IJREAT International Journal of Research in Engineering & Advanced Technology, Volume 1, Issue 1, March, 2013
ISSN: 2320 - 8791
www.ijreat.org

Area and Power Efficient Viterbi Decoder for Storage Devices

Mr.C.Rathina Kumar[1] & Mrs.P.Latha[2]
[1]*PG Scholar, Department of Electronics and Communication Engineering*
[2]*Associate Professor, Department of Electronics and Communication Engineering*
RMK Engineering College, Kavaraipettai, Anna University, Chennai

Abstract
The demand for low power, low cost and less area utilization for Viterbi decoding especially in wireless communication are always required. Viterbi algorithm is widely used as a decoding technique for convolutional codes as well as a bit detection method in storage devices. In this paper the newly proposed Viterbi decoder has an architecture that is incorporated with modification of Branch Metric Unit and Path Metric Unit in Viterbi decoder block which can efficiently reduce the power consumption as well as area. This work analyzes the design complexity of the Viterbi decoder by applying most of the known Very Large Scale Implementation (VLSI) techniques. Through the simulation results we conclude that the proposed Viterbi decoder has strong error correction and low power consumption. Modelsim is used as the software tool for functional verification.

Keywords: Convolutional Encoder, Viterbi Algorithm, VLSI, Branch Metric Unit, Path Metric Unit.

1. Introduction

Most digital communication systems nowadays convolutionally encoded the transmitted data to compensate for fading of the channel, quantization distortions and other degradations effects. For its efficiency the Viterbi algorithm (VA) has proven to be a very practical algorithm for forward error correction of convolutionally encoded messages [1]. When a potentially corrupted sequence of symbols is received, the VA determines the most likely transmitted sequence by exploiting a maximum likelihood criterion.

The idea behind the Viterbi Decoder (VD) is quite simple, in spite of its inherent implementation difficulty. Moreover, there is a wide gap in complexity with the transmission side, where convolutional codes are represented by a state trellis, the decoder is a finite state machine that explores the transitions between states, stores them in a large memory and comes to a final decision on a sequence of transitions after some latency due to the constraint length of the input code. Decisions are usually taken by considering the transition metrics among states, which are updated in terms of either Eucledian or Hamming distance with the error-corrupted received sequence [1]. The performance of convolutional codes strongly depends on their minimum distance, which in turn depends on the constraint length and the coding rate [2]. Section 2 gives a brief description of convolutional codes and the Viterbi Algorithm and introduces the parameters that influencing the design complexity. Section 3 discusses about the design decisions for every sub unit of the decoder. Section 4 discuss about the results of the Viterbi decoder. Finally, Section 5 draws some conclusions and future work.

2. Background

Convolutional codes are commonly described using two parameters the code rate and the constraint length. The code rate (k/n), is expressed as a ratio of the number of bits in to the convolutional encoder (k) to the number of channel symbols output by the convolutional encoder (n) in a given encoder cycle. The constraint length (K) is defined as the number of shifts over which a single message bit can influence the encoder output. The parameter (m) indicates the memory length of the encoder. Convolutional codes are widely used as channel codes in practical communication systems for error correction. The encoded bits depend on the current k input bits and a few past input bits [6]. The main decoding strategy for convolutional codes is based on the widely used Viterbi algorithm. The Viterbi Algorithm (VA) finds a maximum likelihood estimate of a transmitted sequence from a received sequence by using a specific decoding strategy [9].

3. Proposed Work

In the proposed work initially developed convolutional encoder in order to decode the encoded output using Viterbi decoder. The scope of the paper is limited to the classical Viterbi algorithm so the proposed work mainly concentrates on Viterbi decoder block. Generally Viterbi decoder consists of three blocks namely Branch Metric Unit, Path Metric Unit and Survivor Memory Unit.

3.1 Convolutional Encoder

Channel Symbol

Convolutional codes are used as a channel codes in communication systems for long transmission medium error correction. The information bits are fed in small groups of k-bits at a time to a shift register. The output encoded bits are obtained by modulo-2 addition (EXCLUSIVE-OR operation) of the input information bits and the contents of the shift register which are a few previous information bits.

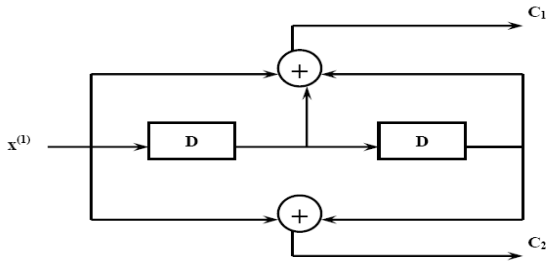

Fig. 1 Block Diagram of convolutional encoder

The general block diagram of convolutional encoder is shown in Fig 1. If the encoder generates a group of 'n' encoded bits per group of 'k' information bits, the code rate R is commonly defined as R=k/n. Fig 1 has the coding rate of R=1/2 where x is the input and c1,c2 are the encoded output. A convolutional encoder may be defined as a finite state machine. Contents of the rightmost (K-1) shift register stages define the states of the encoder so the encoder has four states. The transition of an encoder from one state to another, as caused by input bits, can be depicted using the state diagram. A new input bit causes a transition from one state to another. The path information between the states, denoted as x/c1c2, which represents input information bits as 'x' and the corresponding output bits (c1c2).

3.2 Viterbi Decoder

Viterbi decoding algorithm is the most popular method to decode the convolutional error correcting codes. Coding rate, constraint length and number of bits representing each input value are the key parameters influencing the performance and design of the Viterbi decoder. The Viterbi Algorithm (VA) finds a maximum likelihood estimate of a transmitted code sequence from the corresponding received sequence by maximizing the probability p(r|c) that sequence r is received conditioned on the estimated code sequence c. Sequence c must be a valid coded sequence.

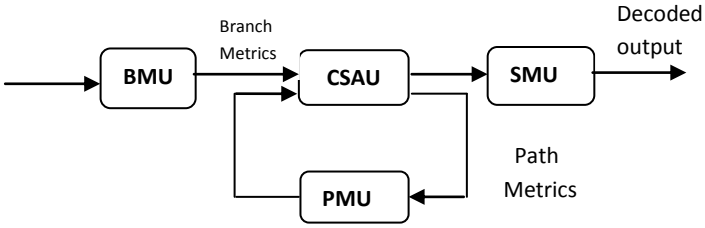

Fig. 2 Block Diagram of Viterbi decoder

The general block diagram of Viterbi decoder is shown in Fig 2 which consists of mainly three units. At the top level, Viterbi decoder consists of three units: the Branch Metric Unit (BMU), the Path Metric Unit (PMU) and the Survivor Memory Unit (SMU).

3.3 Branch Metric Unit (BMU)

BMU is the typically the smallest unit of the Viterbi Decoder. The BMU calculates the distance from the received (noisy) symbols to all legal code words of the transmitted data. As a result branch metrics are produced as the output of Branch Metric Unit. The branch metric value is nothing but the number of bits differs from the original bits. Its complexity increases exponentially with (reciprocal of the coding rate) and also with the number of samples that are processed by decoder per clock cycle. The complexity increases linearly with soft bits. Therefore, the area and throughput of the BMU can be completely described by these two factors.

The branch metrics is calculated by using the look up table based Branch Metric Unit. The look up table is created on the basis of difference in the state transitions that occur between the legal code words and the noisy code words.

Table 1: Look up Table for BMU

Input	Metric
0000	00
0001	01
0010	01
0011	10
0100	01
0101	00

The look up table based BMU results in less power consumption when compared to the design circuit for branch metric calculation. The algorithm used to create the look up table is as same as the hamming distance but the approach used is different.

3.4 Path Metric Unit

The PMU accumulates the distance of single codeword metrics produced by the BMU for every state. Under the assumption that zero or one is transmitted, corresponding branch metrics are added to the previously stored path metrics which are initialized with zero values. The resulting values are compared with each other and the smaller value is selected and stored as the new path metric for each state.

PMU is a critical block both in terms of area and throughput. The key problem of the PMU design is the recursive nature of compare-select-add (CSA) operation (path metrics calculated in the previous clock cycle are used in the current clock cycle). CSA is a retimed variation of the ACS operation. It computes all possible paths originating from a node and based on decision it retains the minimum PM path. Thus, addition and comparison can be processed in parallel.

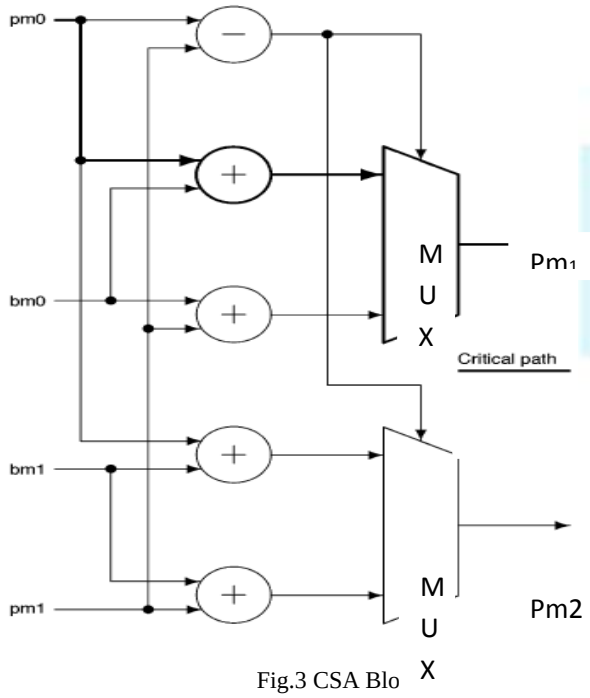

Fig.3 CSA Blo

The Fig 3 shows the proposed CSA block which contains previous metric values as pm0 and pm1 and branch metric values as bm0 and bm1. From the Fig 4 it is clearly known that all the operations that are done simultaneously in which comparison is done at first and smallest values are selected as a result of comparison and then addition operation is calculated in order to get new path metric values. The new path metric values pm1 and pm2 are taken for the critical path.

3.5 Survivor Memory Unit

The survivor sequence update and storage unit (briefly called survivor memory unit, SMU) receives decisions from the PMU and produces the decoded sequence. The implementation techniques for this unit can be divided in two classes they are standard and adaptive. Trace Back Depth (TBD) determines the number of memory access operations required by the SMU. The value of the TBD parameter depends on the transmission channel conditions (or the channel mode used in the simulation). On noisy channels, trace-back path needs longer to merge, so TBD values become larger. On the other side, with fixed trace-back depth and good channel conditions, SMU may run redundant memory access operations, since path tends to merge faster than the. Adaptive SMU implementations are used to reduce the number of these redundant accesses. Only one of the paths, entering into a state, which has minimum accumulated path metric, is chosen as the 'survivor path' for the state. So at the end of this process, each state has one 'survivor path'. The 'history' of a survivor path is also maintained by the node appropriately.

Trace Back (TB) process is used to produce the decoded sequence that has been encoded. The TB stores the decisions in an SRAM and reads them in reverse order during the trace-back cycle. As the update rate of the memory is much less frequent than the update rate of the registers in the register exchange technique, the power consumption is significantly reduced. TB is split in three major operations. If a survivor path is traced back for TBD stages, then the state at which all paths merge can be determined.

Symbols traced back beyond this state can be used as a decoded output. Trace forward technique is used to eliminate the merge state which estimates the starting state in decode operation. This technique uses a set of registers that stores the encoded state. In every clock cycle, the decision bits coming from the PMU are used to update these registers with new states corresponding to the previous trellis stage. After TBD clock cycles, all registers are expected to converge to the same state, which is at the same time the starting state for the decode operation. All these techniques involve several accesses to the memory in the same clock cycle.

4. Simulation Results and Discussion

Modelsim is used as the software tool for software implementation.

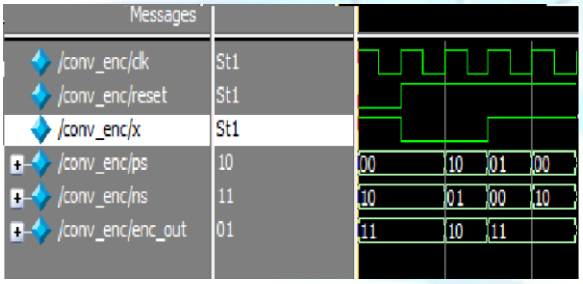

Fig 4.Simultaion Result of Convolutional Encoder

The Fig 4 shows positive clock is given and x is said to be the single bit input which produces two bits as the encoder output along with their corresponding present and next state as shown.

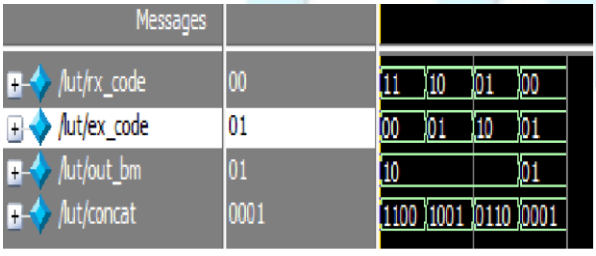

Fig 5.Simultaion Result of BMU Unit

The Fig 5 shows the received code and transmitted code are the two input sequences in which BMU output is calculated by using the look up table based on the state transitions.

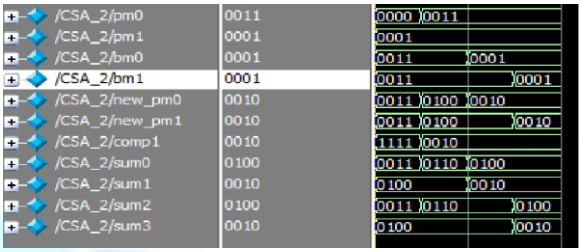

Fig 6.Simultaion Result of CSA Unit

The Fig 6 shows the output of CSA unit which is the shortest path obtained for corresponding inputs as old path metric pm0 and pm1 and the branch metric values as bm00 and bm01. Initially pm0 and bm00 are added then pm1 and bm01 are added then smallest sum value is chosen by comparing both the sum values.

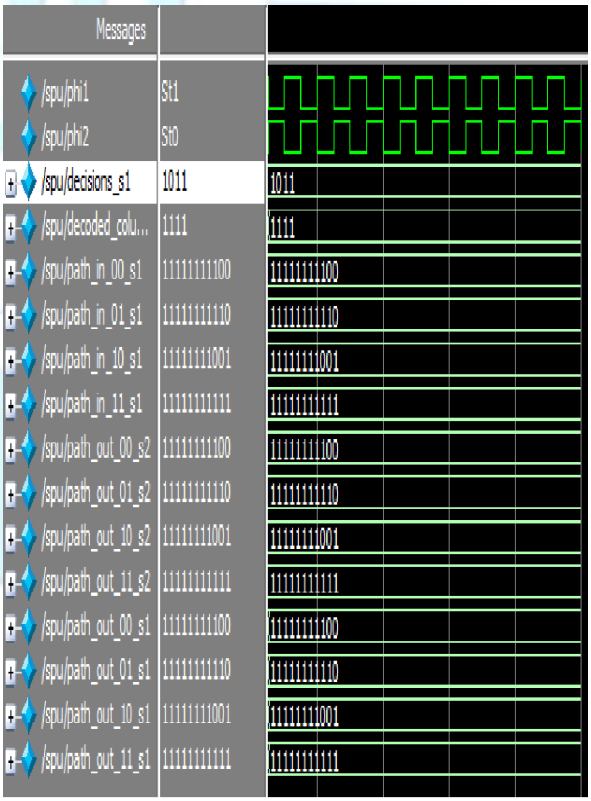

Fig 7.Simultaion Result of Trace Back Unit

The Fig 7 shows the output of the trace back unit is the decoded sequence that has the smallest metric values in the trellis diagram.

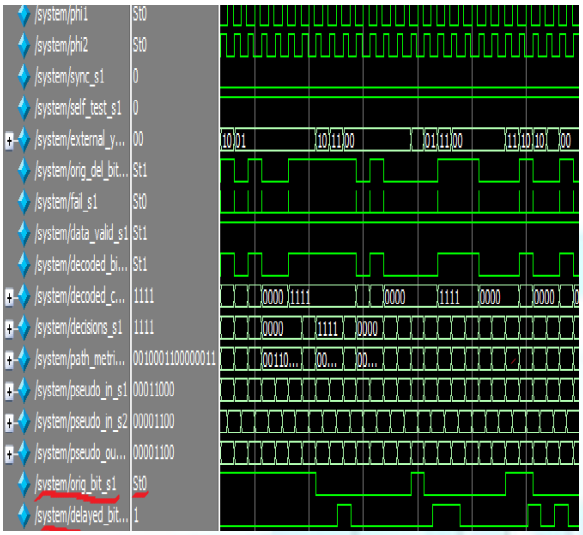

Fig 8.Simultaion Result of Viterbi Decoder

The Fig 8 shows the output of Viterbi decoder is shown as decoded s1 bit which is equal to the original bit s1 that is transmitted at the input side with some delay.

Table 2: Comparison Report

TYPES	No. of Slices	No. of Slice flip flops	No. of LUT's	Power (mw)
Traditional Viterbi Decoder	254	150	402	59.96
Existing Viterbi Decoder	172	102	294	21.473
Proposed Viterbi Decoder	**86**	**56**	**153**	**20.069**

5. Conclusion

In this paper, a comprehensive analysis of the Viterbi decoder design space is presented. This paper summarizes the importance of the different subunits of the decoder depending up on the optimization criteria. Power efficient Viterbi decoder is designed by doing modifications in the BMU and PMU. By using this modified design we can produce the low power consumption and high speed Viterbi decoder for various applications. Design vision is used as the netlist design tool used to calculate the power.

The future work of this paper is Viterbi decoder in the FPGAs. The ability to process parallel data paths within the FPGA takes advantage of the parallel structures of the hardware units in Viterbi decoder and therefore, higher speed can be achieved.

References

[1] R. A. Abdallah and N. R. Shanbhag, "Error-resilient low-power Viterbi decoder architectures," IEEE Trans. Signal Process., vol. 57, no. 12,pp.4906-4917,Dec. 2009.

[2] J. B. Anderson and E. Offer, "Reduced-state sequence detection with convolutional codes," IEEE Trans. Inf. Theory, vol.40, no. 3, pp. 965-972, May. 1994.

[3] F. Chan and D. Haccoun, "Adaptive Viterbi decoding of convolutional codes over memoryless channels," IEEE Trans. Commun., vol. 45, no.11, pp. 1389-1400, Nov. 1997.

[4] Y. Gang, A. T. Erdogan and T. Arslan, "An efficient pre-traceback architecture for the Viterbi decoder targeting wireless communication applications," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 53, no.9, pp. 1918-1927, Sep. 2006.

[5] T. Gemmeke, M. Gansen and T. G. Noll, "Implementation of scalable power and area efficient high-throughput Viterbi decoders," IEEE J. Solid-State Circuits, vol. 37, no. 7, pp. 941-948, Jul. 2002.

[6] R. Henning and C. Chakrabarti, "An approach for adaptively approximating the Viterbi algorithm to reduce power consumption while decoding convolutional codes," IEEE Trans. Signal Process., vol. 52, no. 5, pp. 1443-1451, May. 2004.

[7] J. He, Z. Wang and H. Liu, "An efficient 4-D 8PSK TCM decoder architecture," IEEE Trans. Very Large Scale Integr (VLSI) Syst., vol. 18, no. 5, pp. 808-817, May. 2010.

[8] J. He , H. Liu and Z. Wang, "A fast ACSU architecture for Viterbi decoder using T-algorithm;" in proc. 43rd IEEE Asilomar Conf. Signals, Syst. Comput., pp. 231-255, Nov. 2009.

[9] Jinjin He, Huaoing Liu, Zhongfeng Wang, Xinming Huang, and Kai Zhang, "High-speed low-power Viterbi Decoder design for TCM decoders," IEEE Trans. Very Large Scale Integr (VLSI) Syst., vol. 20, no.4, April 2012.

[10] S. J. Simmons, "Breadth-first trellis decoding with adaptive effort," IEEE Trans. Commun., vol. 38, no. 1, pp. 3-12, Jan. 1990.