Optimized Viterbi Decoder for Convolutional Code

Yunhua Fang

1 Introduction

Convolutional codes are a class of error-correcting codes widely employed in digital communication systems to enhance the reliability of data transmission over noisy channels. These codes function by adding redundancy to the transmitted data, enabling the receiver to detect and correct errors without requiring retransmission. This capability is essential in environments where data integrity is paramount, and retransmission may be impractical or costly. The Viterbi decoder serves as a critical component in the decoding process of convolutional codes. It utilizes the Viterbi algorithm, which operates on the principle of maximum likelihood decoding. This algorithm efficiently identifies the most probable sequence of transmitted data by evaluating the likelihood of all possible paths through a trellis diagram—a graphical representation of the state transitions in the convolutional encoder.

The significance of implementing a Viterbi decoder extends across modern communication and storage systems. By effectively correcting errors introduced by noisy channels, it ensures the integrity and reliability of data, a requirement that is vital in applications such as wireless communication networks, satellite systems, and data storage devices like hard drives. This project centers on the design and implementation of a Viterbi decoder, emphasizing its pivotal role in error correction and examining the trade-offs involved in its realization, including computational complexity and decoding latency. Through this work, we aim to demonstrate how the Viterbi decoder enhances system performance by mitigating the impact of transmission errors.

This report provides a comprehensive overview of the Viterbi decoder for convolutional codes, detailing its theoretical foundations, implementation considerations, and the outcomes of performance evaluations. By exploring these aspects, we seek to highlight the importance of error-correcting codes in achieving reliable data transmission and to illustrate the practical application of the Viterbi algorithm in addressing real-world challenges in communication technology.

2 Background

Convolutional codes are a type of error-correcting code designed to improve the reliability of data transmission over noisy communication channels. Unlike block codes, which divide data into fixed-size blocks for encoding, convolutional codes operate on a continuous stream of data. They introduce redundancy by generating additional bits based on a sliding window of input bits. This process is implemented using shift registers and modulo-2 adders (essentially XOR operations). At any given moment, the output depends not only on the current input bit but also on a set of previous input bits. The number of bits influencing each output is defined by the constraint length of the code. For example, a constraint length of 3 means the current bit and the two preceding bits are used to produce the output. The code rate, such as 1/2, indicates

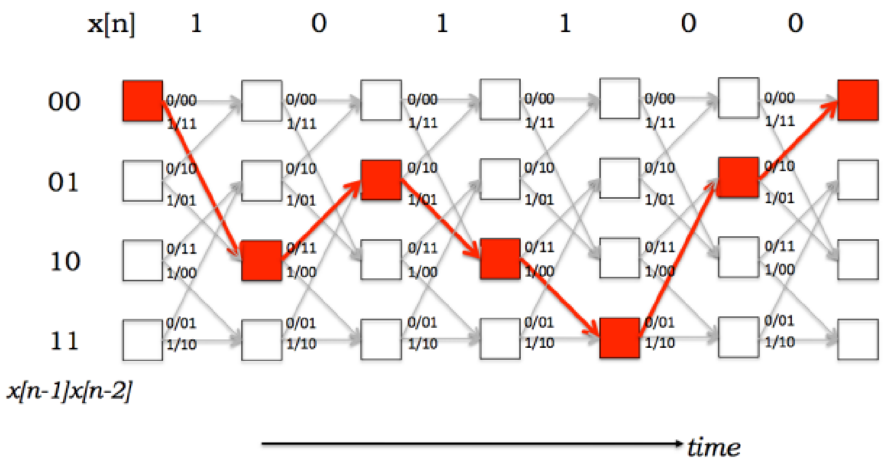


Figure 1: Trellis

the level of redundancy: for every input bit, two output bits are generated. This redundancy allows the receiver to detect and correct errors, making convolutional codes particularly effective for real-time data transmission.

Soft decision decoding is an advanced technique used to decode error-correcting codes like convolutional codes. Unlike hard decision decoding, which interprets received signals as strict binary values (0 or 1) based on a threshold, soft decision decoding takes advantage of the analog information in the received signal—such as signal strength or probability estimates. This method assigns a likelihood or confidence level to each possible bit value, providing a more nuanced view of the data. By incorporating this reliability information, soft decision decoding significantly enhances error correction performance, especially in noisy environments where signal quality varies. Compared to hard decision decoding, it can deliver a coding gain of 2-3 dB, meaning it performs better at lower signal-to-noise ratios. This improvement is critical for applications requiring robust communication, such as satellite systems or wireless networks.

The Viterbi algorithm is a key tool for decoding convolutional codes because it efficiently determines the most likely sequence of transmitted bits from a noisy received signal. Convolutional codes create a complex dependency between input and output bits due to their sliding window approach, making direct decoding computationally challenging. The Viterbi algorithm addresses this by using a trellis diagram, a graphical representation of all possible state transitions of the encoder over time. Each path through the trellis corresponds to a potential sequence of transmitted bits. Instead of evaluating every possible sequence—which would be exponentially complex—the algorithm employs dynamic programming to keep track of only the most promising paths at each step. This reduces computational overhead, making it practical for real-time systems. Additionally, the Viterbi algorithm can integrate soft decision inputs, leveraging the reliability data to improve accuracy. Its combination of efficiency and optimality makes it indispensable for decoding convolutional codes in modern communication systems, ensuring reliable data recovery even under adverse conditions.

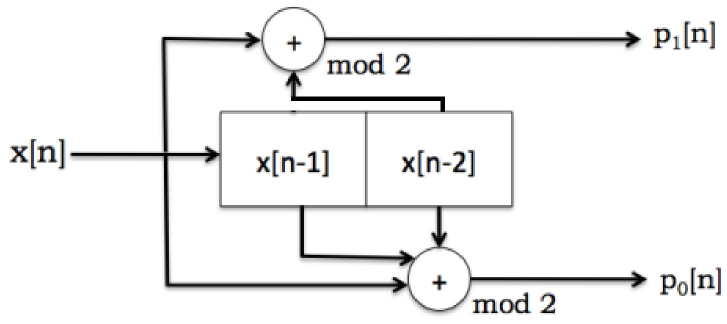


Figure 2: Generator Polynomials

3 Implementation

The Viterbi decoder presented here is designed to decode a convolutional code characterized by a constraint length of (K = 3) and a code rate of ($\frac{1}{2}$). This implies that for each input bit, the encoder produces two output bits, and the decoding process must account for a trellis with four states (since ($2^{K-1} = 2^{3-1} = 4$)). The implementation, crafted in SystemVerilog, emphasizes soft decision decoding to improve error correction in noisy channels. The decoder is built by integrating several key modules—quantizer, branch metric calculation (BMC), Add-Compare-Select (ACS), and survivor path management using the register exchange method—each meticulously designed and composed to achieve high performance and efficiency.

The decoding process begins with the quantizer module, which transforms the incoming 8-bit noisy signals into 3-bit soft decision values. This step is critical as it converts the analog-like received signal into discrete levels that reflect the reliability of each bit. For instance, a value close to 0 suggests high confidence in a logical 0, while a value near 7 indicates confidence in a logical 1. This quantization sets the stage for subsequent metric calculations by providing a compact yet informative representation of the received data.

Following quantization, the BMC module calculates the branch metrics, which measure the likelihood of each possible transition in the trellis. For a code rate of ($\frac{1}{2}$), the encoder outputs pairs of bits (00, 01, 10, 11), and the BMC evaluates how closely the received quantized bits match these expected outputs. The metric for each bit is computed as follows: if the expected bit is 0, the metric is the quantized value (q); if it's 1, the metric is (7 - q). These individual bit metrics are summed for each pair to yield the branch metric. To optimize performance, all eight possible branch metrics (corresponding to transitions from the four states) are computed in parallel, ensuring rapid availability of these values for the next stage.

The ACS module forms the core of the Viterbi algorithm, updating path metrics and selecting the most likely predecessor for each of the four states at every time step. For each state, it performs three steps:

Add: Adds the branch metrics to the path metrics of the predecessor states that could transition to the current state.

Compare: Identifies the smallest resulting path metric, indicating the most likely path.

Select: Chooses the predecessor with the smallest metric and records this decision.

Given the trellis structure for (K = 3), each state has two possible predecessors, and the ACS module processes all four states concurrently. This parallel computation is a key optimization, reducing the time required per symbol and making the decoder suitable for real-time applications. The selected decisions also guide the survivor path updates in the next stage.

To manage survivor paths efficiently, the decoder employs the register exchange (Figure 3) method. In this approach, each of the four states maintains its own survivor path—a sequence of bits representing the most likely input sequence leading to that state. At each time step, the path is updated by appending the bit associated with the transition selected by the ACS module. Unlike the traceback method, which requires revisiting past decisions, register exchange updates paths in real-time, eliminating the need for a separate traceback phase. This reduces decoding latency, though it increases memory usage since each state stores its full path history. The final decoded output is typically taken from the state with the smallest path metric at the end of the sequence.

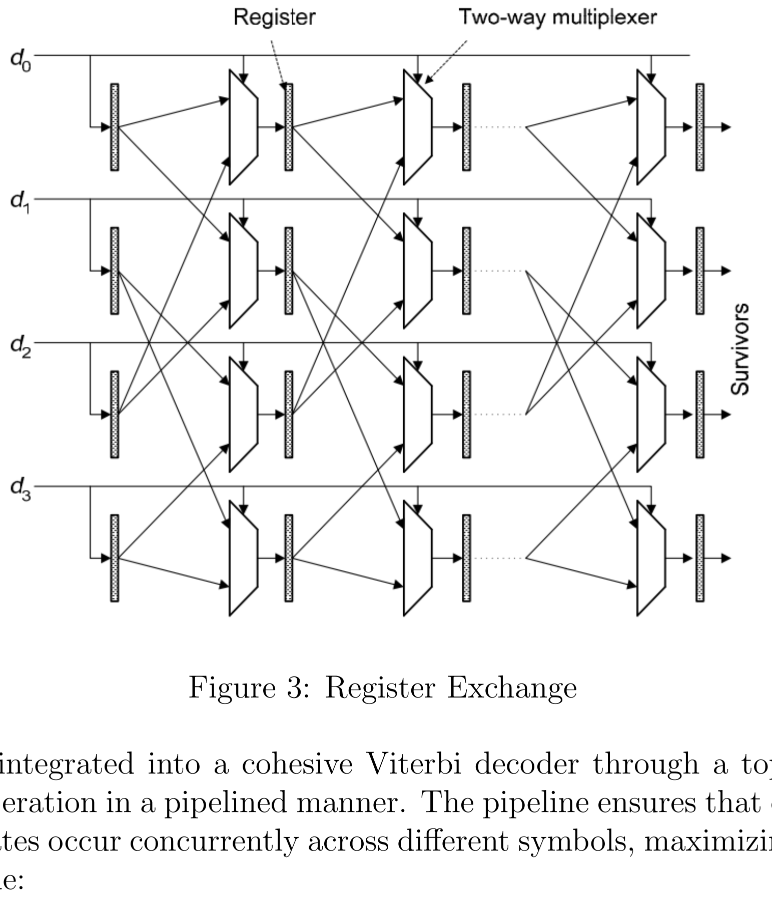
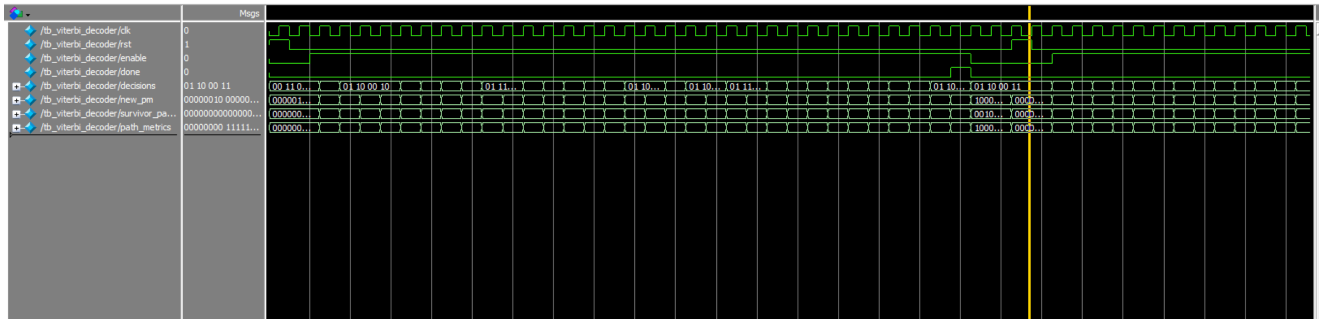


Figure 3: Register Exchange

These modules are integrated into a cohesive Viterbi decoder through a top-level module that orchestrates their operation in a pipelined manner. The pipeline ensures that quantization, BMC, ACS, and path updates occur concurrently across different symbols, maximizing throughput. Key optimizations include:

Parallel Processing: Both BMC and ACS operations are parallelized, minimizing computational bottlenecks.



```
                                        Msgs
/tb_viterbi_decoder/clk                 0
/tb_viterbi_decoder/rst                 1
/tb_viterbi_decoder/enable              0
/tb_viterbi_decoder/done                0
/tb_viterbi_decoder/decisions           01 10 00 11
/tb_viterbi_decoder/new_pm              00000010 00000...
/tb_viterbi_decoder/survivor_pa...      00000000000000...
/tb_viterbi_decoder/path_metrics        00000000 11111...
```

Figure 4: Test signals

Low Latency: The register exchange method avoids traceback delays, making the decoder responsive for time-sensitive applications.

Soft Decision Decoding: Using 3-bit soft values enhances error correction by leveraging signal reliability, outperforming hard decision decoding.

Together, these components and optimizations create a robust and efficient Viterbi decoder, well-suited for digital communication systems requiring reliable data recovery under noisy conditions.

4 Evaluation

To assess the performance and correctness of our Viterbi decoder implementation, we conducted a series of experiments leveraging a robust hardware simulation environment. The evaluation platform consisted of Quartus and ModelSim, two powerful tools used in tandem with SystemVerilog to design, synthesize, and simulate the decoder. Quartus served as the synthesis tool, enabling us to compile our SystemVerilog code into a configuration suitable for a field-programmable gate array (FPGA). For this purpose, we selected the Cyclone V FPGA, a member of a widely adopted FPGA family known for its effective balance of performance and cost. This FPGA provided sufficient logic elements and memory blocks to accommodate the needs of our Viterbi decoder design. We configured the Cyclone V to operate at a clock frequency of 100 MHz, a practical choice that aligns with the demands of real-time data processing in typical communication systems. This setup allowed us to test the decoder's capability to manage high-speed data streams while ensuring accurate error correction.

Following the synthesis in Quartus, we utilized ModelSim to simulate the decoder's behavior under controlled conditions. The simulation process involved generating test input sequences, encoding them with a convolutional code (constraint length K=3, rate 1/2), and introducing noise to emulate a realistic communication channel. We evaluated the decoder with three distinct sequences: an alternating pattern (0xAAAAAAA), a sequence with the first half as ones and the second half as zeros (0xFFFF0000), and a pseudo-random pattern (0x12345678). These sequences were processed by the decoder, and the simulation outputs were analyzed to verify correctness. The results were promising, as the decoder consistently recovered the original sequences by identifying the survivor path with the smallest path metric, a key indicator of the most likely transmitted sequence. The ModelSim logs confirmed that the decoded outputs matched the original inputs after adjusting for bit ordering in the survivor paths, which were reversed to align with the expected format. The path metrics further reinforced the reliability of the decoding decisions, with lower values corresponding to higher confidence in the results. Through these experiments, we validated the Viterbi decoder's functionality and its effectiveness in performing error correction within the simulated FPGA environment.

5 Conclusion

In conclusion, this project successfully implemented a Viterbi decoder for a convolutional code with a constraint length of 3 and a code rate of 1/2, using SystemVerilog and targeting the Cyclone V FPGA. Through careful design and optimization, including parallel Add-Compare-Select operations and the register exchange method for survivor path management, the decoder demonstrated its ability to correct errors and recover original data sequences in simulated noisy environments. This work underscores the critical role of error-correcting codes in modern communication systems, where data integrity is paramount. The project not only validated the theoretical foundations of the Viterbi algorithm but also provided practical insights into its hardware implementation.

Looking ahead, several avenues for improvement and expansion present themselves. Future efforts could focus on extending the decoder to support larger constraint lengths, which would enhance error correction capabilities at the cost of increased computational complexity. Additionally, integrating the decoder into a complete communication system simulation would offer a more comprehensive evaluation of its performance under real-world conditions. Exploring advanced optimizations, such as pipelined traceback or increasing the quantization levels for soft-decision decoding, could further improve the decoder's efficiency and accuracy. This project serves as a solid foundation for understanding and implementing error-correcting codes, and future work could leverage this knowledge to address more complex challenges in digital communications.

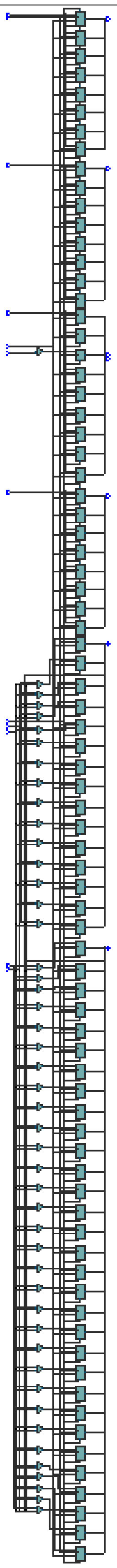


Figure 5: RTL view