Abstract: Convolutional encoding is a forward error correction technique that is used for correction of errors at the receiver end. The two decoding algorithms used for decoding the convolutional codes are Viterbi algorithm and Sequential algorithm. Sequential decoding has advantage that it can perform very well with long constraint length. Viterbi decoding is the best technique for decoding the convolutional codes but it is limited to smaller constraint lengths. It has been widely deployed in many wireless communication systems to improve the limited capacity of the communication channels. The Viterbi algorithm is the most extensively employed decoding algorithm for convolutional codes. The availability of wireless technology has revolutionized the way communication is done in our world today. With this increased availability comes increased dependence on the underlying systems to transmit information both quickly and accurately. Because the communications channels in wireless systems can be much more hostile than in “wired” systems, voice and data must use forward error correction coding to reduce the probability of channel effects corrupting the information being transmitted. The Viterbi Algorithm, an application of dynamic programming, is widely used for estimation and detection problems in digital communications and signal processing. It is used to detect signals in communications channels with memory, and to decode sequential error control codes that are used to enhance the performance of digital communication systems.

Though various platforms can be used for realizing Viterbi Decoder including Field Programmable Gate Array (FPGAs), Complex Programmable Logic Devices (CPLDs) or Digital Signal Processing (DSP) chips but in this project benefit of using an FPGA to Implement Viterbi Decoding Algorithm has been described. FPGAs are a technology that gives the designer flexibility of a programmable solution, the performance of a custom solution and lowering overall cost. The advantages of the FPGA approach to DSP Implementation include higher sampling rates than are available from traditional DSP chips, lower costs than an ASIC. The FPGA also adds design flexibility and adaptability with optimal device utilization conserving both board space and system power that is often not the case with DSP chips.

Keywords: Convolutional encoding, Viterbi decoder, Path Metric, branch metric, FPGA, Xilinx, Modelsim.
1. Introduction:

In digital communication system, error detection and error correction is important for reliable communication. Error detection techniques are much simpler than forward error correction (FEC). But error detection techniques have certain disadvantages. Error detection presupposes the existence of an automatic repeat request (ARQ) feature which provides for the retransmission of those blocks, segments or packets in which errors have been detected.

This assumes some protocol for reserving time for the retransmission of such erroneous blocks and for reinserting the corrected version in proper sequence. It also assumes sufficient overall delay and corresponding buffering that will permit such reinsertion. The latter becomes particularly difficult in synchronous satellite communication where the transmission delay in each direction is already a quarter second. For, as the error rate approaches the packet length, the majority of blocks will contain detected errors and hence require retransmission, even several times, reducing the throughput drastically. Forward error correction may be desirable in place of, or in addition to, error detection for any of the following reasons:

i. When a reverse channel is not available or the delay with ARQ would be excessive.

ii. There transmission strategy is not conveniently implemented.

Keeping in view requirements of communication channels in 3G wireless systems, need of reliable data communication, fast as well as accurate is the main consideration. So main work in this Paper is to develop an algorithm for a Viterbi Decoder to decode an encoded data.

2. Background:

Error correction coding [9] can be used to detect and correct data transmission errors in communication channels. Encoding is accomplished through the addition of redundant bits to transmitted information symbols.

These redundant bits provide decoders with the capability to correct transmission errors. Convolutional codes [7] form a set of popular error-correction codes. In convolutional coding, the encoded output of a transmitter (encoder) depends not only on the set of encoder inputs received during a particular time step, but also on the set of inputs received within a
previous span of K-1 time units, where K is greater than 1. The parameter K is the constraint length of the code.

A typical convolutional encoder of constraint length $K = 3$ is shown in Figure 1. As shown in the figure, the encoding of convolutional codes can be accomplished with shift registers and generator polynomials (XOR functions). A convolutional encoder is represented by the number of output bits per input bit ($v$), the number of input bits accepted at a time ($b$), and the constraint length ($K$), leading to representation ($v, b, K$). Figure 1 depicts a (2, 1, 3) convolutional encoder since the encoder accepts one input bit per time step and generates two output bits. The two output bits are dependent on the present input and the previous two input bits. The constraint length $K$ indicates the number of times each input bit has an effect on producing output bits. Larger constraint lengths, i.e. $K = 9$ or higher, are preferable since they allow for more accurate error correction. Encoding rate $R$ is equal to $b/v$. In many communication systems, a rate of 1/2 is used [9]. Initially, the contents of the encoder shift register are set to zero. The contents are shifted right each time a one-bit value is converted into a two-bit symbol and transmitted. The operation of the encoder can be represented by a state diagram, as shown in Figure 2. Nodes represent the present state of the shift register while edges represent the output sequence and point to the next state of transition. Successive evaluation of state over time leads to the trellis diagram shown in Figure 3.

The diagram is a time-ordered mapping of encoder state with each possible state represented by a point on the vertical axis. Nodes represent the present state of the shift register at specific points in time while edges represent the output sequence and point to the next state of transition. The horizontal axis represents time steps. Branch lines indicate the transition of the present state of the shift register to the next state upon receiving a particular input bit, $b$. The upper branch leaving a node implies an input of 0 while the lower branch implies an input of 1.

![Figure 1: A(2,1,3) Convolution Encoder](image_url)
The function of the decoder is to attempt to reconstruct the input sequence transmitted by the encoder by evaluating the received channel output. Values received at the decoder may differ from values sent by the encoder due to channel noise. The interaction between states represented by the trellis diagram is used by a decoder to determine the likely transmitted data sequence [16] as \( n \)-bit symbols are received. An example received sequence of two-bit \( n \) values appears at the top of Figure 3. The cost of a particular transition edge (branch) is determined from the Hamming distance of the received symbol and the expected symbol, labelled in bold on the transition edge. At each node the cumulative cost or path metric of the path is determined. These values are labeled in bold at each node in the figure. If multiple paths converge on the same state, the lowest cost path is preserved and other paths are eliminated. After a series of time steps, referred to as the truncation length (TL), the lowest-cost path, also known as minimum distance path, is determined, identifying the most-likely transmitted symbol sequence. The typical value of the truncation length depends on the noise in the channel and has been empirically found to be 3-5 times the constraint length [9]. Each path in the trellis diagram represents a unique set of inputs, such as the path highlighted in bold, corresponding to the lowest-cost input sequence \( b = (0110) \).

The performance of a decoder is characterized by the number of decoded output bits which are in error, the Bit Error Rate or BER. The BER is the ratio of the number of bits in error to the total number of bits transmitted. For communication fidelity it is desirable to achieve a low BER. For this paper, a typical, maximum BER of \( 10^{-5} \) [9] is targeted.
The Viterbi algorithm [9], the most popular decoding approach for convolutional codes, determines a minimum distance path with regards to Hamming distances applied to each received symbol.

A limiting factor in Viterbi decoder implementations is the need to preserve candidate paths at all \(2^{K-1}\) trellis states for each received symbol. This requirement leads to an exponential growth in the amount of computation performed and in the amount of path storage retained as constraint length \(K\) grows.

Most hardware implementations of the Viterbi algorithm [9] are split into three parts: the branch metric generators (BMG), add-compare-select (ACS) units, and the survivor memory unit.

A BMG unit determines Hamming distances between received and expected symbols. An ACS unit determines path costs and identifies lowest-cost paths. The survivor memory stores lowest cost bit-sequence paths based on decisions made by the ACS units.

### 3. Adaptive Viterbi Algorithm:

The adaptive Viterbi algorithm [4] was introduced with the goal of reducing the average computation and path storage required by the Viterbi algorithm. Instead of computing and retaining all \(2^{K-1}\) possible paths, only those paths which satisfy certain cost conditions are retained for each received symbol at each state node. Path retention is based on the following criteria.

i. 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.

ii. The total number of survivor paths per trellis stage is limited to a fixed number, \(N_{max}\), which is pre-set prior to the start of communication.

The first criterion allows high-cost paths that likely do not represent the transmitted data to be eliminated from consideration early in the decoding process. In the case of many paths with similar cost, the second criterion restricts the number of paths to \(N_{max}\).
A trellis diagram for an adaptive Viterbi algorithm of constraint length 3 is shown in Figure 4 for the same set of received symbols as shown in Figure 3. Threshold value $T = 1$ is preset for this example.

![Figure 4: Trellis diagram for a hard-decision adaptive Viterbi decoder with $T = 1$ and $N_{\text{max}} = 3$](image)

At each stage, the minimum cost (path metric) of the previous stage $d_m$, threshold $T$, and maximum survivors $N_{\text{max}}$ are used to prune the number of surviving paths. Initially, at $t=0$, the decoder state is set to 00. Like the Viterbi trellis, two branches emanate from state 00 to states 00 and 10 at $t=1$ representing encoded transmission 0 and 1 values respectively by the encoder. If the received value at $t=0$ is 00, as shown in the trellis, it is more likely that a $b = 0$, $v = 00$ was transmitted rather than a $b = 1$, $v = 11$ value since both bits of the latter $v$ would have been corrupted by noise. As a result, the path metric of the top branch is 0 and the bottom branch is 2. These are the Hamming distances between the received and expected values shown on the trellis. Since state 00 is the only state at $t=0$, $d_m$ is the path metric of state 00, which is 0. As a result, $d_m + T$ is 1. At $t=1$, the path leading to state 10 does not survive since 2, the current path metric of state 10, is greater than 1, the value of $d_m + T$. As a result, only one branch, the branch leading to state 00 survives at $t=1$. The new $d_m$ used at $t=2$ is the minimum among metrics of all surviving paths at $t=1$. Since only one path survives at $t=1$, $d_m$ is 0, the path metric of state 00. At each stage the process is repeated until the truncation length is reached and the least error path can be identified.

Careful calculation of $T$ and $N_{\text{max}}$ is the key to effective use of the AVA algorithm. If threshold $T$ is set to a small value, the average number of paths retained at each trellis stage will be reduced. This can result in an increased BER since the decision on the most likely path has to be taken from a reduced number of possible paths. Alternately, if a large value of $T$ is selected, the average number of survivor paths increases and results in a reduced BER. As a result, increased decode accuracy comes at the expense of a additional computation and
a larger path storage memory. Generally, the value of $T$ should be selected so that the BER is within allowable limits while matching the resource capabilities of the hardware.

$N_{\text{max}}$ denotes the maximum number of survivor paths to be retained at any trellis stage. The maximum per-trellis stage number of survivor paths, $N_{\text{max}}$, has a similar effect on BER as $T$. If a small value of $N_{\text{max}}$ is chosen, paths which satisfy the threshold condition may be discarded, potentially leading to a large BER. Alternately, if $N_{\text{max}}$ is set to a large value, extra computation and memory are required, potentially with little benefit to BER reduction. As a result, an optimal value for $N_{\text{max}}$ should be chosen to balance hardware size and BER. The M-algorithm [6] is a popular reduced-complexity alternative to the Viterbi algorithm. Like the AVA, complexity reduction is achieved by retaining only the best $M (N_{\text{max}})$ paths at each trellis stage. Unlike AVA, this approach does not use a threshold condition to determine which paths are saved but rather sorts all paths and retains the $M$ lowest-cost paths. This requirement of sorting circuitry adds complexity and delay to the M-algorithm. Since the AVA only requires comparison to a value determined by the metric of the best extended

![Figure 5: Adaptive Viterbi decoder architecture](image)

The Add-Compare-Select (ACS) unit, shown in detail in Figure 6, evaluates the path metric of each path and determines if paths meet AVA conditions for path survival. At each trellis stage, path, sorting is not required. A beam-search algorithm, which is similar to the AVA, was implemented in software in [10]. This approach was used in conjunction with hidden Markov modelling (HMM) of speech.

Unlike previous AVA approaches [4], the standard operation of eliminating the largest-metric path when two survivor paths enter the same trellis state was not implemented in our approach due to hardware complexity. The implemented algorithm more closely resembles a variant of the AVA known as the Simmons T-algorithm [11].
4. Ava Architecture:

To demonstrate the benefit of the adaptive Viterbi algorithm we have developed the first hardware implementation of the algorithm. This architecture takes advantage of parallelization and specialization of hardware for specific constraint lengths and dynamic reconfiguration to adapt decoder hardware to changing channel noise characteristics.

4.1. Description Of The Architecture:

The architecture of the implemented adaptive Viterbi decoder is shown in Figure 5 for the encoder with parameters shown in Figure 1, (2, 1, 3). The adaptive Viterbi decoder accepts two inputs from the channel which represent the outputs of the encoder that have been transmitted. This difference is the Hamming distance between the values. A total of $2^v$ branch metrics are determined by the branch metric generator. For $v=2$ these metrics are labelled $b_{00}$, $b_{01}$, $b_{10}$ and $b_{11}$.

![Figure 6: ACS unit of adaptive Viterbi decoder](image)

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. As shown at the left of Figure 6, the path metrics for all potential next state paths, $di$, are computed by the ACS unit. 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. Present and next state values for the trellis are stored in two column arrays, Present state and NEXTSTATE of dimensions $N_{max}$ and $2N_{max}$ respectively, as shown in Figure 5. There can
be at the most $N_{max}$ survivor paths at any stage. Since each path is associated with a state, the number of present states is $N_{max}$. Each path can potentially create two child paths before pruning as there are two possible branches for each present state based on a received 0 or 1 symbol. Entries in the NEXTSTATE array need not be in the same row as their respective source present states.

Once the paths that meet the threshold condition are determined, the lowest-cost $N_{max}$ paths are selected. To avoid the need for the sorting circuit described in [6] for the M-algorithm, we have developed a novel path pruning approach. Sorting circuitry is eliminated by making feedback adjustments to the parameter $T$. If the number of paths that survive the threshold is less than $N_{max}$, no sorting is required. For stages when the number of paths surviving the threshold condition is greater than $N_{max}$, $T$ 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 $N_{max}$. In Section 7 it is shown that $T$ and $N_{max}$ can be determined through simulation so that $T$ reduction is needed infrequently. Following path reduction, at most $N_{max}$ remaining trellis states are stored in the Present state array in preparation for the receipt of the next symbol.

The register-exchange based survivor memory [12] stores path sequence information and has a two dimensional size of $N_{max} \times TL$, where $TL$ is the truncation length. Each memory location stores an input bit, the decision bit from the ACS. Single-bit storage is performed for each surviving path at each trellis stage. Each row of the survivor memory is associated with a present state and has a valid bit to indicate the existence of a survivor path. Once survivor memory storage reaches the truncation length, the lowest-cost path sequence can be retrieved.

### 4.2. Architectural Model:

The logic area in terms of logic blocks of our adaptive Viterbi decoder can be characterized by an empirical model in terms of parameters $N_{max}$ and $K$. This expression is of the form:

$$Area = AN_{max} + BKN_{max} + C$$

where $A$, $B$, $C$ are constant coefficients. These coefficients were determined by evaluating a set of decoders with constraint lengths between 4 and 14. Through line-fitting, coefficients for $A$, $B$, and $C$ were determined to be 90, 5.6, and 215, respectively.

The first term in the equation accounts for logic blocks which implement path metric comparators and the registers used to store path metrics. The second term accounts for path storage in memory implemented inside the FPGA. Since the truncation length ($TL$) is $5 \times K$
[9], the term depends on both the maximum number of bits stored per trellis state, \( N_{\text{max}} \), and \( K \). Additionally, the width of present state and next state registers increases linearly with \( K \) and the number of registers increases linearly with \( N_{\text{max}} \). The constant term in the expression accounts for logic which is fixed in size in relation to \( N_{\text{max}} \) and \( K \). This includes the branch metric generator, which is dependent on the parameter \( v \) defined in Section 2 and the logic needed to iteratively decrement \( T \) to avoid sorting.

4.3. Suitability Of Dynamic Reconfiguration:

Dynamic reconfiguration of the entire adaptive Viterbi decoder hardware is considered as a means to enhance performance without compromising decode accuracy. The hardware resource requirements of an adaptive Viterbi decoder change in response to channel noise conditions for a fixed \( \text{BER} \).

Based on noise levels at a specific time instant, minimally sufficient hardware resources can be dynamically allocated to meet the \( \text{BER} \) requirements of the application while achieving maximal performance [3]. A significant amount of channel noise demands a large constraint length, such as \( K = 14 \), to achieve a \( \text{BER} \) similar to that achieved by a constraint length \( K = 4 \) for a less noisy channel. It is shown in Section 5 that decoding speed is inversely related to resource requirement. This relationship is exploited to enhance performance through the dynamic allocation of AVA logic resources.

In implementing the AVA, two reconfiguration options, \textit{fine-timescale} and \textit{coarse-timescale} reconfiguration are considered. Coarse-timescale reconfiguration of the adaptive Viterbi decoder, based on parameters such as \( K, T \) and \( N_{\text{max}} \), is performed in accordance to variations in channel noise conditions over seconds.

Reconfiguration at this time scale minimizes the performance impact of millisecond FPGA reconfiguration times. Coarse-timescale reconfiguration is motivated by changing channel noise characteristics from parameters such as weather, distance, or battery-power. These parameters result in a signal-to-noise ratio (\( \text{SNR} \)) that changes relatively slowly (seconds or longer).

When more accurate decoding is required, a lower clock-speed decoder (larger \( K \)) can be used at the cost of reduced decode rate. When less accurate decoding is required, a higher-performance decoder is swapped in. If dynamic reconfiguration was not allowed, the lower-performance decoder would always need to be resident. Coarse-timescale reconfiguration
provides an optimized but variable bit rate and is targeted at data rather than voice applications.

![Figure 7: Number of surviving paths versus decoded symbol for the adaptive Viterbi algorithm for K = 9, r = ½ convolutional code with Nmax set to 2^{K-1} (its maximum) on an unfaded channel](image)

In fine-timescale reconfiguration, configuration contents are changed once every one or two decoded bits (ms) in response to the number of survivor paths retained at specific instants of time. If additional survivor paths are required, the amount of required path storage increases, potentially limiting decoder performance. After experimentation it was determined that fine-timescale reconfiguration is infeasible due to rapid variations (multiple changes per ms) in retained survivor paths over time. Current FPGA architectures \[1\] \[15\] require reconfiguration times measuring milliseconds. Figure 7 shows survivor path variation for a constraint length 9 decoder.

5. Results And Discussions:

Viterbi test Bench is created using Xilinx Webpack. Code is written in Verilog HDL. There are different Modules for the code and that are Viterbi test bench, Viterbi, Viterbi distance, reduce, path, path memory, compute metric, dff, ACS enable, back and compare select. To prove the correctness of our design, the verilog HDL description was simulated & tested using Model Sim Verilog Simulator. Afterwards Xilinx Webpack is used for design entry, synthesis, place & route and floor plan design. What is inputted to the test bench come from the Viterbi Encoder for K=3, rate= ½ (111,101). For this Example the original data is: 010111001010001 + 00 (2 tail bits). The Encoder output is: 00 11 10 00 01 10 01 11 11 10 00 10 11 00 11 + 10 11. We may manually introduce error (s) into decoder input by altering the data (s).
Device selected is 2v40fg256-6.

Number of Slices: 86 out of 256 33%

Number of Slice Flip Flops: 62 out of 512 12%

Number of 4 input LUT’s: 131 out of 512 25%

Number of bonded IOB’s: 10 out of 88 11%

Number of GCLK’s: 1 out of 16 6%

Figure 8 describes simulation waveforms of viterbi decoder. In this all values displayed are in nano seconds (ns). Figure 9 gives Symbol of a Viterbi Decoder. From the figure it is clear that in this viterbi decoder there are six inputs & one output. The inputs are in0, in1, in2, in3, clk & reset & the output is decode out. Figure 10 gives RTL Schematic of Viterbi Decoder.

![Simulation Waveform of Viterbi Decoder](image1)

**Figure 8: Simulation Waveform of Viterbi Decoder**

![Symbol of Viterbi Decoder](image2)

**Figure 9: Symbol of Viterbi Decoder**

![RTL schematic for Viterbi Decoder](image3)

**Figure 10: RTL schematic for Viterbi Decoder**
6. Conclusion And Future Scope:

Viterbi Algorithm is widely used for the elimination of the potential noise in a data stream. Encoding is such that the Viterbi Decoder can remove potential noise in the incoming stream by decoding it. Its streamed input-output, regular architecture and parallel execution favor on an FPGA Implementation. FPGAs open a wide range of opportunities in the solution space that can result in high performance and economic solutions to a DSP problem because they do not map well to software programmable DSP architectures. The algorithm will have an ASIC solution but this may not be an option for reasons of schedule, economics of scale and flexibility. Applications such as data communications and image processing require more processing power but when the fastest DSP processor is not fast enough, the only alternatives are to add multiple DSP processors or to design custom hardware devices.

Multiple DSP processors are expensive, require many components and consume too much power. The performance gain that comes with each additional processor is small when compared to the increase in cost, board space, power consumption, and development time. Custom devices deliver the performance but sacrifice flexibility and require a large engineering investment with no chance to recover from mistakes. FPGAs are the new solution used by many engineers to implement computationally intensive algorithms.

FPGAs offer the best of all the worlds, the flexibility of a programmable solution, the performance of a custom solution, and lowering overall cost. So keeping in view of the advantages of using FPGAs as compared to other customized devices this work can further be increased to the designing & implementation of a generic Viterbi decoder. Some of the generic parameters are basic decoder specifications, metric size, trellis window length, number of surviving paths and pipeline depth. A new Viterbi decoder with new specifications can be realized by only re-synthesizing the code.
References: