either hardware or software form. First the general background of convolutional encoding and Viterbi decoding is reviewed, mainly to define the necessary notation. In a subsequent step the Viterbi algorithm is simplified and cast in a form suitable for real time hardware and/or software implementation. As a direct consequence, it becomes possible to derive the global flowchart as well as the overall architecture of a special purpose machine which realizes the algorithm. Practical hybrid hardware/software realizations of such a machine operating with different degrees of parallelism are then proposed using microprocessors nets. Finally a computer program written for the DEC/PDP-11 series is presented, which emulates the machine in a completely sequential manner. The actual decoding speed of this machine clearly demonstrates the inadequacy of such a simple software implementation of the Viterbi decoding algorithm in the context of real time bit decoding and stresses the usefulness of the hybrid hardware/software realization using microprocessors.

1. INTRODUCTION

Among the different possible techniques used to alleviate the effect of random errors in data communication systems, one of the most powerful is convolutional encoding combined with the optimal trellis search of the Viterbi decoding algorithm. The Viterbi algorithm is also useful in the optimal demodulation of discrete signals transmitted over bandwidth limited channels which are characterized by intersymbol interference. The potential of forward-error-codes, and more specifically of convolutional codes in the context of data transmission through satellite channels has been widely demonstrated [1, 2]. In applications requiring a high level of reliability as well as high efficiency, simple forward error encoding can be proved to be insufficient. A hybrid FEC/ARQ scheme has been recently suggested in [3], and such a system would be useful to combat errors in node to node transmissions for a packet switched data communication network using a satellite link. All these potential applications of the Viterbi algorithm provide sufficient motivations for a thorough investigation of this algorithm as well as of the possible means of implementation using present state technology. This paper is an attempt to give some partial answers to this important problem. In section 2 and 3 the necessary background in convolutional encoding and Viterbi decoding is quickly reviewed. Section 4 presents the Viterbi algorithm in a form suitable for machine implementation and the intrinsic parallelism of the procedure is clearly demonstrated. As a consequence, it becomes possible using microprocessor nets to outline the overall architecture of special purpose machines which realize the operations of the algorithm with different degrees of parallelism. A simple, completely sequential realization of the machine on a DEC/PDP-11 computer demonstrates clearly the inadequacy of the software implementation for high speed real time bit decoding even in the context of relatively weak convolutional codes.

2. CONVOLUTIONAL ENCODING

A binary convolutional code of rate $R = d/v$ and constraint length K can be encoded by using $d(K-1)$-stage shift registers together with v modulo-two adders as shown in Figure 1. In this diagram, at each time t an input branch $\underline{u}_t \triangleq (u_t^1, u_t^2, \ldots, u_t^d), u_t^i \in \{0, 1\}$ $1 \le i \le d$, is transferred to the shift registers through the input switch and a corresponding output branch $\underline{x}_t \triangleq (x_t^1, x_t^2, \ldots, x_t^v)$, $x_t^j \in \{0, 1\}$ $1 \le j \le v$ is computed according to the content of the shift registers and the connections between the modulo-two adders and the different cells of the shift register which specify the code. Defining $m = K-1$ as the memory order of the code, the input/output relation associated with the linear sequential circuit of Figure 1 can be described through a set of K [d x v] matrices of 1's and 0's, $G^\ell \triangleq [g_{ij}^\ell]$, $0 \le \ell \le m$, called the generating matrices of the code by the relations

$$\underline{x}_t = \sum_{\ell=0}^{m} \underline{u}_{t-\ell} G^\ell \qquad (1)$$

where $g_{ij}^\ell =$ 1, if the cell with delay ℓ relative to register number i is connected to the modulo-two adder number j; 0, otherwise

At time $t = 0$, the initial conditions are such that all registers are set to "0". If $U_t = (\underline{u}_0, \underline{u}_1, \ldots, \underline{u}_t)$ is the information sequence up to time t, the corresponding output sequence is $x_t = (\underline{x}_0, \underline{x}_1, \ldots, \underline{x}_t)$, where $\underline{x}_t$ the output branch at time t depends only on the values of $\underline{u}_t, \underline{u}_{t-1}, \ldots, \underline{u}_{t-m}$ according to (1) and it is implied by the initial conditions that $\underline{u}_i = (0, 0, \ldots, 0)$ $\forall\, i < 0$. Defining the state space S as the set of all binary d x m . tuples, and the state value of the encoder at time t as the d x m . tuple $\underline{s}_t = (\underline{u}_{t-1}, \underline{u}_{t-2}, \ldots, \underline{u}_{t-m})$, the dynamical behavior of the encoder can be described by means of the next state transition map

$$Q : S \times \{0,1\}^d \to S \qquad (2)$$

which can be characterized by

$$Q : (\underline{s}_t, \underline{u}_t) \mapsto \underline{s}_{t+1} \triangleq (\underline{u}_t, \underline{u}_{t-1}, \ldots, \underline{u}_{t-m-1}) \qquad (3)$$

The graphical representation of this map together with the input/output map described by (1) is known as the Good-De-Bruijn diagram of the code, and is obtained as follows. Each one of the possible state values is represented by a node on the graph and the different nodes are connected by branches according to Q. Each branch is labelled by the pair $\underline{u}/\underline{x}$ where $\underline{u}$ and $\underline{x}$ are respectively the input and output branches. For illustrative purposes, such a graph is represented in Figure 2 for the rate 2/3, m = 1 code with generating matrices

$$G^0 = \begin{vmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \end{vmatrix} \quad \text{and} \quad G^1 = \begin{vmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{vmatrix}$$

For the purpose of decoding, a maximum likelihood sequence decoder must compare all possible output sequences to the noisy received sequence and choose the one with maximum a posteriori probability. The trellis diagram, as introduced by Forney, is a structured representation of all input/output sequences of the code, and is the key to the Viterbi decoding procedure. At time t = 0, the encoder is in the all "0"'s state. Ordering all the elements in the state space S in a fixed and predefined manner, the trellis diagram is obtained by linking all possible state values at time t to all possible state values at time t + 1 by the corresponding connecting branches, and repeating the procedure for t = 0,1,2,...,etc. The trellis can then be terminated at some time L + m by appending m trailing all 0's input branches to the L x v information bits for resynchronisation purposes (sequence decoder), or unterminated by letting L go to ∞ (bit decoder). As an example, the unterminated trellis diagram for the code of Figure 2 is represented in Figure 3.

3. VITERBI DECODING

The operation of a convolutional encoder can be described by a unique path in the trellis diagram resulting in the output sequence X^U when the input sequence is U. Let Y be the received sequence after transmission of X^U over a Discrete Memoryless Channel (DMC). In these conditions, decoding is then tantamount to finding the path $\tilde{U}$ in the trellis whose sequence of branches $X^{\tilde{U}}$ is "elesest" to the received sequence Y. As a consequence, a sequence error will occur whenever $\tilde{U} \neq U$. Assuming equiprobable input symbols and a sequence decoder operating on a terminated trellis of length L + m, the probability of a sequence error P_E is minimized by applying the maximum likelihood rule.

Observe Y and set $\tilde{U} = U$ if

$$\log[P(Y/U)] \geq \log[P(Y/U')] \forall\, U' \neq U \qquad (4)$$

For DMC, we can write the log likelihood function or "metric" of the path U of length N branches as

$$\Gamma^U_N = \log[P(Y/U)] = \log[P(Y/X^U)] = \sum_{i=0}^{N-1} \gamma^U_i \qquad (5)$$

where the so called branch "metric" γ^U_i is defined as

$$\gamma^U_i = \log[P(\underline{y}_i/\underline{x}^U_i)] = \sum_{j=1}^{v} \log[P(y^j_i/x^{U_i}_i)] \qquad (6)$$

and $\underline{y}_i$ and $\underline{x}^U_i$ are respectively the i^{th} received branch and the i^{th} output branch on the hypothesized path U. Rule (4) can then be rewritten for an information sequence of length L as

$$\text{Find } \tilde{U} \in \Gamma^{\tilde{U}}_{L+m} = \max\{\Gamma^U_{L+m}\} \qquad (7)$$

The Viterbi Maximum Likelihood Decoding (VMLD) rule is then equivalent [4] to the forward dynamic program which solves in the trellis diagram the optimization problem defined by (7) and satisfies the terminal conditions on the state $\underline{s}_{t=0} = \underline{s}_{t=L+m} = \underline{s}_0$ (the null state). Such a decoder is easily described by the following recursive algorithm.

MLVD (Maximum Likelihood Viterbi Decoding) Algorithm.

Step 1: Extend in the trellis all distinct 2^{dxm} paths diverging from the root node $\underline{s}_0$ over m branches and compute their metrics. Call these paths the survivors. Set $\ell = m$.

Step 2: Extend all the 2^{dxm} paths surviving at step ℓ and compute their metrics by adding the corresponding branch metrics. This results in 2^d paths reconverging at each one of the trellis nodes at depth $\ell+1$. For each node at depth $\ell+1$, compare the metrics of the 2^d reconverging paths and retain as survivor the one with the largest metric. (In case of ties pick one at random) Set $\ell = \ell+1$.

If $\ell = L + m$, stop (one of the MLD paths is the surviving path at state $\underline{s}_0$). Else, go to step 2.

In applications requiring high efficiency, the sequence length L must be made large with regard to m. On the other hand, as L is increased, it must be the case that P_E tends to 1 for any practical channel. In this case, as well as in situations where the bit, not the sequence is the basic information entity, the criterion of interest is the bit error probability

$$P_{BE} \triangleq \frac{1}{L} \sum_{\ell=0}^{L-1} \sum_{m=1}^{d} P(\hat{u}^m_\ell \neq u^m_\ell) \qquad (8)$$

which gives the fraction of bits which are wrongly decoded.

Algorithms which minimize P_{BE} have been proposed for the terminated trellis case [5]. These algorithms require receipt of the entire sequence U_{L+m} before decoding begins and so cannot be used without resynchronisation. If high efficiency is required, then L must be made large and the storage requirement which grows linearly with L as well as the decoding log limit the usefulness of these algorithms to short values of L and short constraint length code, in which case the gain obtained over straight Viterbi decoding is of no significance. A real time, unsynchronized minimum bit error probability algorithm operating on an unterminated trellis has been proposed in [6]. It appears essentially that this algorithm does not provide any substantial gain over a simple sub-optimal modification of the MLDVD algorithm, suitable for real time applications. Such an algorithm, termed Real Time Viterbi Decoder (RTVD) can be outlined as follows. Its operation is identical to that of the MLVD except that step 2 is repeated indefinitely. Bit decoding is performed as follows. Given $\Delta > m$, an integer called the decoding lag, whenever $\ell \geq \Delta$ the decoded bits at depth $\ell-\Delta$ are the $\underline{d}$ symbols at depth $\ell-\Delta$ on the surviving path which is the most likely among all survivors at step ℓ.

4. PRACTICAL SCHEMES FOR VITERBI DECODING

It should be clear that the informal description of the Viterbi decoding algorithm as outlined in the preceding section cannot be used as a starting point for any practical method of implementation. As a consequence a major simplification of the algorithm will first be proposed and then a new and formal notation will be introduced to describe efficiently the search procedure in the trellis. It is readily seen that the initialization procedure (step 1) in the MLVD Algorithm of the preceding section is not required. In fact, if we assume that all the surviving paths which terminate in any state $\underline{s} \neq \underline{s}_0$ (the null state)at time $t = 0$ are given a starting biased metric of $-\infty$ (in practice any sufficiently negative value would be suitable), and the starting path at state $\underline{s}_0$ is given a metric of "0", it can be seen that if the algorithm is started in step 2, after m successive applications of this step all the surviving paths will be extending from the root node $\underline{s}_0$ at depth 0. As such this new simplified procedure is completely equivalent to the original one. Let us now introduce the following notation. Any state $\underline{s}_n$ can be represented as the concatenation $\rho.\ \delta_s$ where ρ is the (m-1) d-tuple of the leftmost components of $\underline{s}_n$ and δ_s is the complementary d-tuple. Assuming that data are shifted from left to right, when the path which is surviving at state $\underline{s}_n$ is extended by one branch, it will terminate on the next step on any of the states characterized by $\delta_e.\rho$ where δ_e is the d-tuple which represents the input.

$$\mathcal{S}_{\underline{s}_n} \triangleq \{\underline{s} = \delta_e.\rho:\ \ \delta_e \in \{0,1\}^d\} \qquad (9)$$

is a subset of S which will be called the set of successors of $\underline{s}_n$.

In the same manner, any state s'_n can also be written down as the concatenation $\delta'_e.\rho'$, where ρ' and δ'_e have obvious interpretations. The subset of S

$$\mathcal{P}_{\underline{s}'_n} = \{\underline{s}' = \rho'.\delta'_s:\ \delta'_s \in \{0,1\}^d\} \qquad (10)$$

is called the set of predecessors of $\underline{s}'_n$ and is seen to be composed of all the states such that the surviving paths terminating at these states at some step t-1 will be reconverging at state $\underline{s}'_n$ at step t.

It is readily seen that every pair of states $\underline{s}_1$, $\underline{s}_2$ characterized by the same value of ρ' have identical sets $\mathcal{P}_{\rho'}$ of predecessors. Reciprocally, every pair of states with the same value of ρ have identical sets $\mathcal{S}_\rho$ of successors. As a consequence it is possible to define two equivalence relations on S as follows.

Definition 1: Two states $\underline{s}_1$, $\underline{s}_2 \in S$ are $\mathcal{S}$-equivalent iff $\mathcal{S}_{\underline{s}_1} = \mathcal{S}_{\underline{s}_2}$.

Definition 2: Two states $\underline{s}_1$, $\underline{s}_2 \in S$ are $\mathcal{P}$-equivalent iff $\mathcal{P}_{\underline{s}_1} = \mathcal{P}_{\underline{s}_2}$.

These two equivalence relations define two partitions of S

$$\{\mathcal{P}_{\rho'}:\ \ \rho' \in \{0,1\}^{(m-1)d}\} \qquad (11)$$

and

$$\{\mathcal{S}_\rho:\ \ \rho \in \{0,1\}^{(m-1)d}\} \qquad (12)$$

which are such that $\mathcal{P}_\rho$ and $\mathcal{S}_\rho$ are in a 1-1 correspondence in the sense that every element in $\mathcal{P}_\rho$ is the predecessor of all the elements in $\mathcal{S}_\rho$. Since every set $\mathcal{P}_\rho$ and $\mathcal{S}_\rho$ contains 2^d distinct states, such a correspondence defines a subtrellis implying 2^d predecessors and 2^d successors which is illustrated in Figure 4 for the case $d = 2$.

In order to compute the branch metric corresponding to the state transition from the state $\rho.\delta_s$ at depth n to the state $\delta_e.\rho$ at depth n+1, given that $\underline{Y}^n$ is the received branch from the demodulator, a table of all the possible branch metrics $\gamma(S_e,\ \rho,\ S_s,\ \underline{Y}^n)$ indexed by the concatenation of $\delta_e.\rho.\delta_s.\underline{Y}^n$ is assumed to be available. For systems using soft quantization at the output of the demodulator, the dimensionality of this table might be prohibitive. In this case it is possible to operate with two smaller tables. One table indexed by $\delta_e.\rho.\delta_s$ contains the corresponding transmitted branch symbols, the other table associates symbols submetrics which reflect the a priori knowledge of the channel transition probabilities to every combination of transmitted symbols and demodulated symbols. Since the channel is memoryless, according to (6) the branch metric can be computed by summing the symbols submetrics over all the symbols of the branch. The general flowchart of Figure 5 illustrates the operation of a machine which realizes the RTVD algorithm. In this diagram, Γ', δ, ρ, ρ', Γ_{MAX}, Γ^*, δ_e, δ_s are registers, the 2^{md} registers $\Gamma^{(n)}$ contain the metrics of the surviving paths at depth n and the 2^{md} stacks of $(m-\Delta)d$ bits contain the truncated surviving paths at depth n. The portion of the flowchart within the dashed area represents the basic subtrellis extension. At the end of the full extension of the trellis, $\delta'.\rho'$ points to the stack from which the decoded d bits are collected. Following the foregoing discussion, the initialization is such that all stacks are set to "0"s and the metrics of all paths are biased negatively except the metric at the root node which is set to 0. Since the parent states and the descendent states in the basic subtrellis belong respectively to $\mathcal{P}_\rho$ and $\mathcal{S}_\rho$ for some value of ρ, and $\{\mathcal{P}_\rho\}$ and $\{\mathcal{S}_\rho\}$ are distinct partitions of S, two sets of memory registers A and B are required for reading and writing the information relative to an extended path terminating at a given state. The use of these two sets can be alternated in the sense that whenever the set A is used as the base for extension at some step n, the set B will be used to memorize the information relative to the survivors at this step and the role of A and B will be interchanged on the next step. On the other hand, the fact that the operations on the different sets $\mathcal{P}_\rho$ and $\mathcal{S}_\rho$ are identical implies that the operation of the machine can be carried out in parallel. As a consequence the machine of Figure 5 can be broken down into $2^{(m-1)} \times d$ elementary identical submachines which perform the same sequence of operations on the different sets $\mathcal{P}_\rho$, $\mathcal{S}_\rho$. Such submachines can either operate sequentially by using a single processor or in parallel by using a bank of 2^d processors. This concept is illustrated on Figure 6 for the case $d = 1$. Each one of these processors must perform a certain number of state transitions and has access to a ROM which contains the metrics of the branches relative to these transitions. It is a fact that complete parallelism requires the use of a number of processors which grows exponentially with the code complexity and as such some sensible

trade-off between speed and cost/complexity must be made by the designer. In fact it is possible to operate with a varying degree of parallelism by requiring the use of $2^{\nu xd}$ basic submachines ($0 \leq \nu \leq m-1$) which operate in parallel and each machine must perform a sequence of $2^{(m-1-\nu)xd}$ basic extensions. Such a semi-parallel operation of the machine necessitates the partitioning of the state space into $2^{(\nu+1)d}$ sets $\{\mathcal{Q}_\alpha\}$ which contain the same number of states, namely $2^{(m-1-\nu)xd}$ and have the following property: 2^d sets $\mathcal{Q}_\alpha$ $\alpha = \alpha_0, \alpha_1, \ldots, \alpha_{2^d-1}$ can be grouped together so that for every element in $\mathcal{Q}_{\alpha_0}$ which belongs to $\mathcal{P}_{\rho_0}$, all the 2^d-1 remaining elements in $\mathcal{P}_{\rho_0}$ belong to

$$\alpha_1, \quad \alpha_2, \quad \ldots, \quad \alpha_{2^d-1}$$

As a consequence, if the memory space of the machine used as the base for extension, say set A, is distributed so that a memory bank of $2^{(m-1-\nu)xd}$ nodes (a node includes the storage space for the stack of the surviving path and the corresponding metric) is associated to each one of the sets $\mathcal{Q}_\alpha$, $2^{(\nu+1)xd}$ nodes can be extended in one step by using the $2^{\nu xd}$ basic submachines and the procedure can be repeated $2^{(m-1-\nu)xd}$ times. Since $\{\mathcal{P}_\rho\}$ and $\{\mathcal{Y}_\rho\}$ are distinct, if the memory space B for writing the results of the extension is distributed in the same manner as A, it will be necessary in order to be able to do the writing operation in parallel to rearrange memory B by swapping some of the nodes between the different storage banks. Figure 7, 8, 9 illustrate the concept for the case $d = 1$, $m = 3$ with a varying degree of parallelism. It is readily seen that when one basic submachine is used, two read-write cycles are necessary to rearrange the memory space B, only one cycle is needed when two parallel machines are used. The full parallelism case requires a completely distributed memory space, and moreover only one set of memory banks is required provided sufficient delay is allowed between the read-write operation.

Recent advances in LSI microprocessors technology suggest that very low cost "SLICE" processors with cycle times well under 100 ns are coming into operational use [7]. Such microprocessors could be most efficiently used as the processors in the basic subtrellis extension machine. The main advantage of using microprocessors in place of hard-wired special purpose processors resides in the fact that the overall cost of construction can be considerably reduced by mass production, or equivalently, for the same cost the degree of parallelism and hence the speed can be increased. Moreover changing the code connections does not require any additional programming and/or hardware rewiring, since only the metrics ROM's contents need be modified.

Among the general class of machines which has been introduced, the one which uses one single processor is obviously the slowest since in this case even the basic subtrellis extension must be carried out sequentially. Such a machine can be easily realized on a minicomputer. In order to investigate this possibility, the machine of Figure 5 has been programmed on a DEC-PDP/11-20 minicomputer. Extensive use of the macros capability of the MACRO-11 assembly language has led to a very modular implementation exploiting very effectively the full speed capability of the machine. To the authors knowledge, this is the first instance where a general rate ($R = d/v$) Viterbi decoder has been programmed in assembly language. The reader is referred to [8] for the details of implementation. Actual decoding speed for the case $R = \frac{1}{2}$ are reported in Table 1.

	Δ - m		
m	16	32	48
2	2366	2036	1792
3	1284	1092	952
4	674	568	494
5	346	292	252
6	176	148	128

Table 1: Decoding speed in bits/sec for the software RTVD. ($R = \frac{1}{2}$, PDP/11-20) for different decoding lag values and varying degree of complexity.

It appears that for codes of average complexity ($k = 7$), the decoding speed falls short from any useful value in the context of data transmission. Although it should be possible to improve on the speed by using faster minicomputers, it does not appear possible with this sheme to gain more than an order of magnitude.

5. CONCLUSION

This paper has presented a thorough investigation of some practical schemes to implement the Viterbi Algorithm for the decoding of convolutional codes. A general class of machines based on microcomputers nets and using different degrees of parallelism has been presented. These machines easily outperform simple completely software realization using a single processor and could even compete with special-purpose machines when used with codes of more than average complexity ($7 \leq K \leq 10$).

ACKNOLEDGEMENTS

The first author would like to thank Dr. D. Haccoun and Mr. G.E. April, both from the Ecole Polytechnique de Montréal for stimulating discussions as well as helpful suggestions during the preparation of this manuscript.

REFERENCES

[1] D.G. Forney, "Coding and its Application in Space Communications", IEEE Spectrum, June 1970.

[2] J.A. Heller, I.M. Jacobs, "Viterbi Decoding for Satellite and Space Communication", IEEE Transactions on Communication Technology, Vol. COM-19, No. 5, October 1967.

[3] J. Conan, D. Haccoun, H.H. Hoang, "Performance of ARQ, FEC and Hybrid ARQ/FEC error Control Schemes for High Speed Data Transmission on Satellite Channels", Proceedings of the Canadian Communications and Power Conference, pp. 216-319, October 20-22, 1976, Montreal, Québec, Canada.

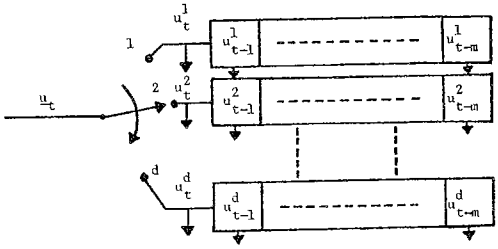
[4] J.K. Omura, "On the Viterbi Decoding Algorithm", IEEE Transactions on Information Theory, Vol. IT-15, January 1969.

[5] L.R. Bahl, J. Cocke, F. Jelinek and J. Raviv, "Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate", presented at the 1972 Int. Symp. Information Theory, Asilomar, Calif. Jan. 1972.

[6] L.N. Lee, "Real-Time Minimal-Bit-Error Probability Decoding of Convolutional Codes", IEEE Transactions on Communication Technology, Vol. COM-22, No.2, February 1974.

[7] W. Blood, "MI0800 - The High Performance LSI Processor Family", Application Note, Motorola Inc., 1976.

[8] R. Oliver, "Décodage en Temps Réel des Codes Convolutionnels par l'Algorithme de Viterbi", Mémoire de Maîtrise en Sciences Appliquées, Département de Génie Electrique, Ecole Polytechnique de Montréal, Sept. 1976.



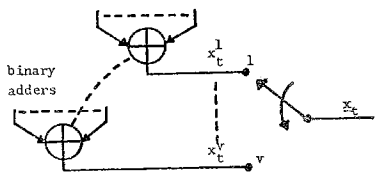


Figure 1: General rate d/v binary convolutional encoder

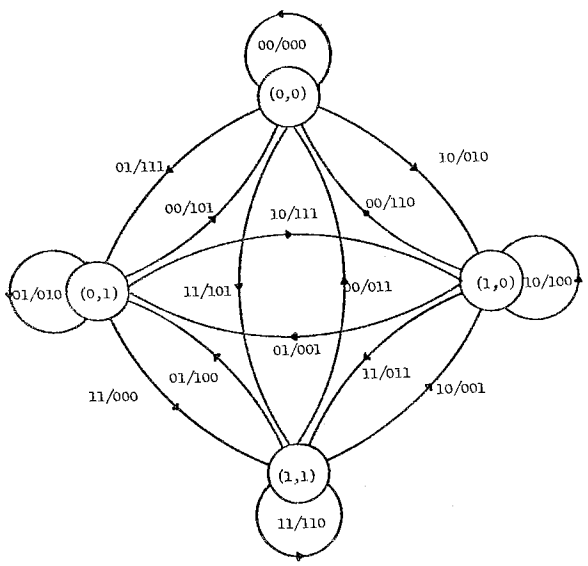


Figure 2: Good-de-Bruijn diagram for the R = 2/3, m = 1 code defined by $G^0 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$, $G^1 = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$

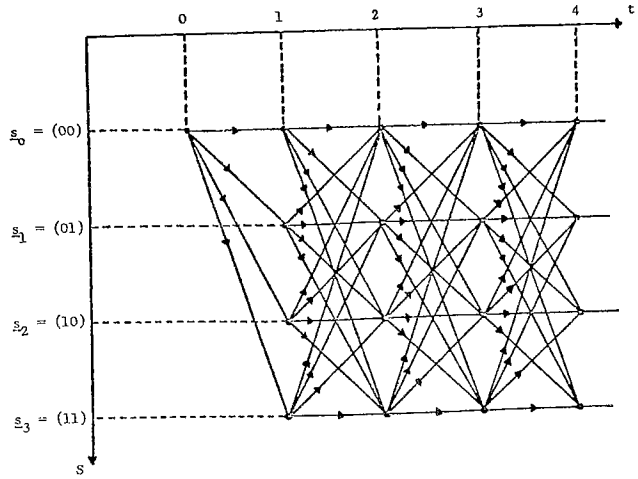


Figure 3: Trellis diagram for the R = 2/3, m = 1 code of Figure 2.

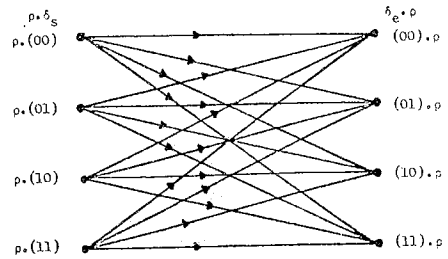


Figure 4: Basic subtrellis for the case d = 2.

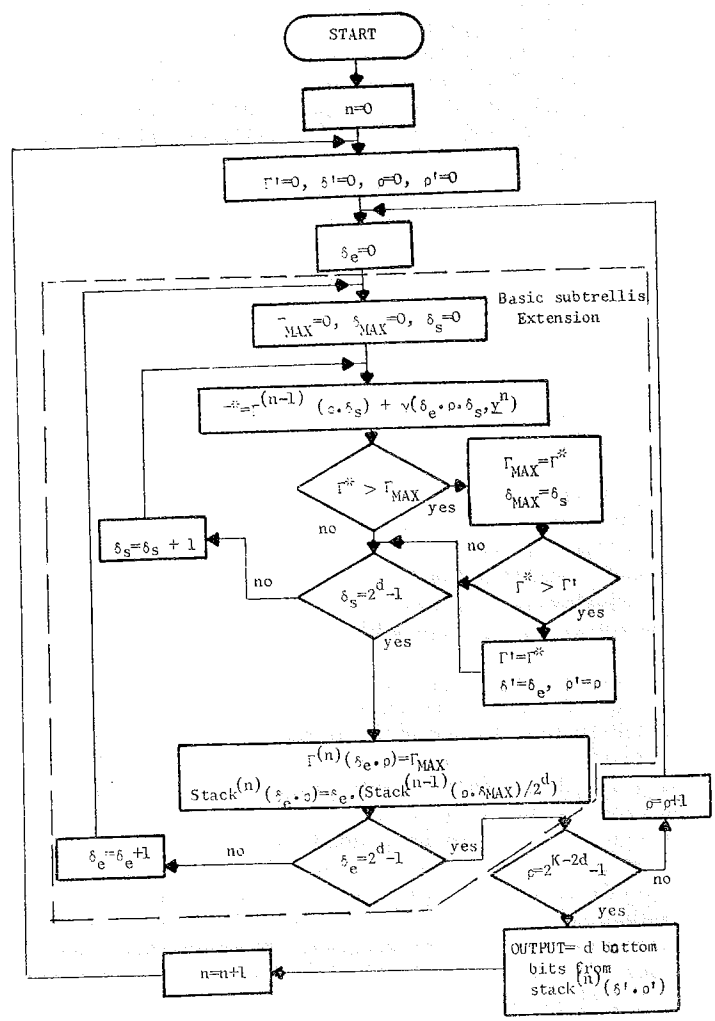


Figure 5: Flowchart of the Viterbi Real Time Decoder

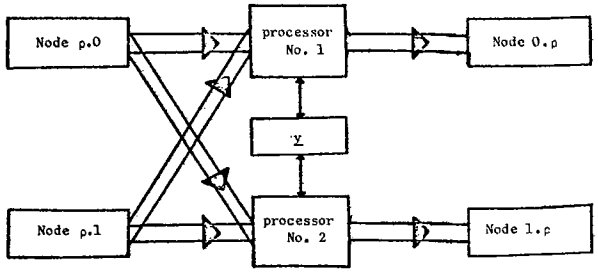


Figure 6: Basic subtrellis extension machine (d = 1)

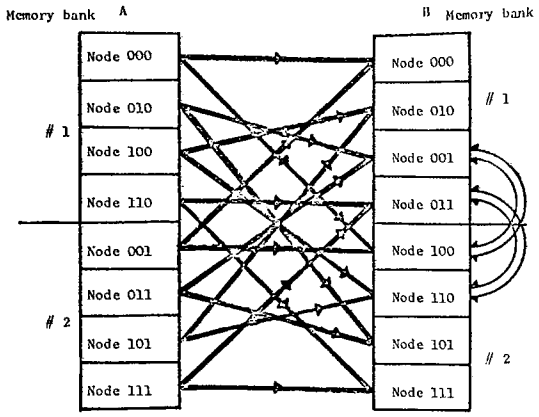


Figure 7: One basic machine with four cycles (d = 1, m = 3)

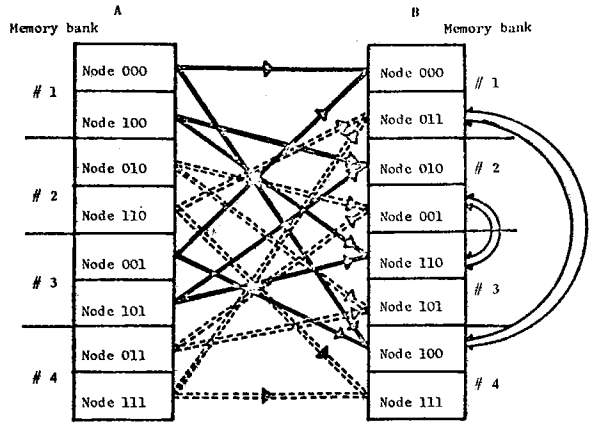


Figure 8: Two basic machines with two cycles (d = 1, m = 3)

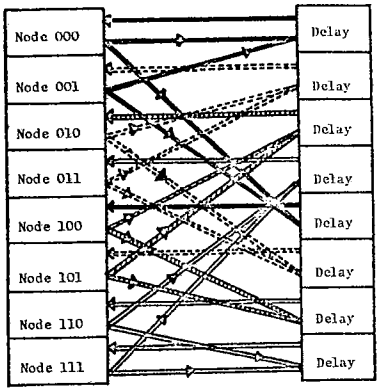


Figure 9: Full parallelism trellis extension using four basic machines with one cycle (d = 1, m = 3)