DESIGN OF HIGH SPEED LOW POWER VITERBI DECODER FOR TCM DECODERS

J. Arunkumar¹
jarun_468@yahoo.co.in
¹PG Scholar, Dept of DSCE, Sree Nidhi Institute of Science and Technology, Ghatkesar, Rangareddy Telangana,

CH. Sathyanaryana²
sathyanarayanac@sreenidhi.edu.in
²Associate Professor, Dept of ECE, Sree Nidhi Institute of Science and Technology, Ghatkesar, Rangareddy Telangana.

Abstract: Viterbi Decoder (VD) employed in digital wireless communication plays a dominant role in the overall power consumption of trellis coded modulation (TCM) decoder. Power reduction in VD could be achieved by reducing the number of states. A pre-computation architecture with Talgorithm was implemented for this purpose, and when we compare this result with full Trellis VD, this approach significantly reduces power consumption without degrading decoding speed much. Low power design of VD for TCM systems with reliable delay is presented in this paper. This work focuses on the realization of convolution encoder and adaptive Viterbi decoder (AVD) with a constraint length (K) of 9 and a code rate (k/n) of ½ using field programmable gate array (FPGA) technology. The performance of the implemented AVD is analyzed by using ISE 10.1 and Modelsim simulations.

Keywords: Trellis Coded Modulation (TCM), Adaptive Viterbi Decoder (AVD), VLSI.

I. INTRODUCTION

The Viterbi decoding algorithm, proposed in 1967 by Viterbi, is a decoding process for convolutional codes in memory-less noise. The algorithm can be applied to a host of problems encountered in the design of communication systems. The Viterbi Algorithm (VA) finds the most-likely state transition sequence in a state diagram, given a sequence of symbols. The Viterbi algorithm is used to find the most likely noiseless finite-state sequence, given a sequence of finite-state signals that are corrupted by noise. In order to reduce the power consumption, and increase the speed, an asynchronous technique that is Delay Insensitive Null Convention Logic (NCL) for Viterbi Decoder (VD) and its Encoder using Dual rail signal [5] is proposed in this paper. In VLSI design the major cause for the power dissipation is the dynamic power dissipation about 80 to 90 percent of total power dissipation. NCL reduces the dynamic power consumption in terms of reducing the switching activity and also it reduces the Glitch power significantly, thereby achieving the lower power. The basic Viterbi algorithm [12] was applied in digital communication systems, speech and character recognition. It focused on the operations and the practical memory requirement to implement the Viterbi algorithm in real-time. Based on the data generated and decoded [10] from the zero Hamming distance path, unnecessary computations in the Viterbi decoder was avoided. Speed and power were not considered. Low-power bit-serial Viterbi decoder chip [1] for the code rate r = 1/3 and the constraint length K = 9 (256 states) was discussed. The add compare-select (ACS) module was based on the bit serial arithmetic and implemented with the pass transistor logic circuit. The Scarce state transition (SST) scheme [14] employed a simple pre decoder followed by a pre encoder to reduce the transitions of the Viterbi decoder.

II. LITERATURE SURVEY

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. Therefore, the computational overhead and decoding latency due to predecoding and re-encoding of the TCM signal become high. In our preliminary work [9], we proposed an add-compare-select unit (ACSU) architecture based on precomputation for VDs incorporating T -algorithm, which efficiently improves the clock speed of a VD with T -algorithm for a rate-3/4 code. In this work, we further analyze the precomputation algorithm. A systematic way to determine the optimal precomputation steps is presented, where the minimum number of steps for the critical path to achieve the theoretical iteration bound is calculated and the computational complexity overhead due to pre-computation is evaluated. Then, we discuss a complete low-power high-speed VD design for the rate-3/4 convolutional code [1]. Finally ASIC implementation results of the VD are reported.

III. VITERBI DECODER

![Fig. 1 Viterbi Decoder block diagram](image)

A typical functional block diagram of a Viterbi decoder is shown in Fig. 1. First, branch metrics (BMs) are calculated in the BM unit (BMU) from the received symbols. In a TCM decoder, this module is replaced by transition metrics unit (TMU), which is more complex than the BMU. Then, BMs are fed into the ACSU that recursively computes the PMs and outputs decision bits for each possible state transition. After that, the decision bits are stored in and retrieved from the SMU in order to decode the source bits along the final survivor path. The PMs of the current iteration are stored in the PM unit (PMU).

**Implementation of a Viterbi decoder**

The major tasks in the Viterbi decoding process are as follows:
1. Quantization: Conversion of the analog inputs into digital.
2. Synchronization: Detection of the boundaries of frames and code symbols.
3. Branch metric computation.
4. State metric update: Update the state metric using the new branch metric.
5. Survivor path recording: Tag the surviving path at each node.
6. Output decision generation: Generation of the decoded output sequence based on the survivor path information.

Figure 2 shows the flow of the Viterbi decoding algorithm, which performs the above tasks in the specified order.

This section discusses the different parts of the Viterbi decoding process. Analog signals are
quantized and converted into digital signals in the quantization block. The synchronization block detects the frame boundaries of code words and symbol boundaries.

The branch metric computation block compares the received code symbol with the expected code symbol and counts the number of differing bits. An implementation of the block is shown in Figure 3.

**IV. PRECOMPUTATION ARCHITECTURE**

**Precomputation Algorithm**

\[
p_{\text{opt}}(n) = \min \{ p_0(n), p_1(n), \ldots, p_{2k-1}(n) \} = \min \{ \min [p_{0,0}(n-1) + B_{0,0}(n), p_{0,1}(n-1) + B_{0,1}(n), \ldots, p_{0,p}(n-1) + B_{0,p}(n)], p_{1,0}(n-1) + B_{1,0}(n), p_{1,1}(n-1) + B_{1,1}(n), \ldots, p_{1,p}(n-1) + B_{1,p}(n), \ldots, p_{2k-1,0}(n-1) + B_{2k-1,0}(n), p_{2k-1,1}(n-1) + B_{2k-1,1}(n), \ldots, p_{2k-1,p}(n-1) + B_{2k-1,p}(n) \}.
\]

Now, we group the states into several clusters to reduce the computational overhead caused by look-ahead computation. The trellis butterflies for a VD usually have a symmetric structure. In other words, the states can be grouped into m clusters, where all the clusters have the same number of states and all the states in the same cluster will be extended by the same Bs. Thus (1) can be rewritten as

\[
P_{\text{opt}} = \min \{ \min (P_{s}(n-1))_{\text{in cluster } 1} + \min (B_{s}(n))_{\text{for cluster } 1}, \min (P_{s}(n-1))_{\text{in cluster } 2} + \min (B_{s}(n))_{\text{for cluster } 2}, \ldots, \min (P_{s}(n-1))_{\text{in cluster } m} + \min (B_{s}(n))_{\text{for cluster } m} \}.
\]

The minimum (Bs) for each cluster can be easily obtained from the BMU or TMU and \( \min(P_{s}) \) at time \( n-1 \) in each cluster can be precalculated at the same time when the ACSU is updating the new \( P_{s} \) for time \( n \). Theoretically, when we continuously decompose \( P_{s}(n-1), P_{s}(n-2), \ldots \), the precomputation scheme can be extended to Q steps. Where \( q \) is any positive integer that is less than \( n \). Hence \( P_{\text{opt}}(n) \) can be calculated directly from \( P_{s}(n-q) \) in \( q \) cycles.

**Choosing Precomputation steps**

Through a design example that, \( q \)-step pre-computation can be pipelined into \( q \) stages, where the logic delay of each stage is continuously reduced as \( q \) increases. As a result, the decoding speed of the low-power VD is greatly improved. However, after reaching a certain number of steps, \( q_{b} \), further precomputation would not result in additional benefits because of the inherent iteration bound of the ACSU loop. Therefore, it is worth to discuss the optimal number of precomputation steps.

In a TCM system, the convolutional code usually has a coding rate of \( R/(R+1) \), \( R=2,3,4, \ldots \), so that in (1), \( p=2R \) and the logic delay of the ACSU is \( T_{\text{ACSU}}=T_{\text{adder}}+T_{\text{p-in_comp}} \), where \( T_{\text{adder}} \) is the logic delay of the adder to compute \( P_{s} \) of each candidate path that reaches the same state and \( T_{\text{p-in_comp}} \) is the logic delay of a p-input comparator to determine the survivor path (the path with the minimum metric) for each state. If
T-algorithm is employed in the VD, the iteration bound is slightly longer than TACSU because there will be another two input comparator in the loop to compare the new Ps with a threshold value obtained from the optimal Path metric and preset T as shown in (3)

$$T_{bound} = T_{adder} + T_{p\_in\_comp} + T_{2\_in\_comp}. \quad (3)$$

**Fig.4 Rate 3/4 convolutional encoder.**

**One Step Precomputation**

For the convenience of our discussion we define the left most register in Fig. 3 as the most significant bit (MSB) and right most register as the least significant bit (LSB). The 64 states and path metrics are labeled from 0 to 63. A careful study reveals that the 64 states can be partitioned into two groups odd numbered Ps (when „LSB” is 1) and even numbered (when „LSB” is 0). The odd PMs are all extended by odd Bs (when Z₀ is „1”) and the even PMs are all extended by even Bs (when Z₀ is „0”). The minimum P becomes:

$$P_{opt}(n) = \min \{ \min (\text{even Ps (n-1)}) + \min (\text{even Bs(n)}), \min (\text{odd Ps (n-1)}) + \min (\text{odd Bs(n)}) \}.$$
In this section, we address an important issue regarding SMU design when T-algorithm is employed. There are two different types of SMU in the literature: register exchange (RE) and trace back (TB) schemes. In the regular VD without any low-power schemes, SMU always outputs the decoded data from a fixed state (arbitrarily selected in advance) if RE scheme is used, or traces back the survivor path from the fixed state if TB scheme is used, for low-complexity purpose. For VD incorporated with T-algorithm, no state is guaranteed to be active at all clock cycles. Thus it is impossible to appoint a fixed state for either outputting the decoded bit (RE scheme) or starting the trace-back process (TB scheme). In the conventional implementation of T-algorithm, the decoder can use the optimal state (state with \( P_{opt} \)), which is always enabled, to output or trace back data. The process of searching for \( P_{opt} \) can find out the index of the optimal state as a byproduct. However, when the estimated \( P_{opt} \) is used [8], or in our case \( P_{opt} \) is calculated from PMs at the previous time slot, it is difficult to find the index of the optimal state.

**V. SIMULATION RESULTS**

**Viterbi Decoder (Top module)**


**VI. CONCLUSION**

The precomputation architecture that incorporates Talgorithm efficiently reduces the power consumption of VDs without reducing the decoding speed appreciably. This algorithm is suitable for TCM systems which always employ high-rate convolutional codes. Both the ACSU and SMU are modified to correctly decode the signal. Compared with the full-trellis VD without a low-power scheme, the precomputation VD could have low power consumption with reliable decoding speed. A reusable Viterbi decoder was carried out by adopting the Process Element technique.

**REFERENCES**

[1] Trellis coded modulation” Intuitive guide to principles of communicaton www.complex toreall.com