International Journal of Computer Communication and Information System (IJCCIS)
– Vol2. No1. ISSN: 0976–1349 July – Dec 2010

Hardware Implementation of Viterbi Decoder for Wireless Applications

Bhupendra Singh[1], Sanjeev Agarwal[2] and Tarun Varma[3]

Deptt. of Electronics and Communication Engineering,
[1]Amity School of Engineering and Technology, Noida, India
Email: [1]bsingh.tech@gmail.com
[2,3] Malaviya National Institute of Technology, Jaipur, India
Email: [2]san@mnit.ac.in, [3]tarun.varma.jaipur@gmail.com

***Abstract*—In 2G mobile terminals, the VD consumes approximately one third of the power consumption of a baseband mobile transceiver. Thus, in 3G mobile systems, it is essential to reduce the power consumption of the VD. In this report the register exchange (RE) method, adopting a pointer concept, is used to implement the survivor memory unit (SMU) of the VD. For the implementation part, hardware implementation of MLVD through Synopsys Design Compiler Synthesis is done. For synthesis UMC-180nm Library is used.**

***Index Terms*— Viterbi Decoder, SMU, ACSU, RE, MLVD**

I. INTRODUCTION

The register exchange (RE) method, adopting a pointer concept, is used to implement the survivor memory unit (SMU) of the VD. The method entails assigning a pointer to each register or memory location. The contents of the pointer, which points to one register, is altered to point to a second register, instead of copying the contents of the first register to the second. When the pointer concept is applied to the RE's SMU implementation[2], there is no need to copy the contents of the SMU and rewrite them, but one row of memory is still needed for each state of the VD. Thus, the VDs in CDMA systems require only 256 rows of memory, hence reducing the VD's power consumption. Also, if the initial state of the convolutional encoder is known, the entire SMU is reduced to only one row. Because the decoded data is generated in the required order, even this row of memory is dispensable. The zero-memory architecture is called the MemoryLess Viterbi Decoder (MLVD)[6], and reduce power consumption.

Another problem of the VD, which is addressed in this report, is the Add Compare Select Unit (ACSU) which is composed of 128 butterfly ACS modules.

The ACSU's high parallelism has been previously solved by using a bit serial implementation. The 8-bit First Input First Output (FIFO) register, needed for the storage of each path metric (PM), is at the heart of the single bit serial ACS butterfly module. A new, simply controlled shift register is designed at the circuit level and integrated into the ACS module.

A. *Structure of Viterbi Decoder*

The four functional blocks of VD in term of implementation, including branch metric unit (BMU), add-compare-select unit (ACSU), feedback unit (FBU) and survivor memory unit (SMU).

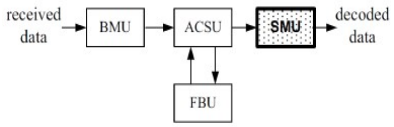


Fig.1 Functional Block of VD

The BMU calculates branch metric of each branch according to maximum likelihood of the received data. The ACSU makes the sum of branch and path metrics, then compares and selects the survivor path metric and the decision bit. The FBU stores the survivor path metric for ACSU to be used in the next cycle. The SMU produces the decoded data based on the decision bit and the survivor path metric. The SMU marked by boldfaced letters in Fig. 1 significantly influences latency, power and chip area in a VD.

B. *Viterbi Decoding Algorithms*

In 1967, Viterbi developed the Viterbi Algorithm (VA) as a method to decode convolutional codes [1]. The VA uses the trellis diagram to decode an input sequence, as demonstrated in Figure 2. The VA[4], which uses a hard decision format, is exhibited in Fig.2. A node is assigned to each state for each time stage.The transition between two states is represented by a branch, which is assigned a weight, referred to as a branch metric (BM). The BM is a measure of the likelihood of the transition, given the noisy observations. The BMs that are accumulated along a path form a path metric (PM). For the two branches entering the same state, the branch with the smaller PM survives, and the other one is discarded. Then two methods can be used to extract the decoded bits: the trace back (TB) or the register exchange (RE)[3].

C. *Trace Back (TB) Method*

At the last stage of the trellis diagram Fig 2, the TB method extracts the decoded bits, beginning from the state with the minimum PM, S0. From this state and tracing backwards in time by following the survivor path, which originally contributed to the current PM, a unique path is identified.

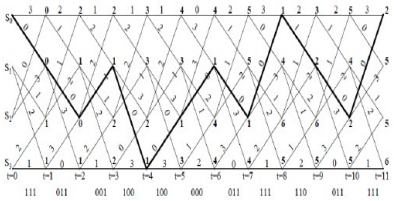


Fig.2 Trace Back (TB) Viterbi Decoding

D. *Register Exchange (RE) Method*

In the RE approach, a register is assigned to each state. The register records the decoded output sequence along the path from the initial state to the final state. This is depicted in Fig 3. At the last stage, the decoded output sequence is the one that is stored in the survivor path register, the register assigned to the state with the minimum PM.

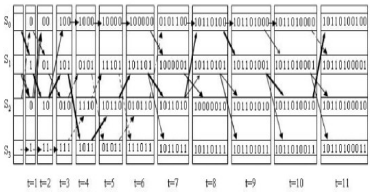


Fig.3 Register Exchange (RE) Method

II. IMPLEMENTATION

A. *Viterbi Decoder Implementation*

The Viterbi decoder is introduced by the flow chart in Fig.5. With more specification, we will introduce it with the micro architecture of the hardware.

Here, we will introduce the Next state ROM, BMU block, ACS block, trace-back block and decode-data block one by one as shown in the Fig. 6.

The Finite State Machine (FSM) of our Viterbi decoder is composed by 5 states and 11 possible conditions shown in fig.4.

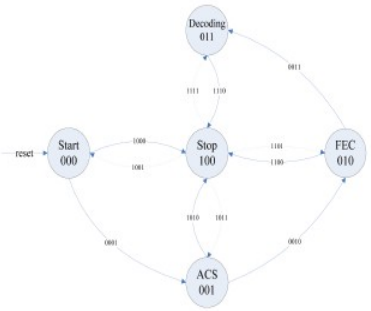


Fig.4 The Finite State Machine of Viterbi decoder

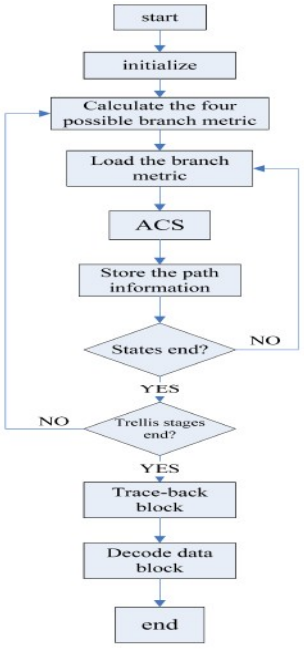


Fig.5 Flow Chart of the Viterbi decoder

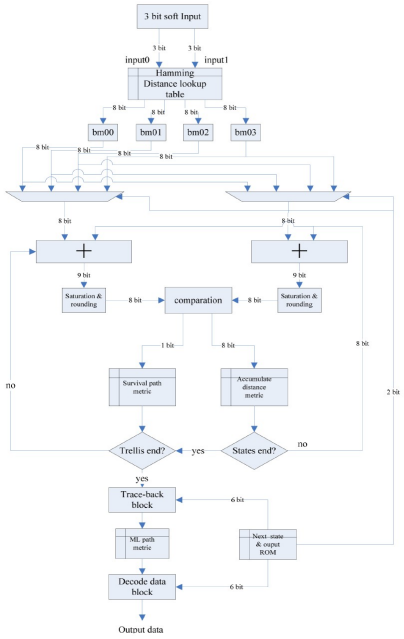


Fig.6 Function allocations in micro architecture of VD

When the reset signal is set high, Viterbi decoder needs to initialize the Next state Rom. The architecture of the Next state ROM is shown in the fig.7.

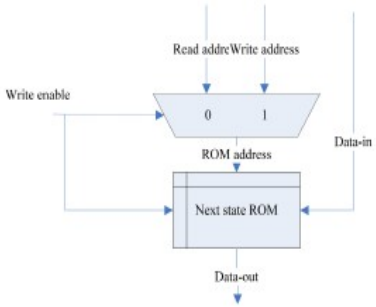


Fig.7 The Next State ROM hardware implementation

Fig.8 shows the ACS module of viterbi decoder.

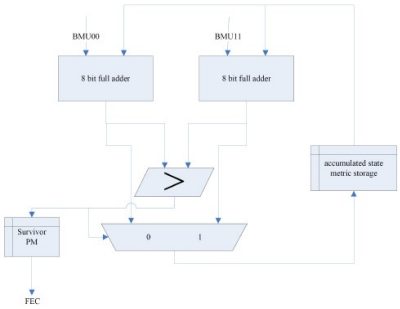


Fig.8 The ACS Module

The total working of decoder can be summarized as follows:

1) The BM Unit (BMU) which calculates the BMs;
2) The Add Compare Select Unit (ACSU) which adds the BMs to the corresponding PMs, compares the new PMs, and then stores the selected PMs in the Path. Metric Memory (PMM); at the same time, the ACSU stores the associated survivor path decisions in the Survivor Memory Unit (SMU);
3) The SMU which stores the survivor path decisions; then the TB mechanism is applied to the SMU.

B. *Architecture of MLVD*

The diagram presented in Fig.9 shows the RE approach with pointer implementation (the upper register carries the pointer and the lower register carries the decoded bits)[5]. The first row of memory decodes the data, if an initial state, S_0, is assumed. The last row records the decoded data, if an initial state, S_{255}, is assumed, and so on. At the end of the decoding process, the row which has the lowest PM is chosen to be the decoder output.

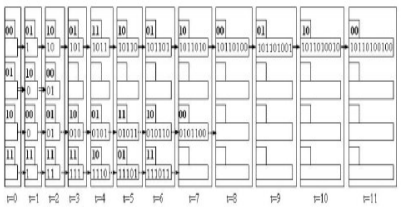


Fig.9 New RE approach with pointer implementation

If the initial state is known, then there is no need for the storage of the other rows except one row next to that state. The new VD implementation is called, the MemoryLess Viterbi Decoder (MLVD)[6].

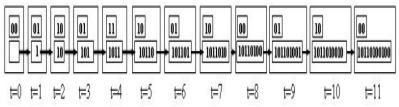


Fig. 10 MLVD approach with pointer implementation

The MLVD is an extra low power design for a VD with the only restriction of resetting the encoder register at each L of the encoded data bits and providing the necessary synchronization. The block diagram of the MLVD, designed in VHDL, is shown in Fig.11[6].

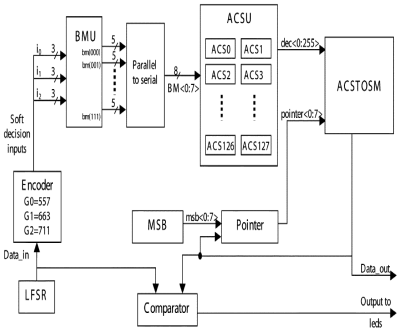


Fig.11 MLVD block diagram

Table I VD Specification

Constraint Length	K= 9
Coding Rate	r = 1/3
Generator Polynomials	G_0= 557, G_1=633, G_2= 711
Decision Level	3-bit Soft Decision
Path Metric	8-Module Arithmetic
Target Speed	2 Mbps

In order to have a built-in self-test design, a Linear Feedback Shift Register (LFSR) and a comparator are added. The LFSR produces the random input for the encoder, whereas the comparator compares the delayed version of the LFSR with the output of the MLVD.

III SIMULATIONS AND RESULTS

To calculate the power estimation, cost values for the MLVD operations are provided in Table 2. It shows the maximum power consumption is in ACSU block and also take the maximum area.

For hardware implementation of the design, we continue with ASIC flow. For that we have synthesized the design using Synopsys Design Compiler. Various area and power report has been generated to summarize the need of hardware size for the decoder.

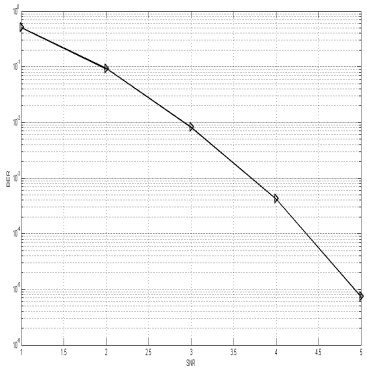


Fig.12 SNR vs BER for MLVD

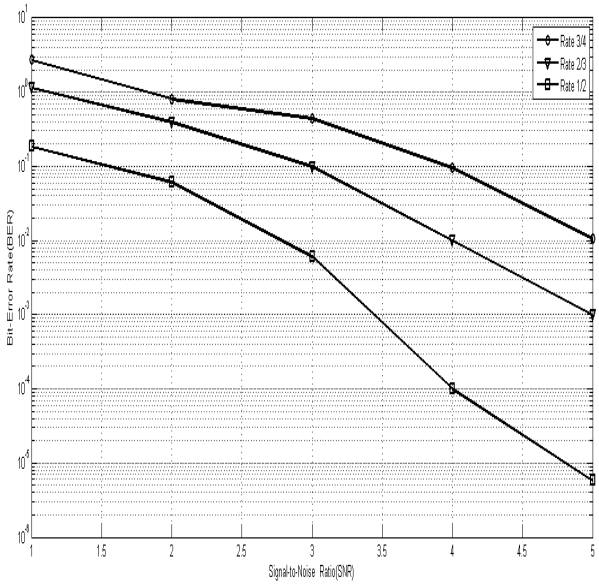
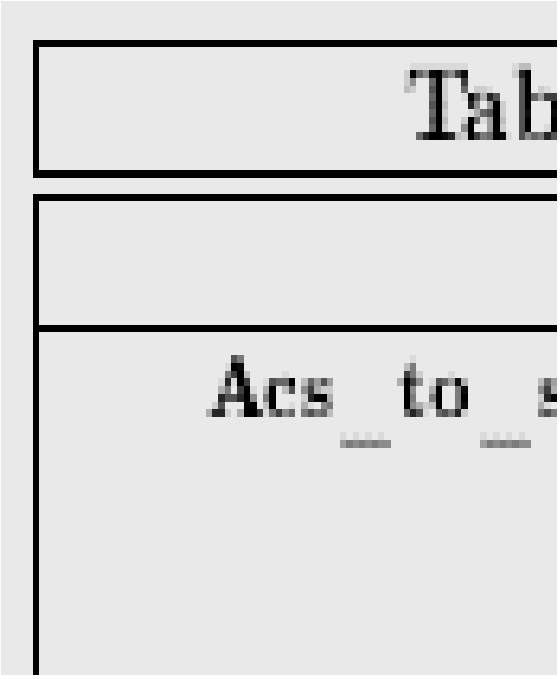


Fig.13 BER For Different Rates 3/4, 2/3 and 1/2

Table II Table for Synthesis of MLVD using Synopsys Design Compiler



Encoder	Number of ports: 12 Number of nets: 33 Number of cells: 30 Number of references: 9 Combinational area: 377.390991 Noncombinational area: 406.421967 Total cell area: 783.812988	Cell Internal Power = 255.2176 uW (91%) Net Switching Power = 26.1067 uW (9%) Total Dynamic Power = 281.3242 uW (100%) Cell Leakage Power = 4.0973 nW
LFSR	Number of ports: 3 Number of nets: 15 Number of cells: 13 Number of references: 3 Combinational area: 32.256001 Noncombinational area: 603.184998 Total cell area: 635.440979	Cell Internal Power = 490.7943 uW (83%) Net Switching Power = 99.4185 uW (17%) Total Dynamic Power = 590.2127 uW (100%) Cell Leakage Power = 2.1631 nW
Parallel_to_Serial Block	Number of ports: 50 Number of nets: 50 Number of cells: 8 Number of references: 8 Combinational area: 903.159790 Noncombinational area: 3251.415283 Total cell area: 4154.576172	Cell Internal Power = 2.8454 mW (85%) Net Switching Power = 517.1058 uW (15%) Total Dynamic Power = 3.3625 mW (100%) Cell Leakage Power = 14.0189 nW
Pointer	Number of ports: 19 Number of nets: 90 Number of cells: 79 Number of references: 14 Combinational area: 954.777649 Noncombinational area: 1032.192139 Total cell area: 1986.969971	Cell Internal Power = 1.1462 mW (81%) Net Switching Power = 267.8141 uW (19%) Total Dynamic Power = 1.4140 mW (100%) Cell Leakage Power = 7.9279 nW
Viterbi_Decoder	Number of ports: 16 Number of nets: 590 Number of cells: 12 Number of references: 12 Combinational area: 95718.250000 Noncombinational area: 335350.750000 Total cell area: 431074.843750	Cell Internal Power = 88.4137 mW (99%) Net Switching Power = 1.2357 mW (1%) Total Dynamic Power = 89.6495 mW (100%) Cell Leakage Power = 1.6913 uW

CONCLUSIONS

Among the several architectures that are available to realize the VD, the RE method with pointer concept is conceptually the simplest, fastest, and most commonly used in VDs with only small values of k. By reinforcing the initial state of the convolutional encoder and synchronizing the VD with the resetting procedure, a design, the MLVD, with the highest power reduction is realized. The new MLVD is a memoryless high speed, low latency, and low power variation to the VD with an approximated BER of 10^{-5} and an SNR of 5 dB. The MLVD along with a convolutional encoder is implemented on a Xilinx ISE chip to demonstrate both the design's functionality and feasibility of implementation. Design synthesis results shows the hardware utilization and power consumption of the resources on FPGA and chip.

REFERENCES

[1] Viterbi, "Error bounds for convolutional codes andasymptotically optimum decoding algorithm," IEEE Transactions on Information theory, vol. It-13, no. 2, pp. 260_269, April 1967.

[2] Dalia A. El-Dib and M.I. Elmasry, "Modified Register-Exchange Viterbi Decoder for Low-Power Wireless Communications," IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 51, no. 2, pp. 371- 378, February 2004.

[3] S. B. Wicker, "Error Control Systems for Digital Communication and Storage". Prentice Hall, 1995.

[4] G. Forney, "The viterbi algorithm", Proceedings of the IEEE, vol. 61, no. 3, pp. 268_278, March 1973.

[5] Dalia A. El-Dib and M.I. Elmasry,"Low power register-exchange Viterbi decoder for high speed wireless communications," IEEE International Symposium on Circuits and Systems, May 2002, pp. 737-740.

[6] S A. El-Dib and M.I. Elmasry, "Memoryless Viterbi Decoder: an extremely low power Viterbi Decoder," IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 51, no. 3, pp. 371- 378, February 2004.