Design Of A Fast Sequential Decoding Algorithm Based On Dynamic Searching Strategy*

Wen-Wei Yang

Li-Fu Jeng

Chen-Yi Lee

Dept. of Electronics Engineering, National Chiao Tung University,
1001, University Road, Hsinchu 300, Taiwan, ROC

Tel: 886-35-712121 ext.3286; Fax: 886-35-724361; Email: cylee@cc.nctu.edu.tw

ABSTRACT

This paper presents a new sequential decoding algorithm based on dynamic searching strategy to improve decoding efficiency. The searching strategy is to exploit both sorting and path recording techniques. By means of sorting, we can identify the correct path in a very fast way and then, by path recording, we can recover the bit sequence without degrading decoding performance. We also develop a conditional resetting scheme to overcome the buffer overflow problem encountered in conventional sequential decoding algorithms. Simulation results show that for a given code, decoding efficiency remains the same as that obtained from maximum likelihood function by appropriately selecting sorting length and decoding depth. In addition, this algorithm can be mapped onto an area-efficient VLSI architecture to implement long constraint length convolutional decoders for high-speed digital communications.

1. INTRODUCTION

Convolutional coding with Viterbi decoding is widely accepted as an efficient method to achieve a significant power gain on digital communication channels with low to moderate signal to noise ratios. Generally speaking, for information bit rates above 5 Mbits/s, the decoder must be based on a fully parallel implementation of the Viterbi algorithm, which often demands one complete cycle in each clock interval. The speed requirements impose timing constraints on each module in the decoder, such as in add-compare-select (ACS) module, and path storage and selection (PSS) module. As constraint length ν becomes large, the hardware complexity will increase a lot and hence the design cost becomes huge or even not feasible for present technology. Due to this consideration, it is very often found that the constraint length is limited, say 7 in many available hardware solutions found in the literature [1,2,3].

On the other hand, sequential decoding [4,5] has low computation complexity in search of a correct path. However the decoding rate becomes lower because a correct path can only be identified after several trials on incorrect paths due to error bits. The principal sequential decoding algorithms are the Fano [4] and the stack-based [6]. Although these algorithms are useful for longer constraint length codes, they suffer from "buffer overflow" which derives from an inability to maintain uniform decoding rates while searching message sequences. Previously, we presented a dedicated memory structure [7] to speed up the decoding rate. However the buffer overflow still

remains to be solved. We recently develop a new high-speed data sorter [8] which produces a sorted sequence right after input samples are given. This technique provides us a very powerful hardware solutions for those applications which require massive sorting operations. In principle, the sequential decoding algorithm can be regarded as a set of sorting operations working on the survived nodes to identify a candidate node. And from this candidate node, further tracking can be performed until all input sequences are decoded. This modified sequential decoding or named fast sequential decoding (FSD) algorithm will be discussed in more detail in Section 2. In addition, a path recording technique is also developed to overcome back-tracking of correct bit sequences and to improve throughput rate. Section 3 provides some strategies to improve coding gain. Some simulation results obtained from this algorithm will also be provided to illustrate the decoding efficiency of this modified algorithm.

2. THE FAST SEQUENTIAL DECODING ALGORITHM

The modified version is named fast sequential decoding or FSD algorithm. It mainly consists of sorting and path recording kernels which will be explained below.

2.1. The Sorter Kernel

We assume that both hard decision and soft decision methods can be handled. However, to make this algorithm more readable, we only consider the hard decision method and use (2,1,3) code as an example to illustrate the decoding process.

The code generator is given in Figure 1(a). We first construct the trellis diagram indicating the diverting path at each node as shown in Figure 1(b). For correct transmission of one bit, the cost will be decreased by -1, while for incorrect transmission, the cost will be increased by 10 if p is 97% [5]. Thus one of 3 different costs (-2, 9, 20) will be added to each diverted node according to the difference between transmitted and received patterns. For example, if "10" is transmitted and "11" is received, then the cost to be added will be 9.

If an error-free channel is considered, then the correct path can easily be identified if we follow the node based on the minimum weight as shown in Figure 2(a). It can be found that search of the correct path is now formulated as a minimum tracking scheme working on the weights of all survived nodes. If we start from the root and select the node with the less cost of the two candidate nodes, we can always find the correct path and then the transmitted sequence can be fully recovered. Thus it is not necessary to track all nodes as those required in maximum likelihood function since only the node with minimum weight is needed for each decoded bit.

If we take noise effect into account, then the scheme just mentioned cannot be exploited. This is because the noise may cause the scheme to select an incorrect path which cannot be

*Work supported by the National Science Council of Taiwan, ROC, under Grant NSC-83-NSPO-B-RDD-009-01.

Fig.1. (a) gives an example of a (2,1,3) encoder and (b) shows the corresponding trellis diagram.

Fig.2. Path search by the fast sequential sorter decoding scheme using the (2,1,3) code given in Figure 1. (a) Without noise effect and (b) With noise effect. Note that the mask area indicates the survived nodes after decoding a certain bit sequences. It can be found that more storage space is requested for transmission on noisy channel.

revised back to the correct one. This is shown in Figure 2(b), where 2 error bits are occurred consequently. The correct path cannot be immediately identified because of lower weights resulting from the error bits. However if the decoding depth is taken further, the weight from the incorrect path becomes higher than those of the correct path. Then after a certain steps of bit decoding, we can move back to the correct path and continue estimating the weights of nodes on the correct path. This implies that a sorter can be used here to track the minimum weight of the survived nodes. If no errors occur, only the node with the minimum weight is replaced by the new node which inherently has the new minimum weight. The other node is placed to a certain position according to its weight. The insertion of this node is important because, *by no errors, we mean that two bits are simultaneously correct or wrong*. Therefore it is necessary to keep this node and its weight for possible revisit later. However if only one error occurs, we arbitrarily select one, e.g. "1" path, from two diverting paths since both nodes have the same cost, say 9 in our example. The first element is deleted and then both new weights are inserted at the neighboring positions according to their sorted order. If, by luck, we select the correct path, then the weights of this path are getting lower; otherwise after detecting bit sequences on several nodes, the "0" path will then be selected. From these discussions, we found the search of correct path can be formulated as a sorter-based scheme again. In other words, if we always start from the minimum weight, we can identify the correct path in a very fast way, even there is noise-effect. Thus this sorter-based scheme can be summarized as follows:

1. reset the weights of all nodes and start cost estimation from root;
2. use the weight of the first element to generate two new weights and record the decoded bit; and in the mean time, extract the decoded bit sequence when pre-defined depth is reached.
3. sort the weights of all nodes according to ascending sequence;
4. if input sequences are not terminated, then go back to step 2 for bit decoding; otherwise output the decoded bit sequence.

Since sorting requires a lot of operations, we can reduce this computation complexity by checking the error patterns. If one error bit is detected, we only have to sort once since both candidates have the same weight. However if no or two error bits are detected, we can replace the first element by the new minimum weight and then insert the other weight according to its sorted order. Thus in this way, the required sorting operations can be reduced by half. In summary, the approach is to dynamically search the node with minimum weight from survived tree nodes.

2.2. Speed Up Decoding Process by Path Recording

When the correct path is identified, we need to track back for a certain depth d because of the limit of storage space in the sorter. This back-tracking strategy often requires the use of extra storage space and hence cost generation and accumulation becomes idle or another storage unit is needed to speed up performance.

However this back-tracking strategy can be avoided because when a candidate node is selected, we already know that either "0" or "1" path is assigned to this node as shown in Figure 3. Thus each node is already combined with a possible decoded bit information. The new bit sequence is obtained by shifting

Fig. 3. Illustration of the path recording scheme.

# of traced nodes	sorted weight sequence	decoded bit order	recorded path info
1	-2	1	1
2	20	1	0
3	18	2	11
4	20	1	10
5	-8	3	111
6	16	3	1110
7	18	2	110
8	20	1	10
9	-8	4	1111
10	14	4	1110
11	16	3	110
12	18	2	10
13	20	1	0
14	-10	5	11111
15	12	5	11110
16	14	4	1110
17	16	3	110
18	18	2	10
19	20	1	0

Fig. 4. Illustration of the sorting and path recording techniques for the path search given in Figure 2(a).

up the old bit sequence and inserting one bit (either "1" or "0"). Then this new sequence is written to the position as identified by the sorter. The path information identified by the minimum weight is shifted up one bit and then conditionally loaded into the output buffer. Therefore if the decoding depth d is taken appropriately, then the correct bit can be obtained from the path recorder identified by the first element of the sorter while the weight calculation can still be performed simultaneously. This path-recording strategy is very efficient in generating the correct bit sequence since no back-tracking is needed.

It should be noted too that the decoded bit obtained from the path recorder relies on the condition that the depth d should be reached before it is passed to the output buffer. This is used to ensure that all bits obtained are from the correct path. However since in the decoding process, part of the survived nodes are from incorrect paths and may temporarily stay at the first position of the sorter. If we always extract bit sequence from the first element, error bits will be inserted into the output bit sequence. Therefore we have to make sure that output bits can be extracted from the path recorder only when the specified depth d is reached. Fortunately, a status register which specifies the order of output bit sequence can be used for this purpose. This status register serves two purposes: one is for the address to access the input buffer when input bit patterns are to be detected again, i.e. path tracking from the survived nodes has to be done. The other is for depth detection to see if the specified depth is reached.

Combining both sorting and path recording techniques, we can search the correct path and obtain the output bit sequence in a very efficient way. Figure 4 shows the results for these two techniques, where the data are based on the path search given

Fig. 5. Path loss due to burst errors: (a) the weights of survived nodes remain almost the same so that new weight cannot be stored in the case of 1 bit-error and (b) removing path loss by resetting strategy to allow new weights can be stored.

in Figure 2(a). It is clearly that three important parameters are updated simultaneously. Each time when a new node is detected, one more space is created to record the required information, such as decoded bit order, new weight, and path information.

3. STRATEGY TO IMPROVE CODING GAIN

There are two issues to be considered here: *one is the finite precision and finite length problem; the other is the relation between decoding depth and sorter length.*

We have found that in some cases, the decoding process becomes fail because a loop is detected or the new generated weight cannot be stored in the the sorter due to burst errors. In this case, the weights of survived nodes are almost the same as illustrated in Figure 5(a), which implies that some important paths can never be recovered. Therefore the new produced weights cannot be stored in the sorter and hence a loop is detected or a path is lost. To solve this problem, we apply a reset method to initialize the decoding process by using current decoded bit sequences. This is performed by resetting the sorter content as done in the initial phase so that the most recent nodes can be stored. We can use the previous decoded bit sequences to ensure that correct codes are generated for comparison as shown in Figure 5(b). In the meantime, the finite precision and length problem can be solved because of this reset strategy.

As for decoding depth d and sorter length L , we can derive a rule to specify these two parameters. One requirement is that the decoding depth should be at least double of the constraint length ν to allow decoded sequences be corrected. Thus the decoding depth should be meet the requirement: $d > (2 \times \nu)$. The minimum requirement for the sorter length is $(d + 1)$ to temporarily keep survived nodes on the correct path. However this size is not sufficient because there are 2^{d-1} nodes for depth of d , implying that the sorter length L increases exponentially as d increases. From simulation, we can reduce L by considering that, at most, $(\frac{1}{\nu} \times 2^{d-1})$ nodes are needed, where L can be estimated more accurately by taking into account channel error rate. Thus the selection rules for d and L of a code with constraint length ν are:

- $d > (2 \times \nu)$
- $(d + 1) < L < (\frac{1}{\nu} \times 2^{d-1})$

Fig.6. Storage length L and coding depth d vs. decoding error rate. (a) coding depth vs. decoding error rate with $L = 64$ and (b) storage length vs. error rate with $d = 16$.

3.1. Simulation Results

Two parameters are to be determined. One is the length L of sorter and the other is the depth d of path recorder. Since these two parameters are very related and dependent on channel noise level, we first assume that the L is fixed and then change d to estimate the correction rate at different noise levels. The results are given in Figure 6(a). Then based on a selected depth d , we reduce the length L until errors are detected at different noise levels. Figure 6(b) shows the results of L vs. noise levels at a fixed depth d . For a selected channel with known error rate, a pair of suitable L and d can be determined. However it should be noted that if L or d is below a certain level, the correct bit sequence can never be correctly recovered.

We have tested this algorithm using the output of a VBR coder recently developed in our Laboratory [9]. Simulation results show that the decoding performance is the same as or, in some cases, even better than that obtained from maximum likelihood function. These results are similar to those found in [10]. In addition, this algorithm is also useful for long constraint length since only L and d have to be adjusted. For example, if 2% of error bits and (2,1,7) code are assumed, the best length L of the sorter is selected as 96 for a depth d of 16. That is only the first 96 elements are considered to be the candidate nodes for tracking the correct path. If the length is less than 96, probably some of the error bits cannot be corrected.

4. CONCLUSION

In this paper, we have presented a new sequential decoding algorithm for high-speed digital communications. The use of both sorting and path recording techniques not only solves the low-throughput problem but also provides a solution for long constraint convolutional code designs when mapped onto shift register array architecture. Thus a cost-effective solu-

tion for high-speed convolutional codes can be achieved. We are currently developing a proto-type VLSI chip to implement the proposed algorithm for HDTV and high-speed networking applications.

Acknowledgement: The authors would like to thank their colleagues within the SI2 group of National Chiao Tung University for many fruitful suggestions and discussions.

REFERENCES

- [1] T. Ishitani, K. Tansho, N. Miyahara, S. Kubota, and S. Kato, "A Scarce-State-Transition Viterbi-Decoder VLSI for Bit Error Correction", IEEE Journal of Solid-State Circuits, Vol. sc-22, No. 4, August 1987, pp. 575-581.
- [2] D.J. Coggins, D.J. Skellern, R.A. Keaney, and J.J. Nicolas, "A Comparison of Path Memory Techniques for VLSI Viterbi Decoders", Proc. of VLSI'89, Munich Germany, Aug. 1989, pp. 378-388.
- [3] E. Paaske, S. Pedersen, and J. Sparso, "An Area-Efficient Path Memory Structure for VLSI Implementation of High-Speed Viterbi Decoders", Integration, the VLSI Journal, 1991, pp. 79-91.
- [4] R.M. Fano, "A Heuristic Discussion of Probabilistic Decoding", IEEE Trans. Inform. Theory, Vol. IT-9, pp. 64-73, 1963.
- [5] A.J. Viterbi and J.K. Omura, "Principles of Digital Communication and Coding", McGraw-Hill, 1979.
- [6] J. Felinek, "Fast Sequential Decoding Algorithm Using a Stack", IBM J. Res. Develop., pp. 675-685, Nov. 1969.
- [7] C.Y. Lee, F. Catthoor, and H. De Man, "Breaking the Bottleneck of Sequential Decoding for High-Speed Digital Communication", Proc. of ICASSP, Toronto, May 1991.
- [8] C.Y. Lee, J.M. Tsai, and P.W. Hsieh, "High-Speed Median Filter Designs Using Shiftable Content Address Memory", Revised for publication in IEEE Trans. on CAS for Video Technology, June 1993.
- [9] S.C. Juan and C.Y. Lee, "A New Multi-Path Tree-Search FSVQ Architecture for Image/Video Sequence Coding", in Workshop of IEEE Visual Signal Processing and Communications, Sep. 20-22, 1993, Melbourne, Australia.
- [10] T.M. Gould and J.H. Harris, "Single-Chip Design of Bit-Error-Correcting Stack Decoders", IEEE Journal of Solid-State Circuits, Vol. 27, No. 5, May 1992.