

Viterbi Detector: Review of Fast Algorithm and Implementation

By sheng, Xiaohong

I. Introduction

Viterbi Algorithm is the optimum-decoding algorithm for convolutional codes and has often been served as a standard technique in digital communication systems for maximum likelihood sequence estimation. Given a sequence of symbols, the Viterbi Algorithm finds the most likely state transition sequence in a state diagram.

Implementing high-coding gain Viterbi decoders in high-speed applications is more difficult compared to in low-speed applications because of the massive hardware size, large power consumption and expensive cost in high-speed application. However, it becomes more important to implement high-speed Viterbi decoders in recent years. For example, the application of Viterbi decoders to magnetic storage channels for decoding intersymbol interference has pushed required decode rates to over 100Mb/s for high-end drives. A potential application of even higher rates is in convolutionally coded optical M-ary pulse position modulation (PPM) systems, for which decode rates extend into the Gb/s range. To meet these demands, it is imperative to solve the problems of massive hardware size, large power consumption and slow detecting speed for high-speed application.

This project will focus on a review on increasing the detector speed rather than reducing its power consumption. In general, there are two ways to increase the detector speed. One way is to modify the decoder at the algorithm level and another way is to increase the implementation speed at the hardware level. Both of these two methods will be reviewed in this report.

The Organization of this report is as follows. In section II, the convolutional encode and its Viterbi decoder will be briefly reviewed. Algorithm level methods to increase the detecting speed will be discussed in section III. In this section, three different algorithm modification methods will be introduced. First, linear distances to represent the branch metrics will be briefly introduced. Then, a modified Viterbi Algorithm that decodes convolutional codes with noise adaptive and reduced effort is presented. Finally, a reduced state Viterbi algorithm will be illustrated. The hardware level implementation improvement will be addressed in section IV. In this section, the overall hardware

implementation to solve the problems of Viterbi Algorithm such as slow decoding speed, massive hardware size and large power consumption is briefly introduced first. Then, several hardware implementation methods to increase the Viterbi decoder speed are described in detail. A conclusion is given in section V.

II. Viterbi algorithm and convolutional code

2.1 Convolutional code

Convolutional codes are widely used for digital communication system. Given a set of base of the code, symbol is constructed by the convolution of states and base according to the following equation.

$$\mathbf{C}_j = \sum_{l=0}^m X_{j-l} \mathbf{G}_l \quad (1)$$

where C_j is the j th symbol and X_j is the states at j th time and G_l is the l th code base.

Following is an example for four state convolutional code. The symbol C is supposed to be

composed of two elements, i.e., $C = \{0,1\}$, and base is: $G = \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}$, we get:

$$\mathbf{C}_j = \begin{bmatrix} X_{j-2} & X_{j-1} & X_j \\ 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} X_{j-2} + X_{j-1} + X_j & X_{j-2} + X_j \end{bmatrix} \quad (2)$$

Fig 1. shows the diagram of the convolutional encoder for the above example.

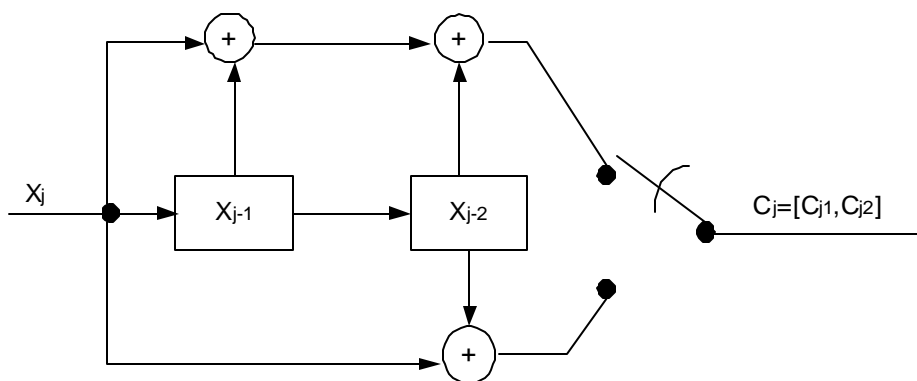


Fig 1. Convolutional Encoder

2.2 Viterbi Algorithm

Viterbi Algorithm (VA) is the optimum-decoding algorithm for convolutional codes. It finds the sequence of symbols in the given trellis that is closest in distance to the received sequence of noisy symbols. This sequence computed is the global most likely sequence.

To compute the global most-likely sequence, the VA first recursively computes the survivor path entering each state. After the survivor paths entering all states are computed, the survivor path that has the minimum path metric is selected to be the global most likely path.

In VA, the branch metric is defined as the distance between the received noisy symbol, y_n and the ideal noiseless output symbol, $C_{i,j}$. The branch metric transition from state i to state j at recursion n is

$$B_{i,j,n} = \sum_{l=1}^K (y_n \oplus C_{i,j})_l \quad (3)$$

The path metric in VA is defined as the distance between the survivor path and the sequence of noisy symbols. $M_{j,n}$, which is the path metric for state j at recursion n , is the most likely path coming into state j at recursion n . It is calculated as follows:

$$M_{j,n} = \min(M_{i,n-1} + B_{i,j,n}) \quad (4)$$

After the most likely transition to state j at recursion n is computed, the path metric for state j , $M_{j,n}$, is updated and this most likely transition is appended to the survivor path of state m at recursion $n-1$ in order to form the updated survivor path coming into state j at the current recursion, n .

The path metrics and the survivor paths are updated for all states at each recursion. At the end of the recursions, the survivor path with the minimum path metric is selected to be the most likely path.

Fig. 2 is a four state of trellis diagram for the Viterbi decoder. The dark bode line is the final survivor path (winner) as it has the minimum path metric.

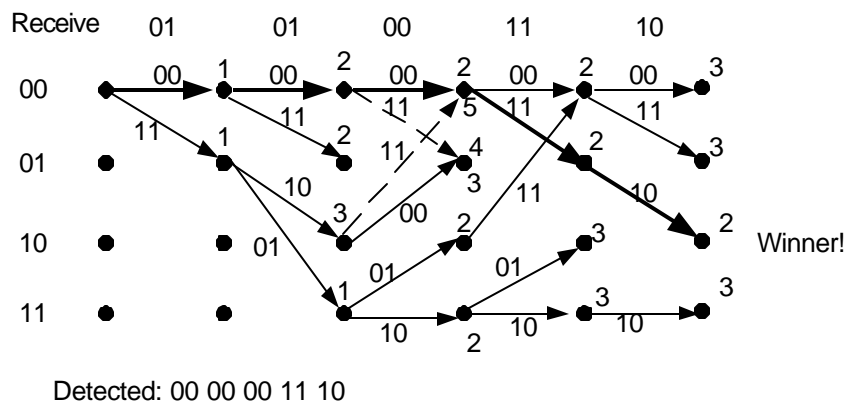


Fig. 2 Four states of a trellis diagram

III. Algorithm level methods to increase the Viterbi decoding speed

VA is the optimal maximum likelihood detection method in AWGN when Euclidean distance is used as a distance measure. Practically, Hamming distance is used to speed up the decoding speed. However, the performance of the VA with Hamming distance as a metric measure is sub-optimal. If we intend to use Euclidean distance to improve the VA performance, we might have to use the duplication of the multipliers to obtain the squared distances for high bit-rate applications. These can increase the decoder complexity significantly. Hui-Ling Lou [1] proposed a method with which linear distances were used to represent the branch metrics, without compromising the Viterbi decoder performance. Therefore, the high complexity caused by the duplication of the multipliers can be avoided while VA performance can still be maintained.

Christian Feldmann [2] proposed a modified Viterbi Algorithm that decoded convolutional codes with noise adaptive and reduced effort while retaining coding gain close to the VA over the range of SNR values for which that algorithm was useful. The algorithm had three levels of prioritized effort. Movement from one level to the next was controlled by parameters that can be selected according to the desired output bit-error-rate performance. The algorithm provided both a method for enhancing the frequency of matches of successive path segments and a means for reducing path metric computations when a match occurs. The algorithm also provided an efficient method for obtaining path metrics when a match did not occur. The number of computations needed to determine path metrics was less than for the normal VA.

With the Viterbi Algorithm, computational complexity increases exponentially with the constraint of length of the convolutional code. So, reducing the constraint length can greatly simplify the implementation of the Viterbi algorithm. In [3], Tong proposed a reduced state Viterbi Algorithm for blind sequence estimation of DPSK sources. The correlation of the deterministic source was estimated from the observation and the Viterbi Algorithm was applied to reconstruct the input symbols. With this estimation, at least half of number of states can be reduced. In [4], Wang Zhaocheng, proposed a reduced state Viterbi Algorithm for multiuser Detection in DS/CDMA systems. It partitioned the total K users into G groups according to their power strengths and interference levels, which reduced the computational complexity per binary decision from $O(2^{K-1})$ to $O(G \cdot 2^{K/G-1})$ for asynchronous DS/CDMA systems in AWGN.

The following two subsections will illustrate the constraint length based modified VA with adaptive effort [2] and reduced state VA algorithm for some special cases [3,4].

3.1 Constraint length based modified Viterbi algorithm with adaptive effort

Christian Feldmann [2] provided a modified Viterbi algorithm that decoded convolutional codes with noise free and reduced effort while retaining coding gain close to the VA over the range of SNR values for which that algorithm was useful. The algorithm was derived from an observation about trellis path segments that, when certain restrictive conditions were met, the state at which two successive and independently derived trellis path segments meet was highly likely to be an error-free state.

The algorithm was applied by collecting sufficient received codeword to define a path segment one constraint-length long. The algorithm groups a small fraction (F1) of the highest ranked path segments in each of two successive time intervals and checks for a match between successive groups. When matches are frequent, which is defined as N_c consecutive matches, path metrics for the top F2 derivative path segments are determined and used to obtain path metrics to states. When there is a match, but there are not consecutive matches, path metrics for the top F3 derivative path segments are determined. When there is no match, maximum path metrics are obtained for a prescribed number (N_s) of states.

The algorithm may be described as the following steps.

- 1) Collect received codeword corresponding to K message bits over the time interval $[t_{i+1}, t_i]$ (see Fig. 3). K is the constraint length. There are 2^K trellis states.
- 2) Determine source and destination states for each of F_1 highest-ranking path segments.
- 3) Determine if there is a path segment match (at t_i) by comparing the source states from step 2 with the destination states of the F_1 highest ranking path segments generated by the codewords received previously in time interval $[t_i, t_{i-1}]$.
- 4) If N_c consecutive path segment matches have been recorded (at $t_{i-1}, t_{i+1}, t_{i-2}, t_{i-3}, \dots$), determine the metrics of destination states (at t_{i+1}) of the first F_2 -ranked path segments, store origin states (at t_i), and return to step 1 for the next time segment.
- 5) Failing step 4, compute the metrics of destination states of all path segments originating in the source state (at t_i) that has the maximum metric.
- 6) If a path segment match was observed in step 3, update the metrics of destination states (at t_{i+1}) from the F_3 highest ranking path segments, store origin states (at t_i), and return to step 1.
- 7) If no path segment match was observed in step 3, update path metrics until a maximum value of metric has been found for destination states (at t_{i+1}), store origin states, and return to step 1.

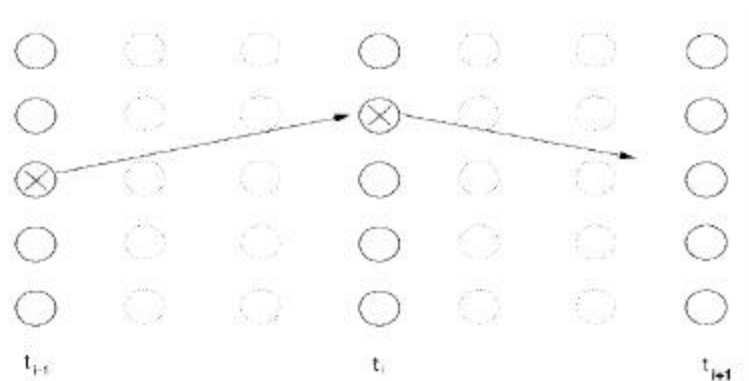


Fig. 3. Conditions of a path segment match. Circles with an X represent trellis states with maximum metric. States separated on the trellis by a constraint length, corresponding to times t_{i-1} , t_i , and t_{i+1} , are shown darkened. Arrows represent path segments over the intervals $[t_{i+1}, t_i]$ and $[t_i, t_{i-1}]$.

Simulations were performed on a $K=7$, rate $r=1/2$ code. It was shown that for 3-bit soft decision detected signals, coding gain within 0.06 dB of Viterbi at 3-dB SNR was achieved for the same

constraint-length code with modest parameter values and computational effort. At values of SNR above 6 dB, the algorithm decoded with very low computational effort.

3.2 A reduced state Viterbi Algorithm

With the Viterbi Algorithm, computational complexity increases exponentially with the constraint of length of the convolutional code. So, reducing the constraint length can greatly simplify the implementation of the Viterbi algorithm. However, there is no generic method to reduce the states in Trellis diagrams. That means, reduced state Viterbi Algorithm is problem oriented. Following will introduce two special cases for this modified VA method. One is for blind sequence estimation of DPSK sources and another is for the multiuser detection of DS/CDMA system.

3.2.1 A reduced-state Viterbi algorithm for blind sequence estimation of DPSK sources

In [5], Tong proposed a blind sequence estimation scheme by exploiting the second order statistics of the source. The correlation of the deterministic source was estimated from the observation and then the viterbi algorithm was applied to reconstruct the input symbols. With this estimation, Tong proposed a way to reduce the number of states in the trellis for DPSK sources without the loss of performance of VA[2]. The model was developed as follows:

First, the received signal at the i th receiver for a QAM data communication system is described as:

$$x_i(t) = \sum s_n h_i(t - nT) + n_i(t) \quad (5)$$

Where s_n is the symbol sequence; T is the symbol baud period; and $h_i(t)$ is the composite channel that includes the pulse shaping filter, the multipath fading channel and the receiver filter. $n_i(t)$ is the additive noise.

When $x_i(t)$ is oversampled by K , i.e., $x_i(t)$ is sampled at $t=l\Delta$, with $\Delta = \frac{T}{K}$, and assuming that $h_i(l)$

lasts a maximum of d symbol intervals, the i th receiver can be expressed by the following equation:

$$x_i(t) = H_i s(t) + n_i(t) \quad (6)$$

Organizing all N receiver together, the input/output relation can be described as

$$\mathbf{x}(t) = \mathbf{H}\mathbf{s}(t) + \mathbf{n}(t) \quad (7)$$

Assume the input sequence is zero mean and orthonormal, and the noise is Gaussian, the autocorrelation of \mathbf{x} is

$$\mathbf{R}_x = E\{\mathbf{x}(t)\mathbf{x}^*(t)\} = \mathbf{H}\mathbf{H}^* + \sigma^2\mathbf{I} \quad (8)$$

\mathbf{R}_x can be decomposed by SVD, and get:

$$\mathbf{R}_x = \mathbf{U}\text{diag}(I_1^2 + \mathbf{s}^2, \dots, I_d^2 + \sigma^2, \sigma^2, \dots, \mathbf{s}^2)\mathbf{U}^* = \mathbf{U}_s(I_1^2, \dots, I_d^2)\mathbf{U}_s^* + \sigma^2\mathbf{I} \quad (9)$$

Define

$$\mathbf{y}(t) = \mathbf{T}_m \mathbf{x}(t) \quad (10)$$

$$r_y^{(k)}(t) = y^*(t)y(t-k) \quad (11)$$

$$\mathbf{r}_s^{(k)}(\mathbf{t}) = \mathbf{s}^*(\mathbf{t})\mathbf{s}(\mathbf{t}-\mathbf{k}) = \sum_{l=0}^{d-1} \mathbf{s}_{t-l}^* \mathbf{s}_{t-l-k} \quad (12)$$

Where:

$$\mathbf{T}_m = \mathbf{U}_s^{-1} \mathbf{U}_s^* \quad (13)$$

$$\mathbf{?} = \text{diag}(\mathbf{?}_1, \dots, \mathbf{?}_d) \quad (14)$$

It has been proven in [5] that, when there is no noise, correlation of $y(t)$ is equal to correlation of $s(t)$, i.e., $r_y^{(k)} = r_s^k$. So, the correlation function of $S(t)$ can be recovered from the observation process and enables a direct application of the Viterbi algorithm to $y(t)$.

An optimal sequence detection defined by $\min_{\{s_n\}} \sum_r |\mathbf{r}_y^{(k)}(\mathbf{t}) - \mathbf{r}_s^{(k)}(\mathbf{t})|^2$ can be achieved by applying the Viterbi algorithm to a M^{d+k-1} state trellis for a M-QAM signaling system where half of the number of states is reduced. (The original algorithm has M^{d+k} states). Note that here $r_y^{(k)}$ is the correlated observation of y with noise.

For differentially encoded DPSK signals, the symbols $s_n = e^{j\theta_n}$ is defined as :

$$s_n = s_{n-k} e^{j\theta_n} \quad (\text{Generally } k=1)$$

Clearly, $e^{j\theta_n} = s_n s_{n-k}^*$. Define $c_n = s_n s_{n-k}^* = e^{j\theta_n}$, then,

$$r_s^{(k)} = \sum_{l=0}^{d-1} c_{n-l}^*$$

The optimal sequence detection $\min_{\{s_n\}} \sum_t \left| \mathbf{r}_y^{(k)}(\mathbf{t}) - \mathbf{r}_s^{(k)}(\mathbf{t}) \right|^2$ turns out to be $\min_{\{c_n\}} \sum_t \left| \mathbf{r}_y^{(k)}(\mathbf{t}) - \mathbf{r}_s^{(k)}(\mathbf{t}) \right|^2$

The output of the VA becomes $\{\mathbf{c}_n\}$ instead of $\{\mathbf{s}_n\}$. The detection can be achieved by applying the VA to a M^{d-1} state trellis for a M-PSK system when $\{\mathbf{c}_n\}$ has the same number of terminal phase as $\{\mathbf{s}_n\}$, and this can always be ensured by proper initialization in the differential encoding. The original bits can now be decoded directly from the symbol $c_n = e^{j\theta_n}$ based on the encoding rule. Thus, the number of states is reduced by a factor of M^k .

3.2.2 A reduced state Viterbi algorithm for multiuser detection in DS/CDMA systems

Optimum multiuser detector was proposed to eliminate the multiple access interference (MAI) in matched filter and increase the system capacity in CDMA systems. However this optimum detector requires a computational complexity that grows exponentially with the number of users $O(2^{K-1})$, which is unacceptable in many practical applications. Therefore, for this optimum multiuser detector in DS/CDMA systems, the states of the trellis to which VA is applied are quite large. In contrast, the memory length is very small. Based on the peculiarity of this trellis, Wang [4] proposed a novel reduced state Viterbi detector called Group Partitioning Viterbi Detector (GPVD), which unified the optimum multiuser VA and the simple successive interference cancellation scheme into one framework and offered a range of thorough trade-off. The basic idea of GPVD was to demodulate the strongest users of Group 1 which had the highest power levels by Viterbi Algorithm first, and then to cancel their signals as far as possible from the received waveform. The process was repeated until the weakest users of Group G which had relatively small power levels were demodulated.

IV. Hardware level methods to increase the Viterbi decoding speed

So far, Lots of methods have been proposed to increase the Viterbi detecting speed in hardware level.

The simplest method to increase throughput is to decompose the input streams into blocks of length M that can be processed in parallel using k conventional Viterbi decoders. However, independent block processing requires knowledge of the initial state metrics which are unknown until the previous block is processed. Hence, without additional information, this approach is of no practical use.

Two practical approaches to block-based decoding are state initialization and interleaving. State initialization achieves block independence by forcing the encoder to a known state at the start of each block. This technique reduces the information rate and adds complexity to the decoder which must obtain block synchronization from the received data. Interleaving achieves block independence by interleaving independently encoded sequences. By definition, the blocks are independent and hence can be decoded independently. However, the encoded sequences are interleaved prior to the addition of noise. This approach is not applicable to digital sequence detection in the presence of intersymbol interference because the encoder is the intrinsic impulse response of the channel and hence cannot be separated from the additive noise component of the channel.

Another block-based approach is the sliding block decoder. This approach is based on modeling the Viterbi algorithm as a time invariant, nonlinear digital filter with finite memory. Table lookup is the proposed method to implement the nonlinear filter. However, this approach is infeasible as a huge ROM is needed even for a very small states application.

Peter J.Black proposed an alternative block-based method named Sliding Block Viterbi Decode (SBVD). It can be applied to any Viterbi decoding problem without constraining the encoding process [6]. The SBVD combined the filtering characteristics of a sliding block decoder with the computational efficiency of the Viterbi algorithm. This approach reduced decode of a continuous input stream to decode of independent overlapping blocks, without constraining the encoding process. A systolic SBVD architecture was implemented by combining forward and backward processing of the block interval [6].

Boo, M. proposed four different solutions for increasing the computation speed of the VA [7]. The first solution was based on the utilization of look ahead techniques to formulate an efficient scheduling method to map the data onto the pipelined ACS unit. The second solution was based on the duplication of some hardware modules in order to break the critical path of the system. The third way was based on the pipelined area efficient solution. And the fourth way was based on the utilization of simplified comparators in order to reduce the complexity of the system's critical path. In all, three of them were based on the utilization of pipelined system combined with efficient scheduling methodologies and the other one was based on the simplification of ACS recursion to cut down the critical path of the system.

Other issues in high-speed application such as massive hardware size, large power consumption are also widely discussed. Kubota, S. [8] proposed two new schemes for high-speed and universal coding rate VLSI Viterbi decoders to reduce hardware size and power consumption. Montse Boo [9] presented another methodology in which an area efficient architecture was implemented, basing on the perfect shuffle, inverse concatenation and inverse decimation operators. However, these topics will not be addressed in this report. Rather, the hardware implementation for increasing the Viterbi Detector will be focused on.

In 4.1, Full-connected M-step trellis (M-step plus 1-step) method to increase the computation speed of the Viterbi Algorithm [7] will be described. And In 4.2, The SBVD approach [6] will be introduced.

4.1 Full connected M-step trellis implementation

The central unit of a Viterbi decoder is the data dependent feedback loop, which performs an add-compare select (ACS) operation. This nonlinear recursion presents the bottleneck for a high-speed parallel implementation. Therefore, the utilization of VA in high-speed applications requires breaking the ACS recursion. Look ahead techniques have been proposed in early to increase the parallelism of the Viterbi algorithm in such a way that more than N ACSs can be enabled at the same time. The strategy is based on the definition of the M-step trellis which is built by observing the transitions in intervals of length Mt , i.e. from nMt to $(n+1)Mt$ with $n=1, 2, \dots, M$. Boo [7] proposed a methodology based on the utilization of this look-ahead techniques to formulate an

efficient scheduling method to map the data onto a pipelined ACS unit. It computes a full connected M-step trellis to exploit in an efficient way the pipelining of a unique 1-step ACS unit where the 1-step trellis is computed. Thus, the computation of a M-step trellis permits finding a scheduling method to break the ACS recursion and, in this way, to compute the 1-step trellis in a pipelined ACS unit with a 100% pipeline stage utilization.

The general structure of the architecture model is presented in fig.4. The 1-step ACS unit has S pipeline stages that can be exploited in an efficient way through computation of the M-step trellis in a M-step ACS unit; i.e. the M-step ACS unit supplies data to the 1-step ACS unit as soon as it is required, without any clock cycles losses.

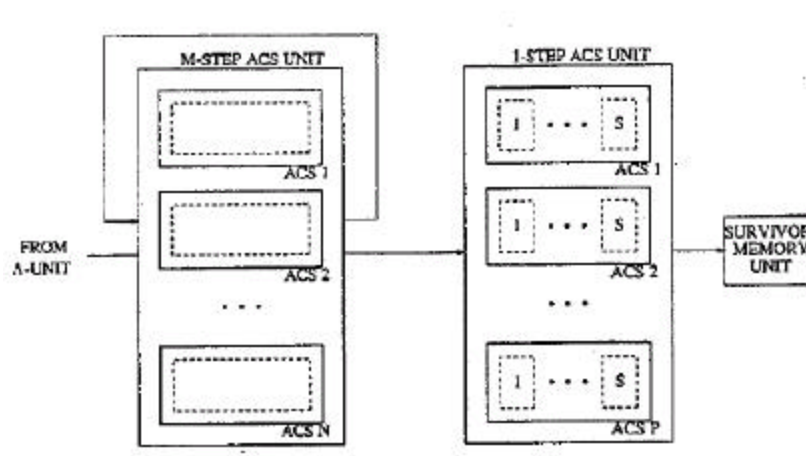


Fig. 4 The 1-step ACS unit has through computation of the M-step trellis in a M-step ACS unit, the M-step ACS unit supplies data to the 1-step ACS unit as soon as it is required, without any clock cycles losses.

M. Boo [7] also proposed another pipelined solution, which was based on the updating of PMs using carry-save adders and on the recirculation of all these possible PM values without waiting until the comparison operation is finished on the pipelined architecture.

The decisions related to each state cannot be realized until the comparison operations are finished. In a pipelined system, all the alternative paths have to be considered in the next stages and eliminated when the comparison operations finished. In this way, the speed of the system is improved by computing the different possible paths to each state, and the selection of the final paths are made afterwards. However, such architectures are not area-efficient, because a k-fold

increase in throughput requires a kN -fold increase in hardware complexity, where N is the number of states. Further throughput increases require an alternative approach to look ahead, if the overall area efficiency is to be maintained. Block-based decoding were proposed to avoid these disadvantages. However, some implementations of this method had some disadvantages (see previous content in section III) that made this method impractical. Peter J.Black proposed a novel block-based detecting method named Sliding Block Viterbi Decode (SBVD) [6] to make it realizable.

4.2 Sliding block Viterbi decoder

Sliding Block Viterbi Decoder (SBVD) approach proposed by Peter J. Black [6] reduced decode of a continuous input stream to decode of independent overlapping blocks, without constraining the encoding process. It combined the filter characteristics of the sliding block decoder with the computational efficiency of a conventional Viterbi decoder. The systolic SBVD implementation proposed by Peter J. was similar to the systolic minimized architecture with some major differences. The minimized method was proposed as an extremely efficient block-based algorithm and claimed to be minimal with respect to hardware complexity. The hardware complexity of SBVD architecture was similar to, or even less than that of the minimized method. Furthermore, SBVD provided a true maximum likelihood algorithm given the constraint of block decoding. Therefore, the decoding performance of the SBVD method always upper bounded the minimized method for the same block length.

The basic idea of SBVD is from the fact that the survivor paths from all possible starting states merge with high probability L iterations back into the trellis [10]. The parameter L is the well-known survivor path length and is typically $5v$ [10]. Similarly, when starting with unknown initial state metrics (typically set to zero), the state metrics after K trellis iterations are independent of the initial metrics. That means the survivor path will merge with the true survivor path as if the initial metrics had been known. The parameter K is the synchronization length and is typically $5v$ [11]. Therefore, the state at time n can be decoded using only information from the interval $n-K$ to $n+L$. Such a decoder is a sliding block decoder, a time-invariant nonlinear digital filter with finite memory and delay.

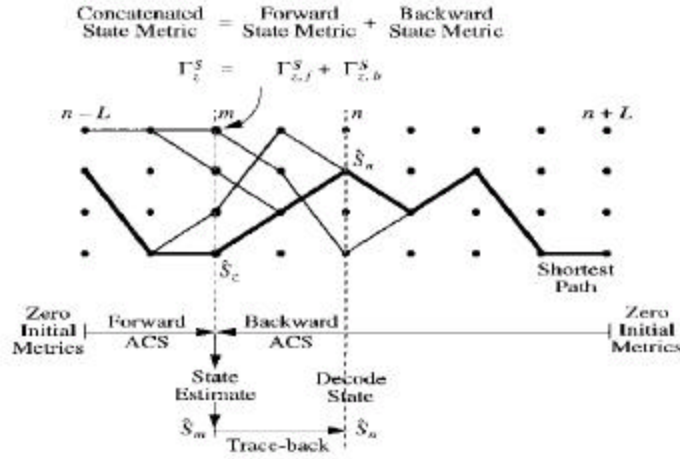


Fig. 5 Hybrid Viterbi algorithm for selecting the shortest path through a trellis of finite length (four-state trellis example).

The sliding block decoder output is equal to the decision of the state at time n that lies on the shortest path (survivor path) through the trellis from $n-L$ to $n+L$, which is equivalent to maximum likelihood decoding given a finite block length. The Viterbi Algorithm is normally run in the direction of increasing time (forward) to avoid storing the entire sequence before decoding can start. Given a finite length trellis corresponding to the block interval, the shortest path can be found by running the algorithm forward or backward (by reversing the trellis transitions). Here, K is chosen to be equal to L to make the reversibility.

A general hybrid Viterbi Algorithm is derived by combining forward and backward processing of the interval. Fig. 5 shows a hybrid Viterbi algorithm for selecting the shortest path through a four-state trellis of finite length. At some trellis iteration m , within the interval, the shortest path must pass through one of the four possible states. Forward processing of the interval $n-L$ to m yields four survivor paths corresponding to the shortest paths from $n-L$ to each state at time m . Similarly, backward processing of the interval $n+L$ to m yields the shortest paths from $n+L$ to each state at time m . For a given state at time m , the shortest path over the interval through this state must be the concatenation of the forward and backward shortest paths for this state and the state metric for a concatenated path is the sum of the forward and backward state metrics. Hence, selecting the state \hat{S}_m at time m with the smallest concatenated state metric yields the starting state for trace-back of

the shortest path. If $m=n$ then \hat{S}_n can be decoded directly. Otherwise trace-back from m to n is required.

Forward and backward algorithms are the special cases of hybrid Viterbi Algorithm, which correspond to $m=n+L$ and $m=n-L$. Minimized method is another special case corresponding to $m=n$.

The SBVD proposed by Peter [6] applied this hybrid VA to the interval from $n-M/2-L$ to $n+M/2+L$ and decode the interval $n-M/2$ to $n+M/2$. It has the following two advantages.

1. The relative area efficiency is increased compared to the relative area efficiency of applying hybrid VA to interval $2L$. Each block of M decodes outputs requires $M+2L$ trellis iterations and hence the relative area efficiency is $M/(M+2L)$. As the decoded block length increases, the SBVD approaches ideal linear scaling. The relative area efficiency of applying hybrid VA to interval $2L$ is $1/2L$.
2. SBVD method is a true maximum likelihood algorithm (under the constraint of using block decoding), selecting the path from the set of all possible paths that is closest to the observed output over the finite observation interval. The minimized method is not a maximum likelihood algorithm because the estimates of the states at either end of the decode block are not based on all of the available data. So the coding gain of the SBVD method always upper bounds the minimized method for the same interval parameters.

This SBVD architecture can be applied to high-speed Viterbi decoding of convolutional codes of any number of states. The hardware overhead is independent of the number of states but proportional to the ratio of the survivor path length L and the block length M which can be made arbitrarily small. This relatively small overhead enables codes of large constraint lengths to be decoded as efficiently as codes of shorter lengths without a theoretical speed limit.

V. Conclusion

In this report, algorithm modification and hardware implementation to increase the Viterbi decoder speed have been introduced.

One way of the algorithm modification is that using linear distances instead of Euclidean distance to represent the branch metrics, without compromising the Viterbi decoder performance [1]. Therefore, the high computation complexity caused by duplication of the multipliers can be avoided while VA performance can still be maintained. A constraint length based modified Viterbi algorithm with adaptive effort [2] is another approach to increase the decode speed. With this method, output error performance, which depends on the selection of values for certain parameters, can be made arbitrarily close to maximum likelihood. This algorithm was based on the phenomenon of path segment matching and the observation that the trellis state at which a match occurs is highly likely to be error-free, on the use of small sets of likely path segments to enhance match conditions, and on development of an efficient procedure for obtaining path metrics. As computational complexity increases exponentially with the constraint of length of the convolutional code, reducing the constraint length can greatly simplify the implementation of the Viterbi algorithm [3,4].

Several methods of hardware implementation to increase the Viterbi decoder speed have been introduced in detail [6,7,10,11]. One solution was based on the utilization of look ahead techniques to formulate an efficient scheduling method to map the data onto the pipelined ACS unit [7]. The duplication of some hardware modules can break the critical path of the system so that decoder speed can be increased [7]. The SBVD [6] approach reduced decode of a continuous input stream to decode of independent overlapping blocks, without constraining the encoding process. A systolic SBVD architecture was presented to combine forward and backward processing of the block interval.

While hardware implementation improvement to increase the Viterbi decoder is generic to most of the application, algorithm modification is problem oriented. For example, linear distances representation for the branch metric produced by Lou [1] was deduced from Trellis Coded Modulation schemes (TCM) with AWGN channel. It hasn't been proved that this method can be used for other cases. Similarly, reduced state implementation produced by Tong [3] is for DPSK sources with AWGN channel. However, algorithm modification is relative cheaper and efficient. For example, with a reduce state Viterbi algorithm for blind sequence estimation of DPSK sources, the states can be reduced at least half. And since computational complexity increases exponentially

with the constraint of length of the convolutional code, half reduction of the state can increase the decode speed greatly.

Some interesting problems that haven't been proposed so far are: can we increase the decoder speed infinitely if we have infinite hardware? If not, what's maximum speed we can achieve? Is there optimal partitions given the size of the source needed to be decoded so that we can achieve maximum decoding speed and use minimum hardware?

Peter [6] claimed that SBVD enabled codes of large constraint lengths to be decoded as efficiently as codes of shorter lengths without a theoretical speed limit. However, the decoder speed of SBVD architecture is still upper bounded by the survivor path length L . Can we increase the decoder speed more? Or say, can we decode the convolutional codes within shorter path length such as $\frac{1}{2}L$ or even shorter? Theoretically it should work if we decode the convolutional codes with shorter path length. But we need to navigate all the possible initial states. That means we need more hardware. Besides, we need to combine all possible combination of two segments with different initial states. And select the optimum one. In this way, we can keep on doing this, and decode the convolutional codes in the path length $\frac{1}{3}L$ or in the path length $\frac{1}{4}L$ and combine all possible combinations of three or four segments with different initial states. And select the optimum one...So, What's the optimum partition? Or is there any optimum partition? If there is, how to mathematically describe it? It is a minimization problem with some constraints. And if there is a solution, how to solve it? Can we use the Lagrange multiplier to solve it? Is the range of variable convex? These are problems which haven't been proposed so far, but which might be interesting and important problems.

References

- [1]. Hui-Ling Lou "Linear distances as branch metrics for viterbi decoding of trellis codes", Acoustics, Speech, and Signal Processing, 2000. ICASSP '00. Proceedings. 2000 IEEE International Conference on , Volume: 6 ,Page(s): 3267-3270
- [2]. Feldmann, C.; Harris, J.H. "A constraint-length based modified Viterbi algorithm with adaptive effort", IEEE Transactions on Communications, Volume: 47 Issue: 11 , Nov. 1999 Page(s): 1611 –1614
- [3]. Tongtong Li; Zhi Ding "A reduced-state Viterbi algorithm for blind sequence estimation of DPSK sources" , Global Telecommunications Conference, 1999. GLOBECOM '99 , Volume: 4 , 1999, Page(s): 2167 -2171 vol.

- [4]. Wang, Z.; G. Ning; Yao, Y., "A reduced state Viterbi algorithm for multiuser detection in DS/CDMA systems" Communication Technology Proceedings, 1996. ICCT'96., 1996 International Conference on , Volume: 2 , 1996 Page(s): 1102 -1105 vol.2
- [5] L. Tong, "Blind sequence estimation", IEEE Trans. Commun., Vol. 43, pp 2986-2994, Dec. 1995
- [6] Peter J. Black, etc. "A 1-Gb/s, Four-State, Sliding Block Viterbi Decoder", IEEE Journal of Solid-State Circuits, Vol. 32, NO. 6, June 1997, pp 797-804
- [7]. Boo, M.; etc, " Pipelined architectures for the Viterbi algorithm", TENCON '97. IEEE Region 10 Annual Conference. Speech and Image Technologies for Computing and telecommunications., Proceedings of IEEE , Volume: 1 , 1997 Page(s): 239 -242 vol.
- [8]. Kubota, S.; Kato, S.; Ishitani, "T Novel Viterbi decoder VLSI implementation and its performance", Communications, IEEE Transactions on , Volume: 41 Issue: 8 , Aug. 1993 Page(s): 1170 -1178
- [9]. Boo, M. etc., "High-performance VLSI architecture for the Viterbi algorithm", IEEE Transactions on Communications, Volume: 45 Issue: 2 , Feb. 1997 Page(s): 168 -176
- [10] G. C. Clark and J. B. Cain, "Error-Correction Coding for Digital Communications", New York: Plenum, 1981, pp. 227–264.
- [11] A. J. Viterbi and J. K. Omura, "Principles of Digital Communication and Coding", New York: McGraw-Hill, 1979, pp. 258–261.
- [12]. Lou, H.-L, "Implementing the Viterbi algorithm", IEEE Signal Processing Magazine , Volume: 12 Issue: 5 , Sept. 1995, Page(s): 42 -52