Design and Implementation of Adaptive Viterbi Decoder for High Speed Applications

Chaitali Brahmbhatt[1], Minal Mishra[2]

[1] Department of Communication System Engineering

[2] Department of Electronics & Communication

Gujarat Technological University, India

[1]Chaitali1208@gmail.com,[2]minal_7k@yahoo.com

ABSTRACT

Error free communication is important aspect of high speed wireless communication. Error detection is possible through convolution code using Viterbi algorithm in VLSI technology. We had compared the BER performance on of our proposed adaptive Viterbi decoder to that of convectional Viterbi decoder. In this paper developed a method to eliminate the search the best winner operation and hence enable high. Throughput implementation simulates in verilog. FPGA implementation on performance based on behavior of noise analysis, power saving at computational and decoder output generation. Modest silicon area reduction area without degradation.

Keywords: **Adaptive Viterbi algorithm, Very large-scale integration (VLSI) architecture, FPGA.**

I. Introduction

As the error-correcting capability of convolution codes is improved by employing codes with larger constraint lengths K, the complexity of decoder is increased. The Viterbi algorithm [2], which is the most extensively employed decoding algorithm for convolution codes, is effective in achieving noise tolerance, but the cost is an exponential growth in memory, computational resources, and power consumption. To address this issue, the reduced-complexity adaptive Viterbi algorithm (AVA) [2], [3] has been developed. The average number of computations per decoded bit for this algorithm is substantially reduced versus the Viterbi algorithm, while comparable bit-error rates (BER) are preserved.

The strategy of adaptation with the changing channel conditions is a promising method to resolve the problem of combined requirement of the low energy, high performance and considerable flexibility in FEC design. Such issues have been proposed an efficient FEC scheme referred to Adaptive Viterbi Algorithm VA (AVA). In this paper, we focus on the Add Compare Select (ACS) unit of AVA, which is the major component for adaptive Viterbi Decoder. In [2], the minimum path metric is stored into the Path Metric Memory Unit (PMU) in the current iteration of the trellis recursion and are read out for calculating the path metric and determining the survivor state in the next iteration. Through reformulated AVA, a significant hardware reduction is received.

The remainder of this paper is organized as follows.
Sections 2 and 3 present the proposed Adaptive Viterbi algorithm and the corresponding VLSI architecture. Design of adaptive Viterbi convolution code decoders are presented in Section 4, and the conclusions are drawn in Section 5. The benefits of our AVA architecture are highlighted by experimental results presented in Section 5.

II. Adaptive Viterbi Algorithm

The well-known VA has been described in literature extensively. The data path of the Viterbi Decoder is composed of three major components: Branch Metric Calculation Unit (BMU), ACS and Survivor Memory Unit (SMU) as shown in Fig (a) received by viterbi encoder output. The branch metrics are calculated from the received channel symbols in BMU and then fed into the ACS which performs Add-Compare-Select for all the states. The decision bits generated in ACS are stored and retrieved in the SMU in order to finally decode the source bits along the final survivor path. The state metrics of the current iteration are stored into the Path Metric Memory Unit (PMU) and read out for the use of the next iteration.

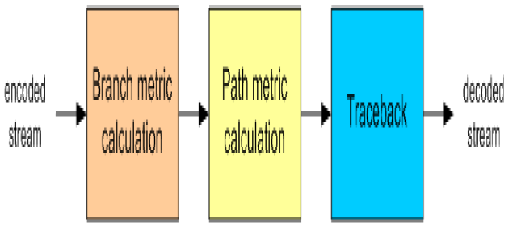

Fig. 1 Ordinary Viterbi Flow

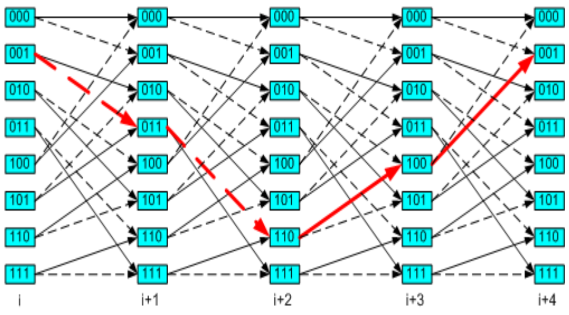

Fig. 2 Adaptive Viterbi Flow

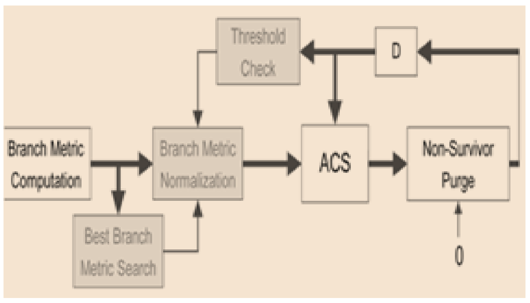
Threshold Check
D
Branch Metric Computation
Branch Metric Normalization
ACS
Non-Survivor Purge
Best Branch Metric Search
0

Fig. 3 Example of Select a Best Survival Path

The adaptive Viterbi algorithm [2] was introduced with the goal of reducing the average computation and
Path storage required by the Viterbi algorithm. Path retention is based on the following criteria.

1. A threshold T indicates that a path is retained if its path metric is less than dm + T, where dm is the minimum
cost among all surviving paths in the previous trellis stage.

2. The total number of survivor paths per trellis stage is limited to a fixed number, Nmax, which is pre-set prior
to the start of communication.

III. AVA Architecture

To explore this architecture exhibits significant parallelism and supports to adaptive decoder hardware to changing channel noise characteristics. A high-level view of the implemented adaptive Viterbi decoder architecture is shown in Figure 1.

The decoder contains a data path and an associated control path. Like most Viterbi decoders [2], A BMG unit determines distances between received and expected symbols. The ACS unit determines path costs and identifies lowest-cost paths. The survivor memory stores lowest the path metric array holds per-state path metrics. The flow of data in the data path and the storage of results is determined by the control path. In the implemented decoder, the expected symbol value (BM select) is used to select the appropriate branch metric from the BMG, as shown at the left in Figure 2. This branch metric value is combined with the path metric of its parent present state to form a new path metric, di. At each trellis stage, the minimum-value surviving path metric among all path metrics for the preceding trellis stage, dm, is computed. New path metrics are compared to the sum dm + T to identify path metrics with excessive cost. Comparators are then used to determine the life of each path based on the threshold, T. If the threshold condition is not satisfied by path metric dm + T, the corresponding path is discarded. Once the paths that meet the threshold condition are determined, the lowest-cost Nmax paths are selected. Sorting circuitry is eliminated by allowing feedback adjustments to the parameter T for each received symbol. If the number of paths that survive the threshold is less than Nmax, no iteration is required. As show in Figure 2, for stages when the number of paths surviving the threshold condition is greater than Nmax, T is iteratively reduced by 2 for the current trellis stage until the number of paths surviving the threshold condition is equal to or less than Nmax. The T value is reset to its original value prior to the processing of the next trellis stage. The output of the ACS units includes path valid signals which indicate which of the 2 * Nmax paths have survived. Most communication systems desire links with predictable performance, which is usually specified by a fixed BER.

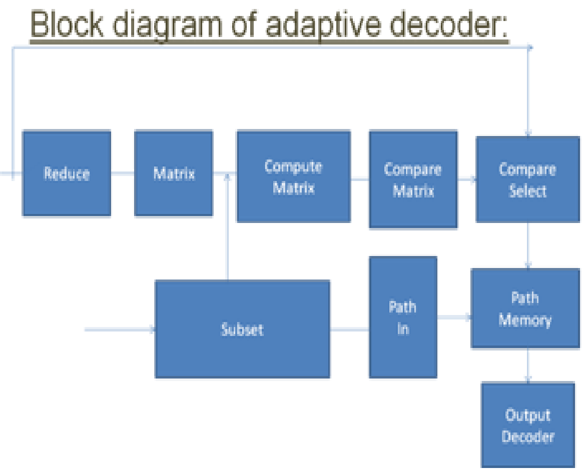

Fig 4: A Block diagram of Adaptive decoder

Although desire decoder accuracy remains constant, channel signal-to-noise ratios can vary widely due to factors such as the propagation distance and the shadowing of the transmitted signal by large objects. In the presence of increased noise power (equivalently, a decreased SNR due to a weaker signal), a higher constraint length code is required to maintain a constant BER.

IV. PERFORMANCE ANALYSIS

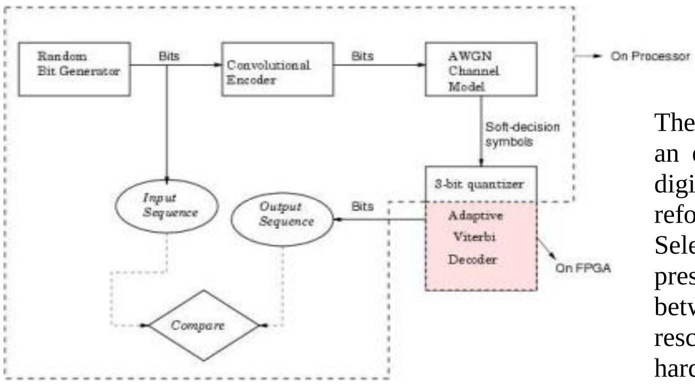

Fig 5: System model performance analysis

Test Platform

To test the practicality of our reconfigurable AVA architecture, a hardware implementation of the decoder was tested as part of a communication system. The communication system model used for experimentation is shown in Figure 3. The Random Bit Generator is a C module that generates a randomized bit sequence to model transmitted data. The convolution encoder can be parameterized to assorted constraint lengths. The modulator converts a coded bit to a real number: 0 -> 1, 1 -> -1 for the binary phase-shift keyed (BPSK) system employed. The output of the modulator is input to the AWGN channel simulator. This block simulates a noisy channel where Gaussian noise is added to the transmitted signal. The amount of noise depends on the signal-to-noise ratio preset by the user. The symbols obtained from the AWGN channel model are quantized before being sent to the decoder as its input. On receiving the input, the decoder attempts to recover the original sequence. All software modeling of the communication system was performed using a 366 MHz Celeron PC. All the existing variants of adaptive Viterbi algorithms (including the one presented above) may lose the correct path [i.e., the maximum likelihood (ML) path] during the decoding due to their nature of non exhaustive trellis search. This inevitably results in decoding performance degradation compared with Viterbi algorithm. As discussed in [1], in order to quickly recover the correct path once it has been lost, the width of the survivor path retention window .Thus, the conventional adaptive Viterbi decoder reduces power by choosing an appropriate that will meet the performance constraints of the system while allowing up to survivors to be kept at each decoding depth [1], [3].

V. CONCLUSION

The use of error-correcting codes has proven to be an effective way to overcome data corruption in digital communication channels. An efficient reformulation based architecture or Threshold Selection in Adaptive Viterbi Decoding is presented. The architecture exploits the inherent between the Add Compare Select Operation and rescale operation in Adaptive Viterbi Decoding, the hardware complexity for the threshold selection in Adaptive Viterbi Decoding is significantly reduced in FPGA technologies, which leads to a corresponding significant reduction in area, power and delay. It should be noted that the proposed technique will also achieve a similar power, area and speed efficiency with different specifications The architecture with its significant reduction of hardware complexity is very attractive for wireless applications with high constraint length such as CDMA systems.

References:

[1] J. Proakis, Digital Communications. New York, N.Y.: McGraw-Hill, 1995.

[2] F. Chan and D. Haccoun, “Adaptive Viterbi decoding of convolutional codes over memory less channels,” IEEE Transaction on
Communications, vol. 45, no. 11, pp. 1389–1400, Nov. 1997.

[3] S. J. Simmons,“Breath-first trellis decoding with adaptive effort,” IEEE Transactions on Communications, vol. 38, no. 1, pp. 3–12, Jan.

[4] D. Matolak and S. Wilson, “Variable-complexity trellis decoding of binary convolutional codes,” IEEE Transactions on Communications, vol. 44, no. 2, pp. 121–126, Feb. 1996.

[5] S. Simmons, “An error bound for reduced-state Viterbi decoding of TCM codes,” IEEE Communications Letters, vol. 3, no. 9, pp. 266–268, Sept.1999.

[6] L. Shang, A. Kaviani, and K. Bathala, “Dynamic power consumption in Virtex-II FPGA family,” in Proceedings, ACM/SIGDA International Symposium on Field Programmable Gate Arrays, Monterey, Ca., Feb. 2002, pp. 157–164.

[7] MT48LC2M32B2 SDRAM Data Sheet, Micron Technologies, Inc., 2003.

[8] Xilinx XC4000 Data Sheet, Xilinx Corporation, 2001, http://www.xilinx.com.
[9] ISE Manual, Xilinx Corporation, 2001, http://www.xilinx.com.
[10] Xilinx Virtex Data Sheet, Xilinx Corporation, 2001, http://www.xilinx.com.
[11] A. Sinha and A. Chandrakasan, "JouleTrack - a web based tool for software energy profiling," in Proceedings, ACM/IEEE 35rd Design Automation
[12] R. Henning and C. Chakrabarti, "Low power approach to decoding convolutional codes with adaptive Viterbi algorithm approximations," in Proceedings, IEEE/ACM International Symposium on Low Power Electronics and Design, Monterey, CA, Aug. 2002, pp. 68–71.
[13] M. Guo, M. O. Ahmad, M. Swamy, and C. Wang, "An adaptive Viterbi algorithm based on strongly connected trellis decoding," in Proceedings, IEEE International Symposium on Circuits and Systems, Scottsdale, AZ, May 2002, pp. 137–140.
[14] G. Fettweis and H. Myer, "High-speed parallel Viterbi decoding: Algorithm and VLSI-architecture," IEEE Communications Magazine, vol. 29, no. 5, pp. 46–55, May 1991.
[15] WILD-ONE Reference Manual, Annapolis Microsystems, Inc., 1999.
[16] L. Shang, A. Kaviani, and K. Bathala, "Dynamic power consumption in Virtex-II FPGA family," in Proceedings, ACM/SIGDA International Symposium on Field Programmable Gate Arrays, Monterey, Ca., Feb. 2002,pp. 157–164.
[17] MT48LC2M32B2 SDRAM Data Sheet, Micron Technologies, Inc., 2003.