Design of High Speed and Low Power Veterbi Decoder for Trellis Coded Modulation Decoders

B.Govind[1] Sk.Subhan[2] D.Venkatrami Reddy[3]
[1]M.Tech(VLSI) Final Year Student [2]Assistant Professor [3]Associate Professor
[1,2,3]Department of Electronics and Communication Engineering
[1,2,3]JNTUH, Madhira Institute of Technology & Science, Telangana, India

Abstract— It is well known that the Viterbi decoder (VD) is the dominant module determining the overall power consumption of TCM decoders. High-speed, low-power design of Viterbi decoders for trellis coded modulation (TCM) systems is presented in this paper. We propose a pre-computation architecture incorporated with -algorithm for VD, which can effectively reduce the power consumption without degrading the decoding speed much. A general solution to derive the optimal pre-computation steps is also given in the paper. Implementation result of a VD for a rate-3/4 convolutional code used in a TCM system shows that compared with the full trellis VD, the precomputation architecture reduces the power consumption by as much as 70% without performance loss, while the degradation in clock speed is negligible.

Keywords: viterbi decoder, VLSI, Trellis coded modulation (TCM)

I. INTRODUCTION

In telecommunication, trellis modulation (also known as trellis coded modulation, or simply TCM) is a modulation scheme which allows highly efficient transmission of information over band-limited channels such as telephone lines. Trellis modulation was invented by Gottfried Ungerboeck working for IBM in the 1970s, and first described in a conference paper in 1976; but it went largely unnoticed until he published a new detailed exposition in 1982 which achieved sudden widespread recognition.

The name *trellis* was coined because a state diagram of the technique, when drawn on paper, closely resembles the trellis lattice used in rose gardens(shown in Fig 1). The scheme is basically a convolutional code of rates (r,r+1). Ungerboeck's unique contribution is to apply the parity check on a per symbol basis instead of the older technique of applying it to the bit stream then modulating the bits. The key idea he termed Mapping by Set Partitions. This idea was to group the symbols in a tree like fashion then separate them into two limbs of equal size. At each limb of the tree, the symbols were further apart. Although hard to visualize in multi-dimensions, a simple one dimension example illustrates the basic procedure. Suppose the symbols are located at [1, 2, 3, 4, ...]. Then take all odd symbols and place them in one group, and the even symbols in the second group. This is not quite accurate because Ungerboeck was looking at the two dimensional problem, but the principle is the same, take every other one for each group and repeat the procedure for each tree limb. He next described a method of assigning the encoded bit stream onto the symbols in a very systematic procedure. Once this procedure was fully described, his next step was to program the algorithms into a computer and let the computer search for the best codes. The results were astonishing. Even the most simple code (4 state) produced error rates nearly one one-thousandth of an equivalent uncoded system. For two years Ungerboeck kept these results private and only conveyed them to close colleagues. Finally, in 1982, Ungerboeck published a paper describing the principles of trellis modulation.

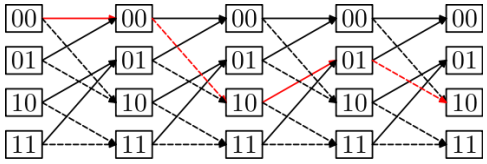

Fig. 1: Trellis Diagram

A flurry of research activity ensued, and by 1990 the International Telecommunication Union had published modem standards for the first trellis-modulated modem at 14.4 kilobits/s (2,400 baud and 6 bits per symbol). Over the next several years further advances in encoding, plus a corresponding symbol rate increase from 2,400 to 3,429 baud, allowed modems to achieve rates up to 34.3 kilobits/s (limited by maximum power regulations to 33.8 kilobits/s). Today, the most common trellis-modulated V.34 modems use a 4-dimensional set partition which is achieved by treating two 2-dimensional symbols as a single lattice. This set uses 8, 16, or 32 state convolutional codes to squeeze the equivalent of 6 to 10 bits into each symbol sent by the modem (for example, 2,400 baud × 8 bits/symbol = 19,200 bit/s).

Once manufacturers introduced modems with trellis modulation, transmission rates increased to the point where interactive transfer of multimedia over the telephone became feasible (a 200 kilobyte image and a 5 megabyte song could be downloaded in less than 1 minute and 30 minutes, respectively). Sharing a floppy disk via a BBS could be done in just a few minutes, instead of an hour. Thus Ungerboeck's invention played a key role in the Information Age.

II. VITERBI DECODER

The requirements for the Viterbi decoder, which is a processor that implements the Viterbi algorithm, depend on the application in which it is used. This result in a very wide range of required data throughputs and may impose area or power restrictions. The Viterbi detectors used in cellular telephones have low data rates (typically less than 1Mb/s) but must have very low energy consumption. On the opposite end of the scale, very high speed Viterbi detectors are used in magnetic disk drive read channels, with throughputs over 600Mb/s but power consumption are not as critical. Since both of these are high volume applications, reduced silicon area can reduce cost significantly. The basic units of Viterbi decoder are branch metric unit, add compare and select unit and survivor memory management unit.

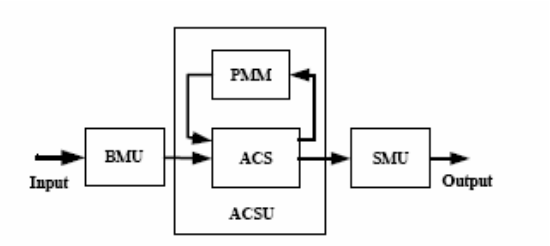

Fig. 2:Block Diagram of Viterbi decoder

A trellis diagram is a time-indexed version of a state machine, and the simplest 2-state trellis is shown in

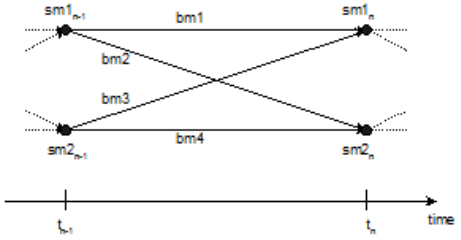

Fig. 2. Each state corresponds to a possible pattern of recently received data bits and each branch corresponds to a receipt of the next (noisy) input. The goal is to find the path through the trellis of maximum likelihood, because that path corresponds to the most likely pattern that the transmitter actually sent.

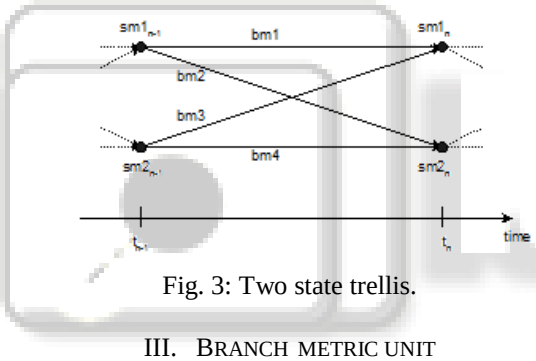

Fig. 3: Two state trellis.

III. BRANCH METRIC UNIT

The first unit is called branch metric unit. Here the received data symbols are compared to the ideal outputs of the encoder from the transmitter and branch metric is calculated. Hamming distance or the Euclidean distance is used for branch metric computation. The branch metric unit takes the fuzzy bit and calculates the cost for each branch of the trellis.

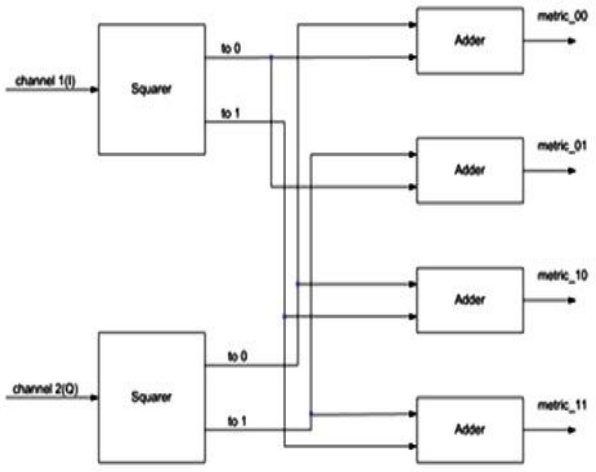

Fig. 4: Branch metric unit

The branch metric uses the Hamming distance for the four possible paths. First we initial four different received hamming distance lookup table. Then each time with check the input symbol, we get the four possible distances. The BMU perform simple check and select operations on the decision bits to generate the output. The detail hardware implementation is shown in the Figure 2

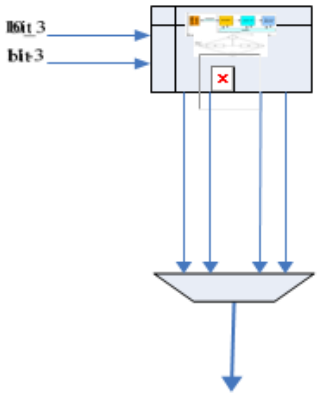

Fig. 5: one possible path hardware implementation example

IV. PATH METRIC COMPUTATION UNIT

The second unit, called path metric computation unit, calculates the path metrics of a stage by adding the branch metrics, associated with a received symbol, to the path metrics from the previous stage of the trellis. The add-compare-select unit is the heart of the Viterbi algorithm and calculates the state metrics. These state metrics accumulate the minimum cost of 'arriving' in a specific state. The branch metrics are added to state metrics from the previous time instant and the smaller sum is selected as the new state metric:

$sm1_n = \min(sm1_{n-1} + bm_1, sm2_{n-1} + bm_3)$

$sm2_n = \min(sm1_{n-1} + bm_2, sm2_{n-1} + bm_4)$

A path metric unit summarizes branch metrics to get metrics for 2^{K-1} paths, where K is the constraint length of the code, one of which can eventually be chosen as *optimal*. Every clock it makes 2^{K-1} decisions, throwing off wittingly nonoptimal paths. The results of these decisions are written to the memory of a traceback unit.

The core elements of a PMU are *ACS* (Add-Compare-Select) units. The way in which they are connected between themselves is defined by a specific code's trellis diagram.

Since branch metrics are always ≥ 0, there must be an additional circuit preventing metric counters from overflow (it isn't shown on the image). An alternate method that eliminates the need to monitor the path metric growth is to allow the path metrics to "roll over", to use this method it is necessary to make sure the path metric accumulators contain enough bits to prevent the "best" and "worst" values from coming within $2^{(n-1)}$ of each other. The compare circuit is essentially unchanged.

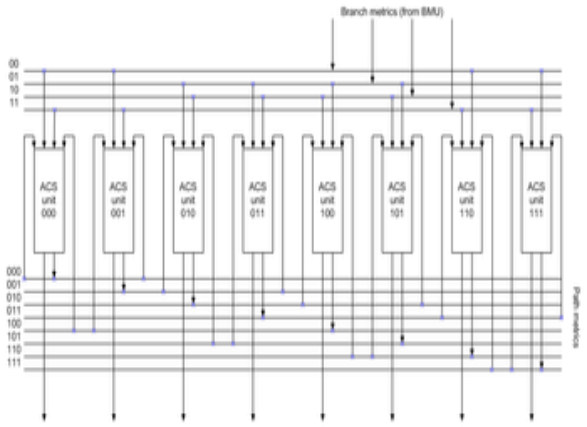

Fig. 6: sample implementation of a path metric unit for a specific K=4 decoder

V. IMPLEMENTING THE ACS UNIT

The ACS unit is responsible for implementing the state metric computation. A direct implementation requires two identical data paths each with two additions, a comparison, and a selection (hence the name ACS). The basic data path is shown in 4.3. Note that although the branch metrics are positive numbers with 5 bits each, the and state metrics are required to have 7 bits. After some amount of time, the state metrics will overflow so a trick called modulo normalization is used. The upshot is that one extra bit is required in the state metric (8 total) and the MSB of a simple subtraction can be used for the compare operation. Lastly, an array of 2:1 multiplexers is used to pick the minimum.

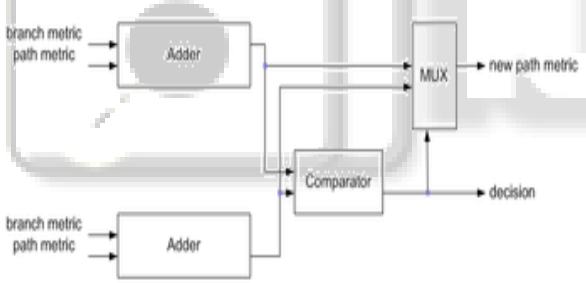

Fig. 7: sample implementation of an ACS unit

When the 4 possible input distance is ready, the ACS block' butterfly module adds the results and the related distance value stored in the state metric storage to get the each two paths for the 64 initial states. The butterfly module is shown in the next figure. Since, each butterfly computes 4 possible paths and selects the two smaller distance paths form. We have totally 32 butterflies.

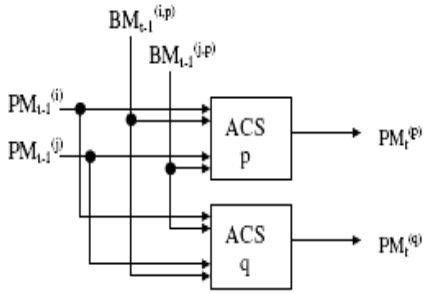

Fig. 8: The Butterfly Module

For each nod (state), the ACS module selects a smaller one as the survival path and stores them to the accumulated state metric storage block and the survivor path metric.

VI. VETERBI ALGORITHM

The Viterbi decoding algorithm proposed in 1967 is a decoding process for convolution codes. Convolution coding, as we all known, has been widely used in communication systems including deep space communications and wireless communications, such as IEEE 802.11a/g, WiMax, DAB/DVB, WCDMA and GSM.

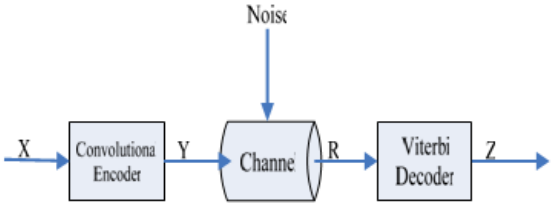

Fig. 9: A Simple Viterbi Decoding System

Viterbi decoding is the best technique for decoding the convolution codes but it is limited to smaller constraint lengths. The basic building blocks of Viterbi decoder are branch metric unit, add compare and select unit and survivor memory management unit. The two techniques for decoding the data are trace back (TB) method and Register Exchange (RE) method.

TB method is used for longer constraint lengths but is has larger decoding time. Also extra circuitry is required to reverse the decoded bits. The RE method is simpler and faster than the TB method for implementing the VD. RE method is not appropriate for decoders with long constraint lengths.

The algorithm can be broken down into the following three steps.

(1) Weigh the trellis; that is, calculate the branch metrics.
(2) Recursively computes the shortest paths to time n, in terms of the shortest paths to time n-1. In this step, decisions are used to recursively update the survivor path of the signal. This is known as add-compare-select (ACS) recursion.
(3) Recursively finds the shortest path leading to each trellis state using the decisions from Step 2. The shortest path is called the survivor path for that state and the process is referred to as survivor path decode. Finally, if all survivor paths are traced back in time, they merge into a unique path, which is the most likely signal path.

A. J. Viterbi proposed an algorithm as an 'asymptotically optimum' approach to the decoding of convolution codes in memory-less noise. The Viterbi algorithm (VA) had already been known as a maximum likelihood decoding algorithm for convolution codes.

As it is showing in the figure, a data sequence x is encoded to generate a convolution code word y. after y is transmitted through a noisy channel. The convolution decoder takes the received vector r and generates an estimate z of the transmitted code word.

The maximum likelihood (ML) decoder selects the estimate that maximizes the probability p (r|z), while the maximum a posteriori probability (MAP) decoder selects the estimate that maximizes p (z|r). If the distribution of the source bits x is uniform, the two decoders are identical. By Bayer's law, we could get: $p(r \mid z)p(z) = p(z \mid r)p(r)$.

The initial part first initializes the Viterbi decoder to the same FSM and the trellis diagram as the convolution encoder.

Then in each time clock, the decoder computes the four possible branches metric's Euclidean distance. For each state, ACS block compute the two possible paths Euclidean distance and select a small one. At the same time, ACS block will record the survival state metric.

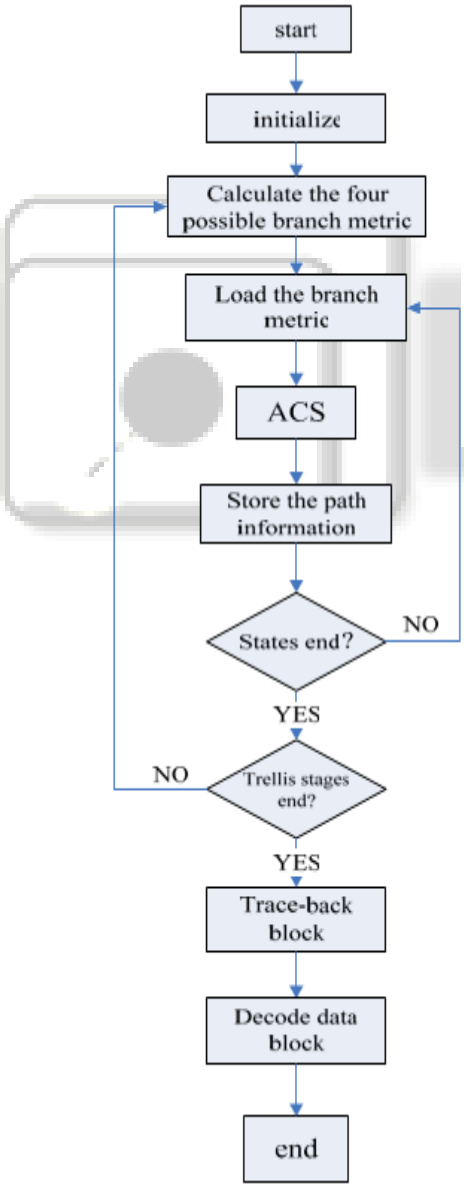

Fig. 10: Flowchart for Viterbi Algorithm

VII. VITERBI DECODING FOR NOISY CHANNEL

Consider now the case in which an error is made on channel and one bit of the received symbol is incorrect. The fourth codeword has error. Figure 2 shows first four steps of decoding process.

Similar steps are followed for all the code words and complete diagram is as shown in Figure. 12

The Figure 12 shows that when the final path is traced back even with one bit of error, output is correct. With the algorithm called iterative Veterbi decoding one can find the subsequence of an observation that matches best (on average) to a given

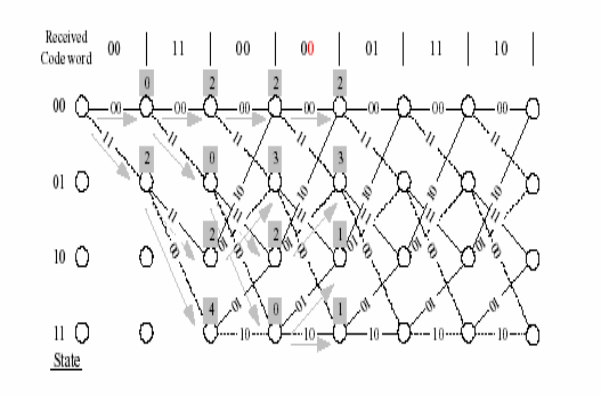

Fig. 11: Decoding for Noisy Channel for Four Code Words

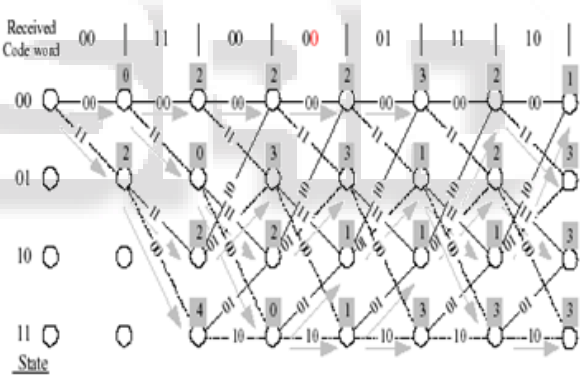

Fig. 12: Complete Decoding Diagram

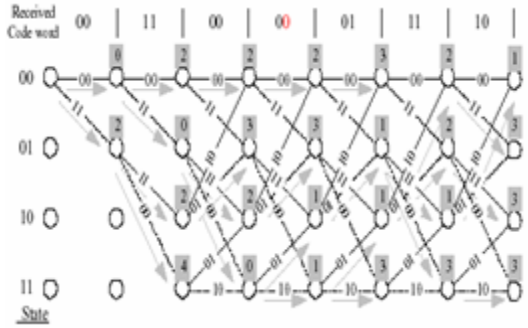

Fig. 13: Decoding for Noisy Channel Using Trace back Method

HMM. Iterative Veterbi decoding works by iteratively invoking a modified Viterbi algorithm, reestimating the score for filler until convergence. An alternative algorithm, the Lazy Veterbi algorithm, has been proposed recently. This works by not expanding any nodes until it really needs to, and usually manages to get away with doing a lot less work (in software) than the ordinary

Viterbi algorithm for the same result - however, it is not so easy to parallelize in hardware.

The Viterbi algorithm has been extended to operate with a deterministic finite automaton in order to quickly generate the trellis with state transitions pointing back at variable amount of history. The performance of the Viterbi algorithm greatly depends on the likelihood and transition probabilities between English characters, the more contextual information those probabilities could extract the better the performance of the Viterbi algorithm.

VIII. RESULTS AND DISCUSSIONS

A. Simulation and synthesis results of encoder

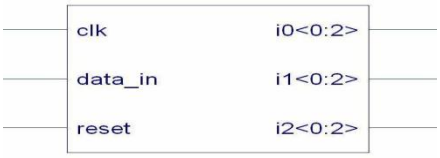

Fig. 14: RTL of Encoder

The above figure shows the RTL of convolution encoder which has one input as data_in and two outputs as i0, i1 and i2.the clock and reset pins are fed as input to the encoder. The state table and state diagram are built based upon the present state and next state of the encoder and the output is fed to the decoder.

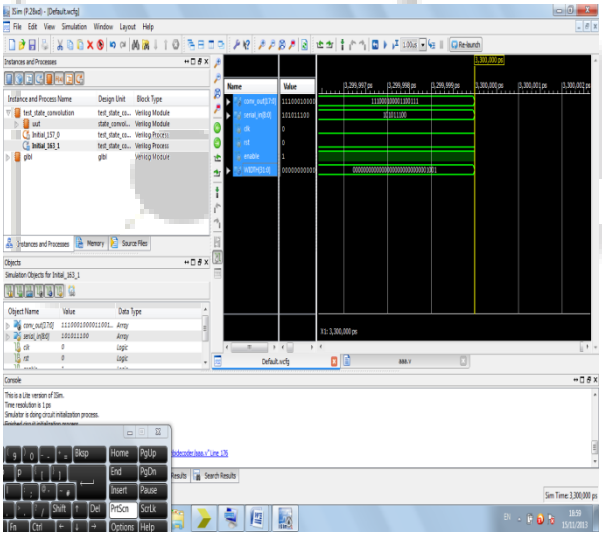

Fig. 15: Simulations of Encoder

The above waveforms are the simulation result of the convolution encoder. The first signal represents the input data given to the encoder. The next signal is the reset signal and next one is the clock signal that enables the process of the block. The signals after the clock signal represent the output with different combinations of input.

B. Simulation and synthesis results of viterbi module

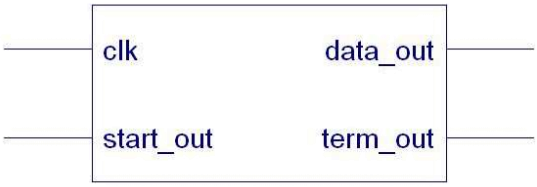

Fig. 16: RTL of Viterbi

All the modules are mapped in this module. The structured programming is used in this module. The input signals are clk, start_out. The output signals are data _out and term_out. The output data will be a serial 12 bit data. The trace back unit output gives the final output without any errors, since the errors are corrected during the traceback path.

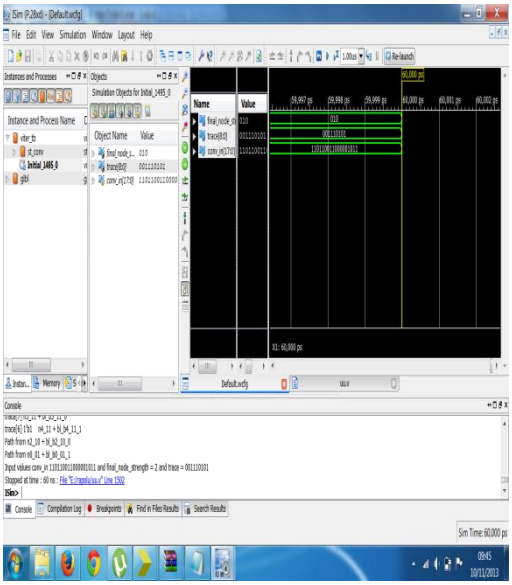

Fig. 17:Simulations of Viterbi Decoder

This waveforms shows the combined output of the branch metric unit, and ACS unit this is the final output of the Viterbi decoder the output of Viterbi decoder is of 12_bits.

C. Synthesis report of convolution encoder

1) Device utilization summary:

Selected Device	: 3s100evq100-5		
Number of Slices	: 3	out of 960	0%
Number of Slice Flip Flops	: 7	out of 1920	0%
Number of 4 input LUTs	: 1	out of 1920	0%
Number of IOs	: 16		
Number of bonded IOBs	: 16	out of 66	24%
IOB Flip Flops	: 1		
Number of GCLKs	: 1	out of 24	4%

D. Timing Summary:

Minimum input arrival time before clock: 2.191ns

Maximum output required time after clock: 4.063ns

Maximum combinational path delay: No path found

IX. SYNTHESIS REPORT OF VITERBI DECODER

HDL Synthesis Report

Macro Statistics

# Adders/Subtractors	: 82
1-bit adder carry out	: 32
3-bit adder	: 50
# Latches	: 5
1-bit latch	: 5
# Comparators	: 49

3-bit comparator equal	: 23
3-bit comparator greater	: 23
3-bit comparator not equal	: 3

X. CONCLUSION & FUTURE SCOPE

As Viterbi algorithm is conceived more interesting and challenging for this research topic, it is considered, and also it has wide variety of applications in digital communications field. This research helps to generate more profits by the developers using Viterbi algorithm. Anyone besides students can easily analyze these Viterbi algorithm concepts and can gain more knowledge about it. This research mainly concerned with implementation of Viterbi algorithm using VHDL coding. Viterbi algorithm has many advantages like low power consumption and main advantage is error correcting using VHDL. Anyone reading this document will have to gain the cognition of working with different tools like Xilinx ISE and MODELSIM. By using FPGA device and hybrid microprocessor the decoding benefits can be achieved in future. Power benefits are provided by the integration of sequential decoding. To reduce the multipath fading which damages the signals, the adaptive array technique may be used for future satellite communication.

REFERENCES

[1] "Bandwidth-efficient modulations," Consultative Committee For Space Data System, Matera, Italy, CCSDS 401(3.3.6) Green Book, Issue 1, Apr. 2003.
[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] C. F. Lin and J. B. Anderson, "□-algorithm decoding of channel convolutional codes," presented at the Princeton Conf. Info. Sci. Syst., Princeton, NJ, Mar. 1986.
[4] S. J. Simmons, "Breadth-first trellis decoding with adaptive effort," *IEEE Trans. Commun.*, vol. 38, no. 1, pp. 3–12, Jan. 1990.
[5] 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.
[6] 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.
[7] J. Jin and C.-Y. Tsui, "Low-power limited-search parallel state viterbi decoder implementation based on scarece state transition," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 15, no. 11, pp. 1172–1176, Oct. 2007.
[8] F. Sun and T. Zhang, "Lowpower state-parallel relaxed adaptive viterbi decoder design and implementation," in *Proc. IEEE ISCAS*,May 2006, pp. 4811–4814.
[9] 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.*, Nov. 2009, pp. 231–235.
[10] 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.

