RTL Design & VLSI Implementation of Convolution Encoder & Viterbi Decoder

K. Rambabu¹, Ch. Prabhavathi²

¹Professor, Dept of ECE, Srinivasa Institute of Engineering and Technology, Amalapuram, AP, India, E-mail: ksrk73@gmail.com.
²PG Scholar, Dept of ECE, Srinivasa Institute of Engineering and Technology, Amalapuram, AP, India, E-mail: prabhavathi42@gmail.com.

Abstract: High-speed, low-power design of Viterbi decoders for trellis coded modulation (TCM) systems is presented in this paper. It is well known that the Viterbi decoder (VD) is the dominant module determining the overall power consumption of TCM decoders. A pre-computation architecture incorporated with T-algorithm for VD, which can effectively reduce the power consumption without degrading the decoding speed much is to be proposed.

Keywords: Trellis Coded Modulation (TCM), Viterbi Decoder, Convolutional Code, T-Algorithm.

I. INTRODUCTION

The use of convolutional codes with probabilistic decoding can significantly improve the error performance of a communication system. Trellis coded modulation schemes are used in many bandwidth efficient systems. Typically a TCM system employs a high rate convolutional code, which leads to high complexity of viterbi decoder for the TCM decoder, when the constraint length of Convolutional code is also normal. For example the rate ¾ convolutional code used in trellis coded modulation system for any application has a constraint length of 7 will be in the complexity of the corresponding viterbi decoder for a rate ½ convolutional code with constraint length of 9 due to the large number of transitions in the trellis. So, in terms of power consumption, the viterbi decoder is dominant module in a TCM decoder. In order to reduce the computational complexity as well as power consumption, low power schemes should be exploited for the VD in a TCM decoder. General solutions for low power viterbi decoder design will be studied in our implementation work. Power reduction in VDs could be achieved by reducing the number of states, (for example reduced state sequence decoding, M-algorithm and T-algorithm) or by over scaling the T-Algorithm has been shown to very efficient in reducing the power consumption. However, searching for the optimal path metric in the feedback loop still reduces the decoding speed.

To overcome this drawback, T-Algorithm has proposed in two variations, the relaxed adaptive VD, Which suggests using an estimated optimal path metric, instead of finding the real one each cycle and the limited-search parallel state VD based on scarce state transition (SST). 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 path metric and the accurate one. On the other hand the SST based scheme requires pre-decoding and re encoding process 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. So, the computational overhead and decoding latency due to pre-decoding and re encoding of the TCM signal become high. An add-compare select unit (ACSU) architecture based on pre-computation for VDs incorporating T-Algorithm, which efficiently improves the clock speed of a VD with T-Algorithm for a rate ¾ code. Now, we further analyze the pre-computation algorithm. A systematic way to determine the optimal pre-computation steps is shown, where the minimum number of steps for 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 VD design for the rate ¾ convolutional code. Finally ASIC implementation results of VD with convolutional encoding are shown.

II. VITERBI DECODER

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).

**Fig. 1 Viterbi Decoder block diagram.**

A. Implementation of A Viterbi Decoder

The major tasks in the Viterbi decoding process are as follows:

- **Quantization:** Conversion of the analog inputs into digital.
- **Synchronization:** Detection of the boundaries of frames and code symbols.
- **Branch metric computation.**
- **State metric update:** Update the state metric using the new branch metric.
- **Survivor path recording:** Tag the surviving path at each node.
- **Output decision generation:** Generation of the decoded output sequence based on the survivor path information.

Fig.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.

**Fig.2. A branch metric computation block.**

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 Fig.3

**III. PRECOMPUTATION ARCHITECTURE**

A. Pre-computation Algorithm

\[ p\text{o}t(n) = \min\{p\text{o}(n),p\text{l}(n),\ldots\ldots,p\text{o}-1k(n)\} = \min\{\min[p\text{o};0(n-1)+B0,0(n),p\text{l}(n-1)+B0,1(n),\ldots\ldots,p\text{o}(n-1)+B0,p(n)], \min[p\text{o};1(n-1)+B1,0(n),p\text{l}(n-1)+B1,1(n),\ldots\ldots,p\text{o}(n-1)+B1,p(n)], \ldots\ldots, \min[p\text{o};2k-1-1,0(n-1)+B2k-1-1,0(n),p\text{l}(n-1)+B2k-1-1,1(n),\ldots\ldots,p\text{o}(n-1)+B2k-1-1,1(n)]\} \]

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 can be rewritten as $P\text{o}t = \min\{\min(Ps(n-1)) \text{in cluster 1} + \min(Bs(n)) \text{for cluster 1),Min}(Ps(n-1)) \text{in cluster 2} + \min(Bs(n)) \text{for cluster 2), \ldots\ldots, \min(Ps(n-1)) \text{in cluster m} + \min(Bs(n)) \text{for cluster m}\}$. The minimum (Bs) for each cluster can be easily obtained from the BMU or TMU and min(Ps) at time n-1 in each cluster can be precalculated at the same time when the ACSU is updating the new Ps for time n. Theoretically, when we continuously decompose Ps(n-1), Ps(n-2),……..., the precomputation scheme can be extended to Q steps. Where q is any positive integer that is less than n. Hence Pots(n) can be calculated directly from Ps(n-q) in q cycles.

**Fig.3. Topology of precomputation pipelining.**

A. Choosing Precomputation Steps

In 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-step further pre-computation 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 pre-computation steps. In a TCM system, the convolutional code usually has a coding rate of $R/(R+1)$, $R=2,3,4,...$, so that, $p=2R$ and the logic delay of the adder to compute $Ps$ of each candidate
path that reaches the same state and Tp-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 as shown in Fig.4. 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

$$T_{bound} = T_{adder} + T_{in\_comp} + T_{in\_comp}$$

$$q_b = k - 1 \cdot R$$

Fig.4. Rate 3/4 convolutional encoder.

IV. ONE STEP PRECOMPUTATION

For the convenience of our discussion we define the left most register in Fig. 5 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: Popt(n) = min {min (even Ps(n-1)) + min (even Bs(n)), min (odd Ps(n-1)) +min(odd Bs(n)) }.

Fig.5. Viterbi decoder with 1-step pre-computation T-algorithm.

V. TWO STEP PRECOMPUTATION

A. ACSU Design

We again need to analyze the trellis transition of the original code. In the 1-step pre-computation architecture, we have pointed out that for the particular code shown in Fig.3, odd-numbered states are extended by odd Bs, while even-numbered states are extended by even Bs. Furthermore, the even states all extend to states with higher indices (the MSB in Fig. 5 is „1”) in the trellis transition, while the odd states extend to states with lower indices (the MSB is „0” in Fig. 5). This information allows us to obtain the 2-step pre-computation data path. This process is straightforward, although the mathematical details are tedious. For clarity, we only provide the main conclusion here.

B. SMU Design

In this section, we address an important issue regarding SMU design when T-algorithm is employed as shown in Fig.6. 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 Popt), which is always enabled, to output or trace back data. The process of searching for Popt can find out the index of the optimal state as a byproduct. However, when the estimated Popt is used [8], or in our case Popt is calculated from PMs at the previous time slot, it is difficult to find the index of the optimal state.

VI. METHODOLOGY AND RESULTS

- Low-Power High-Speed Viterbi Decoder Design to correctly decode the signal.
- Simulation and Synthesis of circuit
- Verification using test benches
- Comparison of obtained results with available literature
- Modification if required to improve results as shown in Figs.7 to 11.

VII. CONCLUSION AND FUTURE SCOPE

In this paper, a comprehensive analysis of the Viterbi decoder design space is presented. The importance of the different subunits of the decoder depending on the optimization criteria. The PMU is critical for throughput while the SMU is critical for latency and power consumption. So far, this work discusses most of the known VLSI implementation techniques for the hard-decision Viterbi algorithm in standard cell CMOS technology and carefully analyzes the tradeoffs and dependencies between different design decisions. To the best of authors’ knowledge, this is the most comprehensive analysis of hard-decision Viterbi algorithm VLSI implementation based on actual designs including post-layout experiments published.

VIII. REFERENCES

RTL Design & VLSI Implementation of Convolution Encoder & Viterbi Decoder


