www.ijsetr.com

International Journal of
Scientific Engineering and Technology Research

ISSN 2319-8885
Vol.07,Issue.02,
February-2018,
Pages:0067-0071

Implementation of Viterbi and Huffman Coding for Area Efficient and High Speed Architecture

KOLLI PRASANNA[1], P. SUDHAKAR[2]
[1]PG Student, Dept of ECE, Vignan Institute of Engineering for Women, Visakhapatnam, AP, India.
[2]Assistant Professor, Dept of ECE, Vignan Institute of Engineering for Women, Visakhapatnam, AP, India.

Abstract: An area efficient and high speed architecture design of Viterbi decoder with encoding rate of 1/2 and constraint length of k = 3 is presented in this paper for the application in satellite communication. Trellis Code modulation system is implemented. The architecture design of high speed Viterbi decoder is implemented and results are obtained using Xilinx tool. Further work is extended using huffman coding. Both the results are been compared. The design architecture using huffman coding takes less area as well as offers high speed which is proved in our project.

Keywords: Viterbi Decoder, Trellis Code Modulation, BMU, Huffman Coding.

I. INTRODUCTION

In media transmission, a convolutional code is a sort of blunder rectifying code that produces equality images by means of the sliding use of a boolean polynomial capacity to an information stream. The sliding application speaks to the 'convolution' of the encoder over the information, which offers ascend to the term 'convolutional coding.' The sliding idea of the convolutional codes offices trellis interpreting utilizing a period invariant trellis. Time invariant trellis interpreting permits convolutional codes to be most extreme probability delicate choice decoded with sensible multifaceted nature. The capacity to perform temperate greatest probability delicate choice interpreting is one of the significant advantages of convolutional codes. This is as opposed to exemplary square codes which are for the most part spoken to by a period variation trellis and in this manner are commonly hard choice decoded. Convolutional codes are regularly portrayed by the base code rate and the profundity (or memory) of the encoder [n,k,K]. The base code rate is normally given as n/k, where n is the information rate and k is the yield image rate. The profundity is regularly called the "limitation length" 'K', where the yield is a component of the past K-1 inputs. The profundity may likewise be given as the quantity of memory components 'v' in the polynomial or the most extreme conceivable number of conditions of the encoder (commonly 2^v). Convolutional codes are regularly depicted as constant. In any case, it might likewise be said that convolutional codes have discretionary piece length, as opposed to that they are persistent, since most certifiable convolutional encoding is performed on squares of information.

Convolutionally encoded piece codes commonly utilize end. The discretionary square length of convolutional codes can likewise be differentiated to great piece codes, which for the most part have settled piece lengths that are dictated by logarithmic properties. The Viterbi deciphering calculation, proposed in 1967 by Viterbi, is an unraveling procedure for convolutional codes in memory-less clamor. The calculation can be connected to a large group of issues experienced in the plan of correspondence frameworks. The Viterbi unraveling calculation gives both a greatest probability and a most extreme a posteriori calculation. A most extreme a posteriori calculation recognizes a code word that expands the contingent likelihood of the decoded code word against the got code word, conversely a greatest probability calculation distinguishes a code word that amplifies the restrictive likelihood of the got code word against the decoded code word. The two calculations give similar outcomes when the source data has a uniform conveyance. Generally, execution and silicon territory are the two most vital worries in VLSI outline. As of late, control dispersal has likewise turned into a critical concern, particularly in battery-fueled applications, for example, phones, pagers and smart phones.

II. LITERATURE REVIEW

General solutions for low-power VD design have been well studied by existing work. Power reduction in VDs could be achieved by reducing the number of states (for example, reduced-state sequence decoding (RSSD) [2], T-algorithm [3] and T -algorithm [4], [5]) or by over-scaling the supply voltage [6]. Over-scaling of the supply voltage usually needs to take into consideration the whole system that includes the VD (whether the system allows such an over-scaling or not), which is not the main focus of our research. RSSD is in general not as efficient as the M-algorithm [2] and T -algorithm is more commonly used than M-algorithm in practical applications, because the T-algorithm requires a sorting process in a feedback loop while M -algorithm only searches for the optimal path metric (PM), that is, the minimum value or the maximum value of all PMs. Algorithm has been shown to be very efficient in reducing the power consumption [7], [8]. However, searching for the optimal PM in the feedback loop still reduces the decoding speed. To overcome this drawback, two variations of the T -algorithm have been proposed: the relaxed adaptive VD [7], which suggests using an estimated optimal PM, instead of finding the real one each cycle and the limited-search parallel state VD based on scarce state transition (SST) [8]. In our preliminary work [9], we have shown that when applied to high-rate convolutional codes, the relaxed adaptive VD suffers a severe degradation of bit-error-rate (BER) performance due to the inherent drifting error between the estimated optimal PM and the accurate one. On the other hand, the SST based scheme requires predecoding and re-encoding processes and is not suitable for TCM decoders. In TCM, the encoded data are always associated with a complex multi-level modulation scheme like 8-ary phase-shift keying (8PSK) or 16/64- ary quadrature amplitude modulation (16/64QAM) through a constellation point mapper. At the receiver, a soft-input VD should be employed to guarantee a good coding gain.

III. VITERBI DECODER

A Viterbi decoder utilizes the VA for interpreting a bitstream that has been encoded utilizing FEC in light of a convolutional code. The Viterbi Decoder is utilized as a part of numerous FEC applications what's more, in frameworks where information are transmitted and subject to mistakes before gathering. The VA is regularly utilized as a part of a wide range of correspondences and information stockpiling applications. It is utilized for deciphering convolutional codes, in base band identification for remote frameworks, and furthermore for recognition of recorded information in attractive plate drives. The necessities for the Viterbi decoder or on the other hand Viterbi indicator, which is a processor that actualizes the VA, rely upon the applications where they are utilized. The piece chart of Viterbi decoder is appeared in Fig. 1. The piece chart comprises of the accompanying modules: Branch Measurements, Add-Compare-Select (ACS), enlist trade, greatest way metric determination, and yield enroll choice [10], [11].

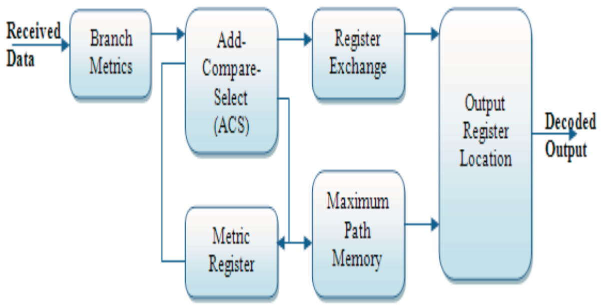


Fig.1. Viterbi decoder block diagram.

IV. IMPLEMENTATION OF VITERBI DECODER

In this section, BMU, ACSU, SPMU and TBU parts of the proposed Viterbi decoder is discussed.

A. Branch Metric Unit

BMU calculates the branch metric. For hard decision ML decoding, the branch metric for a branch is the hamming distance between the received code and the expected code at that instant. BMU comprises of XOR gate and adder. The BMU block is shown in the Fig.2.

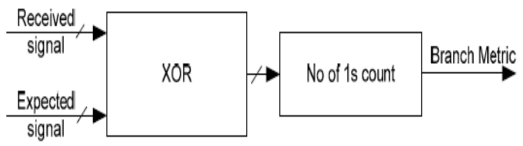


Fig. 2. Branch metric computational unit.

B. Add Compare Select Unit

This structure contains a pair of source and destination states, and four interconnecting branches as shown in Fig 3. In this Fig. 3 the upper or lower branch from each state A(B) is taken, when the corresponding origin input bit is ‘1’ or ‘0’. If the source input bit is ‘1’ or ‘0’, the next state for both A or B is C or D. On the basis of the butterfly notation the architectural block of ACSU can be decomposed as shown in Fig. 3. On the basis of the butterfly notation the architectural block of ACSU [2] can be decomposed as shown in Fig.4. The implementation of two wings of the butterfly module is same except the expected signal. The architecture has shown only one wing of the butterfly module. Branch metric is added with the path metric for state S0 (S2), which is equal to hamming distance between the received signal and the expected signal. The expected input is exclusive-or with received signal. ‘No. of 1’s Count’ Blocks counts the number of 1’s in its input. Adder blocks add the branch metric and state metrics to create a new state metric. Comparator is used to compare the upper path metric and lower path metric for determining the survivor path. Multiplexer select line, connected with the output of the comparator block, selects the survivor path.

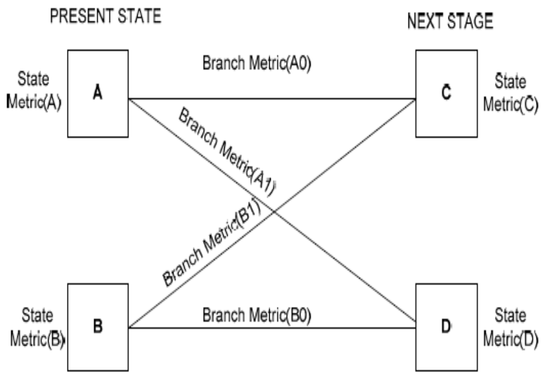


Fig.3. A butterfly structure for ACSU updates corresponding to a node.

C. Survivor Path Memory Unit

This block is needed only for trace back method. Two incoming branches are there for every state except head and tail part of trellis diagram. Between these two branches, it determines which branch will survive (lower or upper). For flagging the survivor path only one bit is enough. ACSU gives one decision bit for every survivor path. All the decision bits are stored in a memory. The size of the memory depends on both the number of ACSU used in proposed architecture and the trellis length. The number of ACSU used in decoder, determines the word length. As in the proposed architecture four ACSU are used, the word length is 4. Trellis length determines the depth of the memory. Here the constraint length of the proposed Viterbi decoder is k = 3. Thus, the trellis length is 16 and the depth of the memory is 32, i.e., twice of the trellis length. Hence, the memory of dimension 32×4 bit is required for the proposed design. In Fig. 5, SPMU is a 32×4 bit duel port memory where all the decision bits are stored. The four ACSU outputs from 4 nodes are fed as input to the SPMU (simple dual port RAM, minimum size has to be 32×4) at each clock cycle. 4-bit Output from the RAM is fed to decode block TBU. This block is shown in Fig. 5.

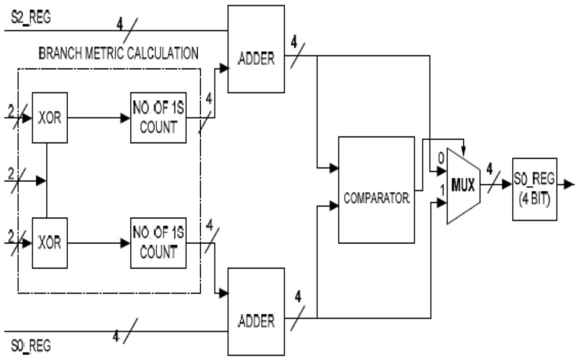


Fig. 4. Architecture of an ACSU.

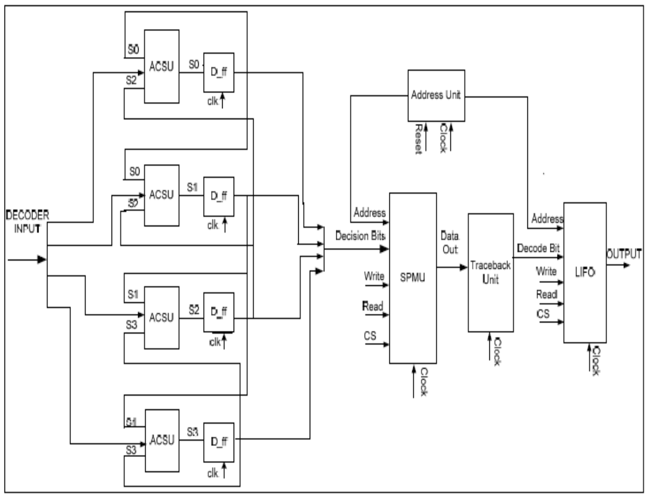


Fig. 5 Architecture of proposed Viterbi decoder.

D. Trace Back Unit

A trellis diagram can be viewed as an extension of the state diagram which explicitly elaborates the passage of time, i.e., it is a combination of time and space representation of state diagram. Fig. 6 shows the trellis corresponding to encoder . In the trellis diagram, different nodes represent the states of the encoder. From an initial state (S0) the trellis maintains the possible transitions to the next states encoded symbols are there for each node, which corresponds to input bit ‘0’ and ‘1’. The maximum number of possible states in the trellis depends on the number of memory used in the encoder. For the present trellis, there are four possible states 00(S0), 10(S1), 01(S2) and 11(S3), and being a radix-2 trellis, each state has two possible output paths attached to input bit ‘0’ and ‘1’. It should be noted that, there is a unique path for every code word. Here the blue color line denotes the survivor path and the red color line denotes the trace back direction as shown in Fig. 6.

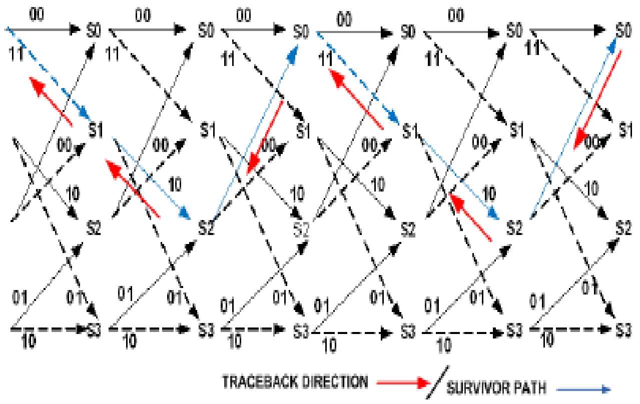


Fig. 6. Trellis diagram of the FSM state diagram shown.

E. Last In First Out

Last In First Out (LIFO) is a 32×1 bit duel port RAM. The LIFO Unit is connected with the output of TBU. The block diagram of LIFO unit is shown in the Fig. 7. In LIFO, the first 16 bits are written in forward direction, addressed by up counter, and then it starts to decode the final output, addressed by down counter. Here ‘Data’ line denotes the decision bits which were decoded by the trace back method. Address line is connected with address unit. Address unit is a 5 bit up and down counter. When the read signal is high it starts to read the signal and when write signal is high, it writes all the bits coming in the ‘Data’ line. Both ‘Write’ and ‘Read’ signals will work if ‘CS’ (chip select) line is high. Here address line is connected to up/down counter which provides the address where the bit is stored.

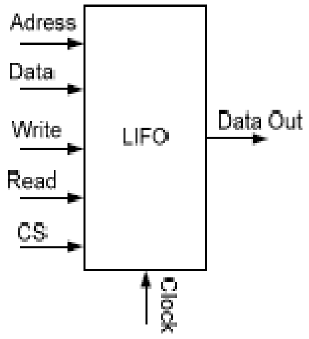
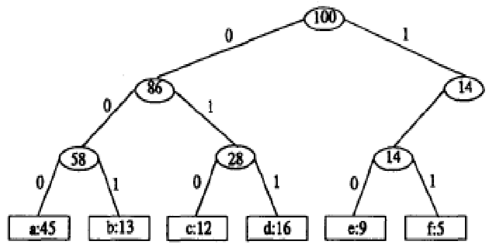


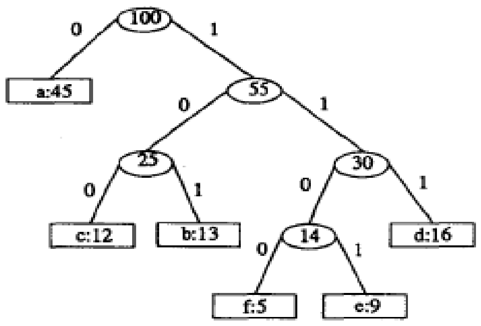
Fig.7. Block Diagram of LIFO Unit.

V. HUFFMAN CODING

Huffman coding is a widely used technique for data compression. It can save from 20% to 90% of the amount of storage or the communication channel bandwidth needed, depending on the characteristics of the input being compressed [13]. No information loss occurs after decoding. Huffman’s greedy algorithm uses a table of frequencies of occurrence of each input symbol to build up an optimal way of representing each symbol by a binary string. Using this, the encoder assigns a variable length binary string to each fixed length input symbol such that the input symbols with higher frequency have shorter lengths. For example, consider coding six symbols [a, b, c, d, e, f]. Assume their frequencies are [45,13,12,16,9,5]. Their Huffman codes are [O, 101,100,111,1101,1100]. Figs. 8(a) and (b) show the trees corresponding to fixed length coding and (variable length) Huffman coding, respectively. In Fig. 8, each rectangle represents a leaf with a symbol and its frequency of occurrence, and each circle represents an internal node with frequency of occurrence of all the symbols in its subtree. The binary label on each edge is used for coding the symbol. Encoding is simple. We just concatenate the codes representing the symbols. For example, three input symbols “abc” are coded as 0 101 . 100 = 0101100 using the tree in Fig. 8(b). Decoding a string obtained by Huffman coding is easy, since no Huffman code is a prefix of any other code. Initial codes can be simply identified and translated back to original symbols by traversing the Huffman tree. For example, string 001011101 can be decoded into “aabe” using the tree in Fig. 8(b). The decoding process needs a convenient representation of the codes so that the initial codes can be easily picked off. A binary tree whose leaves are the given symbols provides such a representation. The binary code for an input symbol can be interpreted as the path from the root to that symbol, where “0” and “1” mean “go to the left child” and “go to the right child,” respectively.



(a)



(b)

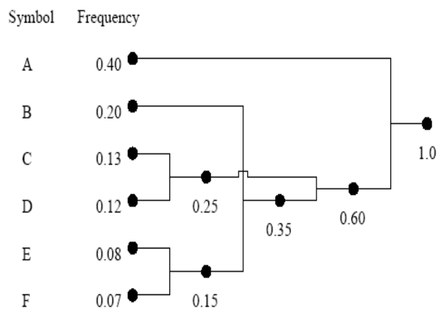
Fig. 8. (a) A tree with fixed length codes. (b) An optimal Huffman code tree can always be represented by a full binaly tree, in which every nonleaf node (internal node) has exactly two children as shown in Fig. 8(b) (the tree in Fig. 8(a) is not a full binary tree). Note that a full binary tree corresponding to an optimal code tree for n symbols has n leaves and n - 1 internal nodes. The length of the encoded input can be represented as B(T) = C, ,f(c)d,(c) where f(c) denotes the frequency of input symbol c in the set of symbols C and d,(c) is the length of the code corresponding to c.

Huffman coding uses a specific method for choosing the representation for each symbol, resulting in a prefix code (sometimes called "prefix-free codes", that is, the bit string representing some particular symbol is never a prefix of the bit string representing any other symbol). Huffman coding is such a widespread method for creating prefix codes that the term "Huffman code" is widely used as a synonym for "prefix code" even when such a code is not produced by Huffman's algorithm.

Example 1: This is an important and widely-used example of a statistical compression algorithm. As with all such algorithms the basic idea is to identify those data items (symbols) that appear most frequently and give them the shortest codes. This is achieved by first creating a binary tree as follows...

- Rank the symbols by their frequency (which should sum to 1) and create a node for each symbol.
- Create a new node by joining the two lowest ranked nodes and summing their rankings together.
- Continue until all nodes on the tree are joined to create a single node of rank 1.

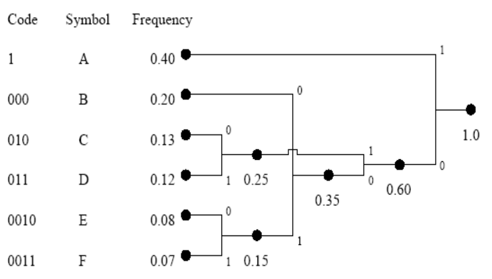
Binary Tree Example:



Having obtained the binary tree it is now possible to calculate the Huffman code for each symbol...

- Starting at the root assign 0 to the branch with the higher value and 1 to the branch with the lower value.
- Once the tree is traversed the code for each symbol may be obtained by following the path from the root to that symbol.

This ensures that that the highest frequency symbols have the shortest codes and that a continuous stream of encoded bits may be uniquely decoded. To illustrate how codes are assigned:



Decoding: Decoding a Huffman code is very easy. For example decode
0110010001110100010011
when
A = 1
B = 000
C = 010
D = 011
E = 0010
F = 0011

Efficiency
It is possible to calculate the average number of bits per character by multiplying the length of each character code by its frequency and then taking the sum.

VI. RESULTS

Results of this paper is as shown in bellow Figs.9 to 12.

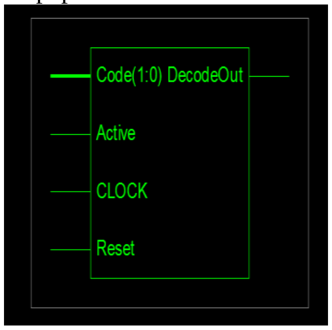


Fig.9. Schematic diagram of VD

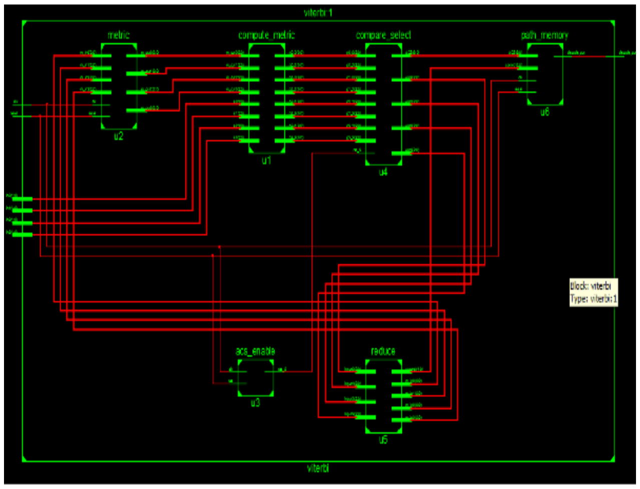


Fig.10. RTL Schematic of VD.

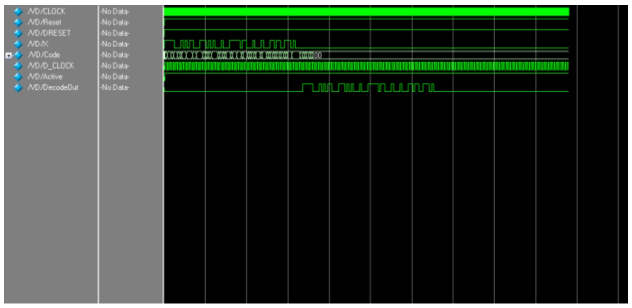


Fig.11. Output waveform of VD.

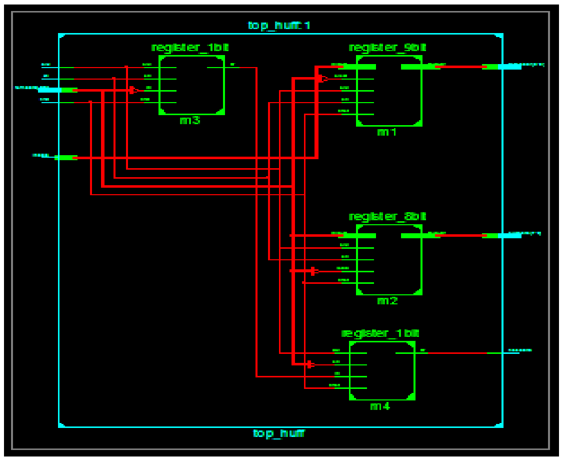
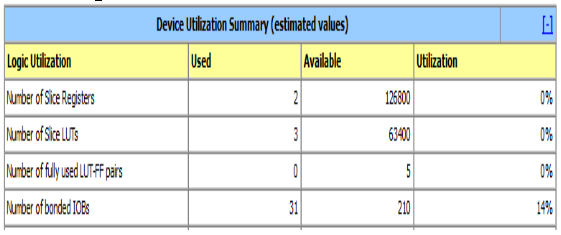


Fig.12. RTL diagram of Huffman.

Synthesis Report:



Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slice Registers	2	126800	0%
Number of Slice LUTs	3	63400	0%
Number of fully used LUT-FF pairs	0	5	0%
Number of bonded IOBs	31	210	14%

TABLE I: Comparison of Area, Time And Frequency

	Viterbi Decoder	Huffman Coding
Area	93 μm^2	37 μm^2
Time delay	4.24ns	0.648 ns
Frequency	235.78 MHz	1542.492

VII. CONCLUSION

A high speed and area efficient Viterbi decoder architecture, mainly for the application in wireless communication, is proposed in this paper. The simulation result is checked in Virtex family using Xilinx 14.7. This calculation is reasonable for TCM frameworks which dependably utilize high rate convolution code. Huffman coding is been implemented and the obtained results shows that the area using huffman coding there is vast reduced in area and time delay compared to viterbi decoder . From the discussion, it can be concluded that our structure takes less area and also offers high speed comparative to the other architectures and is suitable for satellite related application, as payload and speed are very important factors in wireless communication. Further work can be developed by implementation of hard decision adaptive Viterbi decoder for constraint length k=7 at rate 1/2 is can be performed .Same work can be extended for soft decision based adaptive Viterbi decoder which suits multiple quantization level by modifying Branch metric computation unit accordingly.

VIII. REFERENCES

[1] Chien-Ching Lin, Yen-Hsu Shih, Hsie-Chia Chang, and Chen-Yi Lee, 2005. Design of a Power-Reduction Viterbi Decoder for WLAN Applications, IEEE Transactions on Circuits and System-I: regular papers, 52(6), 321-328G.

[2] Irfan Habib, Ozgun Paker, and Sergei Sawitzki,2009, Design Space Exploration of Hard- Decision Viterbi Decoding:
[3] ann S. Yuanand Weidong Kuang, 2004, Teaching Asynchronous Design in Digital Integrated Circuits, IEEE transactions on education,47(3),397-404.
[4] Injin He, Zhongfeng Wang, Zhiqiang Cui, and Li Li, 2009, Towards an Optimal Trade-off of Viterbi Decoder Design, IEEE conferecne,3030- 3033.
[5] Joshi M.V., Gosavi S., Jegadeesan V., Basu A., Jaiswal S.,Al-Assadi W.K.and Smith S.C.2007,NCL Implementation of Dual-Rail 2s Complement 8×8 Booth2 Multiplier using Static and Semi-Static Primitives, IEEE region 5 Technical Conference, April 20-21, Fayetteville,59- 64.
[6] Jun Jin Kong, Keshhab K Parhi., 2004 Low- Latency Architectures for High-Throughput Rate Viterbi Decoder, IEEE Transactions on VLSI System, 12(6), 642-651.
[7]Meilana Siswanto1, Masuri Othman, Edmond Zahedi, 2006 VLSI Implementation of 1/2 Viterbi Decoder for IEEE P802.15-3a UWB Communication, IEEE ICSE2006 Proc., Kuala Lumpur, Malaysia,666 – 670.
[8] Qing Li, Xuan-zhong Li, Han-hong Jiang and Wen-hao He 2008, A High-Speed Viterbi Decoder, Fourth International Conference on Natural Computation IEEE., p.p. 313-316.
[9] Yao Gang, Ahmet T., Erdogan, and TughrulArslan, 2006, An Efficient Pre- Traceback Architecture for the Viterbi Decoder Targeting Wireless Communication Applications, IEEE Transactions on Circuits and Systems-I: regular papers, 53(9),423-432
[10] W. Chen, "RTL Implementation of Viterbi Decoder," Linköping University, Department of Electrical Engineering, June 2006.
[11] S. Hema, V. S. Babu and P. Ramesh, "FPGA Implementation of Viterbi Decoder," Proceedings of the 6th WSEAS Int. Conf. on Electronics, Hardware, Wireless and Optical Communications, Corfu Island, Greece, February 16-19, 2007.
[12] T. H. Corman, C. E. hiserson, and R. L. Rivest, Introduction to Algorithms. McGraw Hill, 1990.

Author's Profile:

Kolli Prasanna pursuing her M.Tech in the department of Electronics & Communication Engineering , Vignan Institute of Engineering for Women, Vizag, AP, India. He obtained her B.Tech(ECE) from Khader Memorial College of Engineering & Technology, Vishakhapatanam.

P. Sudhakar, M.Tech, working as Assistant Professor in the department of Electronics and Communication Engineering , Vignan Institute of Engineering for Women, Vizag, AP, India. His area of interest is signal processing.