CMOS Implementation of Viterbi Decoder

MANJULA SUTAGUNDAR[*] , S.G. KAMBALIMATH[**] AND A.V.SUTAGUNDAR[***]
[*]Department of Instrumentation Technology,
[**] Department of Electronics & Communication,
[***] Department of Electronics & Communication,
Basaveshwar Engineering College,
Bagalkot, Karnataka
INDIA

Abstract: - This paper presents implementation of a soft decision Viterbi decoder suitable for convolution codes with short constraint lengths. The decoder is based on a property of Viterbi algorithm that states "if the survivor paths from all possible states at time n are traced back then with high probability all the paths merge at time n-L where L is the survivor path length". Pipeline structures are introduced in Viterbi decoder, which increases the decoding speed. Because of the pipeline structure there is a little increase in hardware complexity. The implemented Viterbi decoder operates on a block of received data and the block length decides the decoding speed. The decoder is simulated and synthesized on Altera's Quartus-II targeting on Stratix-EP1S10B672C6 by writing structural VHDL Description. The static CMOS transistor level circuit simulation is done using Berkeley SPICE for 0.35μm MOSIS technology.

Key-Words: - Convolutional encoder, Viterbi decoder, Survivor path length, Euclidean distance, Add Compare Select unit, Branch Metric unit.

1. Introduction

Channel coding is often used in communication system to detect and even correct the errors introduced by the communication channel. The idea of the channel coding is to encode the message by introducing some form of redundancy so that the errors are can be detected and even corrected. There are two main types of channel codes, namely *block codes* and *convolutional codes*. Block codes accept a block of *k* information bits and produce a block of *n* coded bits. By predetermined rules, *n-k* redundant bits are added to the *k* information bits to form the *n* coded bits. With convolutional codes, the incoming bit stream is applied to a *K*-bit long shift register. For each shift of the shift register, *k* new bits are inserted and *n* code bits are delivered. Convolutional codes offer an alternative to block codes for transmission over a noisy channel. Convolutional coding can be applied to a continuous input stream (which cannot be done with block codes), as well as blocks of data. Convolutional encoding with Viterbi decoding is a *Forward Error Correction* technique that is particularly suited to a channel in which the transmitted signal is corrupted mainly by additive white Gaussian noise. Convolutional codes are usually 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 into the convolutional encoder (k) to the number of channel symbols output by the convolutional encoder (n) in a given encoder cycle. The constraint length parameter, K, denotes the "length" of the convolutional encoder, i.e. how many k-bit stages are available to feed the combinatorial logic that produces the output symbols. There are basically two types of decoding algorithms for convolutional codes namely Sequential decoding algorithm and Viterbi algorithm. Viterbi decoding algorithm was developed by Andrew J. Viterbi in his seminal paper [1] in 1967. Sequential decoding has the advantage that it can perform very well with long-constraint-length convolutional codes, but it has a variable decoding time. Viterbi decoding has the advantage that it has a fixed decoding time. It is well suited to hardware decoder implementation.

This paper Viterbi decoder presented in this paper is suitable for convolutional codes with short constraint lengths. The decoder operates on a block of received data and the block length decides the decoding speed. Since the decoder operates on a block of received data the parameterized buffers are used so that the block size can be varied according to the requirement

2 Viterbi Algorithm

Viterbi algorithm is a decoding algorithm used for convolutional codes. Convolutional codes are usually characterized by two parameters and the patterns of *n* modulo-2 adders. The two parameters are the *code rate* and *constraint* length. The code rate, *kln*, is expressed as a ratio of the number of bits into the convolutional encoder (k) to the number of channel symbols output by the convolutional encoder (n) in a given encoder cycle. The decoder is implemented for rate 1/3 convolution encoder and the encoder is shown in the fig 1.The three modulo-2 adders of the encoder are:

$g_1 = x^2 + 1$ (expressed as: $g_1 = 101$)
$g_2 = x + 1$ (expressed as: $g_2 = 011$)
$g_3 = x^2 + x + 1$ (expressed as: $g_3 = 111$)

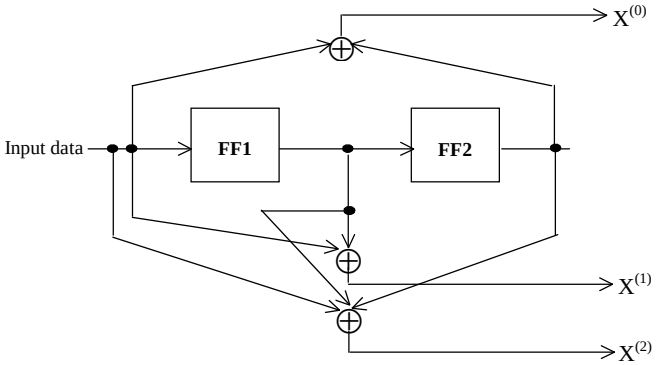


Fig 1. A Rate 1/3 Convolutional Encoder

The encoding cycle starts when the shift register clock edge occurs. The output of the left-hand flip-flop (FF) is clocked into the right-hand flip-flop, while the previous input bit is clocked into the left-hand flip-flop, and a new input bit becomes available. If the input sequence is: 010011110010 then the output sequence will be: 000,101,011,111,101,110,001,001,100,111,101 assuming that the output of both the flip-flops in the shift register are initially cleared. The encoder shown in Figure 1 is a 4 state encoder and it can be described using the trellis diagram shown in Fig 2.

The Trellis is a time-indexed version of the state diagram. Each node corresponds to a state at a given time index, and each branch corresponds to a state transition. Associated with each branch is the input bit and the output symbol corresponding to the state transition. The Viterbi algorithm uses maximum likelihood decoding that involves finding the valid code symbol, whose distance is minimum from the received code symbol. Viterbi algorithm involves the process of finding the minimum cost route through the Trellis diagram where the cost of each transition is known. The received symbol is compared with the transition values and their distances are computed. In soft decision Viterbi algorithm

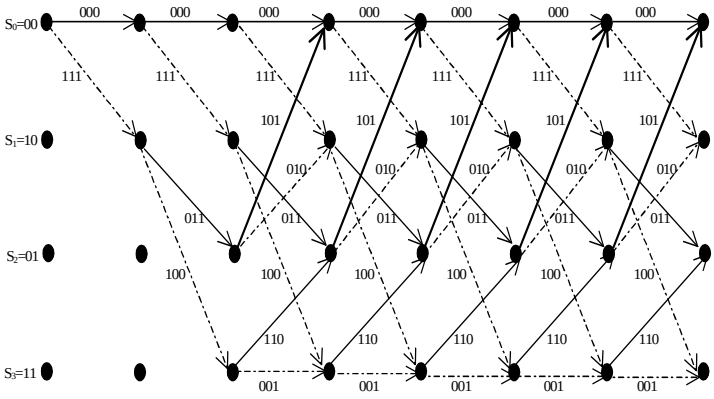


Fig 2. Trellis diagram for the four state encoder of Figure 1

Euclidean distance is used as a measure for computing the distance between the received code symbol and the transition values. When the input symbol is X and encoder symbol is Y, the Euclidean distance is calculated from the formula $(X-Y)^2$. [2]. These distance values are called branch metrics. Each state (node) in the current time stage can be reached from two states from the previous time stage. Each node has a value that corresponds to the sum of branch metrics when moving in trellis diagram. These values are called path metrics. In each node the smaller one of these sums is selected and the state where the transition came from is stored to the memory. This operation is known as *add-compare-select (ACS)* operation. This ACS operation is repeated for every node in the current time stage. When the Viterbi decoder has received the whole message or a predefined number of symbols it starts the trace-back operation. In the trace-back operation the minimum value of all path metrics is selected and moving backwards in the trellis diagram starts. When the trace-back depth is received we know a route in the trellis diagram. Due to these state transitions it is known which input bit sequence caused this route in the trellis diagram. After the result is output the decoding sequence can be started again / continued.

The two important characteristics of Viterbi decoder used for the implementation of the decoder are:

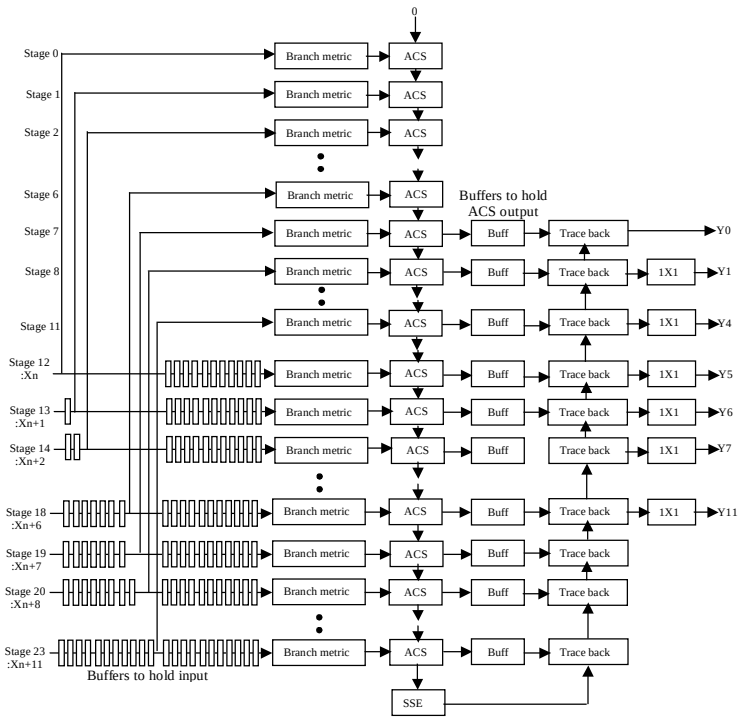
- A property of the trellis, which is used for survivor path decoding, is that if the survivor

paths from al1 possible states at time *n* are traced back, then with high probability, al1 the paths merge at time *n-L,* where *L* is the *survivor path length and* is typically *5v* [3.]. Once the survivor paths have merged, the traced path is unique independent of the starting state and future ACS iterations.

- When starting with unknown initial state metrics (typically set to zero), the state metrics after *J* trellis iterations are independent of the initial metrics, or equivalently, the survivor path will merge with the true survivor path as if the initial metrics had been known. The parameter *J* is *the synchronization length and* is also typically 5*v* [4].

2. Implementation of Viterbi Decoder

The implemented Viterbi decoder is based on the property of the Viterbi algorithm, which states that if the survivor paths from al1 possible states at time *n* are traced back, then with high probability, al1 the paths merge at time *n-L,* where *L* is the *survivor path length.* Based on this property it is proposed in [5] that the state at time *n* can be decoded using only the information from the interval *n-L* to *n+L.* The proposed Viterbi decoder performs decoding of



a block of finite length of received channel symbols. The block diagram of the Viterbi decoder for a block length of 12 is shown in Fig 3.

Fig 3. Pipeline Structured Viterbi Decoder for a Block Length of 12

In each stage the ACS calculation for the corresponding channel symbol X_c is done and the path metric for each state is found. These path metrics are fed into next stage and the and again the ACS calculation is performed for the next channel symbol. The procedure is repeated until the end of the block. Then a *survivor path estimation* block is used to compare the path metrics of four states and select the state with the minimum path metric as the starting state for the trace back process. From the trace back process the original message can be retrieved. The method is equivalent to best state survivor path decoding, hence the surviving path length can be reduced to *L = 2.5v* [3]. So for a block length of *2L + M* the decoded bits from *L* to *L+M* are the most likely original message. The decoder outputs *M* bits per clock cycle, hence only *M* channel symbol pairs need to be read from the input stream during the same clock cycle . So only M new channel symbol pairs need to be fed into the decoder each cycle- The rest of the *2L* symbols have been buffered at the previous cycle. If we let *M = 2L* , we can make fil1 use of the pipeline buffer resources through sharing.

2.1.Major Components of Viterbi Decoder

The implemented decoder consists of five main blocks namely Branch metric unit (BMU), Add-Compare-Select (ACS) unit, Trace back unit (TB), Survivor state estimation (SSE) unit and Parameterized Buffers. A brief description of each block provided below.

Branch Metric Unit (BMU): The branch metric unit is used to calculate branch metrics. The branch metrics are difference values between received code symbol and the corresponding branch words from the encoder trellis. These encoder branch words are the code symbols that would be expected to come from the encoder output as a result of the state transitions. The Euclidean distance is used as the measure for calculating the branch metrics. When the input symbol is X and encoder symbol is Y, the Euclidean distance is calculated from the formula $(X-Y)^2$. Inside the branch metric unit all possible branch words are saved. Since there are eight branches in one iteration, for one trellis

iteration each branch metric unit accepts one 3-bit symbol and produces eight branch metrics corresponding to eight branches of the trellis. A branch metric is calculated by first finding the difference between the received 3-bit symbol and the stored branch word for that particular branch and then squaring the difference.

Add-Compare-Select: The Add-Compare-Select block receives two branch metrics and the path metrics. An ACS module adds each incoming branch metric of the state to the corresponding path metric and compares the two results to select a smaller one. The path metric of the state is updated with the selected value, and the survivor path information is recorded in the form of 1-bit decision. Four such ACS units are combined to form the ACS block for one iteration. Since the path metrics grow unboundedly, normalization is required. But this can be avoided using modulo arithmetic approach proposed in [6]

Survivor State Estimation: The survivor state estimation unit is used to find the starting state for the trace back unit. At the end of each block, the state with minimum path metric is selected as the starting state for trace back. The logic unit known as survivor state estimation unit does this. The four path metrics are compared by generating six possible pair-wise comparisons and combining the comparison results to form the minimum path metric selection.

Trace Back: When the Viterbi decoder has received the whole message or a predefined number of symbols it starts the trace-back operation. In the trace-back operation the minimum value of all path metrics is selected and moving backwards in the trellis diagram starts. When the trace-back depth is received we know a route in the trellis diagram. Due to these state transitions it is known which input bit sequence caused this route in the trellis diagram.

The trace back unit begins the trace back operation from the state given by the survivor state estimation unit. The trace back unit uses four 1-bit decision outputs given by the ACS unit to compute the present state from the previous trace back unit stage or survivor state estimation unit. If the current state is S_n, then the trace back unit appropriately combines the 1-bit decision and 2-bit current state to produce the estimated state S_{n-1}. S_{n-1} *(1)* is the decoded output bit.

Parameterized Buffers: Buffers are used in the pipeline structured Viterbi decoder design for several reasons such as pipelining of input symbols, storage of decision bits from the ACS unit and storage of output bits

3. Results

The Viterbi decoder was simulated and synthesized using Altera's Quartus-II tool, targeting on Stratix - EP1S10B672C6 by writing structural VHDL description. The static CMOS transistor level circuit of the various blocks of the Viterbi decoder are also simulated using BERKLEY SPICE for 0.35µm MOSIS technology. Results for some sample data is provided in Table 1 Here the *Input Message* represents the input to the encoder, *Encoded Message* represents the output of the encoder, which will be transmitted to the decoder. Because of the noise in the channel the data received by the decoder may not be same as that transmitted by the encoder. The *Received Message* is the actual message received by the decoder, which may not be same as *Encoded Message*. *Decoded Message* is the output of the decoder. The output waveforms for sample data 1 and sample data 2 are shown in Fig. 4 and Fig. 5.

Note: The bold symbols in the received message indicate the position at which error has occurred.

TABLE I
Results of VHDL Simulation

Sl No	Input Message	Encoded Message	Received Message	Decoded Message
1	101111100000	111,011, 010,100, 001,001, 001,110, 101,000, 000,000	111,011, 01**1**,100, 001,001, 001,11**1**, 101,000, 000,000	101111 100000
2	111111000000	111,100, 001,001, 001,001, 001,110, 101,000, 000,000	111,100, **1**01,001, 001,001, 001,11**1**, 101,000, 000,000	111111 000000
3	101010000000	111,011, 010,011, 010,011, 101,000, 000,000, 000,000	111,011, 0**0**0,011, 010,011, 101,0**1**0, 000,000, 000,000	101010 000000
4	010111000000	000,111, 011,010, 100,001, 110,101, 000,000, 000,000	000,111, 011,010, 100,001, **0**10,101, 00**1**,000, 000,000	010111 000000

The snap shots from the Quartus-II simulator for the first 2 test vectors in the table is given in Fig 4 and Fig 5

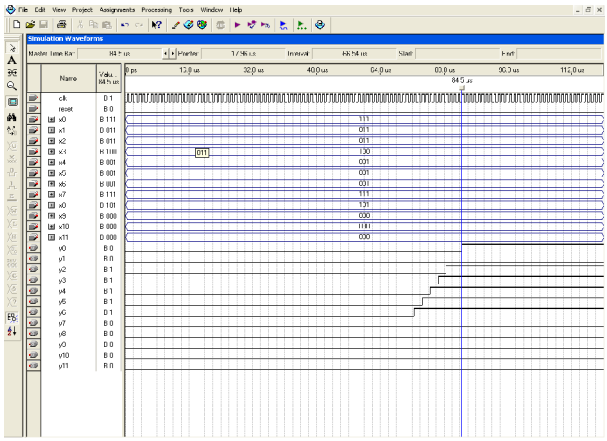


Fig 4. Output waveforms for test vector 1

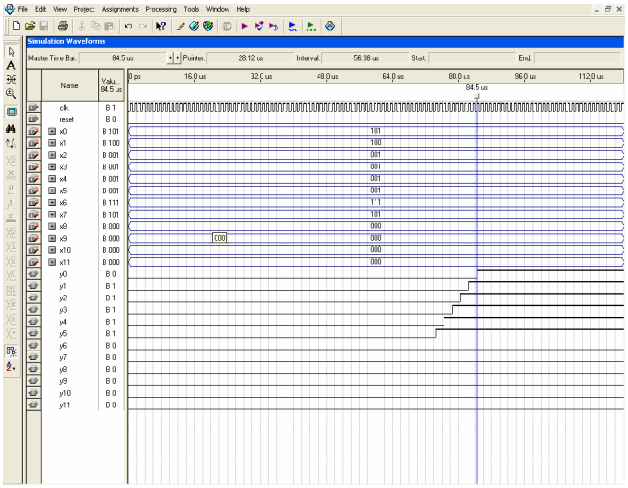


Fig 5. Output waveforms for test vector 2

The SPICE simulation result for one of the ACS unit and BMU are given in Fig 6 and Fig 7

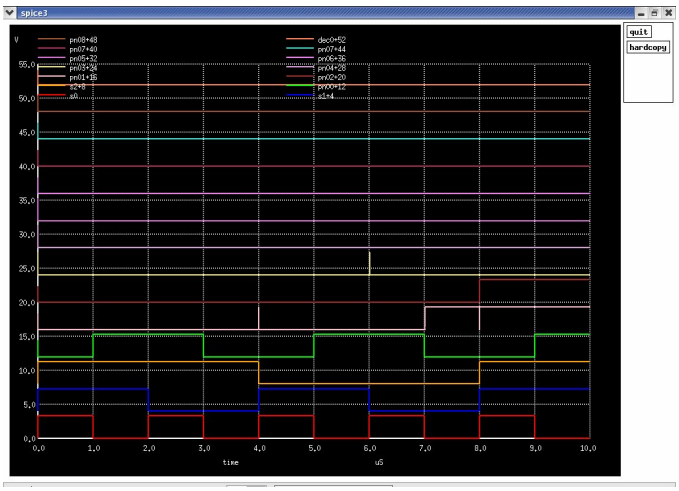


Fig 6. SPICE Simulation result for one of the ACS unit

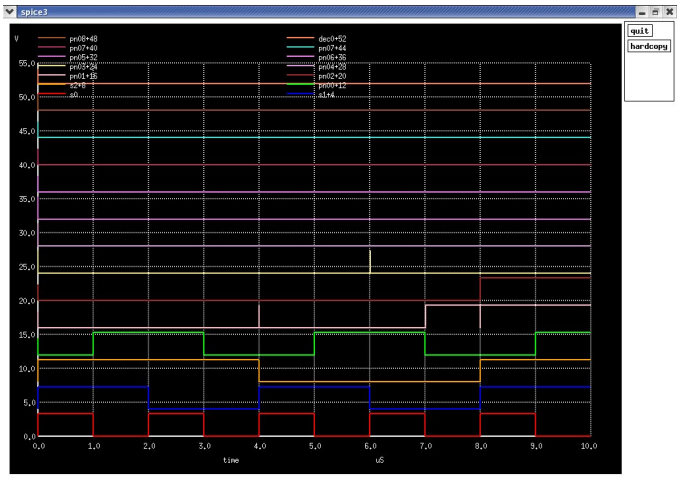


Fig 7. SPICE simulation result for BMU

4. Conclusion

The proposed Viterbi decoder makes use of the constraint length property of the Viterbi algorithm. The Pipeline structure and the concurrent calculation method increase the decoding speed. Also as compared to the traditional Viterbi decoder the memory requirement is reduced. The implemented Viterbi decoder is suitable for convolution codes with small constraint lengths. The performance of the decoder depends on the decoding length and how fast the serial to parallel decoder can deliver the input bits.

References:

[1]. A.J.Viterbi, "Error bounds for convolutional coding and an asymptotically optimum decoding algorithm", *IEEE tran. Inform. Theory, vol.II-13*, Apr. 1967

[2]. M.S.Ryan, G.R.Nudd, “The Viterbi Algorithm” *Warwick Research Report RR238* , February 1993

[3]. G. C. Clark and J. B. Cain, “Error-Correction Coding for Digital Communication”. *NewYork: Plenum*, 198 1

[4]. A.J. Viterbi and J. K. Omura, “Principles of Digital communication and Coding”. *New York: McGraw- Hill*, 1979

[5]. K. H Tzou and J. G. Dunharn, " Sliding block decoding of convolutional codes", *IEEE Trans. Commun., vol. COM-29*, Sept. 1981

[6]. A. P. Hekstra, " An alternative to metric rescaling in Viterbi decoders," *IEEE Trans. Commun.*, vol.3 7, Nov. 1989.