Implementation of CRC and Viterbi algorithm on FPGA

S. V. Viraktamath¹, Akshata Kotihal², Girish V. Attimarad³

¹ Faculty, ² Student, Dept of E&CE, SDMCET, Dharwad,
³ HOD Department of E&CE, Dayanand Sagar College of Engineering, Bangalore, Karnataka, India
Email: svvmath@gmail.com

Abstract— Cyclic Redundancy Codes (CRC) code provides a simple, yet powerful, method for the detection of errors during digital data transmission and storage. Convolutional Coding and Decoding (CODEC) is a Forward Error Correction (FEC) technique that is particularly suited for a channel in which the transmitted signal is corrupted mainly by Additive White Gaussian Noise (AWGN). The Viterbi Algorithm (VA) has been widely applied for decoding convolutionally encoded data in digital communication systems over the last 30 years. In this paper the implementation of CRC and Viterbi decoder on FPGA is presented. CRC-32 and Viterbi hard decision decoding algorithm for rate 1/2 implemented on FPGA. Also for higher SNR at the decoder side the concept of serially concatenated CRC- Convolutional Coding (CC) with lookup table is also proposed.

Keywords— CRC, FPGA, Viterbi, Trellis, Constraint length.

I. INTRODUCTION

The evolving world of telecommunications requires increasing reliability and speed in communications. Reliability in information storage and transmission is provided by coding techniques. Information is usually coded in a bit streams and transmitted over the communication medium, channel. The communication media is prone to errors due to noise present in the analog portion of the channel. Therefore errors have to be detected and corrected while decoding. CRC has the advantages of easy coding and decoding as well as strong abilities of checking errors and correcting errors. Therefore, it was widely used in the field of communications. Reliability in information storage and transmission is provided by coding techniques. CRC is an error-detecting code designed to detect accidental changes to raw computer data, and is commonly used in digital networks and storage devices such as hard disk drives. Blocks of data entering these systems get a short check value attached, derived from the remainder of a polynomial division of their contents; on retrieval the calculation is repeated, and corrective action can be taken against presumed data corruption if the check values do not match. The CRC was invented by W. Wesley Peterson in 1961. CRC is an error detecting code that is widely used to detect corruption in blocks of data that have been transmitted or stored.

The Error Control Coding techniques (ECC) rely on the systematic addition of redundant bits at the transmitting side. The task of channel coding is to encode information sent over a communication channel in such a way that in the presence of channel noise, errors can be detected and possibly corrected. There are two coding methods - backward error correction codes and forward error correction codes. Backward error correction codes requires only error detection, if an error is found then the transmitter is requested to retransmit the message. Forward error correction codes require the decoder to be capable of correcting errors. There are several error correcting codes and these are classified under two basic categories namely block codes and convolutional codes.

Convolutional codes [1] differ from block codes [2] in the sense that bit streams are not partitioned into binary words instead redundancy is added continuously to the whole stream. Convolutional codes are widely used error control coding technique in channel coding because of low complexity and error controlling capability. Viterbi decoding algorithm [3, 4] is the simplest and best algorithm for decoding of convolutional codes. The Viterbi algorithm first appeared in the coding literature in a paper written by Andrew J. Viterbi in 1967 [5]. Since then, due to its easiness in implementation, it has been applied to many different areas related to decoding problems. The 8-bit parallel CRC-32 is proposed in [6] to meet the high throughput of USB3.0. Exhaustive survey of all CRC polynomials from 3 bits to 15 bits is presented in [7].

A set of 35 new polynomials in addition to 13 previously published polynomials are also described. The method that realizes the ability of multiple bits error correction using cyclic redundancy check codes is presented in [8]. The structures of 8-bit CRC are presented in [9]. The joint decoding scheme of serially concatenated CRC and convolutional code (CC) has been investigated in [10 11].

This paper is organized as follows. Section II gives the proposed work. Section III gives the CRC coding. The
Viterbi decoder is discussed in section IV. The results of the proposed model are discussed in section V. Next section concludes the paper.

II. PROPOSED WORK

The basic block diagram of a system used for the simulation is shown in Fig.1. The output of the source encoder is given to the convolutional encoder. The output of the convolutional encoder is given to the CRC encoding; the output of which will be modulated and sent on the channel. At the receiver the reverse process is done. The CRC encoding, decoding, convolutional encoding and decoding blocks are simulated and implemented on FPGA.

The concept of look up table may also be used at the receiver for Viterbi decoding.

III. CRC CODING

CRC are specifically designed to protect against common types of errors on communication channels, where they can provide quick and reasonable assurance of the integrity of messages delivered as shown in Fig.2. However, they are not suitable for protecting against intentional alteration of data. The selection of generator polynomial is the most important part of implementing the CRC algorithm. The polynomial must be chosen to maximize the error-detecting capabilities while minimizing overall collision probabilities. CRC is divided into the following types: Code CRC-12, code CRC-16, code CRC-CCITT, and code CRC-32. Code CRC-12 is usually used to send 6-bit string. Code CRC-16 and code CRCCCITT is used to send 8-bit string, and code CRC-16 is mainly used in America, however, code CRC-CCITT is often used in European countries. Code CRC-32 is often used in a kind of synchronous transfer which is called Point to Point transfer. In the proposed system CRC-32 is simulated and implemented. The Convolutional encoder described in this system is of rate 1/2, and of constraint length-3. Programming is done in VHDL for both coding and decoding and the same is implemented on FPGA. Input to encoder is entered by the user. The output of the encoder is stored in the matrix form, and the same is fed to the decoder part.

IV. CONVOLUTIONAL ENCODER AND DECODER

Channel encoding and decoding considered in this paper is convolutional encoding and Viterbi decoding. Encoding and decoding algorithm are implemented on FPGA. The encoder and decoder algorithms are given below:

A. Convolutional Encoder

1. Read next input bit and take it into the register by right shift by neglecting right most bit.
2. Compute two output bits by XOR operation and store the result.
3. Repeat steps 2 and 3 for all the input bits.

The trellis for Viterbi decoding algorithm is shown in Fig. 3. The minimum hamming distance in the last column is zero for the received input bits, as there are no errors in the received bit. In general the minimum hamming distance should be chosen then path should be traced back.

B. Viterbi Decoder

1. Calculate the output table using the generator polynomials; initialize the next state table.
2. Initialize the weight matrix of the trellis.
3. Calculate the weight matrix for the received message with the help of output table and the next state table.
4. Find the initial path and process it and find the final path or survival path in trellis.
5. Decode the message with the help of output table and next state table.

The Viterbi algorithm is implemented. The constraint length is fixed in the implemented algorithm. In the trellis if the next state weights are same, proposed algorithm is smart enough to choose the correct path. There is no trace back instead while calculating the weight matrix, parallel survival path is also found hence the computation time becomes very less. The behavioral descriptions of the encoder and decoder algorithms are written using VHDL.
The input to and output from the system are bit streams. A \((n, k, m)\) convolutional encoder accepts \(k\)-bit blocks of input sequence and produces \(n\)-bit blocks of output sequence. It consists of \(m\) k-stage shift registers and \(n\) modulo-2 adders. The outputs of \(n\) modulo-2 adders are sequentially sampled to produce the encoded sequence. A \((2, 1, 2)\) convolutional encoder is considered for simulation and implementation. Basically, decoding of convolutional codes is comparison of different paths in trellis. The trellis diagram for \((2, 1, 2)\) convolutional encoder is as shown in Fig.3.

![Trellis for convolutional encoder (2, 1, 2)](image)

In hard decision decoding technique, the Hamming distance is computed by simply counting how many bits are different between the received encoded bits and the actual output bits. The Hamming distance values are computed at each time instant for the paths between the states at the previous time instant and the states at the current time instant are called ‘branch metrics’. For the first time instant these results are saved as “Accumulated Error Metric” values, associated with the states. For the second time instant on, the accumulated error metrics will be computed by adding the previous accumulated error metrics to the current branch metrics. When two paths enter the same state, the one having the best metric (i.e. lower branch metric) is chosen, this path is called the ‘surviving path’. Then by seeing the path and using output table and next state table decoding is done. This procedure is repeated for all encoded bits and at every comparison makes a hard decision as to whether one or zero is transmitted. The output of hard decision decoder is compared with original message bits for verification.

The programming environment for convolution is based on VHDL. The behavioral description is analyzed simulated and synthesized onto Xilinx FPGAs. After the description code is verified to be functionally correct by simulation, it is translated onto a Xilinx net list form. The net list is then mapped onto FPGA architecture by automatic partition, placement and routing tool to form a loadable FPGA object module. A static timing analysis tool is then applied to the object module to determine maximum operating speed.

V. RESULTS AND DISCUSSION

In this section, simulation results are presented. Figure 4 shows the simulated output of the CRC encoder, more than hundred bits of input bits considered. The CRC decoder outputs with errors and without errors are shown in fig. 5 and Fig. 6 respectively.

![Figure 4. CRC Encoder output](image)

![Figure 5. CRC Decoder output without error.](image)

![Figure 6. CRC Decoder output with error.](image)

The synthesis report of encoder and decoder of CRC-32 are shown in figures 7 and 8 respectively.

![Figure 7. Synthesis report of CRC Encoder](image)

![Figure 8. Synthesis Report of CRC Decoder](image)

The behavioral description of the Viterbi decoding algorithm is simulated and synthesized using Xilinx ISE version 6 and functionally verified on the ModelSim simulator version SE 5.5a. The simulation results of the convolution encoder are shown in Fig. 9. The encoder input is port “a” in the form of bits and the encoded outputs are obtained at port “y”. The simulation results of convolution decoder are shown in Fig. 10 and 11. The decoder receives the input vector...
of the encoded data, the decoder is also provided with reset so that at any point of time the decoder can be reset. The decoder receives the data on every event of the clock pulse at ‘a’ and the output is obtained at port “y”.

The Fig. 10 describes the output of the decoder without the errors being introduced in the input. Fig.11 describes the output of the Viterbi decoder with errors being introduced in the input data. Error detecting and correcting capability of the Viterbi decoder can be observed by comparing Fig.10 and Fig.11.

![Fig.9: Simulation results of convolution encoder](image)

![Fig.10: Simulation results of Viterbi decoder without errors in the input](image)

![Fig.11: Simulation results of Viterbi decoder with errors introduced in the inputs](image)

The device utilization summary for the convolutional encoder and decoder are shown in table 1 and table 2 respectively.

So far the implementation of CRC and Viterbi algorithm is discussed in this section. After convolutional encoding block the CRC encoder block is used as shown in figure 1. At the receiver the CRC is calculated and compared with the received CRC. If it is same as received CRC then it indicates that there is no error in the received bits. Instead of decoding using Viterbi algorithm the lookup table may be used to get back the message bits. The look up table reduces the decoding time. If the CRC calculated at the receiver is different than received then Viterbi decoding may be used to decode the message bits.

VI. CONCLUSION

In this paper, the code has been verified for all possible lengths of message polynomial and generator polynomials. The CRC-4 to CRC-32 are successfully implemented on FPGA. The simulation results are presented. The implementation of convolution encoder and an implementation of the convolution decoder using Viterbi decoding algorithm have been presented. The algorithm for convolution encoder and decoder has been modeled using VHDL. The device utilization has been found satisfactory. The code has been tested for different combinations of inputs using ModelSim simulator. The concept of lookup table helps to detect the errors at the receiver.

REFERENCES


