FPGA Implementation Of Power Efficient Low Latency Viterbi Decoder

Nirmal Bhatt
Pursuing ME(WCSN)
Department of ECE,
GEC, Surat, India

Prof. Milind Shah
Asso. Professor
Department of ECE,
GEC, Surat, India

Prof. Bhavesh Asodariya
Asst. Professor
Department of ECE,
SSASIT, Surat, India

Abstract

Convolution encoding with Viterbi decoding is a powerful method for forward error correction. It has been widely deployed in many wireless communication systems to improve the limited capacity of the communication channels. The Viterbi algorithm, which is the most extensively employed decoding algorithm for convolution codes. In this paper, we present a field-programmable gate array implementation of power efficient low latency Viterbi Decoder with a constraint length of 7 and a code rate of 1/2.We also proposed One Pointer algorithm for Trace back Implementation of survivor sequence memory management for low power decoder design.

Keywords - *ACS, Butterfly structure, Convolutional code, One Pointer, Register Exchange, Trace Back, Viterbi Algorithm.*

1. Introduction

In recent years the continued rise of portable data-devices such as cell phones, PDAs and laptops has driven enormous growth in the area of wireless communications. Whenever data is sent over a wireless channel, it is subjected to degradation due to multipath fading and noise [3]. Depending on the amount of degradation, the effect can be a loss or corruption of the original data during transfer. In order to alleviate this problem and to ensure the reliable transfer of data, the typical solution has been the use of an error correction coding scheme.

This is also the building blocks for other wireless devices working on GSM, Wi-Fi, DAB, DVB standards. Design of CC encoder is simple but much more hardware will be utilized in the designing of decoder. As the constraint length of the code increases its error correcting capability will also be improved but at the cost of power and memory requirement[1]. So, our aim is to design optimum CC decoder for power efficient and delay sensitive application.

In this paper we are going to discuss the convolutional encoder of coding rate 1/2, constraint length 7 and generator field G = [1111001; 1011011] and for decoding the CC code we proposed viterbi decoder for the same parameters.

Verilog HDL is used for coding the design. Xilinx ISE 13.4 foundation series is used for simulation. Spartan 3E XC3S500E FPGA has been used for implementing the design.

2. Convolutional Encoder

Convolutional codes differ significantly from block codes in structural form and have much more powerful error correcting capability[3].Convolutional coding can be applied to a continuous input stream (which cannot be done with block codes), as well as blocks of data. Convolutional codes are usually characterized by two parameters and the patterns of n modulo-2 adders (XOR gates). The two parameters are the code rate and constraint length. The code rate, k/n, is expressed as a ratio of the number of bits into the convolutional encoder (k) to the number of channel symbols output by the convolutional encoder (n) in a given encoder cycle. The constraint length K denotes the "length", i.e. how many bits are available to feed n modulo-2 adders.

In fact, a convolutional encoder can be viewed as a finite state machine having $2K$-1 states. Its output is function of presents state and present input. The trellis is a time indexed version of the state diagram. Each node corresponds to a state at a given time index, and each branch corresponds to a state transition. Associated with each branch is the input symbol and output symbols corresponding to the state transition. Given a known starting state, every input sequence corresponds to a unique path through the trellis. The

convolutional encoder, shown in Figure 1, coding rate is 1/2, constraint length is 7 and generator field G = [1111001; 1011011].

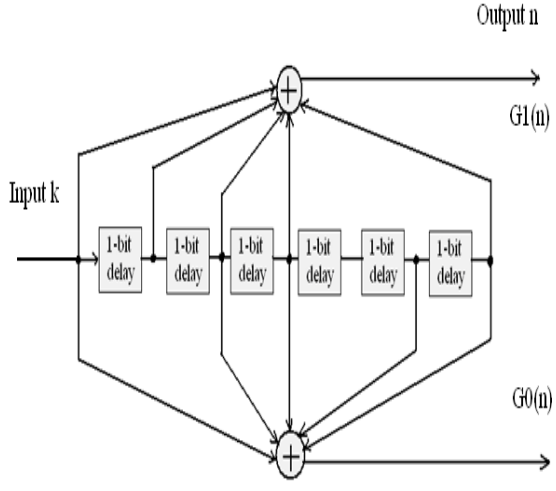

Fig. 1 Convolutional Encoder (K=7, rate-1/2)[5]

3. Viterbi Decoder

A Viterbi decoder uses the Viterbi algorithm for decoding a bitstream that has been encoded using Forward error correction based on a code. There are other algorithms for decoding a convolutionally encoded stream(for example, the Fano algorithm).The Viterbi algorithm is the most resource consuming, but it does the maximum likelihood decoding[6]. Fig.2 shows the block diagram of viterbi decoder. It consists of the following modules:

Branch Metric Unit, ACS, State Metric Storage, Survivor Path Metric, Traceback and Output Decode Block.

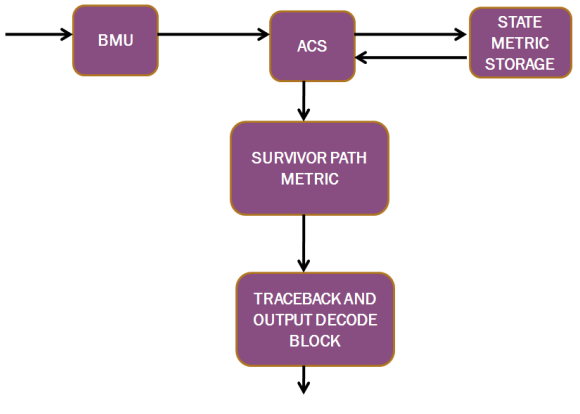

Fig. 2 Block diagram of Viterbi decoder[1]

3.1 Branch Metric Unit

At the decoder side, we receive analog signal. Analog signal is a mixture of transmitted signal and noise signal. Now received signal is given to the 3-bit quantizer. Depending on the signal strength, it assigns the value shown in Table I.

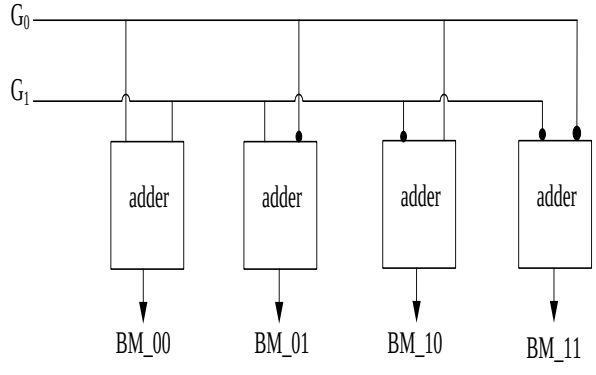

Fig. 3 Branch Metric Calculation[1]

Now BM calculates four values as shown in Fig.3 based on received signal.

BM_00 = $G_1 + G_0$
BM_01 = G_1 + (2's complement of G_0)
BM_10 = (2's complement of G_1) + G_0
BM_11 = (2's complement of G_1) + (2's complement of G_0)

Suppose encoded bit 10 (G_1, G_0) is transmitted but due to effect of noise we receive 1 as a weak 1 and 0 as a strong 0. Value for weak 1 is assigned as -1(100) and strong 0 is assigned as +3(000) shown in Table I.

So as per the above equation BM_00 = -1 + 3 = 2, BM_01 = -1 + (-3) = -4, BM_10 = 1 + 3 = 4, BM_11 = 1 + (-3) = -2.

Table 1 3Bit Soft Quantization Value

Range	Value	Meaning
+1	010	relatively weak 0
+2	001	relatively strong 0
+3	000	strongest 0
0	011	weakest 0
-1	100	weakest 1
-2	101	relatively weak 1
-3	110	relatively strong 1
-4	111	strongest 1

3.2 Add Compare Select Unit (ACS)

ACS is also known as a Butterfly module as shown in Fig.4. The trellis diagram for a rate 1/n convolutional encoder consists of butterfly structures. This structure contains a pair of origin and destination states, and four interconnecting branches.

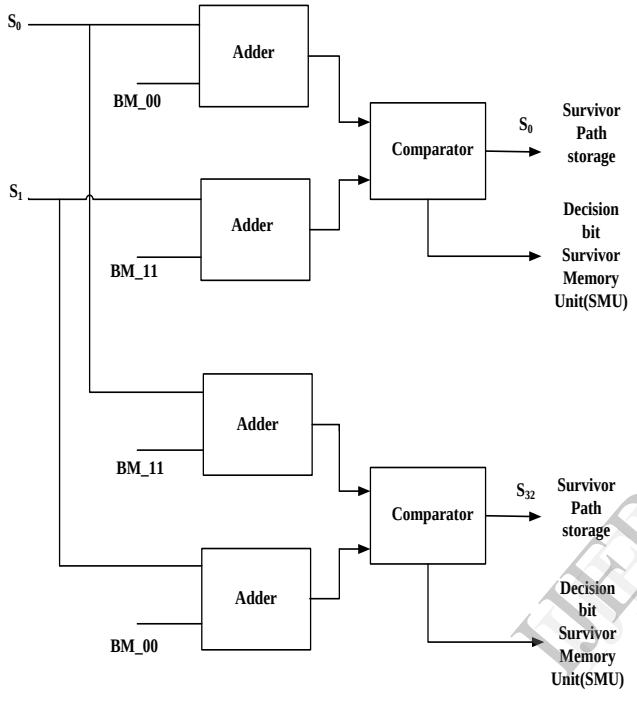

Fig. 4 ACS Module

As shown in Fig. 4, S0 and S1 are present states and S0 & S32 are next states. With 64 states convolution encoder, we have 32 butterfly units. Each butterfly unit generates two survivor decision bits. ACS unit adds BM (Branch Metric) to partial path metric to form lower and upper state metric. It than compares state metric of two incoming states and selects one with maximum state metric value. The branch metric to be added with the path metric for state S0 (S1) is the Hamming distance between the expected code symbol "00" ("11") and the received input. The received input is XORed with the expected code symbol. The branch metric is added to the partial path metric to calculate the new path metric. Two path metrics, the upper one and the lower one, are compared to select the survivor path, and the resultant metric of the selected path updates the path metric of state S0 (S1). The lowerwing is identical to the upper wing except that the expected values differ.

If new state metric is updated by upper state metric then 0 is stored as a decision bit while it is updated by a lower state metric then 1 is stored as a decision bit. Suppose S0 is greater than S1, S0 is selected as a new state metric and decision bit 0 is stored. Otherwise S1 is selected as a new state metric and decision bit 1 is stored.

3.3 Path Metric Calculation and Storage

The Path Metrics are updated in two steps. First for each of the two incoming paths, the corresponding Branch Metric is added to the Path Metric of the original state. The two sums are compared and the larger one is stored as the new Path Metric and the corresponding path is stored as the best path.

3.4 Register Exchange Method for SMU

The register exchange (RE) method is the simplest conceptually and a commonly used technique. Because of the large power consumption and large area required in VLSI implementations of the RE method[9], the trace back method (TB) method is the preferred method in the design of large constraint length, high performance Viterbi decoders[8]. In the register exchange, a register assigned to each state contains information bits for the survivor path from the initial state to the current state[2]. In fact, the register keeps the partially decoded output sequence along the path, as illustrated in Figure 5.The register of state E1 at t=3 contains '101'. This is the decoded output sequence along the hold path from the initial state.[2]

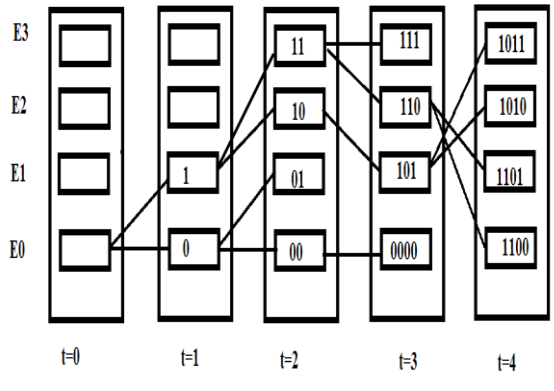

Fig. 5 Register Exchange Method for SMU[2]

The register-exchange method eliminates the need to trace back since the register of the final state contains the decoded output sequence.

However, this method results in complex hardware due to the need to copy the contents of all the registers in a stage to the next stage. The survivor path

information is applied to the least significant bit of each register, and all the registers perform a shift left operation at each stage to make room for the next bits. Hence, each register fills in the survivor path information from the least significant bit toward the most significant bit. The scheme is called shift update. The shift update method is simple in implementation but causes high switching activity due to the shift operation and, hence, results in high power dissipation.[2]

3.5 Traceback Unit

Traceback begins after completing the metric update of the last symbol in the frame. For frame by frame Viterbi decoding, all that is needed by the traceback algorithm are the state transitions. The starting state is state 00. For sliding window decoding, the starting state is the state with the largest state metric.

Two basic operations are performed in the traceback SMU (Survivor Memory Unit)[7].These operations are explained below:

Writing New Data (WR): Survivor vector is generated by the ACS unit it needs to be written into the memory. Each individual survivor is written into a memory position that corresponds to its state. The write pointer moves forward (from the left to the right in Figure 6) through the memory as the ACS unit moves on to each new stage in the code trellis.

Decode Read (DR): Once *D* (or more) survivor vectors have been written to memory, the SMU needs to start tracing a path back to the origin of the trellis in order to decode the output bits. Functionally decode read operation is identical to the traceback read operation, except that the pointers read from memory are decoded to form the output of the SMU. Here, we omit the traceback read operation because we know the starting point (state 0) of the traceback operation. We decode data directly. Decode read starting point is the end of the frame (state 0). This state has a decision bit either 0 or 1 in SMU. This decision bit is now extracted and is stored into FILO buffer. Depending on the decision bit we can find previous state and its decision bit. Process is continuing till the starting point of the frame. Decode read operation frees up memory space for use by the write operation. Since the memory size in hardware is limited, the write operation must fill the space freed up by decode read operation.

Several methods have been devised for trace back[7]. One well-known method, where the termination state is unknown, is to start the traceback from the state that has the best path metric. This method has fastest convergence rate, requires shortest memory traceback and has lowest potential latency. However finding best state requires 2*K*-1 path metric comparisons which would increase area significantly. The other methods, *k*-pointer even and 1-pointer algorithm eliminate need of comparator since traceback can be started from any arbitrary state. However, at least *D* step traceback of Survivor memory is needed before producing decoded bits where *D* is called traceback depth (or traceback length). The k-pointer even algorithm [4] requires 2*k* memory banks, each of size *D* / (*k*-1) columns, and uses *k* read pointers and 1 write pointer to memory.

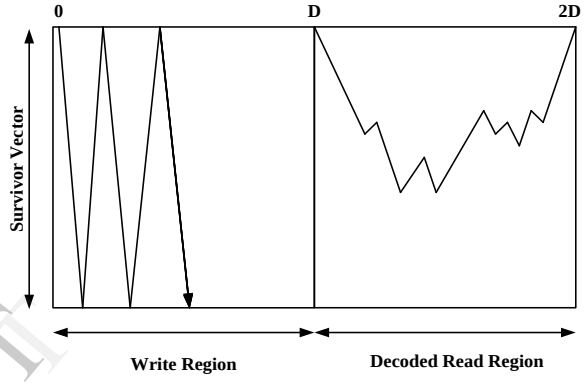

Fig. 6 Trace back operation in survivor memory

The One-Pointer Algorithm[4] differs significantly from the k-Pointer even algorithm. Instead of utilizing k read pointers to perform the required k reads for every column write operation, we chose to use a single read pointer, but accelerate read operations, so that every time the write counter advances by one column, k column reads occur. This acceleration of read operations is made possible by the fact that among the three operations, writing new data, trace back read and decode read, writing new data is by far the most time consuming. This observation is particularly important as 2K-1 bit are written every trellis-stage-time, as opposed to only k bits being read.

3.5.1 One Pointer Algorithm

In 1-pointer algorithm 2 memory banks, each D column long, are required, for a total of 2D columns. As shown in Figure7. Write data and decode read operation are started at time 0. Decode pointer has double speed compare to write pointer. At the time *D* / 2, write pointer wrote *D* / 2 data into memory bank 1 while decode read pointer read out *D* data in memory bank 2. Memory bank 2 is now free to write data. Between D/2 to D time decode read pointer is going into sleep mode while write pointer writes total D data into memory bank1. Now at the time *D*, write pointer wrote *D* data into memory bank 1 while decode read

pointer stayed in sleep mode. Now memory bank 1 is ready to decode data. So write pointer is starting to write data into memory bank 2 and decode read pointer is starting to read the data in memory bank 1 and above process is repeating. Total latency is 3*D* / 2 time. After this time output is stored in 64-bit FILO register and continuously coming out.

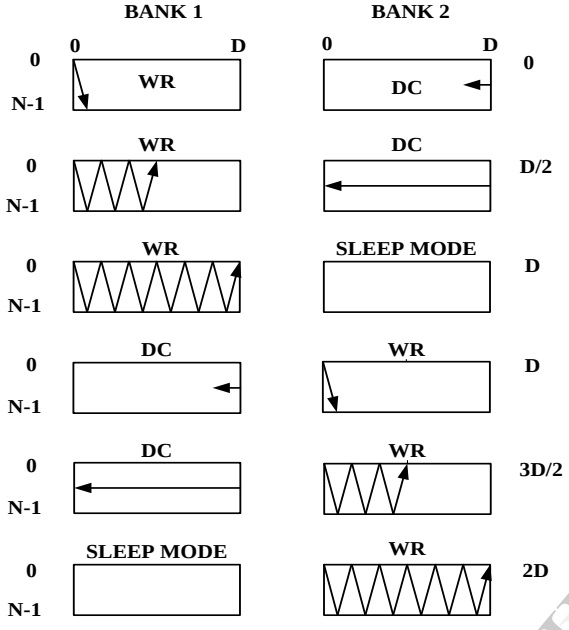

Fig .7 One Pointer Algorithm

4. FPGA Implementation Results

Verilog is used for coding the designs and FPGA flow is taken for implementation. The design is targeted to Spartan 3E (90nm process technology with 1.5V power supply and device package fg320) FPGA using Xilinx ISE 13.4 foundation series. Chipscope Pro is used to see the results from FPGA kit. Xpower Estimator 11.1 is used to estimate the power of the design.

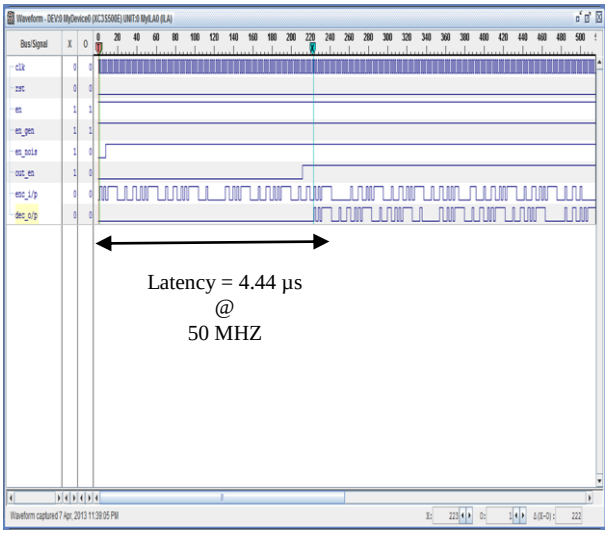

Fig. 8 Encoder i/p and Viterbi Decoder o/p waveform

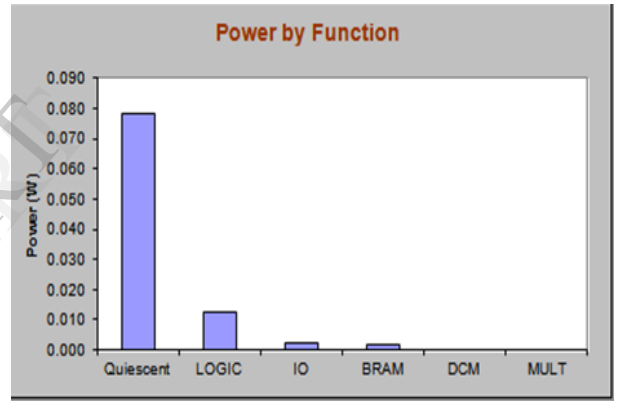

Fig. 9 Power by Function

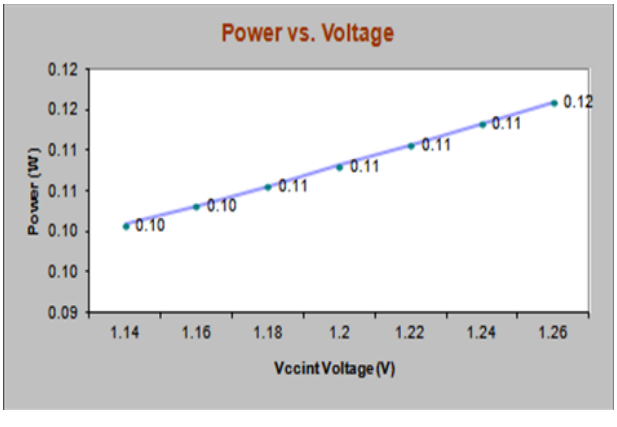

Fig. 10 Power by Voltage

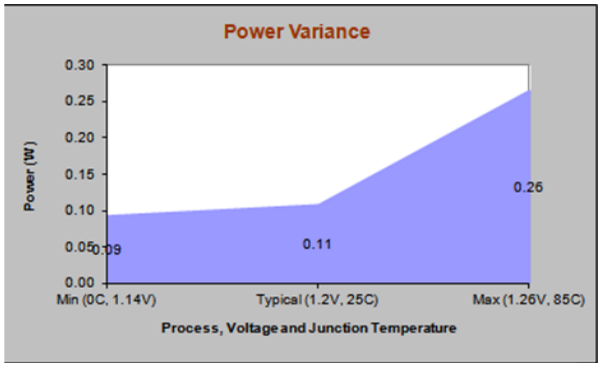

Fig.11 Power by Variance

Table 2 Power Summary

Power Summary of Viterbi Decoder@50Mhz	
Optimization	None
Data	Production
Quiescent(w)	0.078
Dynamic (w)	0.030
Total (w)	0.108

Table 3 FPGA Performance Comparison of Viterbi Decoder

Parameter	Implemented Design		Paper Ref.(TB2F) Norm1 [11]	Paper Ref.(TB2F) Norm2[11]
FPGA Device	Spartan 3E	Virtex-II Pro	Virtex-II Pro	Virtex-II Pro
Decision	Soft	Soft	Soft	Soft
Area (Slices)	3731	3478	2136	2156
BRAM's	2	2	2	2
Max. Freq.	97 MHz	159 MHz	158 MHz	196 MHz
Max. Data rate	48.5 Mbps	79.5 Mbps	79 Mbps	98 Mbps
Min. Latency	2.29µs @97MhZ and 4.44µs @50Mhz	2.22µs @ 100Mhz	2.468 µs @ 158MHz	-
Total Power	108mw @50Mhz and 129mw @97Mhz	272mW @ 100MHz	1068mW @ 158MHz	1081mW @ 196MHz

5. Conclusion

The main consideration of design is to decrease the power dissipation and latency. The two techniques used for decoding the convolutional code are the register exchange method and trace back method. The Trace back method is used for larger constraint length but take more time than register exchange but the power consumption in the method is very less so that we design the Viterbi decoder with the trace back method. To decrease the power we used architectural level difference in ACS module. Conventional module consumes more power but to design ACS in Butterfly structure consume less power. One Pointer algorithm of Traceback is not only useful in saving the power but by increasing the speed of the read pointer twice than write pointer we can also reduce the latency.

6. Acknowledgment

We take this opportunity to express our deepest gratitude and appreciation to all those who have helped us directly or indirectly towards the successful completion of this paper.

7. References

[1] Wei Chen, "RTL Implementation of Viterbi Decoder," *Master's thesis, Linkoping Uni. Rep, pp.6-25*, 2006.

[2] Samirkumar Ranpura and Dong Sam Ha, "Low-Power Viterbi Decoder Design for Wireless Communications Applications," Int. ASIC conference, Sept. 1999, Washington, D.C.

[3] B. Sklar, *Digital Communications: Fundamentals And Applications*, Prentice-Hall, 2nd Edition, 2002.

[4] G.Feygin and P.G.Gulak, "Survivor Sequence memory management in viterbi decoders," *CSRI Tech.Rep.262, Univ.of Toronto*, Jan1991.

[5] Z. M. Patel, "VLSI implementation of IEEE802.11a Physical layer baseband", M.Tech Dissertation, IITB, *Powai*. 2009.

[6] Viterbi A. J. "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory13* (2): 260–269, April, 1967.

[7] M. Horwitz and R. Braun, "A Generalized Design Technique for Traceback Survivor Memory Management in Viterbi Decoders", *Proc. Symposium on Communication and Signal Processing Grahamstown,* South Africa, pp. 63–68, 1997.

[8] R C S Morling and N Haron, "Novel Low-Latency Low-Power Viterbi Decoder Traceback Memory Architecture," *IEEE MELECON,* Dubrovnik, Croatia, May, 2004.

[9] Sang-Ho Seo, and Sin-Chong Park, "Low latency and power efficient VD using Register Exchanged state- mapping Algorithm," Proceedings of the 9th International Database Engineering & Application Symposium (IDEAS'05).

[10] S.W.Shaker, S.H.Elramly, K.A.Shehata, "FPGA Implementation of a Viterbi Decoder for WiMAX Receiver," International Conference on Microelectronics(ICM*)*, Marrakech, 2010.

[11] F. Angarita, M. J. Canet, "Architectures for the Implementation of a OFDM-WLAN Viterbi Decode," *Journal of Signal Processing Systems* 52, 35-44, 2008. Springer Science.