International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-1 Issue-5, June 2012

Design and Implementation of Viterbi Encoder and Decoder using FPGA

Chitra. M, A.R Ashwath, Roopa. M

Abstract*— In *this paper, we present an implementation of the Viterbi algorithm using the Hardware Description Language and Implemented on FPGA. We begin with a description of the algorithm. Included are aspects of design specifications that must be considered when implementing the Viterbi algorithm as well as properties of Verilog HDL that can be used to simplify or optimize the algorithm. Finally, we evaluate the performance of the Viterbi algorithm implemented on FPGA.

Index Terms—HDL-Hardware Descriptive Language, *FPGA-Field Programmable Gate Array.*

I. INTRODUCTION

The Viterbi algorithm is a dynamic programming algorithm for finding the most likely sequence of hidden states – called the Viterbi path – that results in a sequence of observed events, especially in the context of Markov information sources, and more generally, hidden Markov models. The forward algorithm is a closely related algorithm for computing the probability of a sequence of observed events. These algorithms belong to the realm of information theory.

The main purpose of this study is to yield the gains obtained by the developers with the usage of Viterbi algorithm. This research mainly centers on the grandness of Viterbi algorithm in the practical applications with the HDL code. This research helped the people who are in the field of decoders as it is one of the efficient methods for reducing the errors while communication procedure is in advance. Here HDL code is used in order to implement the Viterbi. algorithm in a proper way. Apart from various codes, researcher selected HDL code for this research as it offers the high capability in designing the electronic systems. Viterbi algorithm is an approach towards finding the most common sequence of hidden states in all listed states. It is dynamic programming algorithms that find the probability of all observed sequence for each combination|.

Manuscript published on 30 June 2012.
***** Correspondence Author (s)
Chitra .M, Electronics and communication, Dayananda sagar college of Engineering, Bangalore, India
Dr. A.R. Ashwath, Telecommunication Engineering, Dayanada sagar college of engineering, Bangalore, India,
Roopa. M, Electronics and communication Engineering, Dayanada sagar college of engineering, Bangalore, India

II. VITERBI ALGORITHM

The Viterbi algorithm is one of the standard sections in number of high-speed modems of the process for information infrastructure applicable in modern world. The dynamic algorithm includes some path metrics so as to compute the path sequence transmitted earlier the name Viterbi algorithm arrived after Andrew Viterbi and is represented as VA for reorganization, record of huge possibility decodes as well as least reserved decoding are generally similar in a defined binary symmetric channel. Kia, J. (2005) explains Viterbi algorithm as a "dynamic algorithm that uses certain path metrics to compute the most likely path of a transmitted sequence".

The basic performance of the Viterbi decoder is analyzed with the block diagram shown below. It consists of three main blocks branch metric unit, add compare select and trace back unit. The unit of branch metric will calculate all the branch metrics and then processed to add, compare for selecting the surviving branches as per the branch metrics finally the decoded data bits are generated by the trace back unit.

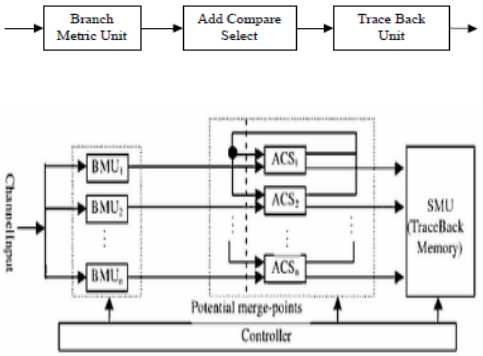

Figure 1: Basic block diagram of Viterbi Decoder

The overall performance of the Viterbi algorithm is analyzed with the help of conventional codes. The simulated block diagram explains the operation of detecting and correcting the coding errors in normal communication system. The transmitted bits of data are encoded in the first block with conventional code that is (CC encoder) which are modulated by means of binary pulse-amplitude modulation (PAM) so as to tune those bits into antipodal bits and process to the additive white Gaussian noise (AWGN) channel thus obtained data combined with noise is supplied to soft decision Viterbi algorithm (SDVA) which only accepts the antipodal data at the input for decoding and produces the output decoded bits.

Implementation of the Viterbi algorithm is supported with two main steps the initial step is to select the trellis from the bits that are achieved at the input at the receiver. A simple trellis figure shows with four stage points for transmission, each state is represented with a dot and the state transition is shown as edge of branch. Each and every branch is known as the branch metric as it is associated at Euclidean distance with the symbol towards final transition.

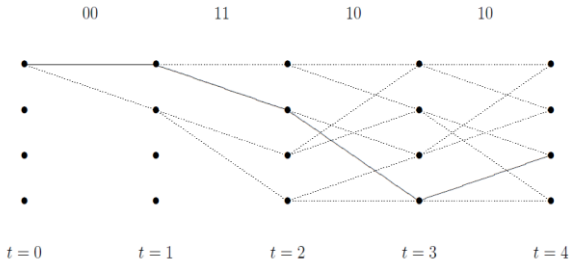

Figure 2: Viterbi algorithm Trellis.

For calculating the branch metric can be obtained with the trellis using the Euclidean analysis as follows:

BM (rr, bb) = (r0-b0)2+ (r1+ b1)2= r02−2r0b0+b02+r12−2r1b1+b12 = r0b0+r1b1…….. (1)

rr = symbol received at the input

bb = branch symbol

Both rr and bb are dependent on the used for conventional encoder. Under the basic assumption that there is no noise in the data and the value of r and b will vary between -1 to +1, the range of branch metric will range within -2 to +2.

In case rr = bb branch metric would be 2, similarly r0 = - b0 as well as r1 = - b1 and BM= -2. The path metric (λ) in the minimum Euclidean distance in the trellis does not required the actual value the original order of the floating point pair numbers is

$$\lambda new = \lambda prev + r0b0 + r1b1 \quad \text{------- (2)}$$

The path metric λ is the shortest distance among cumulative state, thus distance of the path (Euclidean distance) is inversely proportional to the branch metric. After complication of generating a trellis it is necessary to find survivor path with maximum pathmetric. In the Figure 4 the solid black line is the survivor path.

II. DESCRIPTION OF VITERBI ALGORITHM

Viterbi algorithm is basically implemented to decode the errors found in convolution encoded sequence. As discussed the Viterbi algorithm will make use of trellis structure in finding the coded sequence based on the transmission signals. Since each and every code sequence will follow based on the trellis process of encoding data. Considering an example of trellis diagram of half rate, three convolution encoder K=3 and 15 bit messages with four possible states shown in 4 horizontal rows with dotes

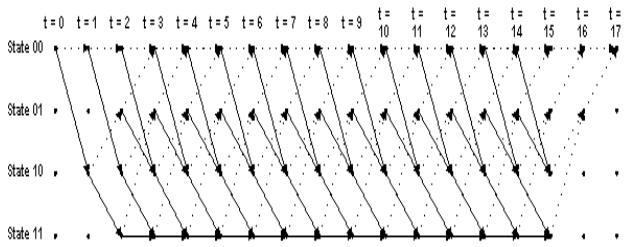

Figure 3: Trellis diagram of Viterbi algorithm

18 columns shows the time instants from t0 to t17 both t=0and t=17 has the four dot column which is initial and final state situations while encoding messages. The state transition is shown with a dotted line at zero input. The figure shows the state of trellis, which reach the encoding of messages.

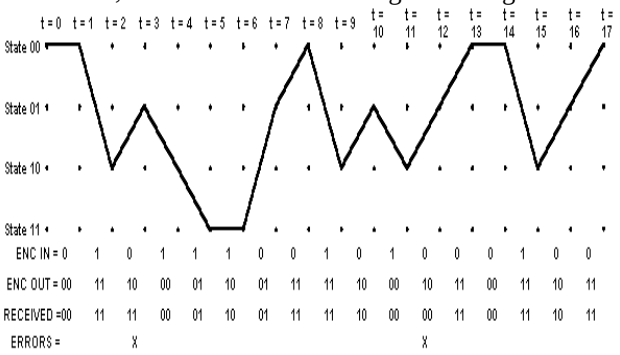

Figure 4: States of the trellis diagram of Viterbi algorithm

Maximum likelihood (ML) sequence will be obtained from the track of paths that occur for Viterbi algorithm is essential for processing information coding. The common aid to analyze the Viterbi algorithm is the trellis diagram. It is the decoding algorithm used with convolution code. The input, output, receiver and error details are shown at bottom to the figure.

The receiver pair of the channel after collecting the complete information regarding the process the Viterbi decoder is ready to function with the bit that are to be transmission following some steps. Initially number of state is to be selected at a negligible collected error metric then save the number state arrived. Accurate performs of step from the initial trellis is to be achieved. All the state sequence state numbers are to be saved. Work forward with record of all the selected states which are saved in the previous steps which are to be built in by the entire encoded convolution.

III. ADVANTAGES OF VITERBI ALGORITHM

All the trellis occurred are arbitrarily solved even in presence of two or more simple errors in input string using Viterbi algorithm. At the same time the presence of more errors, the decryption will be low even then the algorithm works effectively this is the main advantage in implementation of Viterbi algorithm. The usage of this Viterbi algorithm is found to be advantageous due to its cost effectiveness in modulated minimize at the same time the functional performance in some situation would modulate in maintaining the original cost. Emerging linear functioning of linear pulse distance is due to convenient source sequence.

IV. VITERBI ENCODER

A. *Rate of Encoder*

The code rate, is expressed as a ratio of the number of bits into the convolution encoder (k) to the number of channel symbols output by the convolution encoder (n) in a given encoder cycle. A rate 1/2 encoder is implemented in the design.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

B. Constraint Length (k)

The constraint length parameter, K, denotes the "length" of the convolution encoder, i.e. how many k-bit stages are available to feed the combinatorial logic that produces the output symbols. Closely related to K is the parameter m, which indicates how many encoder cycles an input bit is retained and used for encoding after it first appears at the input to the convolution encoder.

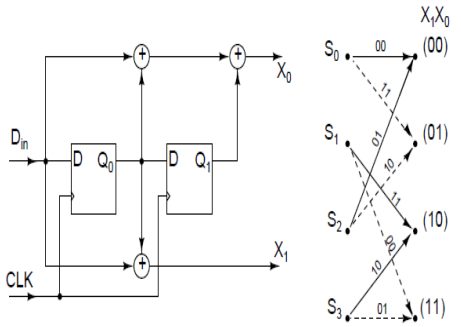

Figure 5: An example of encoder with K=3

V. VITERBI DECODER

A. Design specification

1. Code Rate = ½, Constraint Length (K) = 9.
Some consequences of using a constraint length of 9 are: The amount of branch metric is $2^K = 512$ branches. . The amount of state metric is $2^{K-1} = 256$ states. . The depth of trace back process is at least = 5 * (K-1) = 40. The version 1.0 of VDK9R1/2 does the trace back with a depth of 63 stages.
2. Polynomial generator: 753_8 and 561_8.
3. Hard Decision. Hard decision distance calculation is used on Version 1.0.

B. Design consideration

1. Output_Data_Rate
Convolution codes using constraint length = 9 are mainly used on CDMA applications. Therefore, the output data rates should be at least 9.6 kbps.
2. Area efficient VLSI implementation
The not-very-high speed requirement would enable us to try to minimize the area used for the design. On Version 1.0, the minimization is achieved by using only 4 ACS processors to do the Add-Compare-Select operations.
3. Memory requirement
The largest part of a Viterbi decoder is memory. Let's take an example using constraint length value of 9. If every metric is 8 bits wide, the memory size needed to save all the metrics is: 256 * 8 = 2.048 bits. Double it (because we have to save the current and the next metrics), we need 4.096 bits of memory space. For the survivors, with a trace back depth of 45 (which is the minimum depth we have to use), there survivor memory size is: 256 (states) * 40 (depth) = 10.240 bits. Because of its size, it would be better if we put the memory "outside" the design. The Viterbi decoder will provide an interface to the memories, and the physical implementations of the memories are not the main considerations on our design

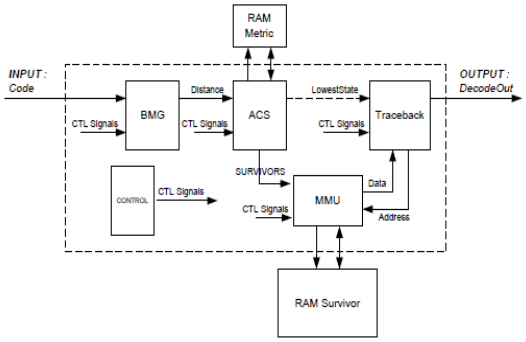

Figure 6: VDK9R1/2 Block Diagram

There are 5 main components of the Viterbi decoder: BMG (Branch Metric Generator) Unit, ACS (Add Compare Select) Unit, Trace back Unit, MMU (Memory Management Unit) and its Survivor Memory, and Control Unit. Each component's function will be briefly described below

C. Control unit

Control Unit is used to provide control signals for the other components.

D. Brach Metric Generator (BMG)

BMG Unit block diagram is shown on Figure

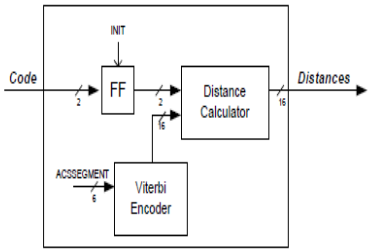

Figure 7: BMG Unit Block Diagram

The BMG receives Code signals, calculating its distance with all possibility of branch metric, giving the output of Distances signal. All possible values of branch metric is generated by the Viterbi Encoder block. The value generated is depending on the value of ACS Segment. The Distance Calculator block computes the hard-distance of Code with the branch metric from Viterbi Encoder block.

E. Add Compare Select Unit (ACS)

The ACS Unit block diagram is shown on Figure

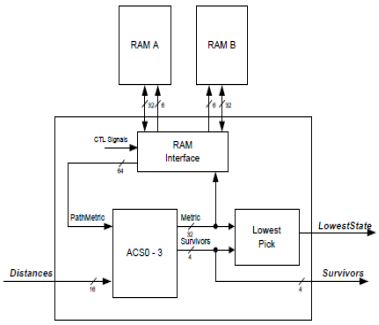

Figure 8: Add Compare Select Unit Block Diagram

The ACS Unit consists of ACS0-3 block, Lowest Pick block, and RAM Interface.ACS0 - ACS3 block consists of 4 ACS processors. The ACS processor's architecture is called Modified Comparison Rule as described in (5)

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

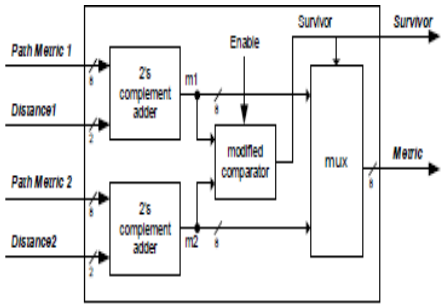

Figure 9: An ACS Processor

The Modified Comparison Rule processor uses two 2's-complement adders and two XOR gate to work. Note that there is an Enable signal on the comparator. This signal is come from the Control Unit. Because of there were only 4 ACS processors available, the input to the ACS has to be arranged so that all state could be processed efficiently. On VDK9R1/2 Version 1.0, the states to be processed are arranged sequentially. Meaning the first cycle will update state 0000 0000, 0000 0001, 0000 0010, and 0000 0011. The next cycle will update state 0000 0100, 0000 0101, 0000 0110, and 0000 0111, and so on, until the 64th cycle will update state 1111 1100, 1111 1101, 1111 1110, and 1111.

F. Lowest Pick

The Lowest Pick block is used to determine which of all 256 metrics the smallest one is and what state has those smallest metric. The value of state acquired will be used by the trace back to indicate in which state the trace back process should begin. The Lowest Pick block consists of two parts. The first part is used to find the smallest metric among every 4 outputs of ACS processor. The second part will compare the output of the first part to the previous smallest metric.

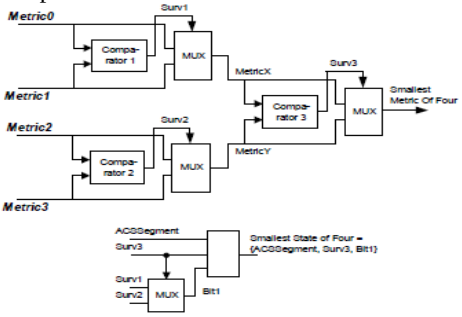

Figure 10: The first part of Lowest Pick Block

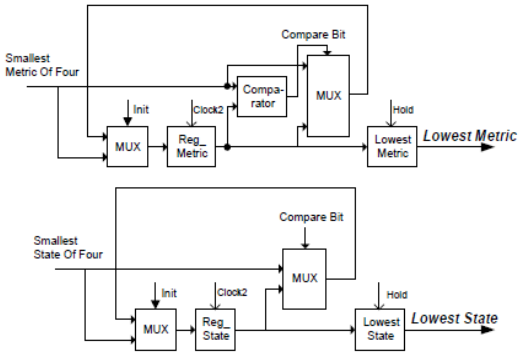

Figure 11: The second part of Lowest Pick Block

G. Metric Memory

The metric values are saved on Metric Memory. Two blocks of RAM needed as we have to know the current metric values and save the next metric values we've just calculated.

The RAM interface for ACS Block currently has not been coded tidily. There were some considerations such as the need of large data bus (32 bits), the relatively small size (2 Kbytes).

The RAM Interface job is to do the following process: Select which block of RAM used to read the metric and which one is used to write the metrics we've just calculated? (On the source code, MM Block Select signal do this). . Upon RESET, the values of metrics on both block of RAM have to be set to 0. . The update process on the ACS processor happened on every rising edge of Clock1. Therefore, the read process from the RAM (to get metric values) has to be performed on the previous rising edge of Clock2. The write process to RAM (to save the calculated perform) could be performed on falling edge of Clock2.The timing diagram could be shown as of Figure (7) below,

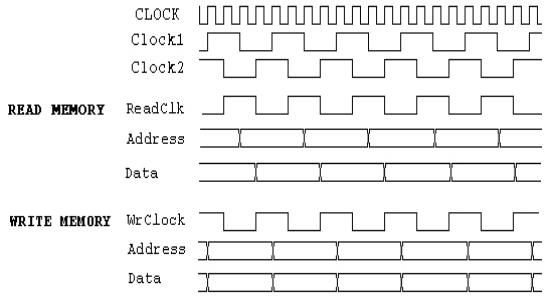

Figure 12: Metric Memory Timing Diagram

H. Trace back unit

The trace back algorithm implemented on VDK9R1/2 Version 1.0 could be described as follows:

1. On time *t*, select a node where the trace back process will start. In VDK9R1/2 the starting node is the state which has smallest metric value. This value is coming from the ACS Unit.
2. From the survivor memory, get survivor value of those nodes.
3. The previous state (on level *t –1*) is calculated by shifting left the node value by 1 bit, the LSB part then will be filled with the survivor value gathered from the memory.
4. Back to step 2 for the state on level *t-1*. Continue until node on level *t – DEPTH*.
5. The decoded values are the MSB of node on level *t – DEPTH*.

   Let's take a simple example using a K=2 convolutional code case. Suppose that on level *t*, we take node 00 as the starting point (*step 1*). The survivor data for state 00 is 1 (*step 2*). Then we know that the surviving branch must come from state: 01 (*step 3*). Back to step 2 for state 01 on level t-1 (step 4). The survivor value of state 01 level t-1 is 1 (*step 2*), then we know that the surviving branch must take node 00 as the starting point (*step 1*). The survivor data for state 00 is 1 (*step 2*). Then we know that the surviving branch must come from state: 01 (*step 3*). Back to step 2 for state 01 on level t-1 (step 4). The survivor value of state 01 level t-1 is 1 (*step 2*), then we know that the surviving branch must come from state: 11 (*step 3*). And so on until level *t-DEPTH*.

   The Trace back Unit block diagram is shown on

figure(13)

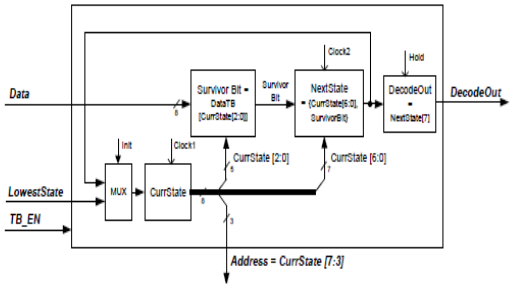

Figure 13: Trace back Unit Block Diagram

When there was an Init signal, the Lowest State value will be used as Current State value. Trace back Unit will then send Address to the MMU, requesting survivor data from the survivor memory. Because the width of survivor memory data bus is 8, and there were 256 states, there will 32 blocks of survivor data. The Address signal from the Trace back Unit, which is 5 bit width, will be used to select one block out of 32 possible blocks. The Current State [2:0] then will be used to select 1 Survivor Bit from the 8 bit Data taken from memory. The Next State value is then determined as the concatenation of Current State [5:0] and Survivor Bit. When there was a Hold signal, the Trace back Unit will output a Decode Out signal.

I. Memory Management Unit(MMU)

The survivor memory is used to save the survivors data calculated by the ACS Unit. The survivor data will be later used by the Trace back unit to find decoded data. The MMU block will control the operation of survivor memory. Assume that we use a typical RAM which has input ports: Address, Enable, Write Clock, Read Clock, and RW Select, and an inout port Data. The RAM timing diagram could be simplified as of shown at Figure 14**.**

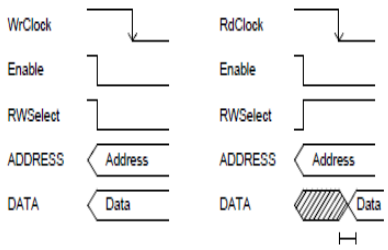

Figure 14: Survivor RAM Simplified Timing Diagram

To perform a write operation, we have to put the data and address at Data Bus and Address Bus, and then give the WrClock signal. To ensure the signal validity, there have to be enough time after we put data and address on the line before the WrClock is given.To perform a read operation, put an address, give a RdClock signal, and after a slightdelay, the data will be available on Data Bus. The size of the RAM needed to perform a trace back operation with a DEPTH of 63 is (63+1) * 256 bit = 16.384 bit. One extra page is used to save the survivor of current Code. The rest are filled with complete

sets of survivor from the previous operations. Using a Data Bus width of 8, there will be (2log16384 – 2log8) = (14 – 3) = 11 bit wide Address Bus.
The block diagram of MMU is shown on Figure (15)

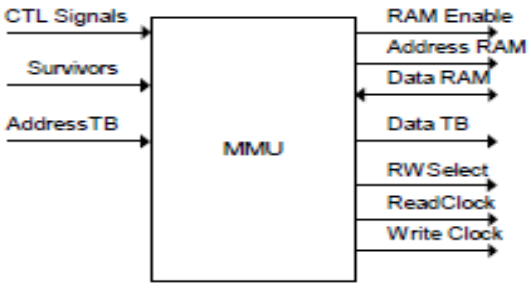

Figure 15: MMU Block Diagram

The write operation on the survivor memory will occur every 2 falling edge of Clock1. Remember that the number of ACS is only 4, while the Data Bus is 8 bit wide. The read operation will occur on every rising edge of Clock1.The survivor memory timing diagram is shown on Figure (16)

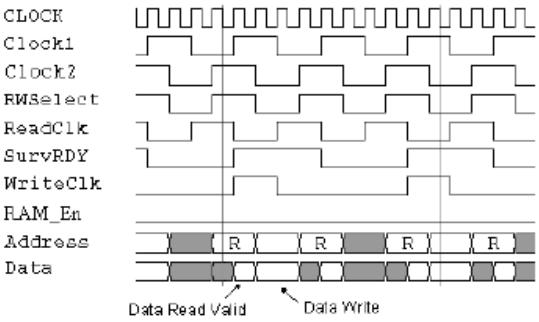

Figure 16: Survivor Memory Timing Diagram

The Survivor Memory Addressing Scheme Write Operation. The address for write operation is assembled from the combination of the value of ACSPage [5:0] and the value of ACSSegment [5:1].

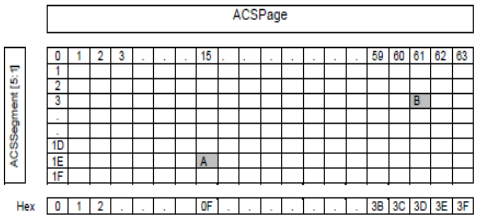

Figure 17: An Example

Let's take an example, for position A, the address would be: {0F16, 1E16} = {00_11112, 1_11102} = 1FE16. For position B, the address is {3D16, 316} = {11_11012, 0_00112} = 3A316.
For read operation, the address is assembled from the TBPage signal from the MMU, and AddressTB signal coming from the Traceback Unit. Å Address = {TBPage, AddressTB} Upon an Init signal, the value of TBPage is loaded with the value of ACSPage–1. The resulting Address, the combination of TBPage and AddressTB, will then be used to read the survivor data. DataRAM will then be sent into the Traceback Unit. The Traceback Unit will provide the next AddressTB signal. By decreasing the TBPage one on every falling edge of Clock2, we will be able to read the survivor on level *t – 1* node.

1) Are not supported by adequate data and critical details.

IV RESULTS

For clock signal when set to value '1' then it results in a continuous signal. When the signal is set to reset value '0' then it is a dc signal, and there will be no changes in the signal. When srst that is system reset value is set to '0'. For an encoded bit input value is set to '0' then the results can be obtained as
100110100111000110111000
When an encoded valid input value is set to '0' then the result can be obtained as
101010101010101010101010
When an encoded symbo0 value is given as '0' then the results are
110111000000001111111111
For an encoded symbol1 if the value is set to 1 then the analyzed results are
111011110001111000111111
Now for a decoded symbol0 when the value is set to 7 then there will be periodic changes obtained in a de-mux as 7 0 7 0 7and for decoded symbol1 when the value is given as 7 then the output is the demux value given as 0 7 0 7 0 7 0 7
When decoded valid input value is given as '0' then the output is continues signal. For a pattern value when given as '3' then there will be no changes in the signal and similar appears when the decoded bit output, decoded valid output and decoded output error value is set to '0'.When glb_seed value is set to 000E4048 then there appear a demux in the result. For ccnt, value is set to 1 then the result1041010101010101010101010101010 and finally for buffer output count, total count and sim done value when set to '0' the n there will be no changes in the signal.
Implementation of Viterbi encoder can also be simulated with the help of active HDL simulation environments. Active-HDL is an Aldec product developed using FPGA with HDL simulator.
The below figures can represent the active-HDL simulation results.

A. *Viterbi encoder*

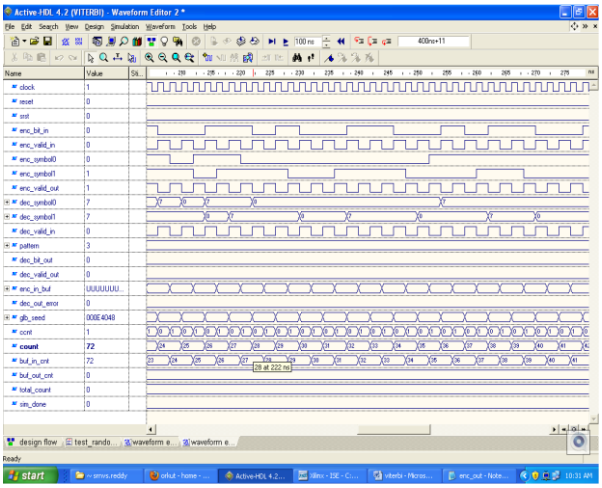

Figure 17: Simulation result of Viterbi Encoder

B. *Viterbi decoder*

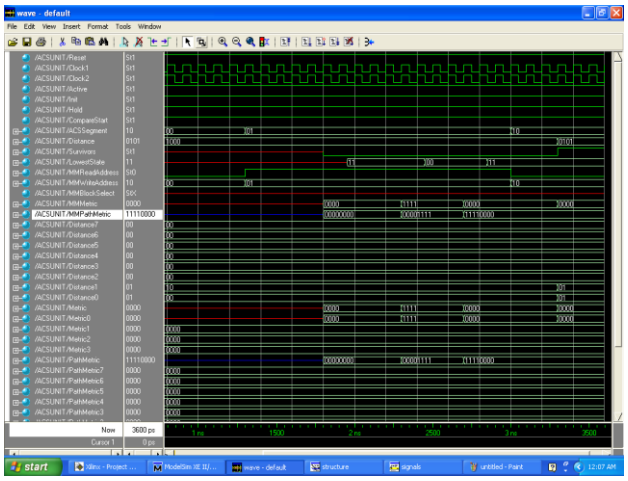

Figure 18: Simulation result of ACS Unit

C. *BGM Unit*

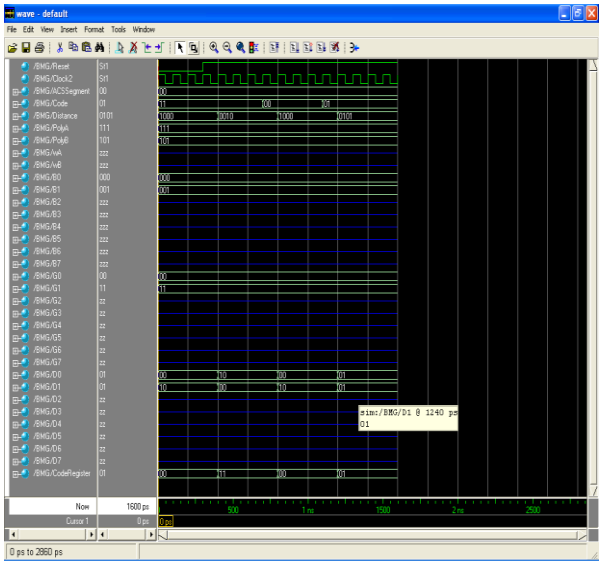

Figure 19: Simulation result of BMG Unit

D. *Control Unit*

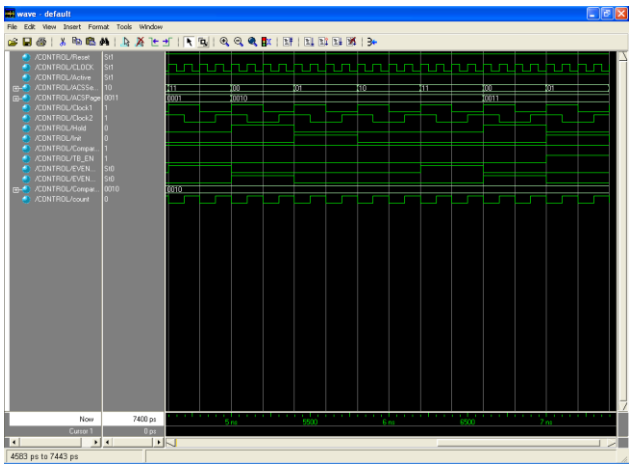

Figure 20: Simulation result of Control Unit

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

E. *Viterbi decoder*

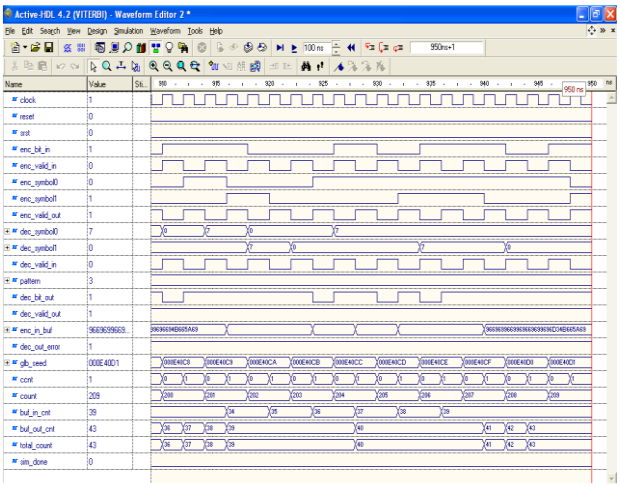

Figure 21: Simulation result of Viterbi Decoder

F. *Design Layout*

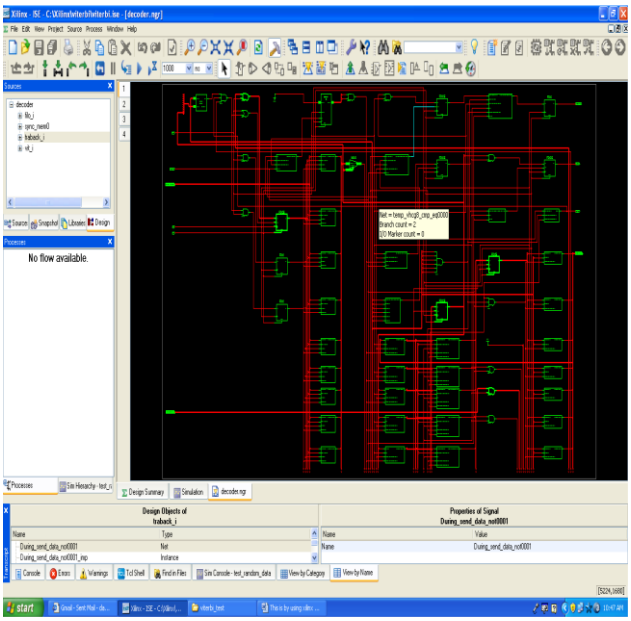

Figure 22: A design layout of the project

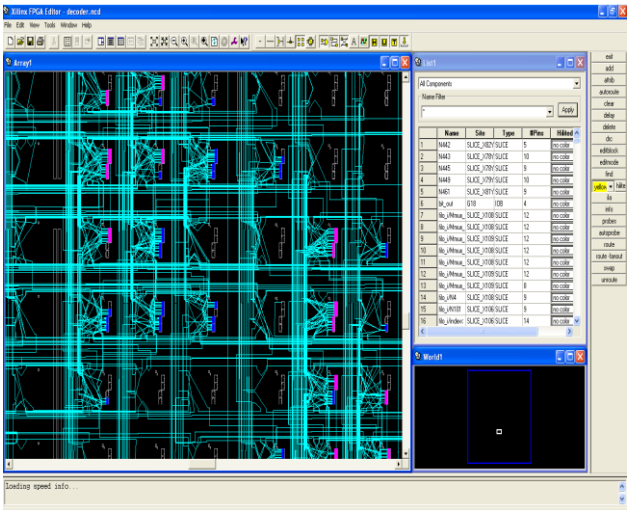

Figure 23: A design layout and implementation of the Viterbi decoder.

V. CONCLUSION

Viterbi algorithm is successfully implemented using Verilog HDL hardware and tools like Xilinx and FPGA. Results that were obtained are to be observed and the entire developed code working process, its design and synthesis results can be obtained very easily using Xilinx ISE and FPGA editors. The Adaptive Viterbi algorithm requires very higher memory locations and logical programming capable performance the operations, so there is a need to less occupied memory Viterbi algorithm architecture has to be developed. The algorithm parameters are needed to be chosen and also the required variations for the noise level predict better results.

REFERENCES

[1] Implementation of the viterbi algorithm using functional programming languages: By Tristan Bull, Department of Electrical Engineering & Computer Science, University of Kansas, Lawrence.[An IEEE paper].
[2] IEEE transactions on magnetics, vol. 44, no. 1, January 2008: "Viterbi Detector for Non-Markov Recording Channels" by Sam Gratrix, Robert Jackson, Tom Parnell, and Oleg Zaboronski..
[3] Y. K. P.Cheung, G. A. Constantinides, J.T. D. Sousa (2003) Field programmable logic and applications: 13th International Conference, FPL 2003, Lisbon, Portugal. New York:Springer.
[4] Y. K. P.Cheung, G. A. Constantinides, J.T. D. Sousa (2003) Field programmable logic and applications: 13th International Conference, FPL 2003, Lisbon, Portugal. New York:Springer.
[5] Shung, C.B. on his paper: "VLSI Architecture for Metric Normalization in the Viterbi Decoders", Proceeding ICC, Vol.4; 1990.

Chitra.M. received B.E, Degree from V.T.U. in 2007, worked as a lecturer in Acharya Patashala College of Engineering for three years. Currently pursuing M.Tech Degree from V.TU.

Roopa. M. received B.E, Degree from U.V.C.E Bangalore University, Bangalore In 1990, M.E, Degree from U.V.C.E, Bangalore University, Bangalore, India in 1994, pursuing Ph.D in the Department of Applied Electronics, Gulbarga University, Gulbarga, INDIA, currently she is working as Assistant Professor in the Department of Electronics and communication, Dayananda sagar college of engineering, Bangalore, INDIA, her research interests include Design of Digital circuits, VLSI Design and Embedded systems.

Dr.A.R.Aswath, received B.E Degree from Mysore University in 1991, M.Tech Degree from M.I.T Manipal in 1996, M.S. Degree from B.I.T.S. Pilani in 2002, Ph.D degree in 2010 from Dr. M.G.R University. Currently he is working as Professor & Head in Telecommunication Department, Dayanand Sagar College of Engineering, Bangalore, India. His main research Interests include Analysis and design of Low Power VLSI Circuits, Embedded System Design and Image Processing.