Design Trade-offs in the VLSI Implementation of High-Speed Viterbi Decoders and their Application to MLSE in ISI Cancellation

Jelena Dragaš
jelena.dragas@epfl.ch
March 24, 2011

Advisors: Andreas Burg (TCL-EPFL) andreas.burg@epfl.ch
Christoph Roth (IIS-ETHZ) rothc@iis.ee.ethz.ch
Alessandro Cevrero (LSM-EPFL) alessandro.cevrero@epfl.ch
Abstract

A 64 state fully parallel Viterbi decoder in 90 nm CMOS technology compatible with WLAN IEEE 802.11 standard is described. Optimisation of different decoder architectures ranging from radix-2 to radix-16 for reaching multi-Gb/s throughput is performed and trade-offs w.r.t. energy efficiency and area are laid out. Best simulated throughput after placement and routing reaches the value of 3.1Gb/s, corresponding to 1.03 GHz operating frequency. In addition, Viterbi decoder implementing MLSE for ISI cancellation in a POF-based data transmission system is described and implemented in 90nm CMOS technology.
# Contents

1 Introduction 1

2 Viterbi Algorithm 3
   2.1 Overview .................................................. 3
   2.2 Implementation of the Algorithm ......................... 5
   2.3 Application Examples ..................................... 6
      2.3.1 Convolutional Codes Decoding ....................... 6
      2.3.2 Maximum Likelihood Sequence Estimation in ISI cancellation . . . 10

3 Mapping to Hardware 13
   3.1 System Overview .......................................... 13
   3.2 Radixes .................................................... 14
   3.3 Data Windowing ........................................... 15
   3.4 Data Quantisation ......................................... 18
      3.4.1 Modulo Normalisation ................................ 18
      3.4.2 Fix-point Metrics Representation .................... 20
   3.5 Viterbi Decoder - Reference Architecture ............... 23
      3.5.1 Branch Metrics Unit (BMU) ........................... 24
      3.5.2 Add-Compare-Select Unit (ACS) ....................... 25
      3.5.3 Register Exchange Unit (RE) ......................... 27

4 Evaluation for High Speed 31
   4.1 Performance Metrics and Voltage Scaling Model .......... 31
   4.2 Evaluation Flow ............................................ 32
   4.3 Evaluation of Different Radix Architectures ............... 34
   4.4 Data Interleaving .......................................... 37
   4.5 VLSI Implementation Results for Radix-2, -4, -8 and -16 Architectures after Interleaving ............................. 38
      4.5.1 Throughput ............................................. 39
      4.5.2 Energy Efficiency ...................................... 40
      4.5.3 Area .................................................... 42
   4.6 Conclusion and Comparison to Related Studies ............. 43
5 Application of a Viterbi Decoder in a POF Data Transmission System 49
  5.1 System Model ......................................................... 49
  5.2 Performance Evaluation for Different System Parameters ................. 51
  5.3 Hardware Mapping and Synthesis Results .................................. 54
  5.4 Conclusion .............................................................. 56

6 Conclusion ................................................................. 61

A Block Diagrams and Chip Interface ...................................... 63
  A.1 Handshake Interface .................................................. 63
  A.2 Block Diagrams ....................................................... 65

B Project task ............................................................... 71

C Presentation .................................................................. 79

Bibliography .................................................................. 102
## List of Figures

2.1 System model for Viterbi algorithm application. .......................... 4
2.2 Trellis diagram for a four-state process. ................................. 4
2.3 Convolutional encoder implementing WLAN IEEE 802.11 standard .... 7
2.4 Data transmission system employing Viterbi decoder for convolutional codes decoding. ......................................................... 7

3.1 Data transmission system implementing Viterbi decoder. ............... 13
3.2 a) 8-state radix-2 trellis, b) 4-state subtrellis decomposition, c) 8-state radix-4 trellis 14
3.3 Trellis diagram showing the convergence of the survivor paths. ........ 15
3.4 Simulation results of $Eb/N_0$ sweep for different code-rates. .......... 16
3.5 Traceback length sweep for 1/2 code-rate on a radix-2 architecture, with a fixed SNR value. ......................................................... 17
3.6 Survivor path length sweep for 5/6 punctured coding at a constant SNR value. 18
3.7 Graphical example of modulo normalisation. ............................... 19
3.8 Hardware realisation of modulo normalisation. ............................. 20
3.9 Fixed-point representation of metrics in the design. ....................... 21
3.10 Number of integer bits for fixed-point representation of input LLR data. 22
3.11 Number of fractional bits for fixed-point representation of input LLR data. 23
3.12 Viterbi decoder building blocks. ............................................ 24
3.13 Branch metrics unit, radix-4 architecture. ................................. 25
3.14 Add-Compare-Select unit, radix-4 architecture. ........................... 26
3.15 ACS module of ACS unit, radix-4. .......................................... 28
3.16 Register Exchange, radix-4. ................................................... 29

4.1 Area distribution in a radix-4 design. ....................................... 36
4.2 Throughput - energy-efficiency trade-off for different radix architectures. 37
4.3 Hardware efficiency for different radix architectures. .................... 38
4.4 Voltage scaling of different radix architectures. ........................... 39
4.5 Pipelining stages in ACS module; a) starting module structure, b) module structure after introducing one pipeline stage. ....................... 40
4.6 The impact of interleaving on the throughput for different radix architectures. 41
4.7 The impact of interleaving on energy efficiency. ........................... 42
4.8 Power distribution in the three-stage interleaved radix-8 architecture. 43
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.9</td>
<td>Throughput — energy-efficiency trade-off in radix-8 architecture with three-stage interleaving.</td>
<td>44</td>
</tr>
<tr>
<td>4.10</td>
<td>The impact of interleaving on area.</td>
<td>45</td>
</tr>
<tr>
<td>4.11</td>
<td>Area distribution in the three-stage interleaved radix-8 architecture.</td>
<td>45</td>
</tr>
<tr>
<td>4.12</td>
<td>Throughput — area trade-off in radix-8 architecture with three-stage interleaving.</td>
<td>46</td>
</tr>
<tr>
<td>5.1</td>
<td>POF-based data transmission system.</td>
<td>50</td>
</tr>
<tr>
<td>5.2</td>
<td>POF’s impulse response for $f_s = 1GHz$ and $L = 25m$.</td>
<td>52</td>
</tr>
<tr>
<td>5.3</td>
<td>POF’s impulse response for $f_s = 1GHz$ and $L = 100m$.</td>
<td>53</td>
</tr>
<tr>
<td>5.4</td>
<td>BER as a function of $Eb/N_0$ for $f_s = 1GHz$ and 64 states Viterbi decoder, without metrics quantisation.</td>
<td>54</td>
</tr>
<tr>
<td>5.5</td>
<td>BER as a function of $Eb/N_0$ for $f_s = 1GHz$ and 1024 states Viterbi decoder, without metrics quantisation.</td>
<td>55</td>
</tr>
<tr>
<td>5.6</td>
<td>BER as a function of $Eb/N_0$ for different sampling rates and 64 states Viterbi decoder, without metrics quantisation.</td>
<td>57</td>
</tr>
<tr>
<td>5.7</td>
<td>Sweep over fractional part width for branch metric.</td>
<td>58</td>
</tr>
<tr>
<td>5.8</td>
<td>Sweep over fractional part width for input signal metric.</td>
<td>58</td>
</tr>
<tr>
<td>5.9</td>
<td>Sweep over fractional part width for input signal metric.</td>
<td>59</td>
</tr>
<tr>
<td>A.1</td>
<td>Handshake protocol — Single-data transfer example.</td>
<td>64</td>
</tr>
<tr>
<td>A.2</td>
<td>Handshake protocol — Multiple-data transfer example.</td>
<td>64</td>
</tr>
<tr>
<td>A.3</td>
<td>Viterbi decoder, detailed structure.</td>
<td>65</td>
</tr>
<tr>
<td>A.4</td>
<td>Branch metric unit, detailed structure.</td>
<td>66</td>
</tr>
<tr>
<td>A.5</td>
<td>Add-Compare-Select unit, detailed structure.</td>
<td>67</td>
</tr>
<tr>
<td>A.6</td>
<td>Add-Compare-Select module, detailed structure.</td>
<td>68</td>
</tr>
<tr>
<td>A.7</td>
<td>Register exchange unit, detailed structure.</td>
<td>69</td>
</tr>
</tbody>
</table>
List of Tables

3.1 Radix-$2^l$ complexity and speed evaluation. ........................................... 15
4.1 Performance comparison of the different radix architectures. .......................... 34
4.2 Overview of the articles on Viterbi decoder. ................................................. 47
5.1 Overview of the maximal lengths of POF for different number of Viterbi states. 56
Chapter 1

Introduction

Throughout the last few decades there exists a growing need for reliable data transmission systems. Even though communication devices are becoming more and more sophisticated and resistant to external interferences, with increasing signal frequencies and tendencies to decrease devices’ power consumption and dimensions, many issues of data transmission become more apparent. The increase of signaling frequencies may lead to significant signal distortion in the systems with already limited bandwidth. Construction of a low-power system may involve lowering the signal levels, thus making them more vulnerable to noise originating in the transmission system, making the task of performing a reliable data transfer even more challenging. This is why many error detection and error correction methods have been developed in order to ensure a satisfiable quality of data transmission, i.e. satisfiable data reliability at the receiver’s end of the system.

One of the methods used for this purpose is a so-called forward error correction method which is based on insertion of a known structure into the data sequence prior to transmission. An example of such system is presented in this report. The system is based on a convolutional code, according to a Wireless LAN IEEE 802.11 standard. Here, a module called Viterbi decoder implementing the Viterbi algorithm [1] [2] is used for data decoding and error correction. A hardware mapping of this algorithm and its physical-level design are described. According to current demands for multi-Gb/s throughput data transmissions, priority is given to obtaining high data throughput. At the same time, the trade-off between the data throughput, energy efficiency and area of the design is explored using an automated reconfigurable design-evaluation script.

In certain data transmission systems, where channel is known to introduce Intersymbol Interference (ISI), if the channel behaviour is known, it is possible to map the task of neutralising the signal distortion to the aforementioned problem of convolutional codes decoding. One such implementation of the Viterbi algorithm is described in the continuation of this report. Here a method called Maximum Likelihood Sequence Estimation (MLSE) in combination with Viterbi algorithm is used with purpose of performing ISI cancellation in a transmission system based on Plastic Optical Fiber (POF).
The first part of this report (Chpt. 2) contains a theoretical background on the Viterbi algorithm and different implementation approaches. A general overview of two different Viterbi algorithm applications — convolutional codes decoding and ISI cancellation based on MLSE are described. In the following chapter (Chpt. 3), mapping of the algorithm to hardware is described. Different issues of hardware mapping, such as data windowing and data quantisation are dealt with in Sec. 3.3 and Sec. 3.4, respectively. Following this, one of the basic architectures of Viterbi decoder is presented in Sec. 3.5. Chpt. 4 contains the evaluation of the results obtained after VLSI implementation of different Viterbi decoder architectures, and the optimisations performed in order to obtain high throughputs. At the same time, the trade-off between throughput, energy efficiency and area is explored. First the performance metrics are introduced which is followed by description of the automatized design estimation flow Sec. 4.2. After this, various optimisation methods are introduced and finally the results of the VLSI implementation are presented in Sec. 4.5. In the final chapter (Chpt. 5) the implementation of the Viterbi algorithm to ISI cancellation in a POF-based data transmission system is presented. First an overview of a POF-based data transmission system is given in Sec. 5.1 which is followed by an exploration of different system parameters based on a Matlab model (Sec. 5.2). Finally, the hardware mapping of the Viterbi algorithm implementing MLSE is given in Sec. 5.3 based on which design synthesis results are presented.
Chapter 2

Viterbi Algorithm

2.1 Overview

Many problems in digital data transmission can be stated in the following manner: Given a sequence of events observed at the output of a memoryless data transmission channel, sequence of events at the input of the channel causing these observations needs to be determined. An optimal solution for cases where the data going into the channel is reflecting a state sequence of a finite-state discrete-time Markov process is Viterbi algorithm [1], [2] which is based on the maximum a posteriori probability (MAP) estimation of a state sequence in Markov process. Markov process can be represented as a shift register of length \( \nu \), with inputs labeled as \( u_n \) at time \( n \), with \( m \) possible values which follow some probability distribution \( P(u_n) \), as depicted in Fig. 2.1. This constellation of parameters describes \( \nu \)-th order, \( m \)-ary Markov process. The contents of the shift register at time \( n \) \((u_{n-1}, u_{n-2}, \ldots, u_{n-\nu})\) determines the current state of the process, \( x_n \). If state sequence up to the time \( n \) is known and is represented by a vector \( x = (x_0, x_1, \ldots, x_n) \), where \( x_i \) represents a state at time \( i \), \( x_i \in \{s_0, s_1, \ldots, s_S-1\} \) and \( S \) is a finite number, the process is called Markov if the probability of it being in a state \( x_{n+1} \) at time \( n + 1 \) depends only on the state \( x_n \) at time \( n \):

\[
P(x_{n+1}|x) = P(x_{n+1}|x_n).
\]

Let us note the transition between states at time \( n \) and \( n + 1 \) as \( \xi_n = (x_{n+1}, x_n) \). Since the number of different states is \( S \), the overall number of possible transitions is \( N_\xi \leq S^2 \). There is a one-to-one correspondence between the state sequence \( x \) and the transition sequence \( \xi = (\xi_0, \xi_1, \ldots, \xi_{n-1}) \), i.e. \( x \leftrightarrow \xi \). Having this correspondence in mind, it follows that the transition \( \xi_n \) corresponds to the current contents of the shift register \((u_n, u_{n-1}, \ldots, u_{n-\nu})\). The elements of the shift register are fed into a block with a known function \( f \), which can be written as:
It is said that the data is transmitted through a memoryless channel which means that the probability of an observation at the output of the channel at time $n$ having the value $z_n$ depends only on the transition $\xi_n$ at time $n$. Having this in mind, given the sequence of observations $z$ the following applies:

$$P(z|x) = P(z|\xi) = \prod_{n=0}^{N-1} P(z_n|\xi_n). \quad (2.1)$$

Since the only observable signal in the system is $z$, the task of the Viterbi algorithm is to find the most probable state sequence $x$ which caused the observed sequence $z$ (i.e. maximum of $P(x|z)$). This is equivalent to finding the most probable sequence of state transitions $\xi$ and finally, since there is one-to-one correspondence between $x$ and $u$ (or between $\xi$ and $u$), to finding the most probable input sequence $u$.

The finite-state Markov process can be represented by a state diagram developed in time, called trellis, in which each node in a stage corresponds to a single state and each stage corresponds to a point in time, as shown in Fig. 2.2. Each transition between states is represented by a branch connecting the nodes, where each of the branches has a weight attached to it. The task of the Viterbi algorithm is to, based on the observed sequence, find the sequence of states in the trellis which has the maximum probability of being the one
appearing in the Markov process. In other words it needs to find the maximum of $P(x|z)$, which is equivalent to finding the maximum of $P(x,z) = P(x|z)P(z)$, or the minimum of $-\ln P(x,z)$, since $-\ln$ is a monotonic decreasing function. From this, it follows that:

$$P(x, z) = P(x)P(z|x) = \prod_{n=0}^{N-1} P(x_{n+1}|x_n) \prod_{n=0}^{N-1} P(z_n|x_{n+1}, x_n).$$

Accordingly, if each transition line in trellis is assigned a "length" (also known as branch metrics) $\lambda$, where:

$$\lambda(\xi_n) \triangleq -\ln P(x_{n+1}|x_n) - \ln P(z_n|\xi_n), \tag{2.2}$$

then the total length (also known as path metrics) of the path ending in state $x_{n+1}$, and containing a state $x_n$ is:

$$\Gamma(x_{n+1}, x_n) \triangleq \sum_{i=0}^{n} \lambda(\xi_i),$$

and the total length of the path corresponding to the sequence $x$ is:

$$\Gamma(x_n) = -\ln P(x, z) = \sum_{i=0}^{n-1} \lambda(\xi_i).$$

This way the task of the Viterbi algorithm can be defined as finding the shortest path between two given states in the trellis, called Viterbi path (which is marked in red in Fig. 2.2). The shortest paths leading to individual states are called survivor paths, and the notation used for a survivor path at time $n$, terminating in state $x_n$ (where $x_n \in \{s_0, s_1, ..., s_{S-1}\}$) is $x(x_n)$. The length of the survivor path terminating in the state $x_n$ is $\Gamma(x_n)$.

### 2.2 Implementation of the Algorithm

The formal statement of the generalised Viterbi algorithm is presented by Alg. 1. Branch "length" ($\lambda$) calculation method differs on the application, and based on this, modules implementing Viterbi algorithm can be divided in two major groups — hard decision and soft decision. The first group of modules have the observations $z$ quantised to only two levels, corresponding to logic 1 and logic 0, and the branch "length" is found as a Hamming distance between the actual observations and all combinations of values these observations can take. The second group uses more quantisation levels for the observed
signal, which means that more precise information on signal’s reliability can be obtained finding the distance between the actual observations and the ideal values these observations can take, for which usually a Euclidian distance is used.

When defining the complexity of Viterbi algorithm’s software or hardware implementation, the required storage resources and arithmetical operations are taken into account. Considering the system shown in Fig. 2.1, at each point in time there are \( m^{\nu+1} \) possible transitions (\( m \) for each state), which means that there are \( m^{\nu+1} \) additions being performed at each time step. The results of the addition for each state need to be compared among each other, in order to find the minimum, which means there are \( m^\nu \) comparisons performed at each time step. Regarding the required memory recourses, it is convenient that for a single state only the length of the survivor path needs to be stored and all the other addition results can be discarded. As for the branch “length” calculation, at each time step only one element of the sequence observed at the input of the receiver (\( z \)) is used, and it is convenient that neither the result of the calculation, nor the observed element need to be stored.

### 2.3 Application Examples

#### 2.3.1 Convolutional Codes Decoding

Signal coding using convolutional codes is employed in signal transmission with the goal of making the signals more resistant to noise which originates in the transmission systems and which can cause significant signal distortion [3]. With this kind of forward error-correction approach introduced in a system, together with the corresponding decoding method, the original signal can be recovered. The extend of the signal recovery depends

---

**Algorithm 1** Formal statement of the Viterbi algorithm

1: \( n \leftarrow 0 \)
2: \( \mathbf{x}(x_0) \leftarrow x_0 \)
3: \( \mathbf{x}(x_i) \) arbitrary, \( i \neq 0 \)
4: \( \Gamma(x_0) \leftarrow 0 \)
5: \( \Gamma(x_i) \leftarrow \infty, i \neq 0 \)
6: **repeat**
7: \( \Gamma(x_{n+1}, x_n) \leftarrow \Gamma(x_n) + \lambda(x), \forall x \)
8: \( \Gamma(x_{n+1}) \leftarrow \min \left( \Gamma(x_{n+1}, x_n) \right), \forall x \)
9: \( \text{update } \mathbf{x}(x_{n+1}) \)
10: \( n = n + 1 \)
11: **until** \( n = N \)
on the power of noise w.r.t. the power of the useful signal. The idea behind this method is to immerse certain pattern into the data bitstream, in order to later on facilitate the extraction of original data from the bitstream received at the other end of a noisy system. Convolutional encoder has the properties of Markov process and a general schematics of the encoder with two outputs is shown in Fig. 2.3. Coding patterns differ by the encoders’ code-rate which is determined as \( p/q \), where \( p \) is the number of bits entering the encoder and \( q \) is the resulting number of bits at the output of the encoder\((q \geq p)\). Equation (2.3) which describes the behaviour of a convolutional encoder with \( J \) outputs, corresponds to a case where there are \( J \) bits formed for each individual input bit (code-rate is \( 1/J \) and the corresponding number of observed symbols at the receiver’s input is \( J \), one for each encoder’s output — \( z^j \)). \( h^j_k \) is \( k \)-th coefficient of the \( j \)-th output of the encoder, where \( j \in \{1..J\} \). The name ”Convolutional” comes from the fact that this method basically represents a convolution in time domain of the input signal and the encoder’s impulse responses. A system employing Viterbi algorithm for convolutional codes decoding is depicted in Fig. 2.4.

\[
y^j_n = \sum_{k=0}^{K} h^j_k u_{n-k},
\] (2.3)
Prior to entering the channel, binary data stream is modulated implementing Binary Phase Shift Keying (BPSK) modulation (i.e. logic 0’s are mapped to (+1) and logic 1’s are mapped to (−1)). The noise superposed to the useful signal in the channel is modeled as an Additive White Gaussian Noise (AWGN) at the input of the receiver \( z_n^j = s_n^j + n_s \). Spectral power density of the noise signal defined on the symmetrical bandwidth is \( N_0/2, \) \( \sigma^2 \) being the variance of the distribution with a 0 mean value, \( n_s = N(0, \frac{N_0}{2}) \). Accordingly, Signal to Noise Ratio (SNR) at receiver’s input is \( SNR = E_b/\sigma^2 \), where \( E_b \) stands for energy per bit of the useful signal. Having in mind the implemented BPSK modulation — \( E_b = 1/2((1)^2 + (−1)^2) = 1 \), meaning that \( SNR = 1/\sigma^2 \).

As mentioned in Sec. 2.1, each of the branches/transitions in the trellis is assigned a metric representing the probability of this transition occurring, based on the observation at a certain point in time. Calculation method for this metric has to correspond to the Markov process occurring in the transmitter, in this case a convolutional encoder. Each branch metric is calculated as a squared distance between the received observation and one of the possible values of the corresponding encoder’s output. This means that in the case of 1/J code-rate encoder, there are \( 2^J \) different combinations of encoder’s outputs, meaning that there are \( 2^J \) branch metrics calculated in every time step \( n \), according to (2.4).

\[
\lambda^l_n = \sum_{j=0}^{J-1} (z_n^j - s(L_j^l))^2, \tag{2.4}
\]

where \( l \in \{0, \ldots, 2^J-1\} \) and \( L_j^l \) represents a \( j \)-th element in a vector \([L_0^l \ldots L_{J-1}^l]\) corresponding to the binary representation of the branch metric’s label \( l \). \( s(L_j^l) \) corresponds to the value bit \( L_j^l \) is mapped to in the modulator. When (2.4) is developed, the following expression is obtained:

\[
\lambda^l_n = \sum_{j=0}^{J-1} (z_n^j)^2 - 2z_n^j s(L_j^l) + (s(L_j^l))^2. \tag{2.5}
\]

Since in the following step, branch metrics are added to the survivor path metrics and the comparison is performed, all the elements of the sum (2.5) that are the same for each of \( 2^J \) branch metrics can be subtracted from the sum. Obviously, elements \( (z_n^j)^2 \) are the same for each metric, as well as \( (s(L_j^l))^2 \) which have a constant value of 1, as a result of BPSK modulation. Branch metrics can be divided or multiplied by a same value without influencing the comparison results. After implementing this, (2.5) gets the following form:

\[
\lambda^l_n = \sum_{j=0}^{J-1} z_n^j s(L_j^l). \tag{2.6}
\]

In order to further simplify the expression for branch metric calculation, it is possible to subtract the metrics \( \lambda^0_n = \sum_{j=0}^{J-1} z_n^j (+1) \) from all the metrics, obtaining in such a way the following expression:
\[ \lambda_n^n = - \sum_{j=0}^{J-1} 2z_n^j L_i^j \]

It is observed that if the metrics would now be divided by \( N_0 \), noise spectral power density, in the obtained expression:

\[ \lambda_n^n = - \sum_{j=0}^{J-1} \frac{2z_n^j}{N_0} L_i^j \]  

(2.7)

the sum elements \( \frac{2z_n^j}{N_0} \) would represent \textit{Logarithmic Likelihood Ratios} (LLRs) of the corresponding observations:

\[ \lambda_n^n = - \sum_{j=0}^{J-1} LLR_n^j L_i^j. \]  

(2.8)

LLR represents a logarithm of a ratio between probabilities of two different models characterising the system at certain point in time, known as the \textit{null model} (starting hypothesis) and the \textit{alternative model} (alternative hypothesis). In binary data transmission, having the observed sequence \( z \) at the input of the module implementing Viterbi algorithm, first hypothesis is that of a logic 1 being transmitted through the channel, the alternative hypothesis is the one of a logic 0 being transmitted instead. It follows that value of LLR at time \( n \) can be calculated as follows:

\[ LLR_n^j = \log \left( \frac{P(z_n^j = 1|y_n^j)}{P(z_n^j = 0|y_n^j)} \right) \]

The calculation of LLRs is done in the module called "demodulator" in Fig. 2.4 and it is performed in the following manner [3]:

\[ LLR_n^j = \frac{2z_n^j}{N_0}, \]

which is in fact the factor found in (2.8).

(2.8) can be more conveniently expressed as a product of input LLRs vector and corresponding branch metric’s label vector:

\[ \lambda_n^n = - [LLR_n^0 \ldots LLR_n^{J-1}] \times [L_i^0 \ldots L_i^{J-1}]^T. \]  

(2.9)

Because of the form in which branch metrics are calculated, this implementation of the Viterbi algorithm is such that its task is to find the longest survivor path metric for each state instead of the shortest one, as introduced in Alg. 1. This fact only affects the type of the comparison used in the algorithm and does not affect any of the remaining steps.

Convolutional encoder compliant with IEEE 802.11 standard for Wireless LAN communication used in the system described in the first part of this report is represented by
Fig. 2.3. It contains a shift register of length $K - 1$, where $K$ is known by the name constraint length and in this case it has a value $K = 7$. Bit-rate in this encoder is $R = 1/2$ with corresponding coefficients (also known as polynomials) being $h_0, h_1 = 133_8$ and $h_0^2, h_0^3 = 171_8$.

In systems where code-rates need to be high because of the limited bandwidth or high cost of bits transmission, the method called puncturing is used. Data puncturing consists of leaving out some bits at the output of the transmitter according to specific pattern. This system requires a de-puncturing unit at the receiver’s input. It is necessary to provide the same information on the puncturing pattern both to transmitter and to receiver, otherwise it is impossible to correctly decode the data. Task of de-puncturing unit is to introduce the neutral value of the signal in the places that correspond to bits left out in the transmitter. As the values at the receiver’s input centre around $-1$ and $+1$, a neutral value in this case is $0$. In this report, several punctured codes with different data rates have been implemented: $R = 1/2$, $R = 2/3$, $R = 3/4$ and $R = 5/6$ according to IEEE 802.11 standard [4].

2.3.2 Maximum Likelihood Sequence Estimation in ISI cancellation

Some data transmission channels show properties of a finite-state Markov process. Transmission systems containing such channel can be modeled as systems with memoryless channels together with an encoder implementing the corresponding finite-state Markov process in the transmitter. In other words, transmission systems containing such channels can be presented as Fig. 2.1. An example of this type of channel is Plastic Optical Fiber (POF) whose transfer characteristic has properties of a low-pass filter. The behaviour of this fiber can be modelled with a Finite Impulse Response (FIR) filter. The filter can be mapped to a convolutional encoder whose constraint length is $K$, which has only one output generated using coefficients: $h_0, h_K - 1$, and which instead of mod2 adder, implements a regular real-numbers adder. Branch metrics calculation in this case is done by employing Maximum Likelihood Sequence Estimation (MLSE) method. The idea behind this method is to emulate the behaviour of the channel in the receiver and then find the negative squared distance between the observation at the input of the receiver and each of the possible ideal outputs of the channel. This way, the most likely transition occurred in the fiber is the one corresponding to the branch metrics having the maximal value. FIR filter behaviour of a POF introduces signal distortion in the form of Intersymbol Interference (ISI), which suggests that the process of signal decoding performed by the Viterbi decoder is in fact process of ISI cancellation. For a discrete data transmission, impulse response of a FIR filter can be given in form of a set of samples/taps carrying the information on the continuous impulse response values at certain points in time ($h^c = (h_0^c, h_1^c, ..., h_K^c)$). In
order to calculate the value of the signal at the output of the channel at time \( n \) (\( \bar{y}_n \)), a convolution needs to be performed between the channel’s impulse response and the signal entering the channel (\( u \)), as presented by (2.10).

\[
\bar{y}_n = \sum_{k=0}^{K-1} h^c_k u_{n-k},
\]

(2.10)

where \( K \) is the length of the channel’s impulse response, and \( n \in \{0, \ldots, N\} \), where \( N \) is the length of the input sequence \( u \).

As mentioned earlier, Viterbi decoder has to emulate the channel in order to correctly recover the data sequence, which means that in this case it has to emulate the behaviour of an FIR filter. If the number of different values signal \( u \) can take is \( m \), then the number of states implemented in Viterbi algorithm is \( N_s = m^{K-1} \). In the trellis diagram, each of the \( N_s \) states in the stage corresponding to time \( n \) has \( m \) possible branches leading to the next stage, where the number of branches corresponds to \( m \) possible values signal \( u \) at the input of the fiber can take at time \( n \). It follows that the branch metrics are calculated as:

\[
\lambda^l_n = - \left( z_n - \sum_{k=0}^{K-1} h^c_k s(L^l_k) \right)^2,
\]

where \( l \in \{0, \ldots, mN_s - 1\} \), \( L^l_k \) represents a \( k \)-th element in a vector \([L^0_k \ldots L^{K-1}_k]\) representing the branch metric’s label and \( s(L^l_k) \) corresponds to the value symbol \( L^l_k \) is mapped to in the modulator. When (2.4) is developed and the constant \( z^2_n \) is subtracted from each metric, the following expression for branch metric calculation is obtained:

\[
\lambda^l_n = 2z_n \sum_{k=0}^{K-1} h^c_k s(L^l_k) - \left( \sum_{k=0}^{K-1} h^c_k s(L^l_k) \right)^2.
\]

(2.11)
Chapter 3

Mapping to Hardware

3.1 System Overview

The mapping of the Viterbi algorithm to a VLSI module called Viterbi decoder is presented in this chapter. The system where this decoder is employed is a Wireless LAN data transmission system which is in accordance with IEEE 802.11 standard. All the modules in the system are presented in Fig. 3.1.

The difference between the system depicted in Fig. 3.1 and the one depicted in Fig. 2.4 is the presence of the quantisation module at the input of the Viterbi decoder. The quantisation module performs quantisation of LLRs calculated in the demodulator, which have continuous values and need to be discretised and quantised in order to be used in a digital circuit such as Viterbi decoder. The next section presents few different architectures of Viterbi decoder based on different trellis diagrams, after which one such architecture is presented. The presented architecture serves as a reference for further architecture exploration and optimisation.

Figure 3.1: Data transmission system implementing Viterbi decoder.
3.2 Radixes

The trellis presented in Fig. 2.2 has the most basic structure, called radix-2 structure, where each node has two branches leading to the next stage, corresponding to the two possible values the encoder’s binary input signal can take. It is possible to include so-called lookahead stages in the trellis diagram in the way presented in Fig. 3.2, so that several time steps are merged into a single one. In this figure an example of lookahead-stages implementation is presented on a 8-state trellis and 2-stage lookahead. The possibility of implementing lookahead stages is owed to the trellis diagram’s regular structure. The trellis diagram obtained by introducing $l$ lookahead stages is called radix-$2^l$ trellis.

The motivation behind using architectures based on higher-radix trellis diagrams is the speedup of the design. Collapsing $l$ stages of trellis into a single one allows for the $l$ decoded bits to be generated in a single clock cycle, instead of them being generated in $l$ consecutive cycles one by one, as in radix-2 configuration. This means that, ideally, a radix-$2^l$ configuration gives $l$ times larger decoding throughput (number of decoded bits in one clock cycle) compared to the radix-2 structure. Clearly, the algorithm implementation complexity is increased and as a result more resources are needed for the calculation and storage of the results, which reflects on the area efficiency. This is depicted in the table Tbl. 3.1 [5]. Note that this table depicts idealised results, especially regarding the speedup. As the complexity of the hardware increases, so does the number of operations needed to be performed in a single clock cycle, which then results in larger signal propagation delay. In some cases, signal delay becomes equal or larger then the clock period and the frequency
### 3.3 Data Windowing

In cases where Viterbi algorithm is used for decoding large sequences ($N$ in (2.1) has a large value), storing the entire survivor path vector $x$ for each state would require too many memory resources. This is why it is assumed that these vectors can be truncated and only a relatively small number ($\delta$) of elements needs to be stored. It is estimated that if $\delta$ is large enough, all the survivor paths corresponding to individual states will pass through exactly the same nodes of the trellis from the time 0 to the time $n - \delta$. An illustration of this is given in Fig. 3.3 corresponding to Fig. 2.2, where all the paths at time 4 contain the same nodes from time 0 to time 2. In marginal cases where the survivors up to the time $n - \delta$ do not completely match, the survivor vector $x(x_{n-\delta})$ can even be chosen arbitrary from the available ones, since the effect of this on the algorithm’s performance, if $\delta$ is large enough, is negligible [2].

The method of determining the length of the stored survivor path ($\delta$ - also known as

<table>
<thead>
<tr>
<th>Radix</th>
<th>$k$</th>
<th>Ideal speedup</th>
<th>Complexity increase</th>
<th>Area efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>0.75</td>
</tr>
<tr>
<td>16</td>
<td>4</td>
<td>4</td>
<td>8</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Table 3.1: Radix-$2^i$ complexity and speed evaluation.

![Trellis diagram showing the convergence of the survivor paths.](image)

Figure 3.3: Trellis diagram showing the convergence of the survivor paths.

of the decoder needs to be lowered, lowering the decoding throughput.
traceback length) during Viterbi algorithm’s execution is called data windowing. As mentioned in Sec. 2.3.1, several punctured codes are explored in the system using convolutional coding as a forward error-correction method. In order to determine which of the different code-rates gives the worst results with respect to Bit Error Rate (BER) at the output of the decoder, each of the codes was simulated on a Matlab model of a radix-4 architecture, while the value of δ was fixed for all code-rates. Obtained results are plotted and can be seen in Fig. 3.4. As expected, the figure shows that the coding that gives the worst results w.r.t. BER is the one corresponding to the code-rate of  \( R = \frac{5}{6} \) because of the large number of bits being omitted in the transmitter, compared to the case of  \( R = \frac{1}{2} \).

Since the focus of this report is to find the optimal architecture w.r.t. requested decoding throughput for the basic system setup, the value of δ will correspond to the system with 1/2 code-rate. A sweep over δ is performed for a fixed SNR value on a radix-2 architecture model, and the resulting plots are shown in Fig. 3.5. As it can be read from the plot, the optimal value of δ to be used here is around 50, since further increase does not introduce any improvement w.r.t. BER, and would only result in increased power and area consumption. As the idea of this report is to give a comparison between different Viterbi decoder architectures, it is necessary to scale all the design parameters in a correct manner. Since for a radix-\( 2^l \) trellis algorithm implementation in each cycle, for each element of the survivor path vector, information on exactly \( l \) bits inputted in the encoder is stored, it is necessary to accordingly scale the length of this vector (δ):
Figure 3.5: Traceback length sweep for 1/2 code-rate on a radix-2 architecture, with a fixed SNR value.

\[ \delta^l = \frac{\delta^1}{l}, \]

where \( \delta^l \) is the length of the survivor path when \( \text{radix-}2^l \) architecture is implemented.

Having in mind the simulation results for radix-2 architecture and the fact that the values of \( l \) considered in this report are \( l \in \{1, 2, 3, 4\} \), the following values are chosen for \( \delta \):

\[ \delta^1 = 48, \]
\[ \delta^2 = 24, \]
\[ \delta^3 = 16, \]
\[ \delta^4 = 12. \]

In order to get the information on the value of \( \delta \) needed to get the optimal results in the worst case puncturing code, meaning \( R = 5/6 \), a sweep is performed for a fixed value of SNR, this time on radix-4 architecture model, and the result is shown in Fig. 3.6. The optimal value is roughly twice as big as the optimal one in case of the basic \( R = 1/2 \) code-rate.
3.4 Data Quantisation

3.4.1 Modulo Normalisation

As mentioned in the beginning of this chapter, it is necessary to perform a quantisation of continuous signals prior to inputting them in the receiver. Since the number of bits for different metrics’ representations is limited, while performing the calculation of path metrics for each state an overflow might occur. When the length of the noisy sequence coming into the decoder is large, the overflow is bound to occur at some point, which could cause errors in the decoded sequence. Overflows don’t present a problem in cases where all the candidate path metrics for a particular state experience overflow at the same time, as it is not their absolute value that is relevant for correct decoding, but their mutual distance. But, since it is more likely that the overflow introduces problems, a metric normalisation must be performed in order to ensure a correct data stream at the output of the decoder.

Various metric normalisation approaches are available in the literature, but the one found to be the most optimal for a VLSI implementation is the Modulo normalisation [6], which is consequently chosen to be implemented in this Viterbi decoder design.

The idea behind the modulo normalisation is for a metric \( m_i \) to be replaced by a normalised metric \( \bar{m}_i \):
3.4. DATA QUANTISATION

Figure 3.7: Graphical example of modulo normalisation.

\[ \tilde{m}_i \triangleq (m_i + C/2) \mod C - C/2, \]

which allows for the comparison of two metrics \( m_1 \) and \( m_2 \) to be mapped into the comparison of the normalised metrics \( \tilde{m}_1 \) and \( \tilde{m}_2 \).

This normalisation can be represented graphically as wrapping the metric \( m_i \) around a circle whose circumference equals \( C \), starting from 0 angle point and moving in the counterclockwise direction. It follows that the angle on the circle corresponding to the normalised metrics has a value of \( 2\pi m_i / C \). Also, it can be seen that the range of the normalised metric is now: \(-C/2 \leq \tilde{m}_i < C/2\). Using this method, the comparison between two metrics is equivalent to comparing the angle between them (moving in the CCW direction) to \( \pi \). An example of this is shown in Fig. 3.7, where \( m_1 < m_2 \) if and only if \( \alpha < \pi \). In order for this method to work correctly, the difference between the two metrics being compared has to be smaller than \( C/2 \) (\(|m_1 - m_2| < C/2\)).

Instead of using regular signed comparison when implementing modulo normalisation, it is possible to present this method in slightly different fashion. The values of the normalised metrics can be represented as:

\[ \tilde{m}_i = \sum_{b=0}^{w-1} m_i^b 2^b, \]

where \( m_i^b \) is the \( b \)-th bit of the metric’s binary representation — \( \tilde{m} = (\bar{m}_i^{w-1}, \bar{m}_i^{w-2}, ..., \bar{m}_i^{0}). \)

A new metric can be defined as:

\[ \hat{m}_i \triangleq m_i \mod C/2, \]

or in another way presented as:
It is possible to show that the comparison of two normalised metrics \( c(\bar{m}_1, \bar{m}_2) \) is equivalent to:

\[
c(\bar{m}_1, \bar{m}_2) = \bar{m}_1^{w-1} \oplus \bar{m}_2^{w-1} \oplus c_u(\hat{m}_1, \hat{m}_2),
\]

where \( c_u(\hat{m}_1, \hat{m}_2) \) represents an unsigned comparison of the metrics \( \hat{m}_1 \) and \( \hat{m}_2 \).

This method can be verified using the example depicted in Fig. 3.7. Implementing (3.1) on the metrics shown in the figure leads to the conclusion that \( \bar{m}_1 < \bar{m}_2 \), which corresponds to the conclusion made based on the value of the angle \( \alpha < \pi \). The corresponding hardware for this method’s implementation is depicted in Fig. 3.8.

### 3.4.2 Fix-point Metrics Representation

To minimize resources needed for hardware or software implementation of the Viterbi algorithm, a fixed-point representation of numbers is used, as depicted in Fig. 3.9.

In order to determine the most optimal design parameters Matlab simulations of system model are used. Since different metrics in the design are mutually dependent, the sweeps of integer and fractional parts’ widths are done only for the input signal, whereas the widths of the remaining metrics are expressed as functions of the former one. Having in mind the calculation of branch metrics given in (2.9), its fixed-point representation parameters are determined as follows:
3.4. DATA QUANTISATION

Table 3.9: Fixed-point representation of metrics in the design.

<table>
<thead>
<tr>
<th>(m_{w-1})</th>
<th>(m_{w-2} \ldots m_{w-F})</th>
<th>(m_{f-1} \ldots m_{f_0})</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\text{sgn})</td>
<td>integer part</td>
<td>fractional part</td>
</tr>
</tbody>
</table>

where \(w_{\text{frac}}\) is the width of the fractional part belonging to metric \(m\), while \(w_{\text{int}}\) is the width of the integer part belonging to metric \(m\) and \(w_m\) is the overall metric’s width including the sign bit. As there is no need for branch metric’s precision to be larger than the one of the LLR since only addition operation is implemented in their calculation, the widths of these two metrics are the same.

\[
w_{\text{frac}} = w_{\text{frac,LLR}},
\]

where \(l\) is the lookahead level implemented in algorithm’s trellis.

As presented in Sec. 3.4.1, in order to implement modulo normalisation correctly, the maximum difference between two path metrics must not exceed \(C/2\). In order to determine the value \(C\), let’s consider the trellis diagram with the constraint length \(K\). Let’s say that at time \(n\) the survivor path of the node \(s_a\) is known and its length is \(\Gamma(s^n_a)\). Considering the shift register in the encoder, it is certain that after at most \(K\) stages in the trellis, there will exist a direct path from the node \(s_a\) in stage \(n\) to the node \(s_a\) in the stage \(n + K\). This means that the new length of the survivor path for the state \(s_a\) will be \(\Gamma(s^{n+K}_a)\). The difference between the two survivor path lengths will be at most \(K\lambda_{\text{max}}\). Since we can claim that at time \(n\) the lengths of all survivor paths for different states are known (the starting values of survivor paths lengths at the time 0 are initialised at the known values), in order to calculate the survivor path of a particular state, we only need to store the sum of lengths of the preceding \(K\) branches for each state. This means that the largest value \(\Gamma(x_i)\) can take is \(K\lambda_{\text{max}}\), and the smallest \(-K\lambda_{\text{max}}\). Accordingly, since the difference between two normalised path metrics cannot exceed \(C/2\), it follows that:

\[
\frac{C}{2} \geq 2K\lambda_{\text{max}},
\]

which, bearing in mind that \(\lambda_{\text{max}} = 2^{w_{\text{BM}}-1}\) and \(-C/2 \leq \bar{m} < C/2\), means that the smallest number of bits used for the path metric representation is:

\[
w_{PM} = 1 + w_{BM} + \lceil \log_2 K \rceil.
\]

Accordingly, fixed-point representation of path metric is:

\[
w_{PM}^{\text{int}} = 1 + w_{BM} + \lceil \log_2 K \rceil = 2 + w_{BM}^{\text{int}} + w_{BM}^{\text{frac}} + \lceil \log_2 K \rceil,
\]
The first parameter needed to be determined is the number of integer bits for LLR representation. With this purpose, the number of fractional bits has been set to a very large value, emulating the behaviour of an unconstrained fractional part, as this part clearly has less significant influence on metrics’ values formation. In order to estimate the sweep range of the LLR integer part width, calculation method of LLR values has to be taken into account, and the fact that typical SNR values used in this system are relatively small (< 10dB). Sweep results are plotted and shown in Fig. 3.10. As it can be seen, BER value does not change when the number of integer bits is increased from 3 to 5 which leads to the conclusion that the optimal integer width is 3 ($w_{\text{LLR}}^{\text{int}} = 3$).

As for the fractional part representation, the simulation is done with the integer part width set to 3 and based on Fig. 3.11 fractional part width is chosen to be 1, setting the overall number of bits for input LLR values to 5.

Clearly, computation methods implemented in the Viterbi decoder do not change and the same decoding results are obtained independent of where the decimal point is positioned, since input values can be scaled up or down multiplying or dividing the inputs by

\[ w_{PM}^{\text{frac}} = w_{BM}^{\text{frac}} = w_{LLR}^{\text{frac}}, \]
2, i.e. shifting the decimal point right or left. It is only the mutual ratio of the input values and not their absolute values what is significant for producing valid results.

As mentioned, the simulations are done for radix-4 architecture, but since the only fixed width is the one of the input signals, and the widths of the other metrics are calculated based on the decoder’s architecture, it is clear that the presented simulation results for input LLR signals representation can be used in other radix architectures as well, as long as the branch and path metric widths are recalculated for each architecture.

3.5 Viterbi Decoder - Reference Architecture

The Reference architecture of the Viterbi decoder explored in this report is taken from [7]. The Viterbi decoder consists of three main building blocks depicted in Fig. 3.12 together with the handshake interface controller for communication with the external modules. This design implements convolutional codes decoding, more precisely it is built in accordance with WLAN IEEE 802.11 standard, and it is based on radix-4 trellis structure. The three main modules in a Viterbi decoder are:
• **Branch Metrics Unit (BMU)**

• **Add-Compare-Select Unit (ACS)**

• **Survivor Path Memory Unit (SPM) — Register Exchange Unit (RE)**

A detailed descriptions of individual Viterbi modules and their functions are presented in subsections that follow.

### 3.5.1 Branch Metrics Unit (BMU)

The Branch metrics unit for each time step generates all possible branch metrics according to the (2.9) introduced in the previous chapter. The actual size of LLR and $L$ vectors is $J = 4$, as $R = 1/2$ code-rate transmission and radix-4 architecture are implemented. This leads to the following equation for branch metrics calculation:

$$
\lambda_i^n = -[LLR_{i,n}^0..LLR_{i,n}^3] \times [L_i^0..L_i^3]^T,
$$

where $[L_i^0..L_i^3]$ corresponds to the binary representation of the output’s label. In other words, each of the outputs of the BMU is calculated as a sum of the inputs whose corresponding bits in the binary output signal’s labels have a non-zero value. The schematic for radix-4 BMU is shown in Fig. 3.13.

As it can be seen, the complexity of this unit is not too great, and the influence it has on overall area and power consumption of the decoder is minimal. Clearly, as the radix level becomes higher, the complexity increases, but since the complexity of the other two units increases at the same time, BMU remains the costliest module in the decoder.
3.5.2 Add-Compare-Select Unit (ACS)

Add-Compare-Select unit is implemented as fully parallel and it consists of 64 ACS modules which correspond to 64 nodes in a single trellis stage (Fig. 3.14). The task of each ACS module is to determine which of the four possible paths is more likely to have led to a particular state which this module corresponds to. To do this, first each individual branch metric is summed with the corresponding path metric to get four candidate path metrics for the current state. Here, the path metric that has the largest value is the one chosen to be the next survivor metrics. When the comparison is done, the label of the sum that represents the most likely path is forwarded towards the next stage in the decoder — survivor path memory unit. This label of the chosen survivor path metric represents the decision signal, which in the case of radix-4 is coded with 2 bits. The structure of ACS module is shown in Fig. 3.15. After comparison and selection, the survivor path metrics are fed back to ACS modules in the following clock cycle. In the case of radix-4 architecture, for each state there are four states that can precede it, and their path metrics are the ones fed as inputs to the module of the current state. In a similar fashion, four branch metrics fed as inputs to an ACS module correspond to the four possible state transitions.

ACS unit is the part of Viterbi decoder that contains speed critical paths of the data.
flow. The reason for this lays behind the fact that these paths are constructed, as the name of the module suggests, from operations such as addition, comparison and selection. Since the goal of this architecture is to achieve the smallest possible operating clock period, implementation of a high level of parallelism is encouraged. This is why a completely parallel comparison between the set of values is implemented as opposed to a tree structure. The tree structure would reference the smallest number of comparators and would give the most optimal results area-wise, but at the same time would introduce high latency as a result of many comparison stages it contains. Let’s say there are $M = 2^m$ values needed to be compared and its maximum chosen using a tree structure. The number of stages would be $\lceil \log_2 M \rceil = m$ and the number of comparators — $m - 1$. As for the fully parallel implementation of comparison, it is done in such a way that the comparison is performed between each two values in the given set and then the logic gates are used to process the output signals and make a decision on the largest value in the set. Number of comparators in this case is $M(M - 1)/2$, which is significantly larger than in case of a tree-like structure, especially for higher radices. Regarding the signal propagation delay through these structures, parallel structure will always have one stage of comparison and few stages of logic gates (the higher the radix, the larger the number of stages of logic) but still giving significantly smaller propagation delay than the tree-like structure. Since the number of bits is limited, the algorithm for calculating the path metrics is adjusted to deal with the overflow occurring during the addition, according to the modulo normalisation.

Figure 3.14: Add-Compare-Select unit, radix-4 architecture.
method introduced in Sec. 3.4.1.

Path metrics which are outputted from the 64 ACS modules are stored in registers found at the input ports of each ACS module, constructing the so-called Path Metric Unit (PMU). Each of the output ports which carry path metric information has a fan-out which corresponds to the radix level (4 in case of radix-4). One of the optimizations done in the decoder is distributing PMU over all 64 ACS modules, thus having the need for $4 \cdot 64 (w_{PM} + w_{PM})$-bit registers with fan outs of 1 instead of having only $64 (w_{PM} + w_{PM})$-bit registers each of which having the fan-out of 4. Even though this introduces some area and power consumption overhead, it gives better results speed-wise. The reason for this is the fact that the smaller the fan-out, the less time is needed for the signals at the input of ACS to be asserted to the valid level. The pattern in which the PMU and BMU output signals are connected to the individual ACS modules corresponds to the trellis diagram and the output signals generation in the encoder.

3.5.3 Register Exchange Unit (RE)

There are two realisations of SPM unit that are most commonly used — Trace Back (TB) [8] and Register Exchange (RE). Both of these structures use decision bits received from ACS unit to recreate the sequence of bits encoded in the transmitter. The first one uses SRAM memory to store the decision bits which, for a high-throughput study, implies MHz or GHz SRAM macros to be used. Having in mind that SRAM compilers are not capable of generating such high speed memories, TB is not suitable for high-speed study. RE on the other hand, uses standard logic cells to emulate the structure of trellis diagram where, in place of the nodes, it contains multiplexers with decision bits as select signals and registers for storing the decision data (Fig. 3.16). This allows for the frequency to be larger than in the case TB is used. One more advantage of RE compared to TB is the fact that the latency of TB is twice that of RE, since TB works on the principle first in-last out (FILO), whereas RE works on the principle first in-first out (FIFO). Disadvantage of RE is the area required for laying out the multiplexers and registers matrix and the large number of interconnections. Apart from the area overhead, RE consumes more power than TB as a consequence of referencing many multiplexers and registers which switch states at each clock cycle. The length of RE unit is usually referred to as traceback length and it corresponds to $\delta$ introduced in Sec. 3.3.

As it is expected that after certain number of time steps (stages of RE) all survivor paths of different states converge to a single one, it does not matter which row in the last stage of RE decoder the output decoded bits are taken from, usually the first row is the one chosen as the output port. As it can be noticed from looking at RE structure, some optimisations are implemented which reduced the number of registers and multiplexers as opposed to a case where exact mapping of the trellis diagram is done. Firstly, the number of stages is reduced by $K - 1$ ($K$ — encoder’s constraint length), based on the fact that the values of registers in the first $K - 1$ stages are constant. Second optimisation is done
in the last stages of the RE, where the registers on the data paths which do not lead to the output port are removed from the module. This way, the area of RE is reduced, as well as the power dissipation which would be introduced by unused registers switching states.
Figure 3.16: Register Exchange, radix-4.
Chapter 4
Evaluation for High Speed

4.1 Performance Metrics and Voltage Scaling Model

In order to be able to easily compare the performance of the designs described in this report to other similar studies, it is necessary to precisely define the metrics to be considered during the evaluation, as well as their units. Since one of the main goals of this report is to pursue high decoding throughputs and explore the trade-offs between area, energy-efficiency and the throughput, metrics considered when evaluating the designs are throughput, energy efficiency/power dissipation and area.

The throughput is measured in Gb/s, which corresponds to the number of informational bits the decoder is able to generate at its output during one second. Area is expressed in $\mu m^2$. As for the energy-efficiency, this metric is expressed in nJ/bit and it is calculated as a product of consumed power in a single clock cycle and the clock period, divided by the number of bits the decoder is outputting in each clock cycle. This way it is possible to determine the quantity of energy which is consumed for decoding a single bit of data, making the comparison between different radix architectures easier.

Performance results of the designs presented in this report are obtained by simulations at fixed supply voltage $V_o=1V$. In order to be able to compare these designs to the others that use different supply voltages, a numerical model performing voltage scaling is employed. The model uses results obtained form simulations at 1 V and it gives approximate results for the energy efficiency of designs for different supply voltage levels.

The total power dissipation of a chip consists of static/leakage power dissipation, internal and switching power dissipation. The most significant contributor to the power dissipation is switching power which has a squared dependency on the supply voltage [9]:

$$P_{\text{switch}} = CV^2 \alpha f_{\text{CLK}} \frac{f_{\text{CLK}}}{2},$$

where $C$ is the capacitance of the node switching the levels, $\alpha$ is activity factor of this node and $f_{\text{CLK}}$ is operating clock frequency of the chip.
The total power dissipation of the chip will be approximated with the switching power dissipation, and if the value of this power at 1 V and 25°C is noted as \( P_0 \), the following dependency applies:

\[
P = P_0 \frac{v^2}{V_0^2}
\]

The dependency of the propagation delay \( T \) on the voltage is following [9]:

\[
T = K \frac{v}{(v - V_0)^\alpha},
\]

\[
K = T_0 \frac{(V_0 - V_T)^\alpha}{V_0},
\]

\( T_0 \) represents the value of \( T \) at the referent supply voltage \( V_0 \) (\( V_0 = 1 \) V). \( \alpha = 1.6 \) is a technology parameter called velocity saturation index, \( V_T \) is the threshold voltage, the value used here is \( V_T = 0.35 \) V.

From the previous two equations we have:

\[
T = T_0 (V_0 - V_T)^\alpha \frac{\sqrt{P}}{P_0} \frac{\sqrt{P}}{(\sqrt{P} V_0 - V_T)^\alpha},
\]

which is the numerical model for voltage scaling used in Matlab simulations.

### 4.2 Evaluation Flow

Design evaluation according to the metrics specified in the previous section is automated using a custom made set of scripts written in Shell command language and Perl. These scripts implement a flow which consists of a sweep over specified clock period range. For each clock point design-specific Tool Command Language scripts are invoked and following actions are thus executed:

- RTL code synthesis (Synopsys)
- Placement and routing of the synthesised design (Candence Encounter)
- Stimuli/VCD-based power simulation (ModelSim/Encounter)
- Statistical power simulation (Encounter)

Finally, after all the results are consolidated, they are plotted in Matlab in order to allow easy overview of the trade-offs needed to achieve certain performance.

This evaluation flow is highly reconfigurable, parametrisable and it is not design-dependent. Different configuration files are used to specify simulation parameters. Clock
sweep range is specified in form of starting clock period value, clock period increment and the number of points to be simulated. Thanks to the different flags implemented in the scripts, it is possible to select the actions to be executed during the clock sweep, as well as the Matlab plots to be generated after the results are consolidated. Flow is not tool version-dependent, the version to be used for each invoked tool can be configured manually. The flow uses specific directory environment which has to be respected. The names of Tcl scripts for design synthesis and/or placement and routing need to be configured prior to running the scripts, since these scripts are design-specific.

Power simulation parameters are reconfigurable. It is possible to specify the start and the end time of the simulations. Also, the activity factors for statistical power simulations can be configured in case a faster estimation of energy efficiency, compared to VCD-based one, is needed. The data on the energy efficiency and the area can be extracted for the whole design and the different modules that can be specified in one of the configuration files.

The script first performs a simulation environment setup, creates different required Tcl and Matlab files to be used later on in the flow. After the entire clock range is swept, the scripts perform a consolidation of the results, extracting them from different report files, outputs of the simulators, and combines them in several files each of which is carrying information on one of the simulation results. The results of the simulations extracted from the reports are: area of the specified modules and achieved clock period after design synthesis; area of the specified modules and achieved clock period after placement and routing; energy efficiency based on switching, internal, leakage and total power dissipation of the specified module after VCD-based power simulation; energy efficiency based on switching, internal, leakage and total power dissipation after statistical power simulation (based on global activity factor, input activity factor and detailed activity factor). Plots in Matlab that can be generated if so specified are:

- Area vs Throughput after design synthesis,
- Area vs Throughput after design placement and routing,
- Frequency after synthesis vs Frequency after placement and routing (with purpose of estimating the effect of the routed interconnections on the signal propagation delay),
- Energy efficiency vs Throughput of the specified modules in the design, the whole design and the clock tree.
Table 4.1: Performance comparison of the different radix architectures.

<table>
<thead>
<tr>
<th></th>
<th>Radix-2</th>
<th>Radix-4</th>
<th>Radix-8</th>
<th>Radix-16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput [Gb/s]</td>
<td>1.287</td>
<td>2.03</td>
<td>1.98</td>
<td>1.85</td>
</tr>
<tr>
<td>Energy Efficiency [nJ/bit]</td>
<td>0.22</td>
<td>0.36</td>
<td>0.54</td>
<td>1.21</td>
</tr>
<tr>
<td>Area [mm²]</td>
<td>0.233</td>
<td>0.54</td>
<td>1.17</td>
<td>2.88</td>
</tr>
<tr>
<td>Hardware Efficiency [GEs]</td>
<td>57730</td>
<td>84824.57</td>
<td>188427.64</td>
<td>496414.78</td>
</tr>
</tbody>
</table>

4.3 Evaluation of Different Radix Architectures

The flow presented in the previous section is used for evaluation of the radix-2, -4, -8 and -16 Viterbi decoder architectures, which are based on the reference radix-4 architecture presented in the Sec. 3.5. Being that one of the main goals of this report is to achieve high speeds for Viterbi decoder, the reference design structure is optimised using gated clocks. Gated clocks are employed as a mean to deal with the large fan-out of one of the handshake interface controller signals which is enabling the flip-flops in the design. The large fan-out is due to the number of flip-flops, which for a radix-4 design reaches the value of around 8 thousand. By introducing gated clock module, the handshake interface controller signal serves as the enable signal for the gated clock output generation, while the placement and routing tool deals with its large fan-out similarly as it does with the regular clock signal. Employing gated clocks in the design allowed for the higher frequencies to be used.

VLSI implementation of different radix architectures is performed, and the best reachable values of data throughput are shown in Tbl. 4.1. Based on the theory, it is expected that the higher the radix is, the higher the throughput is. However, looking at the values in the table, it is obvious that this is not the case. The reason for this is that this regular behaviour observed when introducing additional lookahead stages can be achieved only when the clock period constraints are not too tight. At a certain point, not all architectures are able to reach the same frequencies and this is when the regularity stops. As it can be seen from the table, the best throughput is achieved with radix-4 design, in addition for it being the most optimal when it comes to energy efficiency and the hardware efficiency (for high throughput). Radix-16 architecture is far the worst w.r.t. energy and hardware efficiency, with even smaller throughput compared to radix-8 and radix-4 architectures. This can be explained by taking a closer look into the radix-16 architecture and comparing it to the radix-4 one.
Since the speed critical path of a Viterbi decoder in most cases is located in the ACS unit, it is necessary to take a closer look into it. There are 64 ACS modules working in parallel in this unit. Each of them has two output signals, one containing the survivor path length for the corresponding state and one containing the decision taken when choosing the survivor path. At the input each ACS module has $2^l$ path metric signals and the same number of branch metric signals. This means that the number of sums needed to be calculated is $2^l$, meaning that there is $2^l$ adders working in parallel. Having in mind that the widths of branch and path metrics differ for different radix architectures, according to the formulas presented in Sec. 3.4.2, the higher the radix, the larger the widths of the metrics and consequently the larger the propagation delay of the adders in ACS modules. Following the adders is the stage of parallel comparators. Here as well it stands that the higher the radix, the larger the delay through the comparator stage. The largest difference between two different radix architectures is seen in the combinatorial stages following the comparators, where the output signals of the comparators need to be processed in order to make a decision on the survivor path. In radix-4 architecture, according to the formula presented in Sec. 3.5.2, there are 6 comparators and the decision upon the largest metric can be reached in as little as 4 stages of two-input logic gates (Fig. 3.15 shows the case where three-input gates are used). As for the radix-16 architecture, there are 120 comparators and the same number of signals needed to be processed in order to generate the decision of the survivor metric. The optimisation of this combinational path is done by the tool (Encounter), but it is clear that the number of logical gate stages in this case if far bigger than in case of radix-4 architecture. Consequentially, the propagation delay through this combinational stage is far higher in radix-16 architecture than in radix-4.

At the output of an ACS module there is a selection stage, which based on the decision generated in the previous stage, selects the corresponding survivor path and forwards its value to the output. The actual realisation of this stage is left to be optimised by the tool, but having in mind the structure of a regular multiplexer, it is clear that the higher the number of input signals, and the wider they are, the higher is the propagation delay.

One additional thing needed to be taken into consideration when comparing different radix architectures is the fan-out of the ACS modules. In case of radix-16 this number is 4 times higher than in case of radix-4. Looking at the Fig. 4.3 it is clear that there is a large difference between the areas of these two designs. As it is possible to see from Fig. 4.1 in which the area distribution of radix-4 architecture is presented, most of the design area is in fact consumed by the ACS unit, which means that the increase of the ACS unit area in radix-16 w.r.t. radix-4 ACS unit, suggests a significant increase of interconnections’ lengths of the ACS unit. Together with high fan-out, this results in high parasitic capacitances at the output nodes of ACS modules.

The stated facts serve to explain the order of the radix-4, radix-8 and radix-16 designs in the figures. However, the position of radix-2 architecture is not corresponding to the order of the other three designs. The cause of this is once more the structure of its ACS module. In radix-2 design, there are only two comparators, and consequently only a single gate which is generating the decision signal. This means that here the delay of ACS module depends only on the delays of a flip-flop, adder, comparator and the gates realising
a multiplexer. Since it is impossible to reach higher throughput for radix-2 structure, this leads to the conclusion that the corresponding frequency is the limit of this type of Viterbi decoder architecture, for the used technology.

In order to show trade-offs between the throughput and the energy efficiency for different architectures, Fig. 4.2 is created using the automated design estimation flow, where several clock period values are simulated. Fig. 4.3 shows hardware efficiency comparison for different radix architectures. As in case for the highest achievable throughputs, the constellation of the designs is the same, the radix-2 and radix-4 architectures are the most optimal ones for smaller and larger throughputs, respectively. Radix-8 and radix-16 designs are proven to be non-optimal for any throughput they can reach.

The results of voltage scaling is shown in Fig. 4.4. Scaling is performed in a range from 0.7 to 1.7V. It is done for all mentioned radix architectures, for different designs w.r.t. clock period constraint. The figures present the best achievable throughputs for given range of energy, which are determined from the formulas introduced in the first section of this chapter. Simulated values, corresponding to $V_0=1\,\text{V}$ are marked with dots. As can be seen, the radix-4 architecture is the most optimal architecture for wide range of throughputs. The throughputs achievable with radix-16 come with a price of high energy consumption, i.e. power dissipation, which can prove to be too high for a chip to function properly.
4.4 Data Interleaving

When the reference Viterbi decoder architecture was discussed, it was pointed out that the most challenging part of the data path are the combinatorial stages of ACS module. The usual approach when increase of frequency is wanted is introduction of pipeline stages in the critical data path. However, this produces an improvement of the clock frequency only if the propagation delay of the combinatorial path is not comparable to the one of a flip-flop.

The fact that ACS module contains a feedback loop makes the task of implementing pipeline stages not so straightforward. Since the path metrics added to the current branch metrics at the input of ACS module are the result from the calculation performed in the previous clock cycle, it is not possible to simply place a flip-flop on the path leading from the input to the output of this block. A way to introduce pipeline stages in the ACS module is shown in Fig. 4.5. First figure shows a regular, not pipelined module containing a feedback loop and perfuming a function \( f \) (in this case an ACS module), whereas the second figure represents a structure where pipelining is introduced. As presented, the signal that is fed back into the module performing a function \( f \) is the result of the input signal from two clock cycles ago. In context of the data decoding, the signal coming into
the module has to consist of two independent streams of data, which will be decoded at the same time, but thanks to the pipelining, they will not overlap in any way. Clearly, the pipelining has to be introduced in all modules of the Viterbi decoder, all the registers need to be duplicated. This approach of merging several independent data streams into a single one is called data interleaving. Naturally, the number of pipeline stages is not limited by the functionality, it is only limited by the propagation delay added flip-flops are introducing w.r.t. the propagation delay of the combinatorial stages.

Employing this method can be done only in case of memoryless data transmission channels. In this case a deinterleaver must be employed in the receiver, positioned behind the decoder.

4.5 VLSI Implementation Results for Radix-2, -4, -8 and -16 Architectures after Interleaving

Propagation delay of combinatorial logic in an ACS module can be decreased by implementing data interleaving in the system, and a corresponding number of pipeline stages
4.5. VLSI IMPLEMENTATION RESULTS FOR RADIX-2, -4, -8 AND -16 ARCHITECTURES AFTER INTERLEAVING

in the Viterbi decoder. Two and three stages of interleaving are implemented and the results regarding the throughput, energy efficiency and area are presented in the following subsections. The pipeline registers are positioned using special command in the Synopsys tool, which is performing the optimisation of hardware, balancing the data paths with the goal of reaching the specified clock period.

4.5.1 Throughput

As mentioned, the introduction of interleaving in different Viterbi decoder architectures does not necessarily produce an improvement of the throughput. This is illustrated in Fig. 4.6. As it can be seen, the improvement is obvious only in case of high radixes, where the propagation delay of a combinatorial stages in the pipeline is still high enough so that it is not comparable to the delay of a flip-flop. The increase of design complexity also needs to be taken into account, the placement and routing of the standard cells is more difficult for larger designs. In the same time, the complexity of the clock tree is significantly increased, having in mind that the number of flip-flops in the design is doubled/tripled. Higher fan-out of the clock tree implies that more levels of clock tree need to be implemented, making
the balancing of the clock tree more difficult, thus making the slew of the clock signal between two consecutive registers larger.

In case of a radix-2 architecture, the introduction of the pipeline stages makes almost no difference w.r.t. the throughput, but it introduces larger area and power consumption, which make interleaved designs of radix-2 non-optimal. In case of a radix-4 architecture, the improvement is visible when two-stage interleaving is implemented. However, when three-stage interleaving is implemented, a decrease of throughput can be noticed, which is due to the more complex clock-tree routing. The size of radix-8 and radix-16 critical paths allow for an improvement of the throughput to be observed with introduction of each additional stage of interleaving. It can be seen that the three-stage interleaved radix-8 architecture gives the best throughput from all explored designs. However, after exploring the effects of four-stage interleaving being introduced in this design, it is noticed that there is no noticeable improvement in the throughput, meaning that the three-stage interleaved design is in fact the best possible design w.r.t. the throughput.

4.5.2 Energy Efficiency

As expected, the introduction of interleaving impacts the energy efficiency of the designs. Power dissipation is somewhat increased when additional registers are implemented.
4.5. VLSI IMPLEMENTATION RESULTS FOR RADIX-2, -4, -8 AND -16 ARCHITECTURES AFTER INTERLEAVING

The impact of interleaving on the throughput for different radix architectures.

The impact of interleaving on the throughput

Figure 4.6: The impact of interleaving on the throughput for different radix architectures.

in the design. This can be illustrated on the architecture that is proven to be the best one for obtaining the high throughput — radix-8 architecture. As can be seen in Fig. 4.7, the energy efficiency of different designs do not differ significantly. Even though the number of registers in the design is doubled/tripled when interleaving is introduced, at the same time the buffers that were included in data paths to deal with high signal fan-outs are no longer needed, meaning that the power dissipation is not increased in large extent. In other words, the power dissipation introduced by additional registers is canceled out by the decrease of power dissipation after buffers in data paths are removed.

The maximum throughput in this report is reached in a three-level pipelined, radix-8 architecture. The power distribution in this design throughout different modules is presented in Fig. 4.8, which illustrates that the most power is dissipated in ACS module, as expected. The energy efficiency trade-offs needed to be considered for reaching certain throughputs in this design is shown in figure Fig. 4.9.

In case a low-power Viterbi decoder is required in the system, from the Tbl. 4.1 it can be concluded that the radix level needed to be used in this case is radix-2. This is due to the fact that the combinational logic in ACS module in radix-2 architecture is the shortest and thus it consumes the least amount of power. The significance of the ACS unit in overall power consumption in the decoder design, when no interleaving is introduced in the system, is already presented in Fig. 4.1.
4.5.3 Area

The introduction of pipeline stages leads to some area overhead as presented in Fig. 4.10 on the example of the radix-8 architecture. As mentioned before while discussing the increase of power consumption, the buffers located in the data paths in the non-interleaved design, are mostly unneeded in the interleaved designs due to the fact that the registers are introduced, and the high fan-out problem of some signals is resolved using these registers.

In Fig. 4.11, area distribution throughout different modules in Viterbi decoder is presented, for the case where three-stage interleaving is implemented in the system. As expected, the largest area is consumed by ACS unit.

In Fig. 4.12, an overview of a trade-off between throughput and area for the three-stage interleaved radix-8 architecture is presented.
4.6 Conclusion and Comparison to Related Studies

In this chapter the results of the VLSI implementation of Viterbi decoder are presented. Different optimisation methods are used to achieve best possible throughput, since the main goal of this report is to deal with high-speed Viterbi decoders. In the same time, an overview of different trade-offs needed to be done w.r.t. energy efficiency and area overhead are presented. Design evaluations are done using an automated script which performs design synthesis, placement and routing and power simulations. The script is highly reconfigurable, which enables its usage for different designs, containing different number of modules, while the simulation parameters are set manually prior to script execution. From the presented results it can be seen that the best possible throughput is achieved using three levels of pipelining, in other words implementing a three-stage interleaving in the data transmission system. The achieved throughput is 3.1 Gb/s, in three-level pipeline radix-8 architecture.

Voltage scaling is performed for different radix architectures in order to provide some basic guidelines regarding the possible achievable throughputs if the voltage supply is to be increased.

An overview of the results obtained in different studies dealing with high-speed Viterbi decoders during the last decade is shown in Tbl. 4.2. In the table, the results of the non-interleaved design are presented, i.e. the design with which the other studies are compared is radix-4, with 2 Gb/s throughput. As can be seen from the table, the only comparable results to the 2Gb/s presented in this report is [10] where a throughput of 1.74Gb/s was
reached in 180nm technology. This suggests that the results of the architecture used in this study (MSB first) could get close to the performance of the architecture presented in this report if the 90nm process is to be used. The study [11] shows a reconfigurable Viterbi decoder in 90nm technology, with the highest throughput of 3.8Gb/s, however this is done for a 16-state decoder, with a 1.3V supply voltage. In the same study, for a 64-state Viterbi decoder, achieved throughput is 0.9Gb/s. The hardware efficiency is given in mm²ns, instead of GEns, since different technologies are used in different studies, which means different GE should be used. This way, the scaling of the results for different designs is more straightforward.
The impact of interleaving on area.

Figure 4.10: The impact of interleaving on area.

Area distribution in the three-stage interleaved radix-8 architecture.

Figure 4.11: Area distribution in the three-stage interleaved radix-8 architecture.
Figure 4.12: Throughput — area trade-off in radix-8 architecture with three-stage interleaving.
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>250</td>
<td>0.55</td>
<td>0.55</td>
<td>16</td>
<td>-</td>
<td>2.25</td>
<td>570</td>
<td>1.036</td>
<td>-</td>
<td>-</td>
<td>st.cells</td>
<td>postlayout sim.</td>
<td>2000</td>
</tr>
<tr>
<td></td>
<td>180</td>
<td>0.5</td>
<td>0.5</td>
<td>8</td>
<td>2/4</td>
<td>1.8</td>
<td>400</td>
<td>0.8</td>
<td>4</td>
<td>8</td>
<td>st.cells</td>
<td>IC</td>
<td>2003</td>
</tr>
<tr>
<td></td>
<td>180</td>
<td>1</td>
<td>0.145</td>
<td>16</td>
<td>4</td>
<td>1.3</td>
<td>-</td>
<td>0.1-2</td>
<td>3.3</td>
<td>0.5-8.63</td>
<td>st.cells</td>
<td>IC</td>
<td>2003</td>
</tr>
<tr>
<td></td>
<td>90</td>
<td>3.8-0.22</td>
<td>7.6-0.44</td>
<td>16-256</td>
<td>2/4</td>
<td>1.3</td>
<td>353-441</td>
<td>0.1-2</td>
<td>1.9</td>
<td>0.5-8.63</td>
<td>st.cells</td>
<td>IC</td>
<td>2007</td>
</tr>
<tr>
<td></td>
<td>180</td>
<td>-</td>
<td>-</td>
<td>64</td>
<td>4</td>
<td>1.8</td>
<td>379</td>
<td>0.399</td>
<td>-</td>
<td>-</td>
<td>st.cells</td>
<td>postsynthesis sim.</td>
<td>2008</td>
</tr>
<tr>
<td></td>
<td>90</td>
<td>0.95</td>
<td>1.9</td>
<td>64</td>
<td>4</td>
<td>1.8</td>
<td>-</td>
<td>0.399</td>
<td>-</td>
<td>-</td>
<td>st.cells</td>
<td>postlayout sim.</td>
<td>2011</td>
</tr>
</tbody>
</table>

Table 4.2: Overview of the articles on Viterbi decoder.
Chapter 5

Application of a Viterbi Decoder in a POF Data Transmission System

5.1 System Model

Plastic optical fiber is a type of optical fiber which by its characteristics differs from the widely used glass optical fiber. As opposed to a glass optical fiber, plastic optical fiber is easier and less expensive to install and it has higher tolerance for mechanical damage and electromagnetic interference [15]. On the other hand, POF’s disadvantages are larger dimensions, smaller bandwidth and higher attenuation. Because of these characteristics, POF is typically used for short-distance data transmission (up to 100$m$), in medical and automotive industry, home networks, digital audio/video interfaces and light signs/illumination.

POF-based data transmission system described in this report is shown in figure Fig. 5.1. It consists of a module performing an $M$-ary Pulse Amplitude Modulation (PAM) located in the transmitter ($M \in \{2, 4, 8\}$), a plastic optical fiber and a Viterbi decoder implementing MLSE at the receiver’s end. As already described in Sec. 2.3.2, POF is behaving as an FIR filter while the AWGN originating in the transmission system is modelled as additive noise at the input of the receiver. Using a Matlab model, several different system setups are explored in this report, in order to estimate which set of system parameters is the most optimal for obtaining good quality data transmission (BER at the output should be less than $10^{-3}$) for the specified throughput of $1Gb/s$ and different fiber lengths ($L$). Sampling frequencies implemented in the system and their corresponding PAM modulation levels are: $f_s|_{M=2} = 1GH\underline{z}$, $f_s|_{M=4} = 0.5GH\underline{z}$ and $f_s|_{M=8} = 0.333GH\underline{z}$. Viterbi decoder architectures used for the decoding in 2-PAM, 4-PAM and 8-PAM system setups are radix-2, radix-4 and radix-8, respectively.

In order to make the system performance easily comparable to other similar systems, a unit energy is required for the signal at the output of the modulator, as well as for the
one at the output of the fiber. In order to achieve a unit energy per symbol \( E_s(x) = 1 \) at the output of the modulator it is assumed that the probability of individual symbols \( (s_i) \) appearing in the modulator is equal, hence:

\[
E_s(x) = E \{ x^2 \} = P(s_i) \sum_{i=0}^{M-1} s_i^2 = \frac{1}{M} \sum_{i=0}^{M-1} s_i^2 = 1, \tag{5.1}
\]

where \( E\{x\} \) is a mathematical expectation of \( x \), defined as \( E\{x\} = \int_{-\infty}^{+\infty} xf(x)dx \), \( f(x) \) probability density function of \( x \), and \( M \) is modulation order.

Accordingly, symbol vectors \( (s) \) chosen for the three different modulation orders, implementing Gray’s coding, are following:

\[
s|_{M=2} = (0 1) \cdot \sqrt{2},
\]

\[
s|_{M=4} = (0 1 3 2) \cdot \sqrt{\frac{4}{14}},
\]

\[
s|_{M=8} = (0 1 3 2 7 6 4 5) \cdot \sqrt{\frac{2}{35}}.
\]

The way of obtaining unit energy per symbol at the fiber’s output \( E_s(\bar{y}) = 1 \) is presented in the following lines. Having in mind (2.10) which describes convolution between fiber’s input signal \( x \) and its impulse response \( h^c \), energy of the fiber output signal is:

\[
E_s(\bar{y}) = E \{ \bar{y}^2 \}
= E \left\{ \left( \sum_{k=0}^{K-1} h_k^c x_k \right)^2 \right\}
= E \left\{ \sum_{i=0}^{K-1} \sum_{j=0}^{K-1} h_i^c x_i h_j^c x_j \right\}
= \sum_{i=0}^{K-1} \sum_{j=0}^{K-1} E \{ h_i^c x_i h_j^c x_j \}
= \sum_{i=0}^{K-1} \sum_{j=0}^{K-1} h_i^c h_j^c E \{ x_i x_j \},
\]
5.2. PERFORMANCE EVALUATION FOR DIFFERENT SYSTEM PARAMETERS

As there is no correlation between values of signal $x$ at different points of time, $E\{x_i x_j\}$ differs from 0 only in cases where $i = j$, which means:

$$E_s(\bar{y}) = \sum_{k=0}^{K-1} (h^c_k)^2 E\{x_k^2\}.$$ 

Having in mind (5.1), the conclusion follows:

$$E_s(\bar{y}) = \sum_{k=0}^{K-1} (h^c_k)^2 = 1.$$ 

Fiber’s impulse response ($h^c$) is found by way of calculating inverse discrete Fourier transform of the fiber’s frequency characteristics given by the following expression in its continuous form:

$$|H_{POF}(f)| = \frac{1}{\sqrt{1 + \left(\frac{f}{f_g}\right)^2}}, \quad (5.2)$$

where $f_g$ is cutoff frequency of the filter whose value is calculated from the ratio between fiber’s Bandwidth-Length Product and its length ($f_g = \frac{BWL}{L}$). BWL represents one of the optical fiber’s main properties, which for the POF described in this report has a value $BWL = 30 \text{MHz} \cdot 100\text{m}$.

An example of POF’s impulse response is depicted in Fig. 5.2. This impulse response corresponds to a channel with a memory where a signal inputted at time $n$ impacts the channel outputs until time $n + K$, where $K$ is the length of the impulse response. This effect is called intersymbol interference. By observing the presented equations, it can be seen that the impulse response shape depends on the length of the fiber and the sampling frequency, i.e. the effect of the ISI depends on the fiber’s length and sampling frequency.

5.2 Performance Evaluation for Different System Parameters

As POF’s impulse response can contain large number of non-zero samples/taps, it is necessary to determine the optimal number of taps that need to be taken into account in the Viterbi decoder with purpose of emulating the fiber’s behaviour (Sec. 2.3.2). In other words, it is necessary to determine the order ($N$) of the FIR filter which is to be used in branch metrics calculation in order to emulate the fiber’s behaviour. For obtaining
CHAPTER 5. APPLICATION OF A VITERBI DECODER IN A POF DATA TRANSMISSION SYSTEM

the optimal filter’s order a trade-off between BER at decoder’s output, the design area overhead and power consumption has to be performed.

In Fig. 5.4 the resulting BER plot is presented as a function of $Eb/N_0$ for 2-PAM modulation, with non-constrained metrics representation, 64 state Viterbi decoder and for different channel lengths. Keeping in mind that it is required that $BER < 10^{-3}$, this figure shows that the longest channel for which the decoding quality is at the acceptable level is between 35 and 50m. The reason for this is the shape of the fiber’s impulse response which for longer fibers contains larger number of non-zero taps, causing the effects of ISI to be more prominent. This can be noticed by observing Fig. 5.3 where $L = 100\,m$ and comparing it to Fig. 5.2 where $L = 25\,m$.

A way to deal with the increased ISI is to consider larger number of taps during the branch metrics calculation, i.e. increase the FIR filter’s order. This implies using a larger number of states in the Viterbi decoder. The effects of this can be seen in Fig. 5.5. Here the results for different channel lengths are shown, for $f_s = 1\,GHz$ and 1024 states. After comparing Fig. 5.4 and Fig. 5.5 it is obvious that there is a significant improvement of decoding quality for a 50m fiber. Using a filter order of $N = 10$, instead of $N = 6$ as in case of a 64-state Viterbi decoder, means that 4 additional taps are taken into consideration in the branch metrics calculation, which is enough to bring improvement for 50m channel system. However, MLSE in case of 100m channel does not show any noticeable improvement, which is the consequence of 100m channel impulse response containing a

![Energy of impulse response samples, $f_s=1GHz$, $L=25m$](image)

Figure 5.2: POF’s impulse response for $f_s = 1\,GHZ$ and $L = 25\,m$. 
large number of non-zero taps, i.e. ISI effects are more intense. Fig. 5.5 can be used to estimate the critical length of the fibre above which the MLSE in Viterbi decoder does not perform a satisfiable ISI cancellation — the critical length is around 65 m. In the similar fashion, critical lengths for different number of states in the decoder can be estimated. An overview of this is given in Tbl. 5.1.

In order to see how the sampling frequency affects the decoding quality, simulations for the three different sampling rates are performed on the fiber with a fixed length of \( L = 25 m \) and the plots are presented in Fig. 5.6. As it can be seen, the best results are obtained using the highest sampling frequency \( (f_s = 1GHz) \). The distance between two symbols used in the modulator is smaller in case of a smaller sampling frequencies than in case where \( f_s = 1GHz \). Even though Gray’s coding was used in the modulator in case of 4-PAM and 8-PAM modulation, the distance between two consecutive symbols in the case of 2-PAM is significantly larger, which is the reason why the results of ISI cancellation are better in the case of 2-PAM modulation.

Figure 5.3: POF’s impulse response for \( f_s = 1GHZ \) and \( L = 100m \).
5.3 Hardware Mapping and Synthesis Results

Prior to entering the Viterbi decoder module, the signal from the fiber has to be quantised. Since it is found that the longest fiber for which a correct signal recovery can be performed is around 65m, with 1024-state Viterbi decoder, the quantisation is performed for the 60m long fiber. Determining the widths of integer and fractional parts of different metrics in the decoder is done by way of Matlab simulations, based on (2.11), equation for calculating branch metrics. Having in mind that the branch metrics is carrying the most relevant information, it is this metrics that is quantised first, keeping the others at large widths in order to emulate an unconstrained-widths representation. By observing the values branch metrics is taking, it is determined that the integer part width has to be 4. After fixing the integer part width to 4, a sweep over a range of values for fractional part width is performed and the results are presented in Fig. 5.7. As it can be seen from the figure, the required width is 4, since the desired decoding quality is $BER < 10^{-3}$ and the larger widths do not introduce significant improvement of performance. The improvement of BER at $Eb/N_0 = 30dB$ for 6-bits wide fractional part, compared to the case with ideal metrics in Fig. 5.5 is a simulation artifact.

The next step is to find the exact fix-point representation of input signal ($y$). Again,
it is observed that the required integer part width is 2. After fixing this value for the integer part, from the simulations and its resulting plot (Fig. 5.8) it is determined that the fractional part width of the input signal is 3, since additional bits do not introduce a significant improvement in performance.

Because of the way the (2.11) is implemented in hardware, it is necessary to store the intermediate values of sums, which means that it is necessary to determine the optimal fix-point representation for this metric as well. First a sweep over a range of values is performed for width of the integer part and the resulting plot is shown in Fig. 5.9. The number of bits chosen for the integer part representation is 5, since the corresponding curve is overlapping with the curve for 6 bits, meaning that larger number of bits does not introduce an improvement.

In the similar fashion, based on the simulation results, as well as the estimation by observing the possible values metrics can take, the presentation of the remaining metrics is determined. After synthesis of the 1024-bit design, the area covered by the design was estimated by the Synopsys to be around 15 mm$^2$, which is much larger than the value found to be optimal in this case (around 4 mm$^2$). Looking back to the Tbl. 5.1, it can be seen that in order to get a design which would be compliant with the specified area coverage, one will have to make a trade-off w.r.t. the maximal fiber’s length. It is noticed that in case of a 256-states decoder, the critical fiber’s length is around 55 m, i.e. 10 m shorter than in the case of a 1024-states decoder. It is found that the lost on the fiber’s length is
 CHAPTER 5. APPLICATION OF A VITERBI DECODER IN A POF DATA TRANSMISSION SYSTEM

56

Table 5.1: Overview of the maximal lengths of POF for different number of Viterbi states.

<table>
<thead>
<tr>
<th>No. of states</th>
<th>max length [m]</th>
<th>SNR [dB]</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>45</td>
<td>20</td>
</tr>
<tr>
<td>128</td>
<td>50</td>
<td>23</td>
</tr>
<tr>
<td>256</td>
<td>55</td>
<td>26</td>
</tr>
<tr>
<td>512</td>
<td>60</td>
<td>28</td>
</tr>
<tr>
<td>1024</td>
<td>65</td>
<td>30</td>
</tr>
</tbody>
</table>

not too high, so a 256-states decoder is synthesised. As anticipated, the area of this design is more optimal, and it is around 2.6 mm².

5.4 Conclusion

This chapter presents a study of a Viterbi algorithm employing the maximum likelihood estimation method for intersymbol interference cancelation in case of a data transmission system based on a plastic optical fiber. It is shown that this method of data recovery is not optimal for plastic optical fibers of large lengths (more than 65m) having in mind that a large number of states needs to be implemented in the decoder, which consequently leads to a large area overhead and a large power consumption. Viterbi decoder based on MLSE is suitable for short-distance transmission systems, showing satisfiable ability to perform data recovery (it is possible to ensure BER$10^{-3}$ at the output of the decoder for the values of SNR larger than 15dB). A 1024-states decoder is first implemented in order to reach fiber length of 65m, but as it is presented, after synthesis the consumed area is estimated to be around 15 mm², which would be non-optimal area consumption for most systems. Hence, a 256-states decoder is synthesised, reaching an optimal 2.6 mm² in area and ensuring a maximum of 55 m for POF length.
Figure 5.6: BER as a function of $Eb/N_0$ for different sampling rates and 64 states Viterbi decoder, without metrics quantisation.
Figure 5.7: Sweep over fractional part width for branch metric.

Figure 5.8: Sweep over fractional part width for input signal metric.
Figure 5.9: Sweep over fractional part width for input signal metric.
CHAPTER 5. APPLICATION OF A VITERBI DECODER IN A POF DATA TRANSMISSION SYSTEM
Chapter 6

Conclusion

As the need for high-speed data transmission is always present, in this report a high-speed VLSI implementation of the Viterbi algorithm is presented. Having the high throughput as a main goal, different optimisation methods are introduced and implemented leading to multi-Gb/s Viterbi decoder circuits. The best achieved throughput is 3.1 Gb/s, which is obtained using 90nm CMOS technology, with voltage level of 1 V at a temperature of 25°C. The simulations are performed using Synopsys and Cadence Encounter tools, whereas the design parameters are evaluated using a Matlab model. A trade-off between throughput, energy efficiency and area of the different designs is presented as well. The evaluations of designs’ performances are done using an automated reconfigurable script written in Command Shell and Perl scripting languages. For reaching even larger throughputs, the implementation of a so-called MSB first architecture [16] is recommended. This approach is more complex for the implementation, but it could provide a speed-up w.r.t. the architecture presented in this report.

In the second part of the report, an implementation of the Viterbi algorithm which is employing MLSE is presented. This Viterbi decoder is performing ISI cancelation in a data transmission system based on Plastic optical fiber. Different system parameters are evaluated based on the throughput specifications using a Matlab model. It is concluded that the optimal architecture of Viterbi decoder for achieving specified BER at the receiver is radix-2 with 256 states. Using this Viterbi decoder, it is possible to cancel the ISI in a POF up to 55 m of length. In order to employ the Viterbi decoder in a data transmission system with longer fibers, it is necessary to use larger number of states in the decoder, which in return brings a large area overhead.
Appendix A

Block Diagrams and Chip Interface

A.1 Handshake Interface

The need to implement an interface between the design and the outside world comes from the fact that the speed and the pattern in which data are being written in the decoder differ from the speed and the pattern data are being read from the decoder. The direction of the communication follows the flow of data in the decoder - from the input of the decoder towards the output. Handshake protocol implemented here consists of two signals - Request signal (Req) and Acknowledge signal (Ack). The purpose of the Request signal is to communicate to a stage that the stage preceding it has valid data at its output and that it is ready to forward the data to the following stage. Once the input Request signal is asserted, in case the stage is ready to receive new set of data, it responds by asserting Acknowledge signal which communicates that the data can be accepted.

The hardware responsible for generating Request signal is implemented as a sequential circuit, in form of a Finite State Machine (FSM), whereas the hardware responsible for Acknowledge signal generation is implemented in form of a combinational logic which reacts to the assertion of the received Request signal. The reason behind this is the need to perform data transfers in consecutive clock cycles. This approach, on the other hand, introduces additional speed-critical combinational paths in the design.

The diagrams in Fig. A.1 depict an example of a single data transfer. When the sender requests data transfer by asserting Request signal, the values at its output have to be valid. In case the receiver is ready for accepting the data, Acknowledge signal will be asserted right after the detection of Request signal assertion and the data transfer will occur at the next rising clock edge. After this, since there is no more data to be transferred, Request signal will be deasserted right after the clock edge, which will then result in deassertion of Acknowledge signal. In case where receiver is not ready to accept the data, Request signal remains asserted and the sender holds the values on its output for as many cycles as the receiver needs to get ready to receive the data. After a rising clock edge, if internal logic
signals that the receiver is ready to accept the data, Acknowledge signal will be asserted and data transfer will be performed at the following rising clock edge.

An example of a multiple-data transfer is depicted in the diagrams in Fig. A.2. Here, if after a successful data transfer the sender has more data to send to the next stage, the Request signal will stay asserted. Depending on the status of the receiver, it will respond by leaving Acknowledge signal asserted and performing a data transfer at each consecutive rising clock edge, or by deasserting it and waiting for the following rising clock edge to decide whether it is ready to receive the data or not.
A.2 Block Diagrams

In the figures Fig. A.3 - Fig. A.7 a detailed structures of the Viterbi decoder and its modules are presented. The figures show the exact naming of the interfacing signals between different modules.
Figure A.4: Branch metric unit, detailed structure.
Figure A.5: Add-Compare-Select unit, detailed structure.
Figure A.6: Add-Compare-Select module, detailed structure.
Figure A.7: Register exchange unit, detailed structure.
Appendix B

Project task
Area- and Energy-Efficiency Trade-offs in the VLSI Implementation of High-Speed Viterbi Decoders and their Application to MLSE in POF-based Systems

February 2, 2011

Advisors: Andreas Burg (TCL-EPFL), andreas.burg@epfl.ch
Christoph Roth (IIS-ETHZ), rothc@iis.ee.ethz.ch
Alessandro Cevrero (LSM-EPFL), alessandro.cevrero@epfl.ch

Handout: September 6, 2010
Due: March 4, 2011

Four copies of the written report are to be turned in. All copies remain property of the Integrated Systems Laboratory.
1 Introduction

The Viterbi decoder is one of the most prominent components in digital communication systems and storage devices. The underlying Viterbi algorithm [1] is an optimal solution to the problem of estimating the state sequence of a discrete-time finite-state Markov process observed in memoryless noise. Initially proposed as a method of decoding convolutional forward-error-correction codes, the Viterbi algorithm has become a widely used approach to solve various other problems in the area of digital communications that can be cast in the form of a finite-state Markov process (e.g., intersymbol-interference (ISI) cancellation).

The ever-increasing data rates of modern communication systems have created the need for high-speed Viterbi decoders able to cope with the stringent throughput requirements of respective standards. Recently, the employment of new-generation wireless standards such as IEEE 802.11n [4] or IEEE 802.16e has risen the throughput requirements to the range of hundreds of Mbps. Furthermore, upcoming standards such as WirelessHD or WiGig as well as emerging applications such as communication over Plastic Optical Fiber (POF) [5] links are expected to further rise throughput requirements above the 1 Gbps mark.

While the basic digital VLSI implementation of the Viterbi algorithm is a well-known task, the optimization for high throughput has remained challenging, mainly due to the recursive nature of the algorithm. At the same time, the high mobility foreseen by recent standards has also increased the need for area-efficient and energy-efficient implementations as corresponding circuits are usually operated on mobile battery-powered devices.

2 Project Description

This project is divided into two parts. In the first part, a fully IEEE 802.11n-compliant Viterbi decoder implementation for decoding convolutional codes will serve as a reference design for investigating area-efficiency and energy-efficiency trade-offs in the digital implementation of Viterbi decoders with the main goal of achieving high throughput. Thereby, the main focus of evaluation will lie on the architecture down to the physical level, and the considered technology will be 90 nm CMOS. In order to make the evaluation process as efficient as possible, the VLSI design flow used at IIS (including front end design and back end design) as well as the power analysis of the created physical designs and consolidation of results will be fully automated.

In the second part of this project, the focus of attention will be changed to the task of ISI cancellation in POF-based communication systems. Depending on the fiber length of these systems, strong ISI, introduced due to multimode propagation and other effects, can degrade the error-rate performance of the system considerably. A Viterbi-based maximum-likelihood sequence estimation (MLSE) receiver will thus be implemented to mitigate the effects of ISI. To this end, a MATLAB simulation framework based on PAM modulation as indicated in Fig. 1 will first be developed to simulate the POF transmission and to investigate the influence of fiber length and sampling rate on the uncoded error-rate performance of the system. Thereby, the number of states in the Viterbi decoder will be fixed to 64. A comparison in terms of error-rate performance of different combinations of modulation order and fiber length will lead to the best solution, which will finally be implemented most efficiently in 90 nm CMOS technology based on the insights gained in the first part of the project.
3 Goals

The goals of this thesis are to investigate the area-efficiency and energy-efficiency trade-offs in the digital implementation of Viterbi decoders with strong focus on high throughput as well as to gain insights about the MLSE performance of such a decoder in a high-throughput POF system. The following tasks need to be accomplished during this project:

- Thorough understanding of the Viterbi algorithm and its application to convolutional codes and MLSE.
- Developing a flexible and generic evaluation framework automating the VLSI design flow at IIS and evaluating appropriate power-characterization methods.
- Optimizing a reference Viterbi decoder for high throughput.
- Evaluation of the 64-state Viterbi-based MLSE performance in a POF communication system considering different modulation orders, fiber lengths, and sampling rates.
- Implementing the most promising approach in terms of error-rate performance, hardware-efficiency, and energy-efficiency in 90 nm CMOS technology.

4 Milestones

The following milestones should be accomplished during this project. Note that some milestones can be added or skipped, depending on the project’s status.

1. Thorough understanding of the Viterbi algorithm (e.g., [1, 2, 7]) and its application to convolutional codes (e.g., [7]) and MLSE (e.g., [3, 7]).

2. Complete the existing MATLAB model of the reference Viterbi decoder design for the encoding and decoding of convolutional codes over an AWGN channel. The model should be fully IEEE 802.11n-compliant [4] including the support for all puncturing modes.

3. Explore the arithmetic precision requirements in terms of word lengths and trace-back length based on the MATLAB model and complete the VHDL model of the reference design accordingly.
4. Develop and document a generic evaluation framework that fully automates the VLSI design flow used at IIS including all the steps required to convert a VHDL model to a physical layout in the target technology. The setup should also be able to characterize the power of the considered system and to consolidate the gathered results with focus on area-efficiency and energy-efficiency fully automatically.

5. Based on the implemented evaluation framework, evaluate different power-characterization methods with the goal to enable a fast and efficient power-estimation flow which is as accurate as possible in absolute as well as relative terms.

6. Optimize the reference design for high throughput. Identify the bottlenecks of the reference design in terms of throughput and optimize them with the goal to achieve highest possible throughput. Thereby, consider optimizations on the architecture to the physical level.

7. Compare the different high-throughput approaches using the developed evaluation framework. Considering also supply voltage scaling and related increase in gate delays, identify the best design approach for high-throughput Viterbi decoding.

8. Develop a MATLAB simulation model for communication over POF including the modulator, the POF channel, and the Viterbi-based MLSE block.

9. Study the existing trade-offs in the system focusing on modulation order, sampling frequency, and fiber distance. If time permits, also explore different channel models.

10. Identify the best setup and derive a fixed-point MATLAB model of the system. Explore the resulting arithmetic precision requirements for the Viterbi decoder.

11. Implement the high-speed MLSE Viterbi decoder in 90 nm CMOS. Characterize the design and prepare it for tape-out.

12. Write the final report and prepare the presentation slides.

5 General Recommendations

The following are some recommendations for this Master’s Thesis:

- While coding VHDL, use the IIS standard coding style [8] documented by the Design Zentrum (DZ) web-page [6].

- VHDL coding is greatly simplified and accelerated using the Emacs editor and its famous and widely adopted VHDL mode. This Emacs installation at the institute supports among other powerful features VHDL syntax highlighting, signal and component declaration and instantiation, code beautifying, and automated sensitivity list updates based on the VHDL standard. Since most assistants at the IIS are quite familiar with this editor, they can read and evaluate your VHDL code (and help to solve problems) much faster. Please consult the corresponding FAQ under the following link:
6 Project Realization

6.1 Project Plan

Within the third week of the project you will be asked to prepare a project plan. This plan should identify the tasks to be performed during the project and set deadlines for those tasks. The prepared plan will be a topic of discussion of the first week’s meeting between the student and the advisors. Note that the project plan should be updated constantly depending on the project’s status.

6.2 Meetings

Weekly meetings will be held between the student and the assistants. The exact weekly meeting time and location will be determined to fit the schedule of the assistants. These meetings will be used to evaluate the status and progress of the project. If you like to discuss details of your work, please provide appropriate and up-to-date figures and block diagrams.

6.3 Reports

Documentation is an important and often overlooked aspect of engineering. One short intermediate report and one final report (the Master’s Thesis) are to be completed within this study. Note that the intermediate report should be designed to be part of the final report.

The common language of engineering is de facto English. Therefore, the intermediate and final report of the work is preferred to be written in English. Any form of word processing software is allowed for writing the reports, nevertheless the use of \LaTeX with Tg\text{if} (for block diagrams) is strongly encouraged by the IIS staff.

First Intermediate Report  This report should be written in such a way to become the first part of your final report. It should contain general information about the topic, a description of the problem, explanations of related terminology, and descriptions of similar approaches in literature (with corresponding references to books, papers etc.).

Final Report  The final report has to be presented at the end of the project and two copies need to be handed out and remain property of the IIS. These reports are only accepted when the keys for the ETZ building have been properly returned. Note that this task description is part of your thesis and has to be attached to your final report. A data disc (e.g., CD or DVD) containing all essential files of your project should also be added to the final report.

6.4 Presentation

There will be a presentation (20 min presentation and 5 min Q&A) at the end of this project to present your results to a wider audience. The exact date has to be determined.
References


Zurich, February 2, 2011

Prof. Dr. Andreas Burg

The thesis will not be accepted without returning the keys!
Appendix C

Presentation
Design Trade-offs in the VLSI Implementation of High-Speed Viterbi Decoders and their Application to MLSE in ISI Cancellation

Jelena Dragaš

Advisors: Andreas Burg (EPFL-TCL), Christoph Roth (ETHZ-IIS), Alessandro Cevrero (EPFL-LSM)
Introduction

- Need for reliable, error-free data transmission always present, speed requirements grow rapidly
- High data rates lead to significant ISI in band-limited systems
- Viterbi algorithm (VA) - optimal method for finding the most likely path through a trellis diagram
- Applications of VA:
  - mobile, radio and satellite communications;
  - data storage (hard disk drives);
  - Plastic optical fibres (POF) transmission systems
- Goal: Implement VA in ISI cancellation in POF-based system; Perform exploration of different VLSI architectures for obtaining high throughputs, evaluating the trade-offs w.r.t. energy efficiency and area
Outline

- POF Behaviour
- Viterbi Algorithm
- Mapping VA to VLSI
- Reference Architecture
- High-Speed Evaluation
- Application of VA to POF-based systems
- Conclusion
POF Behaviour Mapped to Markov Process

- POF-based data transmission system:

- Fibre modelled as a low-pass FIR filter:
  \[ |H_{POF}(f)| = \frac{1}{\sqrt{1 + (\frac{f}{f_p})^2}}, \]

- FIR filter – mapped to a finite-state Markov process: 
  - \( h \) – FIR filter impulse response coefficients
Markov process data recovery – Viterbi Algorithm

- Example of a 2-bit shift register Markov process:

- Branch metric ($\lambda$) – measure of the probability of a transition between two states
- Path metric ($\Gamma$) – sum of branch metrics leading to a particular state
- Survivor path – the most probable path from the starting to the current state
Hardware Mapping of the Trellis Diagram (1/2)

- Nodes in trellis mapped to Add-Compare-Select (ACS) unit performing the calculation of the path metrics and selection of the survivor metric.
Hardware Mapping of the Trellis Diagram (2/2)

- Lookahead stages in the trellis diagram
  - \( l \) lookahead stages – Radix-2\(^l\) architecture
  - Ideal speedup – \( l \) times
  - Increase of complexity

Fig – Radix-2 to radix-4
Determining the Survivor Path - Tracing Back the Decisions

- Trace back performed in the Register Exchange (RE) unit, mimicking the structure of the trellis diagram
Viterbi Decoder (VD)

- Viterbi decoder – the hardware module implementing VA
- Three main blocks:

  - Branch Metric Unit (BMU) – calculates branch metrics
  - Path Metric Unit (PMU) – storing the survivor path metrics
ACS Unit – Reference Architecture

- Speed critical as the result of the feedback loop and the complex architecture
Architecture Evaluation Flow

- Automated, reconfigurable script performing clock frequency sweep:
Performance Comparison of the Different Radices (1/2)

- 90nm CMOS process:

<table>
<thead>
<tr>
<th></th>
<th>Radix-2</th>
<th>Radix-4</th>
<th>Radix-8</th>
<th>Radix-16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput [Gb/s]</td>
<td>1.287</td>
<td>2.03</td>
<td>1.98</td>
<td>1.85</td>
</tr>
<tr>
<td>Energy Efficiency [nJ/bit]</td>
<td>0.22</td>
<td>0.36</td>
<td>0.54</td>
<td>1.21</td>
</tr>
<tr>
<td>Area [mm²]</td>
<td>0.233</td>
<td>0.54</td>
<td>1.17</td>
<td>2.88</td>
</tr>
<tr>
<td>Hardware Efficiency [GEqs]</td>
<td>57398</td>
<td>86097</td>
<td>188138</td>
<td>494260</td>
</tr>
</tbody>
</table>

- Length of the combinational path in the ACS grows rapidly with the radix due to the increase in the number of comparators:
  - R – radix:
    \[
    N_{COMP} = \frac{R(R - 1)}{2}
    \]
Performance Comparison of the Different Radices

(2/2)
Dealing with Long Data Paths - Data Interleaving

- Pipelining a module containing a feedback loop
  - Several data streams interleaved into a single one and decoded independently

![Diagram showing data flow and pipelining](image-url)
Impact of Interleaving on Throughput for Different Radices

- Impact of pipelining on different radix architectures, for different pipelining levels

- Best achieved throughput: 3.1Gb/s for radix-8 architecture, with 2-stage interleaving

- Almost no impact on area and energy efficiency – registers “replace” buffers in the data paths
## Comparison of the Viterbi Decoder Studies

<table>
<thead>
<tr>
<th>Paper</th>
<th>Yeo'03</th>
<th>Dinh'03</th>
<th>Anders'07</th>
<th>This study</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technology [nm]</td>
<td>180</td>
<td>180</td>
<td>90</td>
<td>90</td>
</tr>
<tr>
<td>Speed [Gb/s]</td>
<td>0.5</td>
<td>1</td>
<td>2</td>
<td>0.95</td>
</tr>
<tr>
<td>Frequency [GHz]</td>
<td>0.5</td>
<td>0.145</td>
<td>1.9</td>
<td>2</td>
</tr>
<tr>
<td>No. of States</td>
<td>8</td>
<td>16</td>
<td>64</td>
<td>64</td>
</tr>
<tr>
<td>Radix</td>
<td>2/4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Supply Voltage [V]</td>
<td>1.8</td>
<td>-</td>
<td>1.3</td>
<td>1</td>
</tr>
<tr>
<td>Energy Efficiency [nJ/bit]</td>
<td>0.8</td>
<td>0.12</td>
<td>-</td>
<td>0.308</td>
</tr>
<tr>
<td>Area [mm²]</td>
<td>4</td>
<td>3.3</td>
<td>1.9</td>
<td>0.54</td>
</tr>
<tr>
<td>Hardware Efficiency [GEs]</td>
<td>318878</td>
<td>130740</td>
<td>637755</td>
<td>86097</td>
</tr>
<tr>
<td>Cells Type</td>
<td>st.cells, custom clk tree</td>
<td>st.cells</td>
<td>custom cells</td>
<td>st.cells</td>
</tr>
<tr>
<td>Verification</td>
<td>IC</td>
<td>IC</td>
<td>IC</td>
<td>simulation</td>
</tr>
</tbody>
</table>

*scaled*
POF System Parameters Exploration – Impact of the Modulation Level on the Error-Rate Performance

- The same throughput obtained using different sampling frequencies and corresponding radix levels in the Viterbi decoder.

- Optimal architecture – radix-2 - due to the largest distance between symbols.

- Radix-2 architecture most hardware and energy efficient for this throughput.
POF System Parameters Exploration – Impact of the Fibre’s Length on the Error-Rate Performance

- The longer the fibre, the more prominent the ISI is:

Fig - 64-states VD, 2-PAM modulation
POF System Parameters Exploration – Impact of the Number of States in the VD on Data Recovery

- The larger the number of states, the more efficient the ISI cancellation is – Viterbi decoder emulates the fibre’s behaviour more truthfully considering large number of impulse response taps.

![Graph showing BER vs. Eb/N0 for different max lengths and SNRs.](image)

<table>
<thead>
<tr>
<th># of states</th>
<th>max length [m]</th>
<th>SNR [dB]</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>45</td>
<td>20</td>
</tr>
<tr>
<td>128</td>
<td>50</td>
<td>23</td>
</tr>
<tr>
<td>256</td>
<td>55</td>
<td>26</td>
</tr>
<tr>
<td>512</td>
<td>60</td>
<td>28</td>
</tr>
<tr>
<td>1024</td>
<td>65</td>
<td>30</td>
</tr>
</tbody>
</table>

Fig - 1024-states VD, 2-PAM modulation
Design Synthesis Results

- **1024-states Viterbi decoder:**
  - Maximal fibre length: 65m
  - Area after synthesis: 15mm²

- **256-states Viterbi decoder:**
  - Maximal fibre length: 55m
  - Area after synthesis: 3.3mm²
  - Lost 10m in POF’s length, but 4 times gain in area
Conclusion

- Highest achieved throughput without interleaving: 2 Gb/s for radix-4 architecture
  - With interleaving: 3.1 Gb/s for radix-8 architecture

- The increase in radix brings improvement in throughput only when going from radix-2 to radix-4

- Hardware and energy efficient architectures: radix-2 and radix-4 for different ranges of throughput

- Reasonable to use Viterbi decoders in ISI cancellation for fibres up to 55m with 2-PAM modulation
Thank you for your attention!

Questions?
Bibliography


