RESEARCH INVENTY: International Journal of Engineering and Science
ISSN: 2278-4721, Vol. 1, Issue 3 (Sept 2012), PP 15-21
www.researchinventy.com

Design of Convolution Encoder and Reconfigurable Viterbi Decoder

Mahe Jabeen, Salma Khan
Department of Electronics and Communication Engineering
Shadan Women's College of Engineering and Technology, Khairtabad,
Hyderabad, Andhra Pradesh, INDIA.

ABSTRACT- *Error-correcting convolution codes provide a proven mechanism to limit the effects of noise in digital data transmission. Convolution codes are employed to implement forward error correction(FEC) but the complexity of corresponding decoders increases exponentially with the constraint length K. Convolution Encoding with Viterbi decoding is a powerful FEC technique that is particularly suited to a channel in which the transmitted signal is corrupted mainly by Additive white Gaussian Noise. In this paper, we present a Convolution Encoder and Viterbi Decoder with a constraint length of 3 and code rate of 1/2. This is realized using Verilog HDL. It is simulated and synthesized using Modelsim Altera 10.0d and Xilinx 10.1 ISE. The main aim of this paper is to design FPGA based Convolution Encoder and Viterbi Decoder which encodes/decodes the data. This architecture has comparatively simpler code and flexible configuration when compared to other architectures and saves silicon area through efficient device utilization which makes it favorable for FPGA's.*

Keywords- *Convolution Encoder, Viterbi Decoder, Viterbi Algorithm*

I. INTRODUCTION

Convolution coding is a popular error-correcting coding method used in digital communications. A message is convoluted, and then transmitted into a noisy channel. This convolution operation encodes some redundant information into the transmitted signal, thereby improving the data capacity of the channel. The Viterbi algorithm is a popular method used to decode convolutionally coded messages. The algorithm tracks down the most likely state sequences the encoder went through in encoding the message, and uses this information to determine the original message. Instead of estimating a message based on each individual sample in the signal, the convolution encoding and Viterbi decoding process packages and encodes a message as a sequence, providing a level of correlation between each sample in the signal. As the convolution codes are used mostly for the channel encoding of data to achieve low-error-rate in latest wireless communication standards like 3GPP, GSM and WLAN; the use of optimal decoding Viterbi algorithm will suffice.All communication channels are subject to the additive white Gaussian noise (AWGN) around the environment. The block codes can be applied only for the block of data whereas convolution coding has can be applied to a continuous data stream as well as to blocks of data. Convolution Encoding with Viterbi decoding is a powerful FEC technique that is particularly suited to a channel in which the transmitted signal is corrupted mainly by AWGN. It operates on data stream and has memory that uses previous bits to encode. It is simple and has good performance with low implementation cost. The Viterbi algorithm (VA) was proposed in 1967 by Andrew Viterbi [1] and is used for decoding a bit stream that has been encoded using FEC code.

1.1 Motivation of the Paper

Most of the Viterbi decoders in the market are a parameterizable intelligent property (IP) core with an efficient algorithm for decoding of one convolutionally encoded sequence only. In addition, the cost for the Convolution Encoder and Viterbi decoder are expensive for a specified design because of the patent issue [2] .Therefore, to realize an adaptive Convolution Encoder and Viterbi decoder on FPGA is very demanding. Complexity of Viterbi decoding algorithm increases in terms of trellis length. Increasing the trellis length causes the algorithm to take more time to decode. This will cause transmission speed lower but make the transmission more reliable. Reducing the trellis length will increase the transmission speed.

II. CONVOLUTION ENCODER

A Convolution Encoder accepts an input stream of message and generates encoded output streams to be transmitted. In this process for one input bit the encoder generates more than one output bits and these redundant symbols in output bit pattern makes the transmitted data more immune to the noise in the channel. The redundant bits help to decide and correct the errors in received pattern.

For the standardization a terminology was generated as summarized:

M: Length of the shift registers stage in the encoder

Constraint Length (K) = M+1: This number represents the number of input bits required to generate a unique output pattern in the encoder. A constraint length of K=3 means that each output symbol depends on the current input symbol and the two previous input symbols.

Number of States = $2^{(K-1)}$: Defines the maximum number of states that is possible to be mapped by the combinations of the K number of input bits for the convolution encoder.

L: Length of Input Message: 4

R: Convolution Code Rate R=Number of input bits to create a symbol at the output (m)/ Number of output bits in a symbol at the output (n).

For example, 1/2 code rate means each bit entering the encoder results in 2 bits leaving the encoder.

The encoder has n modulo-2 adders, and n generator polynomials one for each adder. This process doubles the number of input bits at the output. For example, a 4-bit input is converted into an 8-bit output, 8-bit input into a 16-bit output and so on.

Generator polynomial: A generator polynomial specifies the encoder connections. In another words, the generator polynomial can be deduced as the mathematical description of the convolution encoder. Each polynomial forming the generator polynomial should be at most K degree and specifies the connections between the shift registers and the modulo-2 adders.

Two generator polynomials are $g1(x)=1+x^2$, $g2(x)=1+x+x^2$ which give g1(x)=(101) and g2(x)=(111).

Convolution encoders can be classified as recursive and non – recursive. Recursive codes are always systematic whereas, non – recursive codes are always non – systematic. [3]

Block Diagram

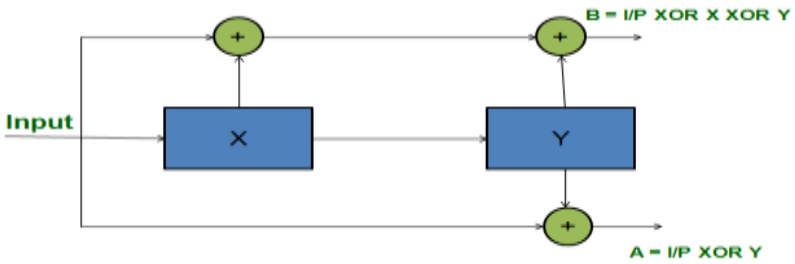

Figure 1 Block Diagram of Convolution Encoder

The block diagram of convolution encoder is shown in Fig 1. To generate the output, the encoder uses three values of the input signal, one present and two past. The set of past values of input data is called a state. The number of input data values used to generate the code is called the constraint length. Each set of outputs is generated by EX-OR ing a pattern of current and shifted values of input data. The pattern used to generate the coded output value can be expressed as binary strings called "Generator Polynomials" (GP). The MSB of the GP corresponds to the input; the LSBs of the GP correspond to the state. The encoder that has been designed is a linear, non – systematic, convolution encoder.

Truth Table

Input	Present State (XY)	Output (AB)	Next State (XY)
0	00	00	00
0	01	11	00
0	10	01	01
0	11	10	01
1	00	11	10
1	01	00	10
1	10	10	11
1	11	01	11

Figure 2 Truth Table of Convolution Encoder

The truth table for this encoder is shown in Fig 2.The present state values are the possible 2 bit combinations for both 0 and 1 input. The output AB is obtained from the expressions in the fig 1. Then, the next state values are calculated by right shifting the input and present state values together by 1 bit position.

Code Trellis

The convolution Encoding can be done using three different techniques as shown below.

- Code tree
- Code trellis
- State diagram

Trellis diagrams are messy but generally preferred over both the tree and the state diagrams because they represent linear time sequencing of events. The x-axis is discrete time and all possible states are shown on the y-axis. We move horizontally through the trellis with the passage of time. Each transition means new bits have arrived. The trellis diagram is drawn by lining up all the possible states (2L) in the vertical axis. Then we connect each state to the next state by the allowable codeword's for that state. There are only two choices possible at each state. These are determined by the arrival of either a 0 or a 1 bit. The arrows show the input bit and the output bits are shown in parentheses. The arrows going upwards represent a 0 bit and going downwards represent a 1 bit. The trellis diagram is unique to each code, same as both the state and tree diagrams are. We can draw the trellis for as many periods as we want. Each period repeats the possible transitions. We always begin at state 00. Starting from here, the trellis expands and in L bits becomes fully populated such that all transitions are possible. The transitions then repeat from this point on. The output of each transition is written on the line within brackets as shown. The state transitions for a given '1' input are denoted by dotted lines while state transitions for a given '0' input are denoted by solid lines. A trellis diagram featuring the present state and next state values is shown in the figure 3.

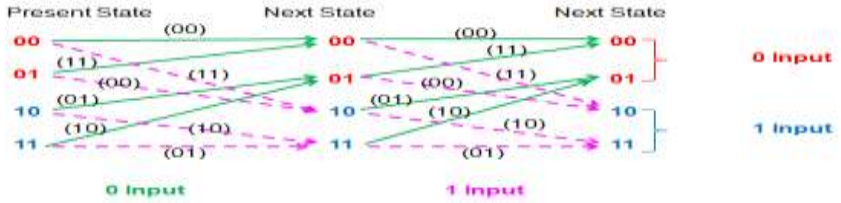

Figure 3 Code Trellis

State Table

The State table is obtained from the code trellis. The arrows emanating from 00 and 01 present state end in 00. Hence we can divide into two states (00 in state 0 and 01 in state 1) and write the respective output values in value 0 and value 1.Repeat the same for other states as well.

State 0	State 1	Value 0	Value 1
00	01	00	11
10	11	01	10
00	01	11	00
10	11	10	01

Figure 4 State Table

The specifications of the above constructed encoder are tabulated below in Table 1

Table 1 Parameters of Convolution Encoder

PARAMETERS	VALUE
Block length	8
Code rate	½
Code vectors	16
Constraint length	3
Dimensions of the code	(8,4)

Example

Consider a 4 bit input to the convolution encoder 0101.The present state is buffer value 00.The output bits are obtained from the expressions according to Fig 1.The next state is obtained by right shifting input and XY bits together. This becomes the present state for the next bit. In this way the output codeword is obtained.

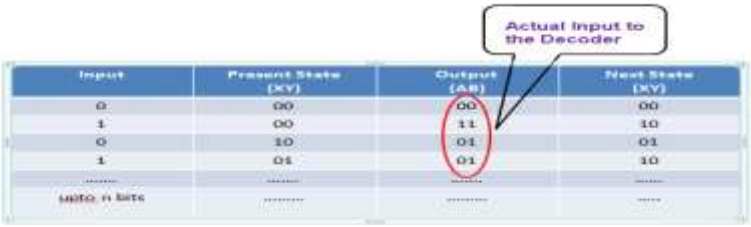

Figure 5 Convolution Encoder Logic Example

In this way the convolution encoder works and the 8-bit output codeword is obtained.

III. VITERBI DECODER

Although convolution encoding is a simple procedure, decoding of a convolution code is much more complex task. Several classes of algorithms exist for this purpose:

- Threshold decoding is the simplest of them, but it can be successfully applied only to the specific classes of convolution codes. It is also far from optimal.
- Sequential decoding is a class of algorithms performing much better than threshold algorithms. Their serious advantage is that decoding complexity is virtually independent from the length of the particular code. Although sequential algorithms are also suboptimal, they are successfully used with very long codes, where no other algorithm can be acceptable. The main drawback of sequential decoding is unpredictable decoding latency.
- Viterbi decoding is an optimal (in a maximum-likelihood sense) algorithm for decoding of a convolution code. Its main drawback is that the decoding complexity grows exponentially with the code length. So, it can be utilized only for relatively short codes. [4]

The Viterbi algorithm works by forming trellis structure, which is eventually traced back for decoding the received information. It reduces the computational complexity by using simpler trellis structure. The Viterbi Decoder is used in many FEC applications and in systems where data are transmitted and subject to errors before reception.

Viterbi decoders also have the property of compressing the number of bits of the data input to half. As a result redundancy in the codes is also reduced. Hence Viterbi decoding is more effective and efficient. The Viterbi decoder designed here is 8:4 decoder. The same logic and concept can also be extended to further number of bits also. Viterbi decoders are based on the basic algorithm which comprises of minimum path and minimum distance calculation and retracing the path.

Block Diagram

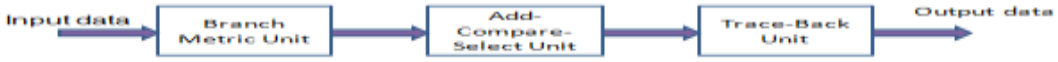

Figure 6 Block Diagram of Viterbi Decoder

The BMU (branch metric unit) receives input data from the channel and computes a metric for each state and input combination. The metric is the Hamming distance for hard-decision encoded data. The ACS (add-compare-select) unit is the second functional unit of the Viterbi decoder. This is based on minimum distance calculations that are obtained from the previous row values. Trace-back unit restores an (almost) maximum-likelihood path from the decisions made by BMU. This is the final stage of the Viterbi decoder where the input that was transmitted by using the convolution encoder is once again retrieved and the 4 bit message is obtained.

Example

Consider the input codeword to the decoder which is 8 bits. Split it into 2 bits each and perform ex-or operations of each 2 bits with values from state table for Value 0 and Value 1.This would result in 4 rows for each of the 2 bits.

Now calculate the minimum distance by taking the minimum value from the result obtained from ex-or operations.

For the second row we take into account the previous minimum distance values which are obtained according to the state value of the respective binary combination of value 0 or 1.These are added to the results obtained from ex-or operations. In order to calculate the path values, take the minimum distance values respective state value. Continue this for all the rows. After obtaining all rows, start with the last row's minimum distance and check the corresponding binary input. If the 2-bit lies in the 0 input side take the input as 0 else take it as 1.Similarly trace back to obtain all the 4 bits of the input and reverse them to get the decoded Output.

IV. SOFTWARES USED

Xilinx ISE is a software tool produced by Xilinx for synthesis and analysis of HDL designs, which enables the developer to synthesize ("compile") their designs, perform timing analysis, examine RTL diagrams, simulate a design's reaction to different stimuli, and configure the target device with the programmer. This design is simulated using ModelSim Starter Edition and synthesized using Xilinx 10.1 ISE. ModelSim SE entry-level simulator, offers VHDL, Verilog, or mixed-language simulation.

Using Hardware Description Languages (HDLs) to design high-density FPGA devices has the following advantages:

1. Top-Down Approach for Large Projects: Designers use HDLs to create complex designs. The top-down approach to system design works well for large HDL projects that require many designers working together. After the design team determines the overall design plan, individual designers can work independently on separate code sections.
2. Functional Simulation Early in the Design Flow: You can verify design functionality early in the design flow by simulating the HDL description. Testing your design decisions before the design is implemented at the Register Transfer Level (RTL) or gate level allows you to make any necessary changes early on.
3. Synthesis of HDL Code to Gates: Synthesizing your hardware description to target the FPGA implementation:
   - Decreases design time by allowing a higher-level design specification, rather than specifying the design from the FPGA base elements.
   - Reduces the errors that can occur during a manual translation of a hardware description to a schematic design.
   - Allows you to apply the automation techniques used by the synthesis tool during optimization to the original HDL code. This results in greater optimization and efficiency.
4. Early testing of Various Design Implementations
5. Reuse of RTL Code [5]

Designing FPGA Devices with Verilog

Verilog is popular for synthesis designs because:

- Verilog is less verbose than traditional VHDL.
- Verilog is standardized as IEEE-STD-1364-95 and IEEE-STD-1364-2001.

Since Verilog was not originally intended as an input to synthesis, many Verilog constructs are not supported by synthesis tools.

V. COMPARATIVE STUDY

1.2 Previous Architectures

There are different approaches of implementation for Convolution Encoder and Viterbi Decoder in the literatures. A Viterbi decoder can be implemented using a DSP or as an ASIC [6]. Implementing the Viterbi decoder as an ASIC is more efficient in terms of power and performance. However, an ASIC is, for the most part, a fixed design, and does not allow for much operational flexibility. A DSP provides a large amount of operational flexibility but this is gained at the loss of performance and power efficiency. These Implementations have fixed constraint length and Code Rate or have partial configuration facility. In order to overcome the performance issue of Convolution Encoder and Viterbi Decoder and have more flexible configuration, FPGA based implementation has been proposed. The advantages of the FPGA approach to DSP Implementation include higher sampling rates than are available from traditional DSP chips, lower costs than an ASIC. The FPGA also adds design flexibility and adaptability with optimal device utilization conserving both board space and system power that is often not the case with DSP chips.

Table 2 Comparative Study between different implementations

	ASIC	**DSP**	**FPGA**
Flexibility	Not much	good	best
Performance	better	good	best
Area Utilization	good	good	best

VI. SIMULATION AND SYNTHESIS RESULTS

Synthesis is a process of constructing a gate level netlist from a register transfer level model of a circuit described in Verilog HDL. Increasing design size and complexity, as well as improvements in design synthesis and simulation tools, have made Hardware Description Languages (HDLs) the preferred design languages of most integrated circuit designers. The two leading HDL synthesis and simulation languages are Verilog and VHDL. Both have been adopted as IEEE standards. The Xilinx ISE™ software is designed to be used with several HDL synthesis and simulation tools that provide a solution for programmable logic designs from beginning to end. [7] RTL View is a Register Transfer Level graphical representation of your design. This representation (.ngr file produced by Xilinx Synthesis Technology (XST)) is generated by the synthesis tool at earlier stages of a synthesis process when technology mapping is not yet completed. After the HDL synthesis phase of the synthesis process, the RTL Viewer can be used to view a schematic representation of the pre-optimized design in terms of generic symbols that are independent of the targeted Xilinx device, for example, in terms of adders, multipliers, counters, AND gates, and OR gates. After the optimization and technology targeting phase of the synthesis process, the Technology Viewer can be used to view a schematic representation of the design in terms of logic elements optimized to the target Xilinx device or "technology," for example, in terms of LUTs, carry logic, I/O buffers, and other technology-specific components.

The technology schematic of Encoder is shown in the Figure 7.

Figure 7 Technology Schematic of Convolution Encoder

The below figures show the device utilization summary synthesis report of Encoder and decoder respectively. When compared to other architectures, this proposed architecture saves silicon area by optimizing the utilization of the chip.

Device utilization summary:

Number of Slices:	16	out of	768	2%
Number of Slice Flip Flops:	24	out of	1536	1%
Number of 4 input LUTs:	20	out of	1536	1%
Number of bonded IOBs:	14	out of	63	22%
Number of GCLKs:	1	out of	8	12%

Device utilization summary:

Number of Slices:	86	out of	768	11%
Number of Slice Flip Flops:	17	out of	1536	1%
Number of 4 input LUTs:	142	out of	1536	9%
Number of bonded IOBs:	14	out of	63	22%
Number of GCLKs:	1	out of	8	12%

The utilization for Convolution Encoder and decoder for slice flipflops is 1% which comprises of total logic to be implemented thus making it an efficient architecture.

VII. CONCLUSION

This architecture has comparatively simpler code and flexible configuration when compared to other architectures and saves silicon area through efficient device utilization which makes it favorable for present day FPGA's. Although this architecture has limitations because of the increasing number of computations in decoding performed at each stage which makes it impractical for convolution codes with large constraint length, it provides a good trade-off between performance and area.

REFERENCES

[1] Viterbi.A.J, "Convolution codes and their performance in communication systems," *IEEE Transaction on Communications*, vol.com-19, pp. 751 to 771, October 1971.

[2] Wong, Y., Jian, W., HuiChong, O., Kyun, C., Noordi, N. "Implementation of Convolutional Encoder and Viterbi Decoder using VHDL", Proceedings of IEEE International conference on Research and Development Malaysia, November 2009.

[3] "A Viterbi Decoder Using System C for Area Efficient VLSI Implementation" Thesis by Serkan Sozen.

[4] http://www.1-core.com

[5] http://www.xilinx.com/itp/xilinx10/books/docs/sim/sim.pdf

[6] Architectures for ASIC implementations of low-density parity-check convolutional encoders and decoders Swamy, Bates, et al. - 2005

[7] http://www.xilinx.com/support/documentation/sw_manuals/xilinx13_1/ug685.pdf

[8] Verilog HDL Synthesis A Practical primer By J. Bhaskar.