

Decoding of Rate K/N Convolutional Codes in VLSI¹

P. G. GULAK, T. KAILATH, A. MONTALVO AND V. P. ROYCHOWDHURY

*Information Systems Laboratory
Stanford University
Stanford, CA 94305*

I. EXTENDED SUMMARY

Decoding convolutional codes can be accomplished by a Viterbi Algorithm based on the state transition diagram of the encoder. This was the original application of the algorithm when first introduced in 1967. Basically, this algorithm provides a dynamic programming solution to the "shortest path" problem in a directed graph, and therefore can be used to obtain the Maximum Likelihood estimates of the state sequence of a finite state machine observed in memoryless noise. Characteristic of the Viterbi Algorithm (VA), is the concept of a trellis diagram. The trellis diagram is a graphical representation of the encoder state diagram drawn as a function of discrete time. Each discrete time step is a single baud interval T , and corresponds to one stage of the trellis diagram. Conceptually² the number of stages in the trellis diagram corresponds to the length of the input data stream to the encoder.

In $GF(q)$, the number of nodes or states at each stage of the trellis is q^v , where q is the cardinality of the input alphabet set and v is the minimum number of memory elements required to implement the encoder. For a rate k/n convolutional encoder, the number of edges in the trellis diagram could be as many as q^{k+v} , and are defined by the possible state transitions in the state diagram of the encoder. Associated to each state are two quantities, the path metric and the survivor sequence, that must be stored. The survivor sequence for state S_j at time p corresponds to the input sequence associated with the path in the trellis diagram, that starts at initial state S_0 and ends at the p stage of the trellis in state S_j , and has minimum path metric. The path metric is a measurement of the unlikelihood of the hypothesis that an

output sequence associated with a particular path in the trellis corresponds to the transmitted output sequence of the encoder. After time t , for t sufficiently large³, the state trajectory defined by the survivor sequence with the smallest path metric will provide estimates of the input symbols to the encoder.

The algorithm proceeds as follows. Once each baud interval, the path metric for state S_j is updated by taking the lowest of the resulting sum obtained by adding the metric associated with the branch connecting states S_i and S_j to the path metric of state S_i for all valid state transition arriving into state S_j . This branch metric is a measure of the unlikelihood of the new received symbol being part of a path arriving at state S_j from state S_i . If P_j^t denotes the path metric of state S_j at time step t (i.e. at stage t in the trellis) and if λ_{ij} denotes the branch metric⁴ of the connecting edge between states S_i and S_j , then the path update procedure can be written as:

$$P_j^{t+1} = \min_{0 \leq i \leq q^v} (P_i^t + \lambda_{ij}) \quad , 0 \leq j \leq q^v \quad (1)$$

The new survivor sequence is obtained by appending to the old survivor sequence the symbol corresponding to the transition of the winning ancestor to this state. Notice, that the VA is a relatively simple algorithm since it consists only of "add, compare and select" operations that must be applied to each state in the trellis. Unfortunately the number of states, and therefore the complexity, grows exponentially (as q^v) with the memory of the encoder. However, we are motivated to use as large a v

¹This work was supported in part by the SDIO/IST, managed by the Army Research Office under Contract DAAL03-87-K-0033, the Department of the Navy, Office of Naval Research under Contract N00014-86-K-0726 (Supplement), and Rockwell International, under Contract INT 6G3052.

²Therefore a potentially semi-infinite diagram can be obtained. But in reality the trellis diagram does not need to contain more than five to six times the memory of the encoder,

³How large t should be is a design criteria since truncating the trellis diagram to t stages when the input sequence is longer than t will affect the probability of error. In the book by Omura and Viterbi such questions are answered for a rate $\frac{1}{n}$ encoder over $GF(2)$, and t five to six times v the probability of error only is degraded by a small portion.

⁴Which can be a Hamming or Euclidean distance between the received symbol at time t and one of the possible transmitted symbols. For those combination of states that do not represent a valid state transition according to the state diagram of the encoder the corresponding λ is taken to be infinite.

as possible since an important property of convolutional codes is that the exponent in the probability of error decoding decreases as v increases⁵. Due to this exponential growth in the number of states, implementation of the VA in sequential Von-Neumann type of systems (such as microprocessors and DSP chips) generally limits the use of the algorithm to low data rate applications or to encoders with small numbers of states.

An alternative, with VLSI technology, is to look for implementations that exploit the parallelism inherent in the VA. In fact a fully parallel implementation of the VA could be realized by providing both: (i) a single processor for each state in the complete trellis diagram, and (ii) of the order of q^k wires from each state to transmit the path metric, the survivor sequence and the branch metric. However for a message of length L , Lq^v processors would be required, which can be extremely large since L is often very large. A somewhat more practical strategy is to use q^v processors globally interconnected according to the state diagram of the encoder. This can be obtained from the previous implementation by recognizing that the trellis diagram can be viewed as a dependence graph for the VA, and a projection along the direction of the state time iteration would result in a signal flow graph, and therefore an array processor implementation, that correspond to the encoder state diagram (see [1] and [2]). In 1984 Cain and Kriete used this approach, along with several tricks in order to keep the branch metric small, to implement a VA at a rate of 10Mbps for a rate $1/2$ feed-forward convolutional encoder with memory $v = 7$ (see [?]). A second approach recently proposed by Chang and Yao (see [?]), for feed-forward encoders of rate $1/n$ and rate k/n , uses q^v processors connected in a linear array with adjacent neighbor interconnections to update the path metric and generate the branch metrics. Chang and Yao obtained this implementation by realizing that, locally at any stage in the trellis, the path metric update can be formulated as a matrix-vector multiplication problem of the following form:

$$P^{t+1} = P^t \circledast \Lambda, \quad (2)$$

where P^t is a $1 \times q^v$ row vector whose j^{th} element is P_j^t , the accumulated path metric associated with state S_j , and Λ is the $q^v \times q^v$ modified adjacency matrix whose i - j^{th} element, λ_{ij} , denotes the branch metric from state S_i to state S_j between two adjacent stages in the trellis diagram. The operator \circledast denotes vector multiplication in the following sense:

$$P_j^{t+1} = (P_0^t + \lambda_{0j}) \bigvee (P_1^t + \lambda_{1j}) \bigvee \dots \bigvee (P_{q^v-1}^t + \lambda_{q^v-1j}), \quad (3)$$

⁵Since the probability of error can be approximated in high SNR as $P_e = N(\text{free})Q(\frac{d_{\text{free}}}{\sigma})$, where the minimum free distance (d_{free}) increases with v but the number of sequences with the same minimum free distance ($N(\text{free})$) also increases with v . Therefore, the probability of error does not strictly decrease exponentially with v and also there will be a point of diminishing return.

where \bigvee denotes the operation of taking the minimum and the $+$ operator is regular addition. The above is of course just another expression for equation (1.1). Once this connection is made, two simple implementations of the matrix-vector multiplication problem in a linear array of processors can be used to implement the VA. Figure 1.0 shows these implementations. In Figure 1.a, R_j , which represents the intermediate result in the minimization of P_j in (.3) stays inside each processor, while the P_i is moving to the right and the λ_{ij} are moving down. All data movements are synchronized, and after q^v steps, R_j will contain the new path metric for state S_j . Figure 1.a only describes one iteration of (.3) and in order to iterate this procedure the new set of branch metrics is fed right on top of the current set and the resulting path metric from the previous iteration (i.e R_j that contains the new P_j) is fed back into the array from the left so that pipelining is possible with no idle period. In Figure 1.b, a slight variation of the above scheme is presented. Here the branch metrics (λ_{ij}) are still fed downward but now the path metrics, P_j , are kept in the processors while the partial results, R_i , are moving to the right. Once R_j emerges from the right end processor, R_j will contain the new path metric for state S_j and is fed into the j^{th} processor. Notice that in general only q^k out of q^v processors are doing meaningful work at any giving time instant.⁶ This low efficiency in the utilization of the processor array will be more accentuated for rate $\frac{1}{n}$ encoders. This also suggest that more efficient implementations can be obtained. Indeed, Chang and Yao exploited some of the properties of the encoder state transition diagram to obtain a strongly connected trellis diagram⁷, and hence one that will keep all the processors busy all the time, that could be used with the VA to decode such convolutional encoders. These properties are derived from the structure of the encoder state transition diagram which for this encoder is described by a de Bruijn graph (or a cartesian product of de Bruijn graphs for rate k/n codes). However even though, in this implementation the number of processors is q^v , the speed gain over a sequential implementation is not q^v as one might expect⁸; in fact under reasonable assumptions the speed gain is only q . We shall show in this talk in that there exist other implementations with the same number of processors and with better speed gains over sequential(uni-processor) implementations. Before doing that we may remark that once a strongly connected trellis is ob-

⁶Since for a rational convolutional encoder to be information preserving the number of memory elements must satisfy the following relation; $k \leq v$.

⁷A strongly connected graph is a graph in which every pair of nodes are connected by a branch.

⁸Since the vector-matrix multiplication problem is a simplification of the matrix-matrix multiplication problem in which a sequential implementation will take $O(N^3)$ time and a implementation with N processors will take $O(N^2)$ time.

tained, then the path metric updating procedure can be accomplished by sorting q^v quantities from the smallest to the largest and retaining the smallest. Therefore, all architectures proposed in the literature for implementing sorting algorithms can be used to implement the VA decoder. Notice that the VA is a regular iterative algorithm (RIA) as defined in [1], so Raos design techniques could be also applied to obtain implementation of the VA in an array of processors with locally interconnections. Since the matrix-vector multiplication algorithm has a dependence graph embedded in a 2-dimensional index space then there will be only four different implementations of it in a linear array of processors with local interconnections (see [1] for details). This is not pursued here. Instead, in this paper, a third approach is proposed in which an array of processors connected according to a de Bruijn graph of the appropriate order, is used to implement the VA. We may also note that, when drawn properly and applied to our problem, the de Bruijn graph degenerates (i.e. is equivalent) to the well known shuffle-exchange graph.

Since 1971, perfect shuffle networks have been utilized in computer science as a way of connecting processors in parallel processing machines (see [3]). Somewhat earlier, Singleton showed that the FFT algorithm could be implemented in an array of processors connected in a shuffle-exchange network. the use of a shuffle-exchange network to decode convolutional codes through a VA should not be surprising. Since the dependence graph (trellis diagram) of any rate $1/n$ feed-forward convolutional encoders can be shown to be isomorphic to the dependence graph of an appropriate FFT, an observation first realized by Forney in 1972 and latter by (see [4]). The use of a shuffle-exchange network to decode convolutional codes through a VA should not be surprising. connected in a shuffle-exchange network. In 1986, Gulak and Shwedyk demonstrated this from first principles, showing that a VA based on a trellis diagram with the following properties can be implemented on a shuffle-exchange network: i) q^v nodes per stage, ii) with each node labeled by a v -tuple of elements from $GF(q)$, iii) each node having q successors with the label of each successor obtained by a shuffle and/or an exchange of the elements of the v tuple used to label the ancestor node. This type of trellis diagram corresponds to an encoder with v memory elements whose time interdependencies are given as follows:

$$X_k^{t+1} = X_{k-1}^t, \quad k = 1, 2, \dots, v. \quad (4)$$

This will be the case for any rate $1/n$ feed-forward (or finite impulse response, FIR) encoder. The state transition diagram for these encoders is described by a de Bruijn graph of appropriate order. Moreover any de Bruijn graph can be mapped to a shuffle-exchange graph of the appropriate order. In this talk, we will extend these results to the case of feedback (or infinite impulse response, IIR) or FIR convolutional encoder with rate

k/n . In particular we shall show that any decoder using the VA should be based on the state transition diagram of the dual encoder instead of the state transition diagram of the encoder.

Table 1 gives a summary of the results for different implementations (see also figure 1).

REFERENCES

- [1] S. K. Rao, *Regular Iterative Algorithms and their Implementations on Processor Arrays*, PhD thesis, Dept. of Electrical Eng., Stanford University, Stanford, California, Nov. 1985.
- [2] S. Y. Kung, *VLSI Array Processor*, Prentice-Hall Inc., Englewood Cliffs, N.J., 1987.
- [3] Harold S. Stone, Parallel processing with the perfect shuffle, *IEEE Transactions On Computers*, c-20, No. 2:153-161, Feb. 1971.
- [4] G. D. Forney, The viterbi, *Proceedings of The IEEE*, 61, No. 3:268-278, March 1973.

Layout Type	Logic Speed for Symbol Interval T	Logic Area
Unprocessed	$O\left(\frac{c}{T}\right)$	$O(v)$
Cascade	$O\left(\frac{c}{T}\right)$	$O(v)$
Ring	$O\left(\frac{c}{T}\right)$	$O(v)$
1-D Mesh (Symm)	$O\left(\frac{c}{T}\right)$	$O(v)$
1-D Mesh (Shuffle)	$O\left(\frac{c}{T}\right)$	$O(v)$
2-D Mesh	$O\left(\frac{c}{T}\right)$	$O(v)$
Shuffle-Exchange	$O(v)$	$O\left(\frac{c}{T}\right)$

