High Speed Viterbi Decoder for W-LAN and Broadband Applications

A. R. Abdul Shakoor, V. Szwarc, and T. A. Kwasniewski*

Communications Research Centre Canada
3701 Carling Avenue, Box 11490, Station H
Ottawa, Ontario, K2H 8S2, Canada.
arafeeq@crc.ca, valek.swarc@crc.ca

*Dept. of Electronics, Carleton University
1125 Colonel By Drive
Ottawa, Ontario, K1S 5B6, Canada.
tak@doe.carleton.ca

Abstract: This paper presents a configurable Viterbi decoder implementation that meets the requirements of the IEEE 802.11b and 802.16a standards. The programmable Very high speed integrated circuit Hardware Description Language (VHDL) design supports a constraint length (K) 7 Viterbi decoder realization with code rates (R) of 1/2 and 1/3, and trace-back lengths (TBL) of 35 and 50 symbols. To assure high throughput, an architecture incorporating 32 Add Compare Select (ACS) units operating in parallel has been selected. Circuit simulation results, based on an Altera FPGA, are presented and confirm a throughput of 160 Mbps.

I. INTRODUCTION

To assure data integrity, error correction techniques are utilized in wireless and satellite communications [1-6]. The data is commonly encoded at the transmitter with convolutional and/or block codes. The selection of the encoder/decoder technology typically reflects tradeoffs that balance implementation complexity and communication overheads with performance. A convolutional encoder for K=7 and R=1/2, recommended by the IEEE 802.11b and 802.16a standards [1,2], is depicted in Fig. 1. The two output bits C1 and C2, which constitute a symbol, are generated and transmitted at every clock cycle. The number of memory element m is one less than the constraint length K. All six flip-flops (F1 to F6) are reset at circuit initialization.

The Viterbi algorithm estimates the maximum likelihood path through the trellis based on received symbols. The decoder reconstructs the actions of the encoder by selecting the most likely path from a structured graph, corresponding to a tree or trellis that describes all possible states as shown in Fig. 2.

The decoding of each bit in a Viterbi decoder requires the execution of a large number of arithmetic operations. Typically, for constraint length K=7, the Viterbi decoder performs 128 add and 64 compare-select operations for every decoded bit. Thus decoder performance depends on the throughput of the ACS unit [3-5].

A number of different Viterbi decoder realizations for rate 1/2 and constraint lengths of K=7 [5], K=6 [3], and K=3 [4] have been reported in the literature. The architectures and implementations exploit in varying degree the parallelism inherent in the Viterbi algorithm. To address the requirements of W-LAN and broadband applications, the decoder was implemented as a reconfigurable VHDL macrocell. The remaining sections consider its architecture, implementation, and results.
II. DECODER FUNCTIONALITY AND ARCHITECTURE

In this section, the architecture and the operation of the proposed 64 state Viterbi decoder, depicted in Figs. 2 and 3, are presented. When a new symbol is received, the Branch Metric unit calculates the Hamming distances for all possible trellis state transitions. The calculation of Hamming distances starts from the "0" state as shown in Fig. 2. The encoder is flushed with zeros at the end of each frame. Thus the decoder, which performs frame by frame decoding, starts from state "0". The computation of branch metrics is based on a comparison of current input symbol with the expected value.

The path metrics, identified in bold face at each node of the trellis in Fig. 2, correspond to the Hamming distances between the current and expected symbols. At stage 1, the second received symbol "01" is processed. First the branch metrics are computed for all 4 incoming paths, then to these branch metrics the two path metrics computed at stage 0 are added. Thus, 4 new path metrics are computed at nodes c, e, f, and g. At stage 2, the third received symbol "00" is processed. First the branch metrics are computed for all 8 incoming paths. To these branch metrics the 4 path metrics computed at stage 1 are added and 8 new path metrics are computed at nodes h, i, j, k, l, m, n, and o. Similarly, 16, 32, and 64 new path metrics are computed at stages 3, 4, and 5 respectively.

From the 6th stage onwards, there are 2 incoming paths per node, of which one is eliminated. The branch metrics are computed for all 128 paths. Let us consider the two branches converging at a representative node. An ACS unit adds the branch metric between node v and w to the partial path metric at node v and also adds the branch metric between node u and w to the partial path metric at node u. Subsequently, the ACS unit compares the two input path metrics at node w. The path with the lowest path metric is the surviving path for that node. Similarly, all other nodes are updated.

The calculations to determine the path metrics, for example at nodes w and y, can be done by a single ACS unit [3]. According to the Viterbi algorithm [8], at each stage only $2^{K-1}$ paths need be retained as survivor paths and the remaining paths can be discarded. The 1st stage path metrics are referred to as "partial path metrics" during the processing of the (i+1)th stage. The survivor path memory (SPM) stores the updated path metrics. One survivor path is stored for each state at each stage of the trellis. Proceeding in this way, the whole 64-state trellis is traversed and survivor paths are stored for each state for 35 stages before decoding starts. For a 64 state decoder with a 35 stage decoding window, 2240 bits are stored in the SPM unit. Once the whole trellis has been scanned and processed, the traceback unit parses the data stored in the SPM unit and outputs the estimated survivor path with a latency of TBL.

III. DECODER IMPLEMENTATION

A. Branch Metric Computation

To increase computational efficiency and reduce complexity of the branch metric module, a new design approach is presented in this section. The Viterbi algorithm implementation requires at each time step $2^{(m+1)}$ distance calculations and $2^m$ comparison operations, where $m$ is the number of memory elements in the convolutional
encoder. Thus, for a K=7 Viterbi decoder where m = 6, the number of distance calculations and comparisons per stage corresponds to $2^6$ and $2^6$ respectively.

For a trace back length T, the number of distance calculations and comparisons can be expressed as $2^{(m+1)} T$ and $2^m T$, respectively. Also, each distance calculation requires "n" binary comparisons for a rate 1/n system. Hence the total number of additions and comparison operations required can be expressed as:

$$Y_a = 2^{(m+1)} T \times n$$

$$Y_c = 2^m T \times n$$

For an information bit encoded by a rate ½ encoder, four possible symbols could be transmitted, namely 00, 10, 01, 11 and these symbols are treated as expected symbols at the receiver. For example, if a "00" is transmitted, then there is a possibility that it could be received as "00" (no error), "10" or "01" (single error), or "11" resulting in a double error.

A considerable number of computations is required to determine distances at each node of each stage. The current design aims to reduce the computational complexity by pre-coding in hardware the Hamming distances so that only a 2-bit comparison is required per stage for the rate 1/2 case. For a received symbol, say 00, the Hamming distance (HD) after comparison with expected symbols is:

$$HD<00,00>=0,$$
$$HD<00,01>=1,$$
$$HD<00,10>=1,$$
$$HD<00,11>=2.$$ 

These Hamming distances are used to process each stage of the trellis and, when the next symbol is received at the next stage, a 2-bit comparison will yield the exact four Hamming distances for that particular stage and so on.

This reduces the computational requirements for determining the Hamming distances at every node and consequently the path metrics for all 64 states at each stage of the trellis.

To determine the computational savings, consider, for example, a traceback length of 50 for the case K=7. The complexity equation after substitution is: $Y_a = 128 \times 50 \times 2$. After applying this technique, the number of additions is reduced to: $Y_a = 4 \times 50 \times 2$, or by a factor of 32.

B. Reconfigurability

Reconfigurability has been implemented both at the VHDL code level and the circuit level. Thus the VHDL code can be configured for decoder operation at rate 1/2 or 1/3. The circuit can then be optimized in terms of hardware resources and/or throughput for the target technology or FPGA device. The circuit can be further configured in terms of the trace back length to better match the decoder's operation to channel conditions. For rate 1/2, the decoder can be set to have TBL of 35 or 50. Adequate performance for a decoder with constraint length K=7 can be obtained with a TBL of 35 [6, 9]. However, a longer TBL may provide a performance improvement for certain channel conditions [6].

C. ACS Architecture and Metric Scaling

The elements that influence the decoder's area and power consumption include constraint length, code rate, window size, ACS path metric register size, and architecture. The first three parameters normally reflect the communication system's specification. The fourth parameter, the path metric register dimensionality, is dependent upon the decoder's architecture and implementation.

To increase throughput, different approaches can be adopted including use of multiple ACS units [3-5, and 8]. One such realization [5] for K=7 incorporates 64 ACS units. Further improvements can be achieved by incorporating parallelism in the Branch Metric unit [5]. The approach adopted here is to process 2 new path metrics in a single ACS unit. Thus 32 ACS units operating in parallel, as shown in Fig. 3, handle all of the 64 states in the trellis.
As the path metric is the cumulative sum of branch metrics, it increases with each addition. To limit the dynamic range of path metrics, a metric normalization method has been adopted [7, 10]. Synthesis and mapping onto an Altera Apex FPGA of the decoder with 8 bit path metric registers resulted in a 15,356 logic cell circuit. To circumvent the problem of overflow, a metric rescaling technique was used [7]. For 6 bit path metric registers, the decoder cell count was reduced to 13,084 logic cells.

circuits, and multiplexers to route the signals. In normal operation the convolutional encoder and the Viterbi decoder can operate independently. The self-test circuit enables on-chip at-speed verification of both the encoder and decoder.

V. SUMMARY

The parallel architecture of the Viterbi decoder incorporating metric scaling resulted in a high throughput area efficient circuit. This implementation required only a 30% increase in overall gate count as compared to a decoder design based on a single ACS.

REFERENCES