ISSN:0975-9646

Implementation of Viterbi Decoder for Convolutional Code in HDL

Nilesh G. Nirmal, Gajanan U.Patil, Prafulla.P.Chaudhari, Mandar.M.Kulkarni
Department of Electronics & Communication Engineering
SSGBCOE&T, Bhusawal
Bhusawal, India

***Abstract*—Convolutional encoding and decoding (Viterbi decoding) is a powerful method for forward error detection & correction. It has been widely deployed in many communication systems to improve the limited capacity and code rate of the communication channels. The Viterbi algorithm, which is the most extensively employed decoding algorithm for convolutional codes. In this paper, we present a Simulation soft core implementation of Viterbi Decoder with a constraint length of three and a code rate of 2/3.**

Keywords-component; Convolutional codes, Viterbi Algorithm, Viterbi decoder, Phase-Shift Key, trellis, Path memory, Simulation.

I. INTRODUCTION

This section describes an ASIC design for a Viterbi decoder using Verilog. Viterbi encoding is widely used for satellite and other noisy communications channels. There are two important components of a channel using Viterbi encoding: the Viterbi encoder (at the transmitter) and the Viterbi decoder (at the receiver). A Viterbi encoder includes extra information in the transmitted signal to reduce the probability of errors in the received signal that may be corrupted by noise.

We shall describe an encoder in which every two bits of a data stream are encoded into three bits for transmission. The ratio of input to output information in an encoder is the rate of the encoder; this is a rate 2/3 encoder. The following equations relate the three encoder output bits (Y_n^2 , Y_n^1 , and Y_n^0) to the two encoder input bits (X_n^2 and X_n^1) at a time nT:

$$Y_n^2 = X_n^2$$
$$Y_n^1 = X_n^1 \text{ xor } X_{n-2}^1$$
$$Y_n^0 = X_{n-1}^1$$

We can write the input bits as a single number. Thus, for example, if $X_n^2 = 1$ and $X_n^2 = 0$, we can write $X_n = 2$. Figure.1 shows a state machine with two memory elements for the two last input values for X_n^1 : X_{n-1}^1 and X_{n-2}^1 is shown. These two state variables define four states: $\{X_{n-1}^1, X_{n-2}^1\}$, with $S_0 = \{0, 0\}$, $S_1 = \{1, 0\}$, $S_2 = \{0, 1\}$, and $S_3 = \{1, 1\}$. The 3-bit output Y_n is a function of the state and current 2-bit input X_n. The following theory describes the rate 2/3 encoder. This model uses two D flip-flops as the state register. When reset (using active-high input signal res) the encoder starts in state S_0.

II. SYSTEM IMPLEMENTATION

A. Viterbi Encoder

This encoder has X_n^2 (msb) and X_n^1 form the 2-bit input message, XN. Example: if X_n^2=1, X_n^1=0, then X_n =2. Y_n^2 (msb), Y_n^1, and Y_n^0 form the 3-bit encoded signal, Y_n (for a total constellation of 8 PSK signals that will be transmitted). The encoder uses a state machine with four states to generate the 3-bit output, Y_n, from the 2-bit input, X_n. Example: the repeated input sequence $X_n = (X_n^2, X_n^1) = 0, 1, 2, 3$ produces the repeated output sequence $Y_n = (Y_n^2, Y_n^1, Y_n^0) = 1, 0, 5, 4$.

The first four rows of Table. I shows the four different transitions that can be made from state S_0 . For example, if we reset the encoder and the input is $X_n = 3$ ($X_n^2 = 1$ and $X_n^1 = 1$), then the output will be $Y_n = 6$ ($Y_n^2 = 1$, $Y_n^1 = 1$, $Y_n^0 = 0$) and the next state will be S_1 .

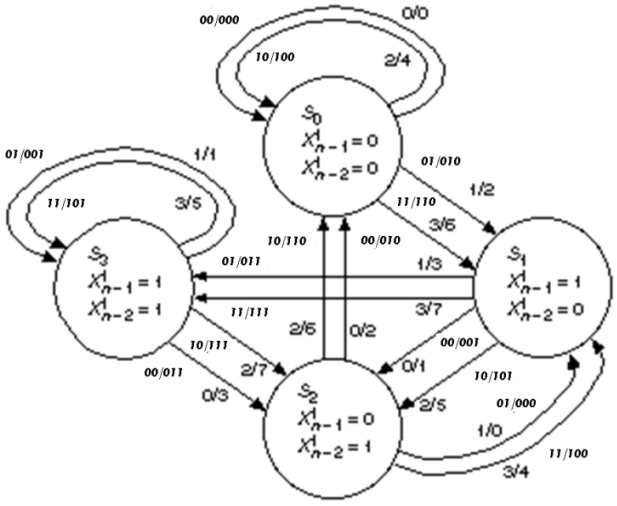

Figure 1. A state diagram for a rate 2/3 Viterbi encoder. The inputs and outputs are shown in binary as $X_n^2 X_n^1$ / $Y_n^2 Y_n^1 Y_n^0$, and in decimal as X_n/ Y_n.

TABLE I. STATE TABLES FOR THE RATE 2/3 VITERBI ENCODER

Present state	Inputs		State variables		Outputs			Next state	
	X_n^2	X_n^1	X_{n-1}^1	X_{n-2}^1	Y_n^2	$Y_n^1 = X_n^1$ xor X_{n-2}^1	$Y_n^0 = X_{n-1}^1$	$\{X_{n-1}^1, X_{n-2}^1\}$	
S_0	0	0	0	0	0	0	0	00	S_0
S_0	0	1	0	0	0	1	0	10	S_1
S_0	1	0	0	0	1	0	0	00	S_0
S_0	1	1	0	0	1	1	0	10	S_1
S_1	0	0	1	0	0	0	1	01	S_2
S_1	0	1	1	0	0	1	1	11	S_3
S_1	1	0	1	0	1	0	1	01	S_2
S_1	1	1	1	0	1	1	1	11	S_3
S_2	0	0	0	1	0	1	0	00	S_0
S_2	0	1	0	1	0	0	0	10	S_1
S_2	1	0	0	1	1	1	0	00	S_0
S_2	1	1	0	1	1	0	0	10	S_1
S_3	0	0	1	1	0	1	1	01	S_2
S_3	0	1	1	1	0	0	1	11	S_3
S_3	1	0	1	1	1	1	1	01	S_2
S_3	1	1	1	1	1	0	1	11	S_3

As an example, the repeated encoder input sequence X_n = 0, 1, 2, 3, ... produces the encoder output sequence Y_n = 1, 0, 5, 4, ... repeated. Table.II shows the state transitions for this sequence, including the initialization steps.

TABLE II. A SEQUENCE OF TRANSMITTED SIGNALS FOR THE RATE 2/3 VITERBI ENCODER

Inputs		State variables		Outputs			Present state	Next state
X_n^2	X_n^1	x_{n-1}^1	x_{n-2}^1	Y_n^2	Y_n^1	Y_n^0		
1	1	x	x	1	x	x	S?	S?
1	1	0	0	1	1	0	S0	S1
0	0	1	0	0	0	1	S1	S2
0	1	0	1	0	0	0	S2	S1
1	0	1	0	1	0	1	S1	S2
1	1	0	1	1	0	0	S2	S1
0	0	1	0	0	0	1	S1	S2
0	1	0	1	0	0	0	S2	S1
1	0	1	0	1	0	1	S1	S2
1	1	0	1	1	0	0	S2	S1
0	0	1	0	0	0	1	S1	S2
0	1	0	1	0	0	0	S2	S1

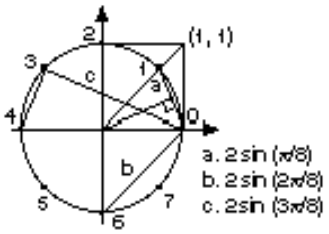

Figure.2. The Signal Constellation for An 8 PSK (Phase-Shift Keyed) Code.

Next we transmit the eight possible encoder outputs (Y_n = 0-7) as signals over our noisy communications channel (perhaps a microwave signal to a satellite) using the signal constellation shown in Figure. Typically this is done using phase-shift keying (PSK) with each signal position corresponding to a different phase shift in the transmitted carrier signal.

B. The Received Signal

The noisy signal enters the receiver. It is now our task to discover which of the eight possible signals were transmitted at each time step. First we calculate the distance of each received signal from each of the known eight positions in the signal constellation. Table.3 shows the distances between signals in the 8PSK constellation. We are going to assume that there is no noise in the channel to illustrate the operation of the Viterbi decoder, so that the distances in Table.III represent the possible distance measures of our received signal from the 8PSK signals.

The distances, X, in the first column of Table.III are the geometric or algebraic distances. We measure the Euclidean distance, E = X_2 shown as B (the binary quantized value of E) in Table.3. The rounding errors that result from conversion to fixed-width binary are quantization errors and are important in any practical implementation of the Viterbi decoder. The effect of the quantization error is to add a form of noise to the received signal.

The viterbi_distances module models the receiver section that digitizes the noisy analog received signal and computes the binary distance measures. Eight binary-distance measures, in0-in7, are generated each time a signal is received. Since each of the distance measures is 3 bits wide, there are a total of 24 bits that form the digital inputs to the Viterbi decoder.

C. Module viterbi_distances

This module simulates the front end of a receiver. Normally the received analog signal (with noise) is converted into a series of distance measures from the known eight possible transmitted PSK signals: s0,...,s7. We are not simulating the analog part or noise in this version, so we just take the digitally encoded 3-bit signal, Y, from the encoder and convert it directly to the distance measures. d[N] is the distance from signal = N to signal = 0 d[N] = (2*sin(N*PI/8))**2 in 3-bit binary (on the scale 2=100) Example: d[3] = 1.85**2 = 3.41 = 110 in N is the distance from signal = N to encoder signal. Example: in3 is the distance from signal = 3 to encoder signal. d[N] is the distance from signal = N to encoder signal = 0. If encoder signal = J, shift the distances by 8-J positions. Example: if signal = 2, in0 is d[6], in1 is D[7], in2 is D[0], etc.

As an example, Table IV shows the distance measures for the transmitted encoder output sequence Y_n = 1, 0, 5, 4, ... (repeated) corresponding to an encoder input of X_n = 0, 1, 2, 3, ... (repeated).

TABLE III. REPRESENTATION OF THE POSSIBLE DISTANCE MEASURES OF OUR RECEIVED SIGNAL FROM THE 8PSK SIGNALS

Signal	Algebraic distance from signal 0	X = Distance from signal 0	Euclidean distance E = X_2	B = binary quantized value of E	D = decimal value of B	Quantization error Q = D - 1.75 E
0	2 sin (0 π / 8)	0.00	0.00	000	0	0
1	2 sin (1 π / 8)	0.77	0.59	001	1	-0.0325
2	2 sin (2 π / 8)	1.41	2.00	100	4	0.5
3	2 sin (3 π / 8)	1.85	3.41	110	6	0.0325
4	2 sin (4 π / 8)	2.00	4.00	111	7	0
5	2 sin (5 π / 8)	1.85	3.41	110	6	0.0325
6	2 sin (6 π / 8)	1.41	2.00	100	4	0.5
7	2 sin (7 π / 8)	0.77	0.59	001	1	-0.0325

TABLE IV. RECEIVER DISTANCE MEASURESFOR AN EXAMPLE TRANSMISSION SEQUENCE

Input X_n	Output Y_n	Present state	Next state	in0	in1	in2	in3	in4	in5	in6	in7
3	x	S?	S?	x	x	x	x	x	x	x	x
3	6	S0	S1	4	6	7	6	4	1	0	1
0	1	S1	S2	1	0	1	4	6	7	6	4
1	0	S2	S1	0	1	4	6	7	6	4	1
2	5	S1	S2	6	7	6	4	1	0	1	4
3	4	S2	S1	7	6	4	1	0	1	4	6
0	1	S1	S2	1	0	1	4	6	7	6	4
1	0	S2	S1	0	1	4	6	7	6	4	1
2	5	S1	S2	6	7	6	4	1	0	1	4
3	4	S2	S1	7	6	4	1	0	1	4	6
0	1	S1	S2	1	0	1	4	6	7	6	4
1	0	S2	S1	0	1	4	6	7	6	4	1

III. TESTING THE SYSTEM

Here is a testbench for the entire system: encoder, receiver front end, and decoder:

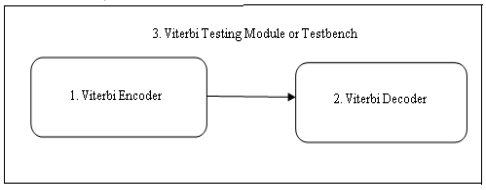

Figure.3. Block diagram of the entire system

A. *Module viterbi_test*

This is the top-level module, viterbi_test, that models the communications link. It contains three modules: viterbi_encode, viterbi_distances, and viterbi. There is no analog and no noise in this version. The 2-bit message, X, is encoded to a 3-bit signal, Y. In this module the message X is generated using a simple counter. The digital 3-bit signal Y is transmitted, received with noise as an analog signal (not modeled here), and converted to a set of eight 3-bit distance measures, in0, ..., in7. The distance measures form the input to the Viterbi decoder that reconstructs the transmitted signal Y, with an error signal if the measures are inconsistent(CDD = counter input, digital transmission, digital reception). The Viterbi decoder takes the distance measures and calculates the most likely transmitted signal. It does this by keeping a running history of the previously received signals in a path memory. The path-memory length of this decoder is 12. By keeping a history of possible sequences and using the knowledge that the signals were generated by a state machine, it is possible to select the most likely sequences.

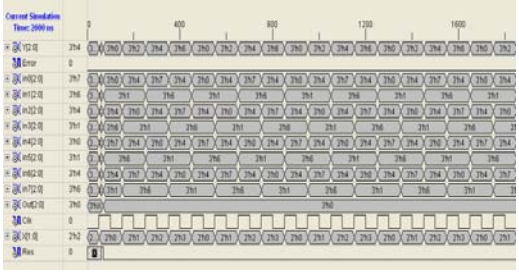

Figure.4. Simulator output from the testbench (displayed using Xilinx ISE).

The system input or message, X[1:0] , is driven by a counter that repeats the sequence 0, 1, 2, 3, ... incrementing by 1 at each positive clock edge (with a delay of one time unit), starting with X equal to 3 at t = 0. The active-high reset signal, Res, is asserted. The encoder output, Y [2:0], changes after the first positive clock edge following the deassertion of the reset. The encoder output sequence is 2, 5, 4, 1, 0…. and then the sequence 5, 4, 1, 0... repeats. This encoder output sequence is then imagined to be transmitted and received. The receiver module calculates the distance measures and passes them to the decoder. The transmitted sequence appears at the output, out [2:0], with 2, 5, 4, 1, 0... exactly the same as the encoder output.

IV. INTERNAL BLOCKS OF DECODER

A. *Verilog Decoder Model*

The Viterbi decoder model presented in this section is written for simulation. The Viterbi decoder makes extensive use of vector D flip-flops (registers).

B. *A D flip-flop module*

We use this model by defining a parameter that specifies the bus width as follows:

dff #(3) subout0(in0, sub0, clk, reset);

The code is not flexible, because bit widths are fixed rather than using parameters. A model with parameters for rate, signal constellation, distance measure resolution, and path memory length is considerably more complex.

Verilog code for a Viterbi decoder. The decoder assumes a rate 2/3 encoder, 8 PSK modulation, and trellis coding. The viterbi module contains eight submodules: subset_decode, metric, compute_metric, compare_select, reduce, pathin, path_memory, and output_decision.

The decoder accepts eight 3-bit measures of ||r-si||**2 and, after an initial delay of thirteen clock cycles, the output is the best estimate of the signal transmitted. The distance measures are the Euclidean distances between the received signal r (with noise) and each of the (in this case eight) possible transmitted signals s0 to s7.

This is the top level of the Viterbi decoder. The eight input signals {in0,...,in7} represent the distance measures, ||r-

si||**2. The other input signals are clk and reset. The output signals are out and error.

C. Module subset_decode

This module chooses the signal corresponding to the smallest of each set {||r-s0||**2,||r-s4||**2}, {||r-s1||**2, ||r-s5||**2}, {||r-s2||**2,||r-s6||**2}, {||r-s3||**2,||r-s7||**2}. Therefore there are eight input signals and four output signals for the distance measures. The signals sout0, sout3 are used to control the path memory. The statement dff #(3) instantiates a vector array of 3 D flip-flops.

D. Module compute_metric

This module computes the sum of path memory and the distance for each path entering a state of the trellis. For the four states, there are two paths entering it; therefore eight sums are computed in this module. The path metrics and output sums are 5 bits wide. The output sum is bounded and should never be greater than 5 bits for a valid input signal. The overflow from the sum is the error output and indicates an invalid input signal.

E. Module compare_select

This module compares the summations from the compute_metric module and selects the metric and path with the lowest value. The output of this module is saved as the new path metric for each state. The ACS output signals are used to control the path memory of the decoder.

F. Module path

This is the basic unit for the path memory of the Viterbi decoder. It consists of four 3-bit D flip-flops in parallel. There is a 2:1 mux at each D flip-flop input. The statement dff #(12) instantiates a vector array of 12 flip-flops.

G. Module path_memory

This module consists of an array of memory elements (D flip-flops) that store and shift the path memory as new signals are added to the four paths (or four most likely sequences of signals). These module instantiates 11 instances of the path module.

H. Module pathin

This module determines the input signal to the path for each of the four paths. Control signals from the subset decoder and compare select modules are used to store the correct signal. The statement dff #(12) instantiates a vector array of 12 flip-flops.

I. Module metric

The registers created in this module (using D flip-flops) store the four path metrics. Each register is 5 bits wide. The statement dff #(5) instantiates a vector array of 5 flip-flops.

J. Module output_decision

This module decides the output signal based on the path that corresponds to the smallest metric. The control signal comes from the reduce module.

K. Module reduce

This module reduces the metrics after the addition and compare operations. This algorithm selects the smallest metric and subtracts it from all the other metrics.

V. CONCLUSION

In this paper, a Viterbi algorithm based on the strongly connected trellis decoding of binary convolutional codes has been presented. The use of error-correcting codes has proven to be an effective way to overcome data corruption in digital communication channels. The Viterbi decoder is modeled using Verilog, and Simulated by Xilinx ISE .We can implement a higher performance Viterbi decoder with such an algorithm. So in the future, with this algorithm with larger code rates we can get better results.

REFERENCES

[1] Ranjan Bose, “Information theory coding and Cryptography”, McGraw-Hill.

[2] J.G. Proakis. ”Digital Communications.” McGraw Hill, second edition, 1989.

[3] Robert G. Gallager, “Information Theory & Reliable Communication” John Wiley & Sons.

[4] Weng Fook Lee, “Verilog Coding for Logic Synthesis”, A John Wiley & Sons, Inc., Publication.

[5] Samir Palnitkar, “Verilog HDL: A Guide to Digital Design And Synthesis, Second Edition”, Prentice Hall PTR.

[6] Volker Kuhn, “Wireless Communications over MIMO Channels-Applications to CDMA and Multiple Antenna Systems” John Wiley & Sons Ltd.