

**A Smart Implementation of Turbo Decoding for
Improved Power Efficiency**

By

Abayomi Jemibewon

Thesis Submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

Electrical Engineering

Approved:

Dr. Brian D. Woerner, Chairman

Dr. Jeffery H. Reed

Dr. Bill Davis

July 7, 2000

Blacksburg, Virginia

Keywords: Turbo Codes, Analog-to-digital converter, Variable resolution

A Smart Implementation of Turbo Decoding for Improved Power Efficiency

Abayomi Jemibewon

ABSTRACT

Error correction codes are a means of including redundancy in a stream of information bits to allow the detection and correction of symbol errors during transmission. The birth of error correction coding showed that Shannon's channel capacity could be achieved when transmitting information through a noisy channel. Turbo codes are a very powerful form of error correction codes that bring the performance of practical coding even closer to Shannon's theoretical specifications. Bit-error-rate (BER) performance and power dissipation are two important measures of performance used to characterize communication systems. Subject to the law of diminishing returns, as the resolution of the analog-to-digital converter (ADC) in the decoder increases, BER improves, but power dissipation increases. The number of decoding iterations has a similar effect on the BER performance and power dissipation of turbo coded systems. This is significant since turbo decoding is typically practiced in a fixed iterative manner, where all transmitted frames go through the same number of iterations. This is not always necessary since certain "good" frames would converge to their final bits within a few iterations, and other "bad" frames never do converge.

In this thesis, we investigate the technical feasibility of adapting the resolution of the ADC in the decoder, and the number of decoding iterations, in order to obtain the best trade-off possible between BER performance and power dissipation in a communication system. With the aid of computer-aided simulations, this thesis investigates the performance and practical implementation issues associated with incorporating a variable resolution ADC into the decoder structure of turbo codes. The possibility of further power conservation resulting from reduced decoding computation is also investigated with the use of a recently developed iterative stopping criterion.

Acknowledgements

I would like to extend my gratitude to my committee chairman and advisor Dr. Brian D. Woerner for giving me the opportunity to work with him. Without his continuous support and encouragement, I would not have been successful with my research. I would also like to express my appreciation to Dr. Jeffrey H. Reed and Dr. Bill Davis for participating in my examination committee. I also want to thank Yufei Wu and Bruce Pucket for continuously assisting me on theoretical and implementation issues of turbo codes. Carrie Aust and Jia Fei also deserve my deepest gratitude for providing me with very useful information and assistance on their respective research projects. I also want to extend my gratitude to all the members of the MPRG, most of whom have assisted me in some form or the other towards the completion of my thesis. Last, but most importantly, I want to thank my family for their continuous unconditional love and support in all my life endeavors.

CONTENTS

Acknowledgements.....	III
1. WIRELESS COMMUNICATION SYSTEMS.....	1
1.1 Evolution of Wireless Communication Systems.....	1
1.2 Requirements of Wireless Communication Systems.....	2
1.3 Error Correction Codes.....	2
1.3.1 Block Codes.....	3
1.3.2 Convolutional Codes.....	3
1.3.3 Turbo Codes.....	4
1.4 Practical Implementation of Turbo Codes.....	5
1.4.1 Quantization.....	6
1.4.2 Fixed Point Representation.....	6
1.5 Smart Decoding of Turbo Codes.....	7
1.6 Thesis Outline.....	7
2. TURBO CODES: FUNDAMENTALS.....	9
2.1 Block Codes.....	9
2.2 Convolutional Codes.....	10
2.3 Recursive Systematic Convolutional (RSC) Codes.....	12
2.4 Turbo Codes.....	12
2.4.1 The Turbo Encoder.....	12
2.4.2 The Turbo Decoder.....	14
2.4.3 Operation of a Turbo Decoder.....	15
2.4.4 Decoding Algorithms.....	16
2.4.5 Maximum A Posteriori (MAP) Algorithm.....	17
2.4.6 Log-MAP Algorithm.....	20
2.4.7 Performance Issues of Turbo Codes.....	22
3. ANALOG-TO-DIGITAL CONVERTERS (ADCs).....	25
3.1 Basic Concepts of Analog-to-Digital Conversion.....	25
3.1.1 Sampling.....	26

3.1.2 Anti-Alias Filtering	27
3.1.3 Quantization.....	27
3.2 Performance Measurements of ADCs	28
3.2.1 Resolution and Accuracy.....	28
3.2.2 Signal-to-Noise-Ratio (SNR)	29
3.2.3 Spurious Free Dynamic Range (SFDR).....	30
3.2.4 Power Dissipation	30
3.3 Practical ADC Architectures.....	31
3.3.1 Parallel Converter Architecture	31
3.3.2 Successive Approximation Architecture.....	32
3.3.3 Sigma-Delta Converter Architecture	33
3.4 Chapter Summary	34
4. ADAPTER RECEIVER DESIGN FOR POWER OPTIMIZATION	35
4.1 Effect of Quantization on BER Performance of Turbo Codes.....	35
4.2 A Variable Resolution ADC.....	38
4.2.1 RSD Cyclic ADC Architecture.....	38
4.2.2 Variable ADC Simulation	39
4.3 Performance considerations of a Variable Resolution Turbo Decoder	39
4.3.1 Performance Considerations for Different Data Rates	40
4.3.2 Link Analysis.....	41
4.3.3 A Personal Communication System (PCS) Link.....	42
4.3.4 Analysis of a Wireless Local Area Network (WLAN) Link.....	44
4.3.5 A Local-Multipoint distribution system (LMDS) network.....	45
4.4 Optimizing Decoding Iterations for further Power Conservation	46
5. CONCLUSION AND FUTURE WORK.....	52
5.1 Future Work	52
Bibliography.....	54
Vita.....	53

LIST OF TABLES

<u>Table 4.1: Power dissipation of variable resolution ADC</u>	39
<u>Table 4.2: Power dissipation of ADC for different data rates</u>	41
<u>Table 4.4: Path loss exponents for different propagation environments</u>	42
<u>Table 4.4: PCS link specifications</u>	43
<u>Table 4.5: Transmitter powers for PCS link</u>	43
<u>Table 4.6: WLAN link specifications</u>	44
<u>Table 4.7: Transmitter powers for WLAN link</u>	44
<u>Table 4.8: LMDS link specifications</u>	45
<u>Table 4.9: Transmitter powers for LMDS link</u>	46
<u>Table 4.10: Energy consumption per decoding iteration</u>	49
<u>Table 4.11: Power savings of the SDR turbo decoder</u>	50
<u>Table 4.12: Energy considerations for SDR-decoding</u>	50

LIST OF FIGURES

1.1	Structure of a turbo encoder.....	4
1.2	Structure of a turbo decoder.....	5
2.1	Convolutional encoder with $K_c=3$	10
2.2	State diagram for sample convolutional code.....	11
2.3	Trellis diagram for example convolutional code.....	11
2.4	A rate 1/2 RSC encoder with $K_c=4$	12
2.5	Turbo encoder using sample RSC encoder.....	13
2.6	Structure of a soft-input soft-output decoder.....	15
2.7	Schematic of a turbo decoder.....	16
2.8	Simulated BER performance of a rate=1/2, $k_c=4$, turbo code for different frame sizes.....	22
2.9	Simulated BER performance for a rate=1/2, $k_c=4$, turbo code for varying number of decoding iterations.....	23
3.1	Functional blocks of an ADC.....	25
3.2a	Spectrum of a bandlimited analog signal.....	26
3.2b	Spectrum of a Nyquist sampled signal.....	26
3.2c	Spectrum of an under-sampled signal.....	26
3.2d	Spectrum of an over-sampled signal.....	26
3.3a	Illustration of ADC resolution for a 2-bit quantizer.....	29
3.3b	Illustration of ADC resolution for a 3-bit quantizer.....	29
3.3c	Illustration of ADC resolution for a 4-bit quantizer.....	29
3.4	Parallel ADC architecture.....	31
3.5	Successive approximation architecture.....	33
3.6	The delta-sigma architecture.....	33
4.1	System block diagram.....	36
4.2	BER for different resolution quantizers.....	37
4.3	BER performance of fixed decoding versus SDR decoding.....	47
4.4	Average number of iterations for fixed and SDR decoding.....	48
4.5	BER performance using different ADC resolutions with and without SDR stopping criteria.....	48
5.1	Smart turbo decoder.....	52

CHAPTER 1

WIRELESS COMMUNICATION SYSTEMS

The last thirty years have seen a dramatic change in the way communication is achieved around the world. Wireless communication has evolved from being an expensive and rare technology for the few in the 70's, to becoming a widespread and economical means for facilitating domestic, commercial, as well as public service communications. One of the major reasons for the continuous growth in the use of wireless communication is its increasing ability to provide efficient communication links to almost any location, at constantly reducing costs with increasing power efficiency. Inevitably, the growth in wireless communication has seen the development of many competing technologies, and this has led to the continuous birth, as well as near-extinction of different wireless standards.

1.1 Evolution of Wireless Communication Systems

First-generation wireless systems were developed to allow a user to maintain continuous access to telephone connections within the service area of a network provider. These systems were characterized by analog transmission of voice signal, and a nonexistent inter-operability between existing standards. Examples of first-generation systems include the total access communication system (TACS) used in the United Kingdom, and the advanced mobile system (AMPS) of the United States [9]. The need for more efficient spectrum use, as well as an interest to exploit improved transmission performance offered by digital encoding and modulation techniques, led to the migration towards second-generation systems. Examples of second-generation systems include the United States Digital Cellular (USDC) standards IS-136 and IS-54, as well as Europe's Global System for Mobile (GSM) communication [9]. Third-generation wireless systems are presently being developed to provide an integrated wide-band communication network capable of providing excellent performance while supporting large numbers of mobile equipment. Design efforts on third-generation standards are presently on going, and different standards organizations are working together to define universally accepted specifications.

1.2 Requirements of Wireless Communication Systems

The goal of the wireless communication provider is to deliver communication links with efficiency comparable to that of wireline systems, at reasonable costs and convenience to as many subscribers as possible. A system with good performance should be able to deliver clear, uncorrupted data/voice/video connection with very little latency, and minimal power consumption. In digital systems, the ability to achieve uncorrupted data transmission is directly proportional to a system's probability of bit error, which is in turn inversely proportional to the power of data transmission. One of the major drives in wireless communication is the effort to increase power efficiency, thus delivering reliable links at lower and lower power levels.

In order for wireless service providers to maximize profit and service efficiency, they need to maximize the capacity of their systems so as to efficiently maintain as many subscribers as possible. Since spectrum allocation is a highly regulated practice in most parts of the world, this calls for effective use of limited available spectrum (high bandwidth efficiency).

Many practices have been developed to assist in the achievement of higher power efficiency, one of which is the implementation of error-correction codes. This practice involves the transmission of extra bits for redundancy in order to reduce the probability of bit error. As expected, the transmission of extra bits requires the use of additional spectrum, which reduces the available bandwidth for transmitting information bits, thereby resulting in a drop in bandwidth efficiency. This illustrates a very important dilemma in wireless communication systems: the tradeoff between power efficiency and bandwidth efficiency. Cost, convenience and mobility issues also need to be considered in order to provide the subscriber with the best wireless communication system possible. Ultimately, each wireless communication provider needs to make the best tradeoff possible between all these requirements in order to remain competitive in today's communication environment.

1.3 Error Correction Codes

The birth of error correction coding came in the late 1940's when Shannon mathematically showed that channel capacity could be achieved when transmitting information through a noisy channel [12]. Channel capacity is a theoretical measure of the fastest rate at which error free transmission can be realized. Traditional modulation techniques deliver performance significantly inferior to that predicted by Shannon's work (Binary phase shift keying by about 10dB in non-fading Gaussian channel), but when incorporated with error correction coding, most digital modulation schemes can achieve performance that approaches channel capacity. Error

correction coding involves the transmission of additional redundant bits in the stream of information bits, in order to allow the detection and correction of some symbol errors at the receiver. Although the use of channel coding improves the error performance of communication systems, the transmission of additional bits results in a reduction in bandwidth efficiency.

1.3.1 Block Codes

Block codes were first introduced in 1946 when Richard Hamming discovered a way to correct detected bit errors during computer simulations [22]. Hamming's method of error correction (Hamming codes) operated by calculating three check bits from a set of four data bits, and feeding all seven bits through a computer algorithm which was then able to correct one single bit error. The performance limitations of Hamming codes were addressed with the subsequent introduction of Golay codes. Golay codes were able to transmit a block of 23 bits, made up of twelve data bits and eleven calculated check bits, with an ability to correct three detected errors in each transmitted frame. The general structure of Hamming and Golay codes, where an n symbol code word is produced by the combination of k q -ary symbols and $n-k$ check symbols, with an ability to correct t errors, is usually referred to as a (q, n, k, t) block code. Since the introduction of Hamming and Golay codes in the late 1940's, many new classes of error correction block codes have been discovered for use in many different applications. Examples of these include the Reed-Muller codes, BCH codes and Reed-Solomon codes.

1.3.2 Convolutional Codes

Despite the performance improvement achieved through the use of block codes, systems utilizing this form of error correction are often burdened with certain fundamental setbacks. The frame oriented nature of block codes not only requires very precise frame synchronization, but each transmitted frame also has to be completely received before decoding, resulting in undesirable system latency. In addition, soft-decision decoding, which is needed to achieve performance bounds predicted by Shannon, is typically not an option for most block code structures. The introduction of convolutional codes in the mid 1950's helped address many of the performance issues faced by block codes [23].

Convolutional codes operate by adding redundant bits to a continuous flow of data bits through a linear shift register. Coupled with the use of the Viterbi algorithm for sequence estimation [15], convolutional codes eliminate the latency issue posed by the frame-by-frame

transmission of block codes. The output of a convolutional encoder is usually in a set of n bits, made up of a linear combination of k input bits and m bits stored in the shift register. The constraint length, denoted by K_c , is the number of bits upon which each encoder output is dependent, and the encoder rate is defined by the ratio of input data bits k to output bits n in each transmission interval. Convolutional decoders can also employ algorithms that exploit the gains of soft-decision decoding, thereby delivering performance closer to Shannon’s predictions. The mapping of convolutionally encoding states to non-binary PSK and QAM signal constellation led to the development of “trellis codes” [30], which are widely used today in data modems. Concatenation of convolutional codes also forms the basis of so-called “turbo coding” techniques that are the subject of this thesis.

1.3.3 Turbo Codes

The introduction of Turbo Codes at the International Conference on Communications (ICC) in 1993 brought the performance of practical coding closer to Shannon’s theoretical specifications. At its conception a Turbo code comprised of the parallel concatenation of two recursive systematic convolutional (RSC) codes [13].

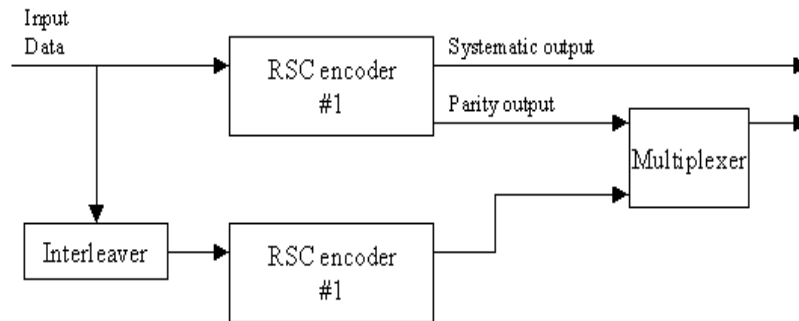


Figure 1.1: Structure of a turbo encoder

As shown in Figure 1.1, an *interleaver* is used to scramble the order of the input bits before feeding them into the second encoder (a *deinterleaver* at the receiver reverses the scrambling effect of the *interleaver*). This is done in order to break up error patterns in the overall code structure. It is important to note that the interleaver within a turbo code has a pseudorandom pattern, known to both the transmitter and receiver, but lacking any regular structure. This serves to break up any recurrent error patterns between the two codes. Since both encoders receive the same input bits (but in different orders), the *systematic* nature (one of the output bits is the same

as the input bit) of the encoders makes the systematic output of one encoder redundant to that of the other. Thus, the systematic output of the lower encoder is usually not transmitted, resulting in an overall code rate of 1/3. This code rate may be increased through a selective removal of bits from the overall code output stream (*puncturing*).

The presence of the interleaver in the structure of the Turbo encoder adds a considerable amount of complexity to the decoding process required to obtain the information bits at the receiver. A sub-optimal iterative decoding algorithm presented in [13] was developed to replace the optimal (maximum likelihood) decoding algorithm, which had been rendered impractical by the soft-output requirements of Turbo Codes. This new decoding algorithm involved decoding each constituent code locally, and then sharing the decoding outcomes in form of *posteriori probabilities* in an iterative manner. This iterative decoding structure is illustrated in Figure 1.2. The iterative decoding continues until the desired performance is attained, however, the marginal gain of every iteration is usually less than that of the one preceding it (following the law of diminishing returns).

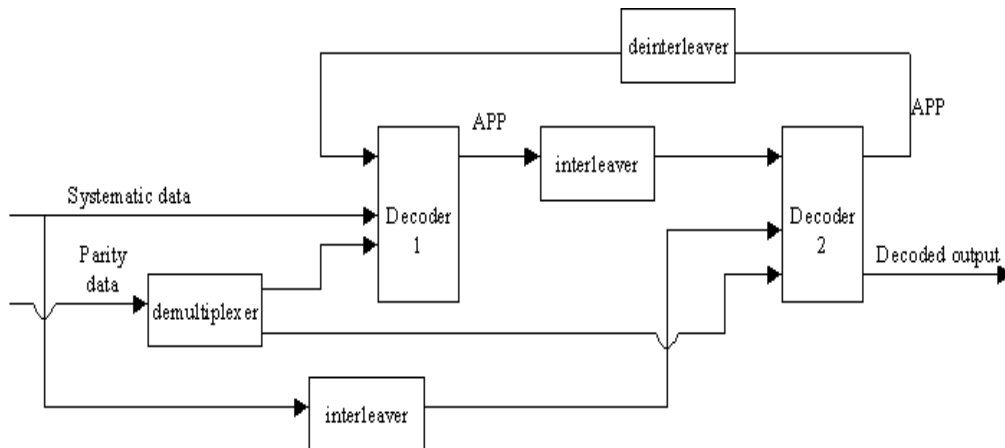


Figure 1.2: Structure of a turbo decoder

After other researchers were able to obtain similar (and in some cases, superior) performance results with different configurations of concatenated codes, it became apparent that the most important breakthrough in the development of turbo codes was the method of iterative decoding and not the specific code structure.

1.4 Practical Implementation of Turbo Codes

One of the major limitations in the performance of turbo codes is the inability to achieve infinite numeric precision. Two main sources of error are encountered when implementing error

correction codes in hardware. The first is the quantization of the encoded modulated signal, which is necessary before decoding, and the second is the limitation of bits available for numeric calculation within the body of the decoder.

1.4.1 Quantization

After encoding, digital information bits are modulated onto a carrier signal, which is typically a continuous RF signal. The continuous modulated signal is then transmitted through a noise-corrupted channel after which it is received for demodulation and decoding at the receiver. Before decoding, the received analog signal must first be digitized to provide finite numeric representation consistent with the number of available quantization bits. Quantization is realized by a systematic mapping of the entire range of the received signal onto 2^n discrete points, where n is the number of available bits. The method used to execute this mapping process depends on the form of quantization chosen; typically, uniform quantization is used in communication receivers for ease of implementation. Due to the approximation errors introduced during quantization, the discretization of received modulated signal leads to a degradation of performance of turbo codes. Although, increasing the number of available quantization bits can alleviate this hit in performance, increasing n would also mean an increase in complexity and cost of digital hardware. It is therefore necessary to trade-off between performance requirement and number of available quantization bits.

1.4.2 Fixed Point Representation

Turbo decoders make use of very powerful soft-input/soft-output iterative decoding algorithms that are numerically intensive. Most real-time decoders implement these algorithms using fixed-point arithmetic, where the amplitude of signals and coefficients alike, have to be discrete. The decoding hardware usually comes with a limited number of bits for numerical representation. Since these bits are also used to represent intermediate calculated values, a larger dynamic range is necessary; the word-length requirement in the internal data path are therefore usually more than those for quantization. With fixed-point representation, error is not only introduced at each rounding stage, but significant discrepancies are also added due to the accumulative approximation of continuous mathematical operations all through the decoding algorithm.

1.5 Smart Decoding of Turbo Codes

The use of newer, more powerful error correction codes accompanies the development of software radio systems, where increasing portions of the receiver are implemented in digital hardware. The increasing role of digital signal processors (DSPs) will result in new tradeoffs in system performance which we investigate in this thesis. As implied in the preceding sections, analog-to-digital conversion inaccuracies, as well as the accumulation of errors due to fixed-point implementation, both contribute to degrading the performance of turbo codes. One way to reduce this degradation effect is by increasing the number of bits available for quantization. But an increase in quantization bits would be accompanied by significant increases in calculation complexity and overall power consumption. It would therefore be favorable to determine the optimum number of bits required to deliver satisfactory performance under different transmission conditions and alter the resolution of the analog-to-digital converter accordingly.

As will be illustrated later on in this thesis, another approach to improving the performance of turbo codes is by increasing the number of decoding iterations. Increasing the number of decoding iterations results in higher calculation complexity, as well as greater system delay. Although, different data frames require fewer iterations to converge, decoding is frequently carried out in a uniform manner, in which all transmitted frames go through the same number of decoding iterations regardless of convergence time. It would be more efficient if the minimum number of decoding iterations required for each transmitted frame could be determined at the decoder, thereby delivering better power efficiency and improved system latency.

In improving the efficiency of digital communication systems, emphasis has often been placed on reducing the transmission power required to deliver a certain level of performance. As software radio architectures become more prevalent, traditional power optimization approach may require modification. It may prove useful to investigate the option of reducing overall power dissipation by increasing the transmission power, while reducing the power dissipated within the receiver (by reducing number of decoding iterations as well as ADC resolution) without significant deterioration in performance.

1.6 Thesis Outline

A brief overview of wireless communication systems is given in Chapter 1. Chapter 1 also contains an introduction to error correction codes and the problems associated with implementing them in hardware. Chapter 2 gives a more exhaustive description of the implementation and performance issues that affect turbo codes. A brief overview of ADC concepts and architectures

are given in Chapter 3. The concepts and simulation results for using a variable resolution ADC and an iterative stopping criterion to improve power consumption of turbo codes are delivered in Chapter 4. Conclusions for this thesis, as well as suggested directions for future research work are given in Chapter 5.

CHAPTER 2

TURBO CODES: FUNDAMENTALS

Shannon showed that reliable communication may be achieved at the minimum possible signal to noise ratio through the use of long random codes [24]. Although the implementation of truly random codes is impractical, cyclic block codes, convolutional codes, and turbo codes offer good practical substitutes.

2.1 Block Codes

A block code C consists of a set of M code words $\{c_0, c_1, \dots, c_{M-1}\}$, each an n -tuple possessing a one-to-one mapping to blocks of an input data stream. The one-to-one nature of the mapping ensures reversal of the encoding process is possible at the receiver. The rate of a (n, k) block code is $r = k/n$, where $M = 2^k$ in the case of binary implementation. The *Hamming distance* between two code words is the number of bits with which the blocks differ from each other, and the minimum distance (d_{min}) of a code is the smallest Hamming distance between two code words. The Hamming weight of a code word is the Hamming distance between itself and the all-zero code word [4].

A linear block code is considered *cyclic* if a cyclic shift of one code word leads to another code word. This means that for every code word $c = \{c_0, c_1, \dots, c_{n-2}, c_{n-1}\} \in C$, there is another code word $c' = \{c_{n-1}, c_0, \dots, c_1, c_{n-2}\} \in C$. Cyclic codes are encoded through the multiplication of the message polynomial $m(D)$, with the generator polynomial $g(D) = g_0 + g_1D + g_2D^2 + \dots + g_{n-k}D^{n-k}$. This multiplication is equivalent to the convolution of polynomial coefficients, and this can be implemented with the use of shift registers [24].

A code is said to be *systematic* if the information bits are embedded within the resulting code word. This is usually implemented through the use of a generator matrix of the form:

$$G = [P \quad I_k], \quad (2.1)$$

Where I_k is the $k * k$ identity matrix, and P is a $k * (n-k)$ matrix. The generator matrix G , has a simple one-to-one correspondence with the generator polynomial $g(D)$.

2.2 Convolutional Codes

Unlike block codes, convolutional codes provide a means to implement continuous encoding of an input stream of data, thereby eliminating the need for fixed data blocks. Convolutional encoders can be implemented as multi-input, multi-output systems, with k bit streams at the input and n bit streams at the output. Encoding commences with the division of the message bits into k input streams (frequently $k=1$), which are passed through a convolutional encoder to provide n output streams. The output data streams are then multiplexed together to produce the output code word C . As in the case of block codes, the rate of convolutional codes is $r = k/n$. In the convolutional encoder, one shift register is required for each input stream, and the total memory required, M_c , is the sum of the lengths of all shift registers associated with each input. The constraint length K_c of the encoder is the maximum number of bits in an output stream that can be affected by any input bit [4]. Figure 2.1 illustrates an example rate $1/2$ convolutional encoder with generator matrix $g^{(0)}=1 + D$ and $g^{(1)}= 1 + D + D^3$ and $K_c=3$.

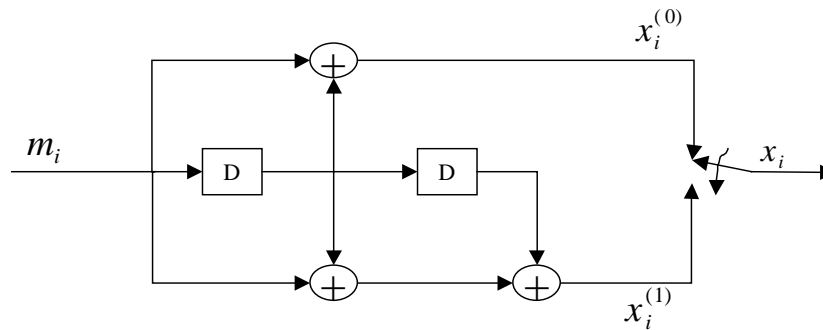


Figure 2. 1: Convolutional encoder with $K_c=3$.

A convolutional encoder with memory M_c , has 2^{M_c} possible states, which are all determined by the contents of the shift register(s). On the reception of an input bit, a transition can be made to one of two states depending on whether a 1 or a 0 is received, and each state transition is accompanied by the emission of an n -bit output. This encoding operation is clearly laid out in the *state diagram* of a convolutional encoder. A state diagram consists of a set of nodes (representing the states of the encoder) connected by links labeled by m/x , where m represents the input bits and x , the output bits corresponding to each state transition. Figure 2.2 illustrates the state diagram of the encoder in Figure 2.1.

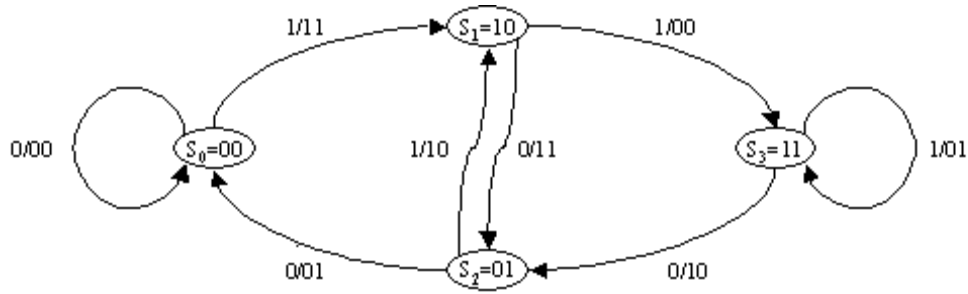


Figure 2. 2: State diagram for sample convolutional code.

The *trellis diagram* is a more elaborate visual tool that incorporates the passage of time to the information already provided by the state diagram. Each node in a trellis diagram is labeled $S_{i,j}$, where i represents a specific encoder state, and j represents a particular instant in time. Figure 2.3 illustrates a sample state diagram for the same encoder given in Figure 2.1.

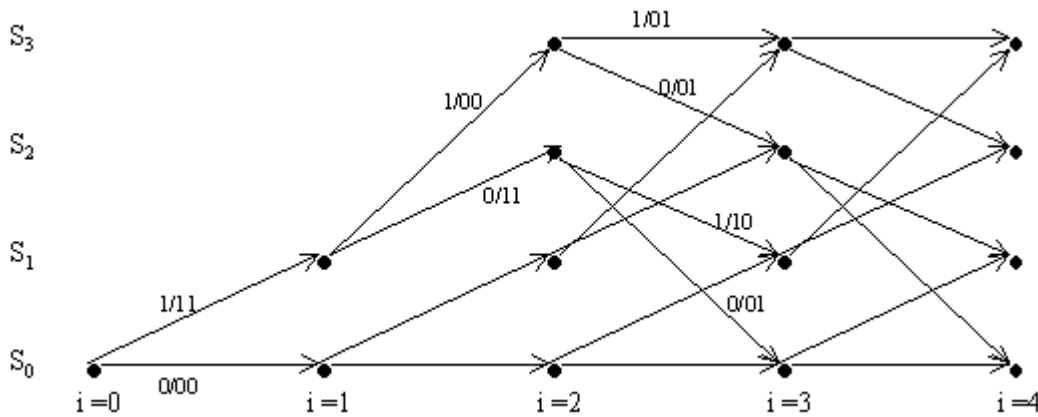


Figure 2. 3: Trellis diagram for example convolutional code.

With convolutional codes, a unique path through the trellis is associated with every code word. This path, known as a *state sequence*, usually begins at state $s_L = S_{0,0}$ and ends at state $s_L = S_{0,L}$. Although convolutional codes are characterized by having variable lengths, it is sometimes necessary, in the case of certain applications, to limit the length of each input frame. This is done by a process known as *trellis termination*, where the last M_c input bits are set to zero in order to force the trellis to terminate at the all zero state, $S_{0,L}$.

2.3 Recursive Systematic Convolutional (RSC) Codes

A good first order performance measure for convolutional codes is the *minimum free distance*, d_{free} , which is the smallest hamming weight of all code words with paths diverging from the all-zero path at time zero. The advantages offered by systematic coding can also be enjoyed with convolutional codes without reducing the minimum free distance. While the systematic information is simply the input bit, the parity information can be obtained by manipulating the generator matrix with the use of feedback equipped shift registers [4]. An example RSC code with constraint length $K_c = 4$ is illustrated in Figure 2.4. RSC codes can also be represented using state and trellis diagrams.

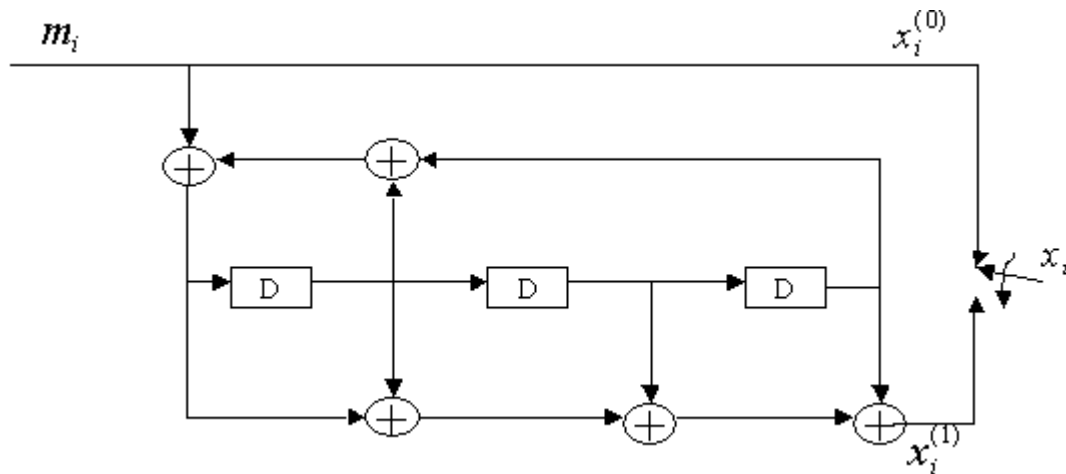


Figure 2. 4: A rate 1/2 RSC encoder with $K_c=4$

2.4 Turbo Codes

Although cyclic block codes and convolutional codes offer simple practical realizations that provide some coding gain, their inherent structure that allows for relatively easy encoding and decoding, prevents the realization of the ‘random’ characteristics necessary to approach Shannon’s coding bounds. The inherent random-like properties realized during the implementation of turbo encoding and decoding helps bring reliable communication closer to the Shannon limit [13].

2.4.1 The Turbo Encoder

The turbo encoder is made up of a parallel concatenation of two RSC encoders that receive the same information bits, but in different orders. The input bits to the top RSC encoder are first

passed through an interleaver before they are fed into the lower RSC encoder. The purpose of the interleaver is to rearrange the input to the second RSC encoder in order to emulate the randomized effect needed for improved code performance. Multiplexing the systematic information with the parity information from both RSC encoders produces the output stream of a turbo encoder. The systematic output of the turbo encoder is taken from the upper RSC encoder, while one parity output is taken from each RSC encoder. Thus, the code rate of a turbo code is typically 1/3, although this can be increased via puncturing. Figure 2.5 below shows the structure of a typical turbo encoder operating with the RSC code structure shown in Figure 2.4.

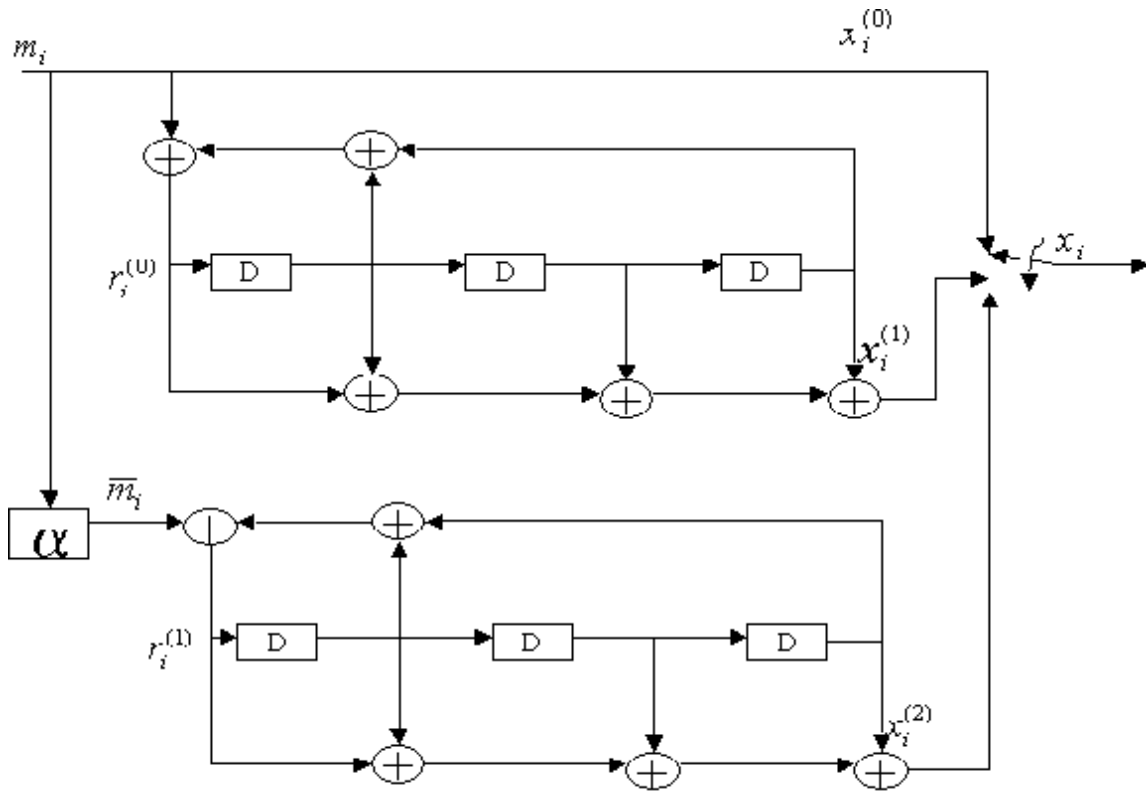


Figure 2. 5: Turbo encoder using sample RSC encoders

Due to its infinite impulse response nature, the output of an RSC encoder is usually has a fairly high Hamming weight. But, there are some input sequences that result in low Hamming weight outputs. By, passing the same input through both RSC encoders in different orders, the probability of both encoders simultaneously producing outputs with low Hamming weights is reduced. Some situations may arise when both encoders produce low weight output sequences, but these occur at a relatively low rate, resulting in superior turbo code performance at low signal-to-noise ratio. Although the Hamming weights of turbo codes are not extraordinarily high,

it is the reduced frequency of low weight code words that allows turbo codes to deliver excellent performance. For this same reason, turbo codes perform better at low signal-to-noise ratios because of a small occurrence of low weight codes, contrary to the case of high signal-to-noise ratio where there usually is a higher occurrence of low weight code sequences.

One major issue faced by the designers of turbo codes is the problem of simultaneously terminating the trellis of both constituent encoders. Due to the recursive nature of the encoders, the terminating bits for each encoder are not known until all information bits have been encoded for that particular encoder. Due to the permutation effect of the interleaver, the tail-bits of one encoder serve as information bits to the other encoder, thereby affecting the tail bits of the other encoder. Thus, to obtain the tail bits for one encoder, the tail bits of the other encoder must be known. This is obviously impractical. Several solutions have been found to alleviate this problem. One option that is widely used is the trellis termination of a single RSC encoder, while the other is left open [25]. Another is to force both encoders back to the all-zero state by using special encoder circuits [14].

2.4.2 The Turbo Decoder

Since the output of each constituent RSC encoder of a turbo code depends only on the last input bit (as well as a convolution operation), the encoding process of a turbo code can be considered as two joint Markov processes. Since the two Markov processes run on the same input data, turbo decoding can proceed by first independently estimating each process, then refining the estimates by iteratively sharing information between both decoders. This means that the output of one decoder can be used as a-priori information by the other decoder. In order to take advantage of this iterative decoding scheme, it is necessary for each decoder to produce soft-bit decisions, which are usually in the form of *log-likelihood ratios* (LLRs). The LLR serves as a priori information that informs on the probability that a transmitted message bit is a one rather than a zero:

$$\Lambda_i = \ln \frac{P[m_i = 1 | y]}{P[m_i = 0 | y]}. \quad (2.2)$$

A decoder that accepts input in the form of a priori information, and produces output in the form of posteriori information is known as a *soft-input, soft-output* (SISO) decoder. The inputs to the SISO decoder consist of systematic information, parity information, and the a priori information

from the previous decoder, while the output is the LLR defined by Λ_i . The structure of a SISO decoder is illustrated in Figure 2.6.

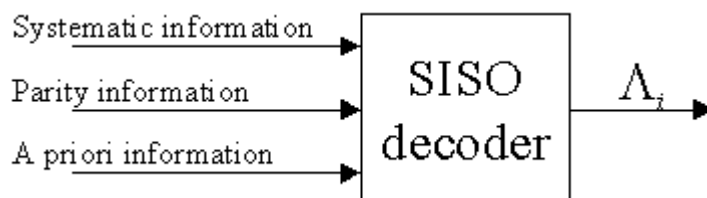


Figure 2.6: Structure of a soft-input soft-output decoder

2.4.3 Operation of a Turbo Decoder

We assume that bits are encoded with a turbo encoder of the same form of Figure 2.5, and transmitted using *binary phase shift keying* (BPSK) modulation. The input-output relationship of a system employing BPSK modulation is given by [1]

$$y = a(2x - 1) + n, \quad (2.3)$$

where a is the fading amplitude and n is a zero-mean Gaussian random variable with variance $\sigma^2 = N_0 / 2E_s$. The LLR of a SISO decoder operating under this model can be expressed as [1]

$$\Lambda_i = \frac{4a_i^{(s)} E_s}{N_0} y_i^{(s)} + z_i + l_i. \quad (2.4)$$

This output is made up of information related to the current decoder's systematic input ($y_i^{(s)}$), the previous decoder's output (z_i), as well as newly obtained probability information known as *extrinsic information* (l_i). In order to avoid positive feedback complications, it is important that only the extrinsic information is passed from decoder to decoder.

The structure of a turbo decoder is illustrated in Figure 2.7. Decoding commences when the first decoder receives the first encoder's parity bit and the systematic channel observations (both weighted by...), as well as the a priori information derived from the second decoder's output. During the first iteration, since the second decoder has not produced any information, the a priori information is set to zero. The first decoder then produces the output LLR from which the extrinsic information is derived by subtracting the weighted systematic and a priori inputs of the first decoder. The extrinsic information is then interleaved to provide a priori information for

the second decoder, which also receives the interleaved systematic observation and the parity bits of the second encoder. In the same way as for the first decoder, the extrinsic information is also derived from the LLR produced by the second decoder, after which it is deinterleaved before serving as a priori information for the first decoder. After the assigned number of iterations, the message bits are estimated by making a hard decision on the deinterleaved output of the second decoder.

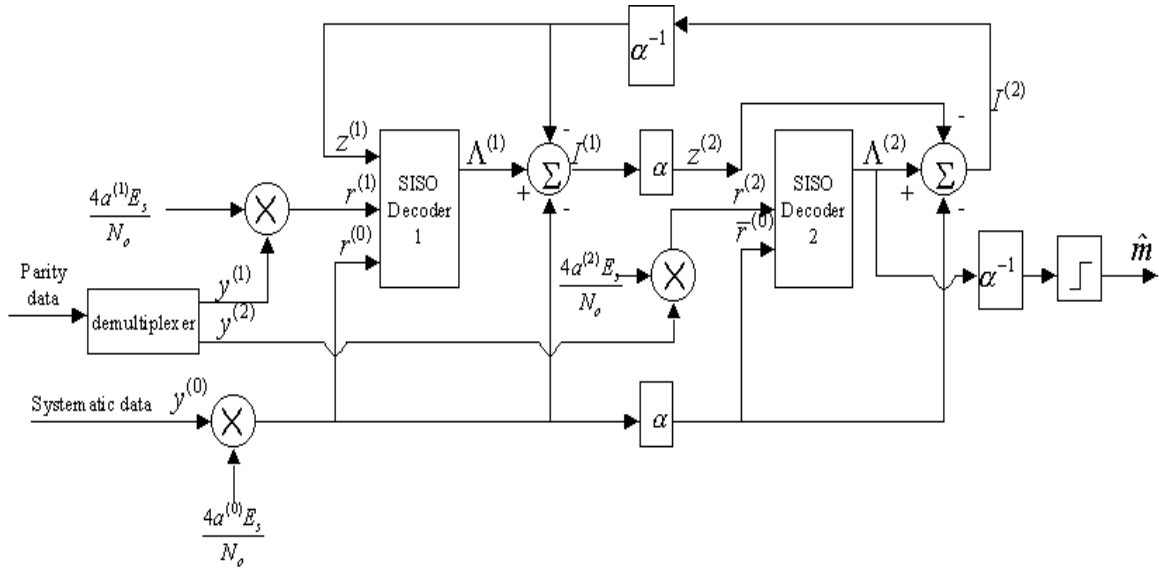


Figure 2.7: Schematic of a turbo decoder.

2.4.4 Decoding Algorithms

Two well-developed and widely used algorithms for determining the state sequence of a trellis encoder are the Viterbi algorithm (VA) [15] and the maximum a posteriori (MAP) algorithm [16]. The Viterbi algorithm is widely used for decoding of convolutional codes, but variants of the MAP algorithm are more suitable for turbo-code decoding. Given the received observation sequence y , the Viterbi algorithm determines the most probable state sequence

$$\hat{s} = \arg\{\max_s P[s | y]\}. \quad (2.5)$$

This implies that, the states estimated by the Viterbi algorithm will always form a connected path through the trellis. The MAP algorithm, on the other hand, attempts to determine each state transition, without regard to the overall sequence of the trellis

$$\hat{s}_i = \arg\{\max_{s_i} P[s_i | y]\}. \quad (2.6)$$

In the case of the MAP algorithm, the resulting trellis solution may contain discontinuities. On the basis of performance comparisons, it has been observed that the VA results in minimizing frame error rate (FER), while the MAP algorithm minimizes the bit error rate (BER). For this thesis, we will only focus on variations of the MAP algorithm.

2.4.5 Maximum A Posteriori (MAP) Algorithm

Given the noisy observations of symbols produced by a Markov process, the maximum a posteriori (MAP) algorithm calculates the a posteriori probability (APP) of each message bit/symbol that must have been transmitted at the encoder. During the decoding sequence, the APPs obtained after every iteration are put into LLR form for manipulation by the next decoder, and hard-decisions are only made after the last iteration. In order to find the APP for each message symbol from the given noisy observations, the MAP algorithm first calculates the probability of each state transition

$$P[s_i \rightarrow s_{i+1}, y] = \alpha(s_i)\gamma(s_i \rightarrow s_{i+1})\beta(s_{i+1}). \quad (2.7)$$

where $\alpha(s_i)$, $\beta(s_i)$, and $\gamma(s_i \rightarrow s_{i+1})$ are defined below, and

$$P[s_i \rightarrow s_{i+1} | y] = \frac{P[s_i \rightarrow s_{i+1}, y]}{P[y]}. \quad (2.8)$$

The term $\alpha(s_i)$ gives the probability of being in a present trellis state after receiving the given channel observations up to a particular instant in time j ($P[s_i, (y_0, \dots, y_{i-1})]$). It can be found by using the forward recursion

$$\alpha(s_i) = \sum_{s_{i-1} \in A} \alpha(s_{i-1})\gamma(s_{i-1} \rightarrow s_i), \quad (2.9)$$

where A is the set of states s_{i-1} connected to s_i . Conversely, given a particular trellis state s_i at a specific instant in time j , $\beta(s_i)$ gives the probability of having the subsequent channel observations from that instant in time till the end of the code block $P[(y_{i+1}, \dots, y_{L-1}) | s_{i+1}]$. The probability $\beta(s_i)$ can be found by using the backward recursion

$$\beta(s_i) = \sum_{s_{i+1} \in B} \beta(s_{i+1})\gamma(s_i \rightarrow s_{i+1}), \quad (2.10)$$

where B is the set of states s_{i+1} connected to s_i . The term $\gamma(s_i \rightarrow s_{i+1})$ is the branch metric, which represents the chances of making a transition from state s_i to s_{i+1} . In the case where states s_i and s_{i+1} are not connected on the trellis diagram, a $s_i \rightarrow s_{i+1}$ transition would be impossible, thus, the branch metric would have a value of zero. The branch metric can be calculated as

$$\gamma(s_i \rightarrow s_{i+1}) = P[m_i]P[y_i | x_i], \quad (2.11)$$

where m_i is the message and x_i , the output associated with the $s_i \rightarrow s_{i+1}$ transition.

After determining the APP for each state transition, the probabilities for the message bits/symbols are then determined by

$$P[m_i = 1 | y] = \sum_{S_1} P[s_i \rightarrow s_{i+1} | y], \quad (2.12)$$

and

$$P[m_i = 0 | y] = \sum_{S_0} P[s_i \rightarrow s_{i+1} | y], \quad (2.13)$$

where $S_1 = \{s_i \rightarrow s_{i+1} : m_i = 1\}$ is the set of all state transitions associated with transmitting a 1, and $S_0 = \{s_i \rightarrow s_{i+1} : m_i = 0\}$ is the set of all state transitions associated with transmitting a 0. The log-likelihood ratio is thus defined as

$$\Lambda_i = \ln \frac{\sum_{S_1} \alpha(s_i) \gamma(s_i \rightarrow s_{i+1}) \beta(s_{i+1})}{\sum_{S_0} \alpha(s_i) \gamma(s_i \rightarrow s_{i+1}) \beta(s_{i+1})}. \quad (2.14)$$

The MAP algorithm is implemented thus [14]:

1) Forward recursion

- a) Create a 2-dimensional array $\alpha(j,i)$, $0 \leq j \leq 2^{Mc} - 1$, $0 \leq i \leq L$, to store recursion results. Array should be initialized as:

$$\alpha(j,0) = \begin{cases} 1 & \text{if } j = 0 \\ 0 & \text{if } j \neq 0 \end{cases} \quad (2.15)$$

- b) Begin with time index $i = 1$, and state index $j=0$.
c) Let $s_i = S_j$ and update α as:

$$\alpha(j,i) = \sum_{s_{i-1}=S_j \in A} \alpha(j',i-1) \gamma(s_{i-1} \rightarrow s_i), \quad (2.17)$$

- d) Increment j , and until $j = 2^{M_c}$ return to step 1(c).
- e) Increment i , and until $i = L$, return to step 1(c).

2) Backward recursion

- a) Create a 2-dimensional array $\beta(j,i)$, $0 \leq j \leq 2^{M_c} - 1$, $0 \leq i \leq L$, to store recursion results. If terminated, array should be initialized as

$$\beta(j,L) = \begin{cases} 1 & \text{if } j = 0 \\ 0 & \text{if } j \neq 0 \end{cases}, \quad (2.17)$$

but, if not terminated, it should be initialized as

$$\beta(j,L) = \frac{1}{2^{M_c}} \forall j. \quad (2.18)$$

- b) Begin with time index $i = L - 1$, and state index $j=0$.
- c) Let $s_i = S_j$ and update β as:

$$\beta(j,i) = \sum_{s_{i+1}=S_{j'} \in B} \beta(j',i+1) \gamma(s_i \rightarrow s_{i+1}), \quad (2.19)$$

- d) Increment j , and until $j = 2^{M_c}$ return to step 2(c).
- e) Decrement i , and until $i = 0$, return to step 2(c).

3) For $i = (0, \dots, L - 1)$, determine LLR by:

$$\Lambda_i = \ln \frac{\sum_{S_1} \alpha(j,i) \gamma(s_i \rightarrow s_{i+1}) \beta(j',i+1)}{\sum_{S_0} \alpha(j,i) \gamma(s_i \rightarrow s_{i+1}) \beta(j',i+1)}, \quad (2.20)$$

where $S_1 = \{(s_i = S_j) \rightarrow (s_{i+1} = S_{j'}) : m_i = 1\}$ is the set of all state transitions associated with transmitting a 1, and $S_0 = \{(s_i = S_j) \rightarrow (s_{i+1} = S_{j'}) : m_i = 0\}$ is the set of all state transitions associated with transmitting a 0.

2.4.6 Log-MAP Algorithm

Although the MAP algorithm produces very accurate APP calculations, it is burdened by a very high computational complexity as well as a high sensitivity to round-off errors during finite-precision implementation. By implementing bulk of the calculations of the MAP algorithm in the log-domain, the *log*-MAP algorithm reduces the computational complexity of the SISO decoder. The reduction in computational complexity is achieved by the replacement of all multiplication operations by addition operations in the log-domain. However, all addition operations also have to be replaced by a maximization operation followed by a correction function in the log-domain [14]:

$$\begin{aligned}\ln(e^x + e^y) &= \max(x, y) + \ln(1 + \exp\{-|y - x|\}) \\ &= \max(x, y) + f_c(|y - x|),\end{aligned}\tag{2.21}$$

where $f_c(|y-x|)$ is a tabulated correction function. The correction term can simply be implemented by performing a search through a pre-determined one-dimensional look-up table.

In replacing the values used in the MAP calculations by their logarithmic counterparts, the logarithm of $\alpha(s_i)$ becomes

$$\begin{aligned}\bar{\alpha}(s_i) &= \ln \alpha(s_i) \\ &= \max_{s_{i-1} \in A} * [\bar{\alpha}(s_{i-1}) + \bar{\gamma}(s_{i-1} \rightarrow s_i)],\end{aligned}\tag{2.22}$$

the logarithm of $\beta(s_i)$ becomes

$$\begin{aligned}\bar{\beta}(s_i) &= \ln \beta(s_i) \\ &= \max_{s_{i+1} \in A} * [\bar{\beta}(s_{i+1}) + \bar{\gamma}(s_i \rightarrow s_{i+1})],\end{aligned}\tag{2.23}$$

and the LLR is determined to be

$$\begin{aligned}\Lambda_i &= \max_{S_1} * [\bar{\alpha}(s_i) + \bar{\gamma}(s_i \rightarrow s_{i+1}) + \bar{\beta}(s_{i+1})] \\ &\quad - \max_{S_0} * [\bar{\alpha}(s_i) + \bar{\gamma}(s_i \rightarrow s_{i+1}) + \bar{\beta}(s_{i+1})].\end{aligned}\tag{2.24}$$

Consequently, the log-MAP algorithm is implemented thus [14]:

1) Forward recursion

- a) Create a 2-dimensional array $\alpha(j, i)$, $0 \leq j \leq 2^{Mc} - 1$, $0 \leq i \leq L$, to store recursion results. Array should be initialized as:

$$\bar{\alpha}(j,0) = \begin{cases} 1 & \text{if } j = 0 \\ -\infty & \text{if } j \neq 0 \end{cases} \quad (2.25)$$

- b) Begin with time index $i = 1$, and state index $j=0$.
- c) Let $s_i = S_j$ and update α as:

$$\bar{\alpha}(j,i) = \sum_{s_{i-1}=S_j \in A} \bar{\alpha}(j',i-1) + \bar{\gamma}(s_{i-1} \rightarrow s_i), \quad (2.26)$$

- d) Increment j , and until $j = 2^{\text{Mc}}$ return to step 1(c).
- e) Increment i , and until $i = L$, return to step 1(c).

2) Backward recursion

- a) Create a 2-dimensional array $\beta(j,i)$, $0 \leq j \leq 2^{\text{Mc}} - 1$, $0 \leq i \leq L$, to store recursion results. If terminated, array should be initialized as

$$\bar{\beta}(j,L) = \begin{cases} 1 & \text{if } j = 0 \\ -\infty & \text{if } j \neq 0 \end{cases} \quad (2.27)$$

but, if not terminated, it should be initialized as

$$\bar{\beta}(j,L) = 0 \quad \forall j. \quad (2.28)$$

- b) Begin with time index $i = L - 1$, and state index $j=0$.
- c) Let $s_i = S_j$ and update β as:

$$\bar{\beta}(j,i) = \sum_{s_{i+1}=S_j \in B} \bar{\beta}(j',i+1) + \bar{\gamma}(s_i \rightarrow s_{i+1}), \quad (2.29)$$

- d) Increment j , and until $j = 2^{\text{Mc}}$ return to step 2(c).
- e) Decrement i , and until $i = 0$, return to step 2(c).

- 3) For $i = (0, \dots, L - 1)$, determine LLR by:

$$\begin{aligned} \Lambda_i &= \max_{S_1} * [\bar{\alpha}(j,i) + \bar{\gamma}(s_i \rightarrow s_{i+1}) + \bar{\beta}(j',i+1)] \\ &\quad - \max_{S_0} * [\bar{\alpha}(j,i) + \bar{\gamma}(s_i \rightarrow s_{i+1}) + \bar{\beta}(j',i+1)], \end{aligned} \quad (2.30)$$

where $S_1 = \{(s_i = S_j) \rightarrow (s_{i+1} = S_{j'}) : m_i = 1\}$ is the set of all state transitions associated with transmitting a 1, and $S_0 = \{(s_i = S_j) \rightarrow (s_{i+1} = S_{j'}) : m_i = 0\}$ is the set of all state transitions associated with transmitting a 0.

2.4.7 Performance Issues of Turbo Codes

The performance of turbo codes is greatly influenced by a variety of factors, and trade-offs often have to be made to best serve different operational situations. The size of the interleaver, for example, is a factor that greatly affects the performance of turbo codes. The fixed structure required by the nature of the interleaver demands that data be grouped together in frames before encoding (block codes in essence). It has been observed that the performance of turbo codes improve as the size of the interleaver, and consequently, the frame size, are increased. One significant point about this is the fact that per-symbol decoding complexity is not affected by the frame size. So, in order to enjoy better performance, the interleaver size can be increased without paying a price in decoder complexity. The price is paid with an increase in system latency. Since the encoder and decoder have to receive all bits in a transmitted frame before commencing operation, longer frames result in longer system delays. Figure 2.8 illustrates the relationship between frame/interleaver size and the performance of turbo codes.

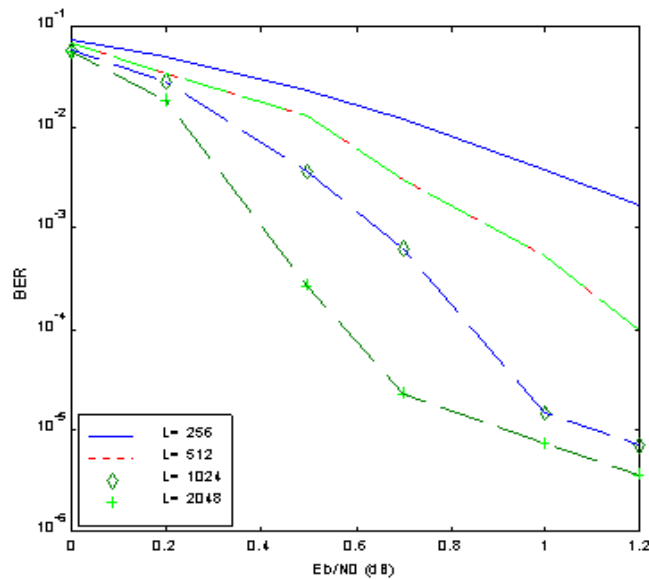


Figure 2.8: Simulated BER performance for a rate=1/2, $K_c=4$, turbo code with $G=\{15,17\}$, and a random interleaver for different frame sizes

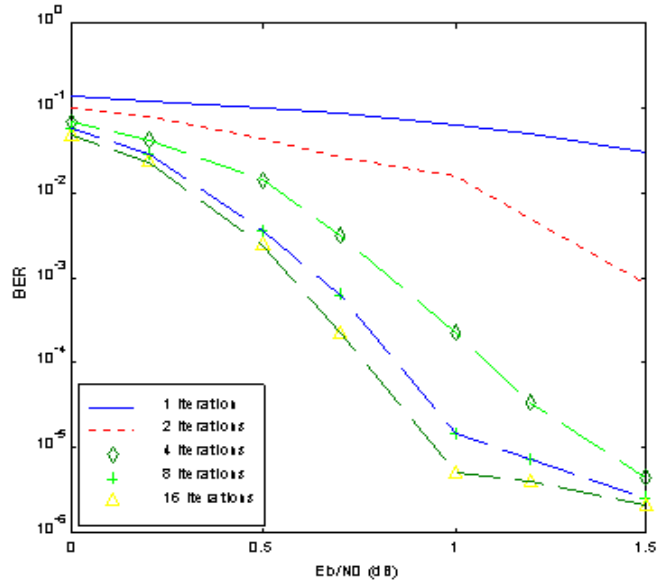


Figure 2.9: Simulated BER Performance for a rate=1/2, $K_c=4$, turbo code with $G=\{15,17\}$, and a random interleaver for varying number of decoder iterations

The number of decoding iterations is another factor that affects the performance of turbo codes. Although dramatic performance gains are noticed after the first few iterations, diminishing returns are observed as the performance gains drop at higher iterations. The price paid when increasing the number of decoding iterations is an increase in decoding complexity as well as a rise in system latency. As more iterations are completed, more calculations have to be performed by the digital processor, leading to greater power consumption and system delay. The effect of varying the number of decoding iterations is shown in Figure 2.9.

As with other codes, a tradeoff can be made between the code rate and code performance. Puncturing practices can be used to increase the code rates of turbo codes in order to achieve better bandwidth efficiency, but performance is reduced. An optimization in the form of puncturing, as well as the type of interleaver used in encoding a turbo code can result in very significant performance gains [26].

There are several other tradeoffs that a turbo code designer has to consider in order to achieve optimum performance for a specific application. As mentioned earlier, the choice of decoding algorithm affects the performance and complexity of turbo codes. The MAP algorithm can deliver better BER performance, but at a higher computational cost than the SOVA algorithm. The type of trellis termination implemented can also affect the tradeoff between code performance and decoding complexity [26].

This chapter has reviewed the basic principles of turbo-codes. The next chapter describes analog-to-digital converter technology. Chapter 4 unites these two topics with a study of the interplay between performance optimization and power efficiency within a software radio.

CHAPTER 3

ANALOG-TO-DIGITAL CONVERTERS (ADCs)

Even though the telecommunication community continues to strive towards an overall digital orientation, most inputs and outputs of communication systems remain analog. As faster, more powerful and less expensive digital hardware are being developed, more of the traditionally analog functions in communication systems have to be executed with software and/or digital hardware. In order to stay competitive in today's telecommunication environment, engineers strive to combine state-of-the-art analog and digital communication techniques with efficient analog-to-digital conversion (ADC) practice. This chapter provides a brief review of current ADC practice, a review which will be exploited in Chapter 4 to examine power efficiency tradeoffs in turbo-code design within a software radio.

3.1 Basic Concepts of Analog-to-Digital Conversion

The function of the analog-to-digital converter is to convert an incoming analog signal, with continuous ranges in amplitude and time, into a digital output signal that is discrete in both time and amplitude. The resulting signal may then be represented with a finite number of bits in a form suitable for digital processing. The basic building blocks of an ADC are illustrated in Figure 3.1. The anti-alias filter band-limits the analog signal (thereby limiting the frequencies present within the signal), the sampling stage is responsible for the time discretization of the filtered signal, and the quantizer executes the discrete amplitude representation of the sampled signal. Each stage within the ADC has its own specific requirements that must be met in order to deliver a particular performance level. Performance of an ADC is usually considered in terms of a combination of factors, including but not limited to, accuracy, jitter, linearity, and power efficiency [27].

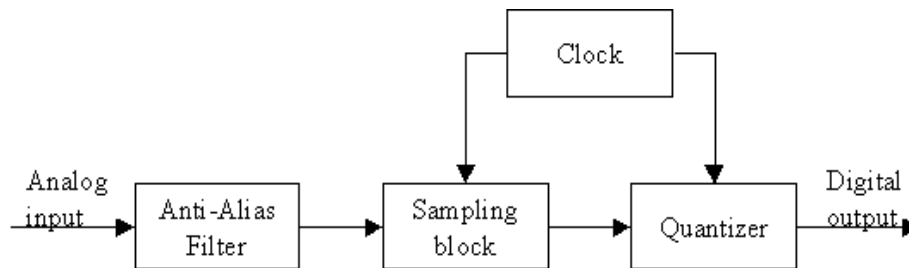


Figure 3.1: Functional Blocks of an ADC

3.1.1 Sampling

The sampling process is the means through which time samples of a continuous signal are taken in order to provide a discrete time signal. If carefully executed, the sampling stage does not introduce any error into the analog-to-digital conversion process. The product of a sampling stage is greatly dependent on the relationship between the frequency components of the sampled continuous signal and the employed sampling rate. After sampling, the spectrum of the original analog signal, $F(f)$, becomes periodic, appearing at integer multiples of the sampling frequency. This is an inherent property of sampling that cannot be avoided. Software radios can exploit this aliasing effect to perform the final stage of demodulation by sampling an intermediate frequency (IF) signal and then subsequently working with a copy of the signal at baseband.

A bandlimited signal is a signal having no frequency components above a given frequency f_{max} . In order to ensure the reconstruction of a sampled bandlimited analog signal, it is necessary that the sampling rate be at least twice the highest frequency component of the analog signal ($2f_{max}$). This theorem is known as the *Nyquist Theorem* [28], and a sampling rate of $2f_{max}$ is known as the *Nyquist sampling rate*. If sampling is carried out at a frequency below the Nyquist frequency, there will be an overlap in the repeated spectrum of the sampled signal (*aliasing*), and the original analog signal cannot be perfectly reconstructed from the sampled signal. As the sampling rate is increased beyond the Nyquist rate, the space between the repeated copies of the original spectrum increases. Figure 3.2 illustrates the frequency domain representation of the sampling theorem.

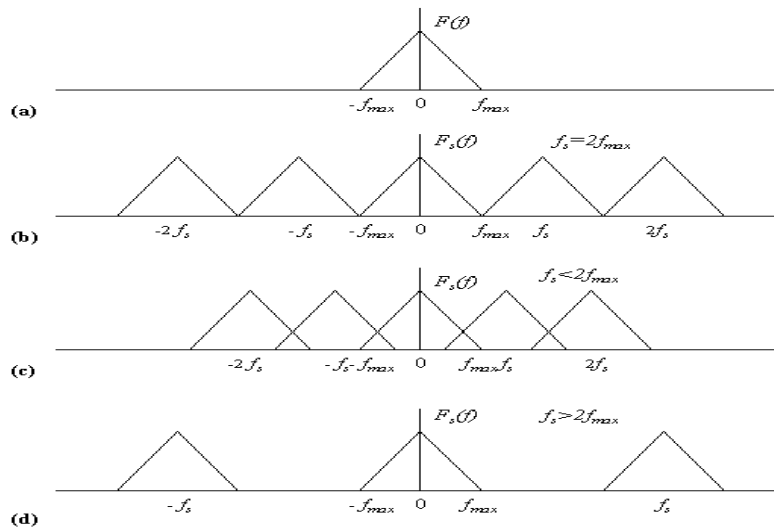


Figure 3.2: Spectrum of: a) Bandlimited analog signal; b) Nyquist sampled signal; c) Undersampled signal; d) Oversampled signal.

3.1.2 Anti-Alias Filtering

To facilitate the practical implementation of subsequent parts of the analog-to-digital converter (particularly the sampling stage), an analog signal must first be band-limited to reduce its frequency content. Practical analog signals usually contain unbounded frequency components, making the requirement of sampling at the Nyquist rate is usually impractical. The job of the anti-alias filter placed before the sampling stage is to pass all desired frequency components up to a certain cut-off frequency (f_{max}), while attenuating all other frequency components.

A problem arises due to the impracticality of realizing an ideal “brick wall” filter. Practical filters provide an attenuation that rises gradually from the cut-off frequency, thereby allowing the transmission of some unwanted frequency components. This means that if sampling at the Nyquist rate, some degree of aliasing will still be experienced even if anti-alias filtering was done before sampling. A good practical filter should have a very steep transition band, and great attenuation in the stop band. But in practice, trade-offs have to be made between stop band attenuation, transition bandwidth, and pass band distortion, and limitations on practical implementation of analog filters make it virtually impossible to achieve a very good combination of all three.

One way of reducing the demand placed on the performance of anti-aliasing filters is to practice oversampling. If sampling is carried out at a rate much greater than the Nyquist rate, then the repeated copies of the original spectrum become increasingly separated from each other (as depicted in figure 3.2c). By doing this, the anti-alias filters will allow the system to use a more gradual roll-off without increasing spectrum distortion due to overlap.

3.1.3 Quantization

Quantization is a process of approximating a continuous signal with an uncountable infinite amplitude range, with a discrete signal made up of a finite set of values. Unlike the sampling process, quantization will always introduce error into the converted signal, regardless of how efficiently it is executed. Uniform quantization, characterized by equal voltage difference between each quantization level, has been found to be simplest and most efficient for digitizing IF and RF signals [3]. Other forms of quantization include logarithmic, adaptive and differential quantization, and these are well described in [14].

In uniform quantization, the entire range of the analog signal from v_{min} to v_{max} (with $v_{min} = -v_{max}$) is divided into $M=2^n$ equally space bins, where n is the number of bits available for quantization. Quantization is achieved by replacing every sampled value of the analog signal by

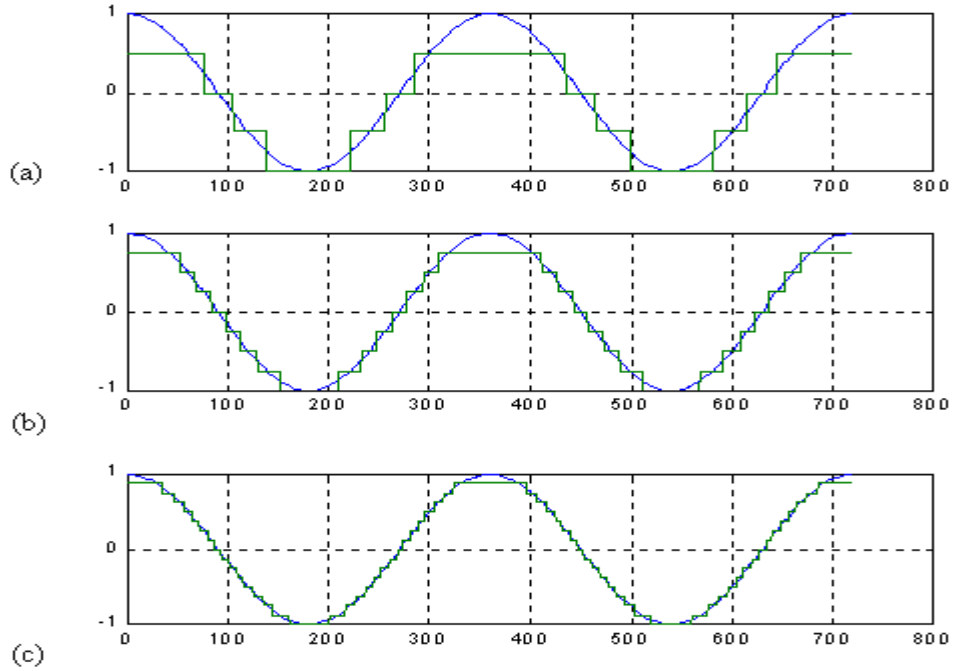
the closest available quantization levels (m_0, m_1, \dots, m_{M-1}). Each quantization bin has a width of δ , where $\delta = (v_{max} - v_{min})/M$, and the noise introduced at each point of quantization can range between $-\delta/2$ and $\delta/2$. In digital hardware, where fixed-point arithmetic is typically used, it is often necessary to implement an automatic gain control on the sampled signal before quantizing, so as to fit the entire range of the incoming signal to the full scale of the quantizer in order to prevent overflow and excessive quantization error.

3.2 Performance Measurements of ADCs

There are many performance considerations that need to be made when searching for the ideal ADC for a specific application. Since different ADC technologies give different performance advantages, tradeoffs typically have to be made in matters of performance, technological availability and cost. Performance issues usually considered include resolution, signal-to-noise-ratio (SNR), power efficiency, and spurious free dynamic range (SFDR) [3].

3.2.1 Resolution and Accuracy

Resolution refers to the degree to which an ADC is sensitive to changes in value of the incoming analog signal. In mathematical measures, the resolution of a uniform quantizer is defined as the width between the quantization levels, δ . Since a uniform quantizer can deliver an accuracy of up to $\delta/2$, and since the range of the quantizer is generally normalized to $(-1,1)$, the resolution of an ADC will simply be determined by the number of bits available for quantization. High resolution ADCs are usually required in applications where small changes in signal amplitude need to be detected. If the needed amount of resolution is not provided, many vital changes in amplitude will go undetected, and this could render many data acquisition systems useless to their users. Typically, the power requirements for ADCs rise exponentially with increase in quantization bits, it is therefore necessary for engineers to determine the minimum amount of resolution required for the satisfactory operation of digitized systems. Figure 3.3 shows the result of digitizing a sinusoidal waveform with different degrees of resolution.



Equation 3.3: Illustration of ADC resolution for: a) 2 bit quantizer; b) 3 bit quantizer; c) 4 bit quantizer

3.2.2 Signal-to-Noise-Ratio (SNR)

As mentioned at the beginning of this chapter, due to the replacement of an infinite valued signal by a finite valued approximation, quantization will always introduce some error into a digitized signal. The error signal, referred to as quantization error signal, is the difference between the incoming analog signal and the quantized signal. For statistical purposes, if we assume that the error signal is uniformly distributed within each quantization level, the mean squared noise power P_n will be defined by:

$$P_n = \frac{\delta^2}{12R}, \quad (3.1)$$

where δ is still the quantization step size, and R is the input resistance of the ADC [3]. Keeping this in mind, if we also assume that the analog input to the ADC is a sinusoid with amplitude equal to the range of the quantizer, the maximum possible theoretical signal-to-noise ratio (SNR) will be:

$$SNR = 6.02n + 1.76 + 10\log_{10}\left(\frac{f_s}{2f_{\max}}\right) (dB), \quad (3.2)$$

where f_s is the sampling frequency, f_{max} is the maximum frequency of the input analog signal, and n is the number of available quantization bits [17,18]. This theoretical SNR is usually approximated as $6n$ (dB), for the case where sampling is done at the Nyquist rate ($f_s = 2f_{max}$), and the 1.76 dB is neglected. This implies that for every additional quantization bit, the SNR of the digitized signal can be improved by approximately 6 dB.

From equation 3.2, it can be observed that as the sampling rate is increased past the Nyquist rate, the SNR also improves. This occurs due to the spreading of the quantization noise power over an increasingly widening frequency band, especially since the quantization noise does not vary with frequency. For this reason, oversampling is often used to realize higher SNR values than would normally be achieved.

3.2.3 Spurious Free Dynamic Range (SFDR)

Spurious free dynamic range (SFDR) is a measure indicating how well an ADC can identify a very small signal in the presence of a much larger signal. While the SNR gives the ratio between the signal power and the noise power, the SFDR is the ratio between the signal power and the peak power of the largest spurious product present within the desired band. The SFDR is a very important performance measure when dealing with radio receivers where the bandwidth of the desired signal may be much smaller than the Nyquist bandwidth. In these types of situations, filtering after digitization may result in larger SNR values, but the obstruction posed by the large spurious products present in the band of the desired signal can only be indicated by the SFDR. Techniques such as dithering, phase plane compensation, projection filtering and others, developed to improve the SFDR performance of ADCs are well discussed in [19,20,21].

3.2.4 Power Dissipation

The power dissipation of an ADC often refers to the total power expended in the realization of a digital word. The power dissipation of an ADC increases with the implemented conversion speed. Furthermore, although high-performance components and more bits are needed to deliver better resolution, power dissipation also increases with ADC accuracy. As will be explained in the following sections, the amount of power required to deliver a given level of conversion performance (speed, accuracy, linearity, etc) varies with different ADC architectures.

3.3 Practical ADC Architectures

The quest to efficiently perform the task of analog-to-digital conversion has led to the development of different design architectures. All ADC architectures function through algorithms that attempt to minimize the difference between the input signal and an approximation signal generated from a reference source voltage. Since every ADC architecture has its own strengths and weaknesses, there is no single one that can best fit all applications, but tradeoffs can be made to find the best fit for every application. ADC architectures commonly used include parallel ADCs, successive approximation ADCs, sigma-delta ADCs, , and many others.

3.3.1 Parallel Converter Architecture

The parallel ADC architecture, also known as the flash ADC architecture, provides the fastest approach to digitizing an analog signal. The parallel architecture uses a resistive divider circuit to generate the voltages at which the quantization boundaries are set. An n -bit parallel ADC would require 2^n+1 resistors to generate the 2^n-1 voltage levels that serve as reference levels for 2^n-1 comparators, to which the incoming analog signal is simultaneously applied. An additional comparator is often necessary to indicate overflow conditions at full scale [11]. The comparator states are converted into an n -bit binary word through the use of an output logic circuitry. Since all bit values are found simultaneously, the speed of the parallel ADC is limited by the comparator switching speed and the propagation delay of the logic gates in the encoding circuit. Figure 3.4 illustrates the general architecture of a parallel ADC.

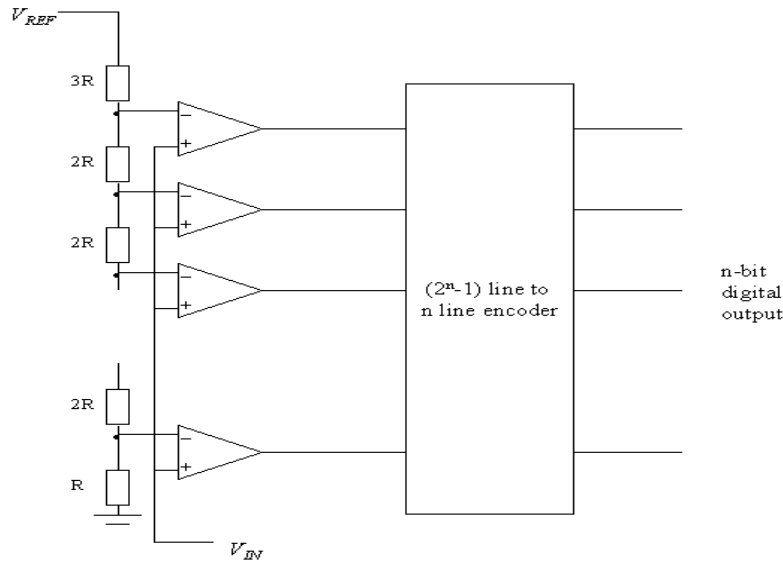


Figure 3.4: Parallel ADC architecture

One of the major limitations of the parallel ADC architecture is the exponential increase in complexity experienced in attempting to achieve higher quantizer resolutions. Each additional bit of resolution would require that the available circuitry be doubled, thereby doubling the power requirements of the ADC. In reality, even more power would be required if the desired increase in resolution is not to affect the overall conversion speed of the ADC. More practical limitations are placed due to the accuracy requirements of all reference circuits that are also doubled with every additional bit. Due to the circuit complexity and practical difficulties associated with realizing higher resolution conversion, parallel ADCs are typically used for lower resolution applications (under 4 bits).

3.3.2 Successive Approximation Architecture

Successive approximation is the most widely used ADC architecture to date. This is because of the big range of conversion speeds and quantization resolutions (between 8 and 16) that can be attained with this architecture. The operation of the successive approximation architecture is analogous to an intelligent tree search algorithm. A series of conversion trials are used to search through a tree containing all quantization levels, with the result of each trial determining the next branch to be followed on the subsequent trial.

At the first trial, the most significant bit (MSB) is applied to a digital-to-analog converter (DAC), and the subsequent analog output is compared to the analog input signal by the comparator. The MSB is retained if the DAC's output is less than the analog input, and discarded otherwise. The control logic applies the next MSB and the same sequence is repeated. This procedure is repeated until the least significant bit (LSB) is attained.

A sample and hold circuit is required to keep the input signal stable to the required accuracy as each bit is being processed. Although the digital logic circuitry can be implemented with an assembly of TTL logic gates and flip-flops, a successive approximation register (SAR) is typically used. The SAR acts as a serial to parallel data converter that takes the output of each comparator function and stores them in the appropriate register positions until the LSB is determined [11]. Figure 3.6 illustrates the structure of the successive approximation architecture.

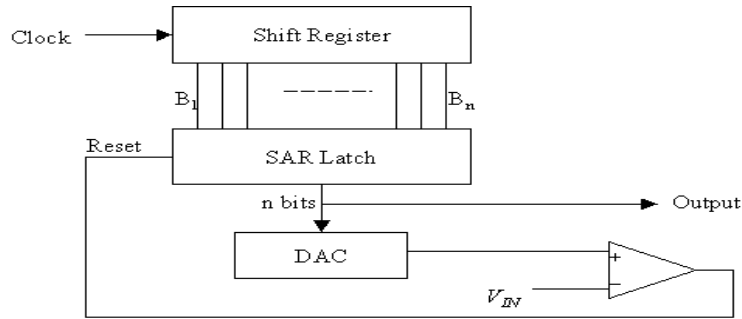


Figure 3.5: Successive Approximation Architecture

Due to the reduced complexity of its circuitry, the power dissipation in the successive approximation architecture is much less than that of the flash architecture, but conversion speeds are much slower, resulting in significantly lower sampling rates.

3.3.3 Sigma-Delta Converter Architecture

The operation of a sigma-delta architecture (also known as a delta-sigma architecture) is based on the translation of high-speed, low-resolution samples of an input signal into a slower, higher resolution digital output [11]. This architecture is also referred as the oversampling DAC architecture because the analog input signals are sampled at a rate much greater than the Nyquist rate. The input samples are first summed up by an integrator, which then passes the summation result to a comparator to decide if the output bit should be a 1 or a 0. The decision of the comparator is serially fed to a digital filter, as well as a feedback loop that operates to push the integrator voltage towards zero. A very large positive input will result in an initial positive comparator output, causing a 1 to be recorded. At the same time, a high reference voltage will be fed back into the comparator where it would be subtracted from the input voltage to produce a new output. As this cycle is repeated, the average value of the DAC converges to the value of the input signal. Figure 3.5 illustrates the operation of the delta-sigma algorithm.

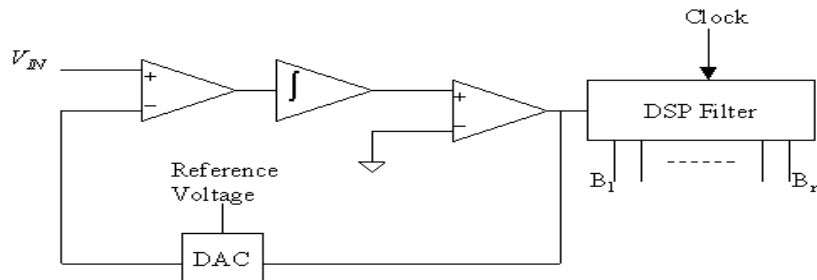


Figure 3.6: The delta-sigma architecture

The sigma-delta converter serves as a middle ground between the flash and successive approximation ADCs with respect to the trade-off between conversion speed and power dissipation.

3.4 Chapter Summary

Analog-to-digital conversion is a crucial step in processing information within a modern digital receiver. It is also a major source of power consumption. In this chapter, we have presented a simple overview of A/D conversion principles, as they relate to power consumption. In the next chapter, we explore the tradeoffs between transmitter power requirements and ADC power consumption in a software radio system employing turbo codes, leading us to the concept of 'smart' Analog-to-Digital conversion.

CHAPTER 4

ADAPTER RECEIVER DESIGN FOR POWER OPTIMIZATION

The performance of digital communication systems is often based upon how much transmission power is required to deliver a given level of bit error rate (BER). The quantization error introduced by the operation of the ADC increases the BER, thereby degrading system performance. As explained in Chapter 3, although increasing the resolution of the ADC will reduce the degree of bit-error rate degradation, it will also increase the overall system power dissipation. In this chapter, we introduce the idea of a smart turbo decoder for software radio implementation. This decoder operates by increasing the transmission power when necessary, but reducing the power dissipated within the receiver by altering the ADC resolution as well as the number of decoding iterations without significant deterioration in bit-error performance. We explore these tradeoffs by examining three typical examples of wireless communication links: Personal Communication Service (PCS), Wireless Local Area Network (WLAN), and Local Multipoint Distribution Service (LMDS).

4.1 Effect of Quantization on BER Performance of Turbo Codes

To date, most of the performance studies done on turbo codes have been performed using floating-point arithmetic; however, fixed-point arithmetic is the more likely choice of implementation in a practical decoder. Ideally, both the amplitude of the received signal and the coefficients within the decoder could be represented to infinite precision, but due to the need for digitization, these values have to be discretized, consequently adding noise to the system. The work done in this thesis is based upon the effects of quantizing the received signal only. In practice, the performance of the decoder is also limited by the number of bits available for internal representation (which is typically much higher than the number of quantization bits). However, that phenomenon is investigated in [1], and is not further investigated in this thesis. This phenomenon has not been incorporated into this thesis work.

The block diagram of the system investigated in this chapter is depicted in Figure 4.1. The input information source is a random bit generator, producing either $d_k = 0$, or $d_k = 1$ with equal probability. The turbo encoder is made up of two RSC encoders, each having a rate of $1/3$, a constraint length of 4, and an octal generator matrix of $\{15,17\}$. The turbo encoder operates with a frame size of 1024, made up of 1021 information bits and 3 termination bits, added to

implement a single-trellis termination (the first RSC is returned to the all-zero state while the second RSC is left open). A simple random interleaver is used to decorrelate the order of the inputs at the constituent RSC encoders. The output of the encoder is made up of the systematic bit of the first RSC encoder and the parity bits of both encoders, resulting in an overall code rate of 1/3. The coded bits are modulated using binary phase shift keying (BPSK), then transmitted through an additive white Gaussian noise (AWGN) channel where they are corrupted by noise of the form $n_k \in N(0, \sigma^2)$.

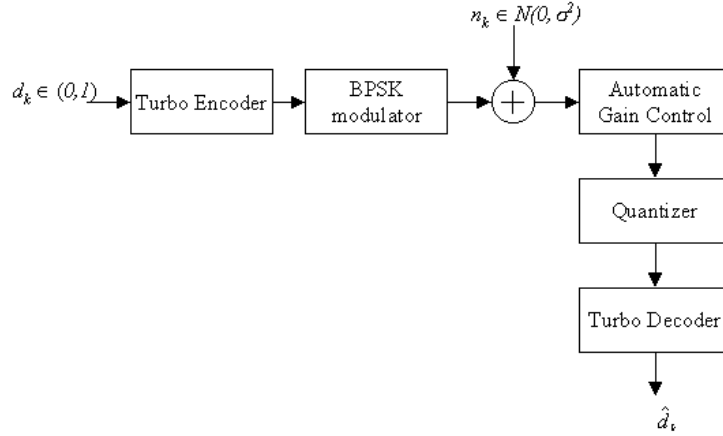


Figure 4.1: System block diagram

On receiving the transmitted signal, the amplitude of the received signal is adjusted by the automatic gain control in order to achieve the best signal-to-distortion ratio (SDR) at the quantizer. For any given input distribution, there is an optimal scaling factor that would maximize the quantizer's SDR, and this is found for every transmitted frame, according to [1]. The scaled signal with range V_{max} to V_{min} (typically $V_{min} = -V_{max}$) is then mapped to integer values between -2^{n-1} and $2^{n-1}-1$, where n is the number of bits available for quantization (resolution). The quantization is implemented by dividing the entire range of the scaled signal into 2^n equal bins, with each bin having a width of $\delta = (V_{max} - V_{min})/2^n$, and bin boundaries at $\{-\infty; V_{min} + \delta/2; \dots; V_{min} + (2m-1)\delta/2; \dots; V_{max} + 3\delta/2; +\infty\}$, $m = 1, \dots, 2^n-1$. After a search through all available bins, each continuous input sample is replaced by the quantization level that corresponds to the sample's resident bin. The quantized signal is finally passed on to the turbo decoder, which uses the Log-MAP algorithm to demodulate and decode the received signal.

The approach taken in this part of the thesis was to observe the effect of changing the number of quantization bits, on the BER curve of the turbo encoded system. The system was first simulated without any quantization (effectively a 32-bit floating point quantization due to the

limitations of the computer), after which it was again simulated using quantizers with resolution values $n = 3, 4, 6$ and 8 . The results of the simulation are illustrated in figure 4.2 below.

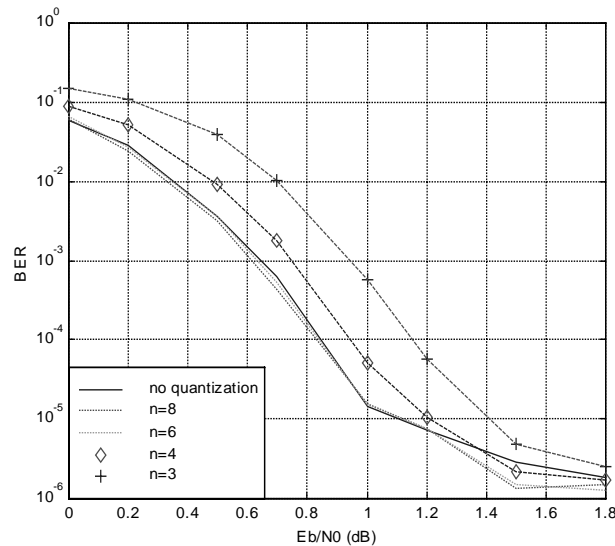


Figure 4.2: BER for different resolution quantizers

Figure 4.2 contains BER curves plotted against E_b/N_0 for each of the simulated quantizer resolutions. Since the internal resolution of the decoder was not considered in the simulations, the noise affecting the decoder is mainly contributed by channel noise and by the quantization error of the quantizer. A couple of observations and inferences can be made from the plot above:

- The BER curves of all simulated resolutions tend to converge roughly at lower E_b/N_0 values. This means that the effects of AWGN tend to dominate in this area of the curve, while quantization effects become more dominant at higher E_b/N_0 values.
- The BER curves of all simulated resolutions again begin to converge from an E_b/N_0 value of about 1.6 dB. This means that, in realizing very low BER values (below 10^{-6}), a 3-bit ADC in the decoder would begin to function as effectively as an infinite-precision implementation.
- The performance of the simulated system improved with increasing quantizer resolution, but not without obeying the law of diminishing returns (it should be kept in mind that in order to achieve this, optimal scaling according to [1] is necessary).
- The difference between the BER curves obtained from using no quantization and using the 6-bit quantizer is negligible. This implies that no more than a 6-bit quantizer is required for efficient decoding of turbo codes. Anything more will result in unnecessary power consumption, for no appreciable gains in performance.

4.2 A Variable Resolution ADC

A very important feature of figure 4.2 is the fact that any BER value below about 0.08 can be attained using any of the simulated quantizer resolutions. In order to use a lower resolution quantizer, the input power would have to be increased so as to transition between BER curves in Figure 4.2. For this reason, it would prove useful to investigate the power savings that may be derived from switching quantizer resolutions in the system of figure 4.1, thus the need for a variable resolution quantizer. To evaluate potential savings, it is crucial to have a model which relates precision requirements to ADC power consumption within an ASIC implementation. In [5], Aust investigated the power consumption characteristics of a variable resolution quantizer based upon a redundant signed-digit (RSD) cyclic ADC architecture.

4.2.1 RSD Cyclic ADC Architecture

Like the conventional cyclic algorithm, the RSD cyclic algorithm uses an iterative approach in determining the digital output of an ADC from a current or voltage input. The RSD cyclic algorithm has the added advantage of being able to tolerate loop offsets and comparator inaccuracies due to its employment of a ternary alphabet $\{-1,0,1\}$, instead of a binary alphabet $\{0,1\}$. However, the output of the RSD cyclic ADC must be represented in binary format specific to the hardware of use. Figure 4.3 illustrates the operation of the RSD cyclic algorithm.

Beginning with the most significant bit, the RSD cyclic ADC uses the Sweeney-Robertson-Tocher division principle [Ginetti] to calculate one bit for every conversion cycle. The loop variable N , is set to the resolution of the ADC, n . The residue current, I_{res} , which starts off as the sampled input current, is multiplied by two, then compared two constant values: $2Q$ and $2P$, where

$$2Q \in [-I \ 0], \quad (4.1)$$

$$2P \in [-I \ 0] \text{ and} \quad (4.2)$$

$$I_{ref} = \frac{(I_{max} - I_{min})}{2}. \quad (4.3)$$

If $2I_{res}$ is less than $2Q$, the bit decision, b_i , will be set to -1 and I_{ref} is added to $2I_{res}$. If $2I_{res}$ is greater than $2P$, b_i will be set to 1 and I_{ref} is subtracted from $2I_{res}$. And if $2I_{res}$ is greater than $2Q$ but less than $2P$, b_i will be set to 0 and I_{res} is unaltered. The loop variable is then decremented by

one, and this algorithm is repeated until the least significant bit has been determined. The algorithm operation is summarized by equation 4.4 below.

$$b_i = \begin{cases} -1 & (I_{res} = 2I_{res} + I_{ref}) & \text{if } 2I_{res} < 2Q \\ 0 & (I_{res} = 2I_{res}) & \text{if } 2Q \leq 2I_{res} \leq 2P \\ 1 & (I_{res} = 2I_{res} - I_{ref}) & \text{if } 2I_{res} > 2Q \end{cases} \quad (4.4)$$

4.2.2 Variable ADC Simulation

In [5] Aust describes a procedure used to investigate the power consumption characteristics of a variable ADC operating with the RSD cyclic algorithm described above. Due to time and cost constraints associated with hardware implementation, the test was not performed with an actual chip. Instead, a simulation software, HSPICE, was employed to simulate the ADC for a sample rate of 1.7 Mbps, for a duration of 12 bit cycles. The power consumption of the ADC was calculated as a function of the maximum input current measured during simulation. The results of these simulations that were used within this thesis report are displayed in figure 4.4.

Table 4.1: Power dissipation of variable resolution ADC

ADC resolution (n)	Power Dissipation (milliWatts)
2	2.62
4	3.42
6	4.21
8	5.39

4.3 Performance considerations of a Variable Resolution Turbo Decoder

As mentioned earlier, with respect to BER performance, there is no significant gain in implementing an 8-bit resolution decoder instead of a 6-bit resolution decoder. So the 6-bit resolution decoder will be taken as the implementation of choice for optimum performance and power efficiency. According to Figure 4.3, in order to transition from a 6-bit quantizer to a 4-bit quantizer without loss in BER performance, the input power increase required would never exceed a value of 0.2 dB. This value can be obtained by observing the largest jump in E_b/N_0

value required to make a horizontal transition between the error curves of the 6-bit and 4-bit turbo decoders. By the same token, changing from a 6-bit implementation to a 3-bit implementation would require an input power increase of no more than 0.4 dB.

From the data in Table 4.1, it can be observed that by switching from a 6-bit resolution to a 4-bit resolution, the power dissipated by the variable resolution ADC drops from 4.21 mW to 3.42 mW (a power saving of 0.79 mW). Converting these power values into milliwatt based decibel format (dBm), we can see that the dissipated power is reduced from 6.2428 dBm to 5.3403 dBm. In other words, reducing the resolution of the variable ADC from 6 to 4 would result in a power saving of about 0.9026 dB.

The performance feasibility of a variable ADC can be accessed from the observations stated in the previous two paragraphs. Worst case scenario, in order to enjoy the 0.9026 dB power savings achieved by switching from a 6-bit resolution to a 4-bit resolution without loss in BER performance, the transmission power must be increased by a value of 0.2 dB. Since the decibel unit only gives a comparative measure, we cannot from these values determine if a variable decoder would be technically feasible. It is necessary that we consider typical digital communication links, so that actual numerical comparisons can be made.

4.3.1 Performance Considerations for Different Data Rates

As stated in Section 4.2, the variable dissipated power values displayed in Table 4.1 are results for a 1.7MHz clock speed. Since the effective operation of different communication links requires that data be transmitted at different speeds, these power values would be of little use in analyzing a multiple of communication systems. Equation 4.5 gives a relationship between clock speed (transmission frequency) and power dissipation of CMOS devices [7].

$$P = \alpha CV^2 f_{clk} \quad (4.5)$$

where

P = dissipated power,

C = gate capacity,

V = supply voltage,

f_{clk} = clock speed, With equation 4.5, the power

α = signal activity. dissipation measurements of a

CMOS device can be translated from one operational speed to the other. For the case of our variable ADC, the digital part of the circuit accounts for only approximately 5% of total power

dissipation (0.3 mW out of 6.35 mW for a 12-bit resolution). Since the analog part of the circuit mainly consumes steady-state power, a variation in data rate (or clock speed), would have very little effect on the analog-portion power consumption. Thus, equation 4.5 will be used to translate only 5% of the total power dissipation values contained in Table 4.1, since the remaining portion is independent of clock speed. The following sections contain link analysis for three communication systems that support different data rates. Table 4.2 below contains ADC power dissipation values for the different data rates.

Table 4.2: Power dissipation of ADC for different data rates

ADC resolution (n)	Power dissipation		
	270 kbps	20 Mbps	115 Mbps
2	2.51	4.03	11.35
4	3.28	5.26	14.82
6	4.03	6.48	18.23
8	5.16	8.29	23.35

4.3.2 Link Analysis

The following section of this thesis contains a detailed link analysis of three existing communication systems. These systems are the Personal Communication System (PCS), Wireless Local Area Network (WLAN), and the Local-Multipoint Distribution Service (LMDS). These three systems support different types of services, thus delivering different data rates and error performances. The objective of the link analysis is to determine whether the power savings achieved by decreasing the ADC resolution overshadows the extra transmitter power used to maintain the same BER performance.

The approach taken was to determine the practical transmitter powers required for a 6-bit and 4-bit decoder to achieve the desired BER performance, and data rate specifications for each system. For efficient communication, we decided to pick a BER operation point of 10^{-5} , which corresponds to an E_b/N_0 value of approximately 1.1 dB in Figure 4.2. It should be noted though, that the transmitter power values would differ from typical system values due to our use of turbo codes. The systems being investigated are second-generation systems that use error correction

codes which deliver anything from 3-6 dB less performance than turbo codes. The transmitter power is calculated as:

$$P_T = \frac{E_b}{N_0} + R_b - G_T - G_R + k + T_S + PL, \quad (4.5)$$

where R_b is the bit rate, G_T is the transmitter antenna gain, G_R is the receiver antenna gain, k is Boltzman's constant, T_S is the system noise temperature and

$$PL \text{ (path loss)} = PL(d_0) + 10n \log\left(\frac{d}{d_0}\right), \quad (4.6)$$

where:

d_0 is a reference distance at which path loss measurements have been made;

d is the actual communication distance;

n is the path loss exponent.

The path loss exponent is a constant that indicates the rate at which communication path loss increases with distance. It changes with respect to the propagation environment, as low as 2 for free space and as high as 6 for highly obstructive indoor applications. Table 4.3 below illustrates typical values for the path loss exponents for different propagation environments [9].

Table 4.4: Path loss exponents for different propagation environments

Environment	Path loss exponent
Free Space	2
Urban area cellular radio	2.7 to 3.5
Shadowed urban Cellular radio	3 to 5
In building line-of-sight	1.6 to 1.8
Obstructed in building	4 to 6
Obstructed in factory	2 to 3

4.3.3 A Personal Communication System (PCS) Link

PCS is a second-generation wireless standard that provides its users the ability to exploit the telephone network for different types of communication needs, regardless of the location of the subscriber or provider. PCS networks provide voice, data, e-mail, and video communication,

with capabilities for message handling, paging, caller id, and more. Tables 4.4 and 4.5 illustrate the link parameters and power calculations obtained for a typical DCS 1800 PCS system [9].

Table 4.4: PCS link specifications

Frequency (f)	1800 MHz
Probability of bit error (P_b)	10^{-5}
Bit Rate (R_b)	270 kbps
Receiver Noise Figure (N)	6 dB
Receiver Noise Temperature (T_s)	29.37 dBk
Transmit Antenna Gain (G_T)	8 dB
Receive Antenna Gain (G_R)	2 dB
Boltzman's Constant (B)	-228.6 dB

Table 4.5: Transmitter powers for PCS link

Environment	n	d (km)	PL	Transmitter Power (mW)		
				6-bit	4-bit	Power Loss
Urban area Cellular radio	2.7	3	117.43	229.81 μ W	235.16 μ W	5.35 μ W
Shadowed Urban Cellular radio	3	3	121.86	637.33 μ W	652.17 μ W	14.8 μ W
	4	3	136.63	19.12 mW	19.56 mW	0.45 mW
	5	3	151.41	0.575 W	0.588 W	0.013 W

Except for the case where $n = 5$, all transmitter power values in table 4.5 are achievable by practical PHS PCS systems. This only means that, with a path-loss exponent of 5 and a path distance of 3 km, the link will not be maintained. By the same token, the extremely small transmitter power values are due to the use of turbo code BPSK modulation, which is not currently available for second generation PCS. This only implies that using a typical PCS transmitter in these situations would deliver more impressive performance. But, for all the other situations, the maximum increase in transmitter power is 0.45 mW. From table 4.2, at the PCS data rate, switching ADC resolution from 6-bits to 4 bits would result in a power savings of 0.75 mW, and this power savings overshadows the transmitter power increase. Moreover, in a cellular downlink, transmitting up to 10 mW extra power would not be a problem for a base-station. The

return link can be implemented in a traditional manner with minimal transmitter power since the base station can decode the received signal with a high resolution ADC. Consequently, a variable ADC can deliver considerable power savings in a PCS mobile unit.

4.3.4 Analysis of a Wireless Local Area Network (WLAN) Link

WLAN systems are typically indoor or very small-range out door wireless systems developed to provide a centralized information network for use in locations such as offices, shops, malls, hospitals and residences. The WLAN standard considered is the IEEE802.11 specification [29]. The 802.11 specifies an operational frequencies around the 2.45 GHz band, supporting data rates between 1-20 Mbps, and providing a link BER as good as 10^{-5} . Tables 4.6 and 4.7 illustrate the link parameters and power calculations obtained for the WLAN system.

Table 4.6: WLAN link specifications

Frequency (f)	2.45 GHz
Probability of bit error (P_b)	10^{-5}
Bit Rate (R_b)	20 Mbps
Receiver Noise Figure (N)	6 dB
Receiver Noise Temperature (T_s)	29.37 dBk
Transmit Antenna Gain (G_T)	0 dB
Receive Antenna Gain (G_R)	0 dB

Table 4.7: Transmitter powers for WLAN link

Environment	n	d (km)	PL (dB)	P_T (mW)		
				6-bit	4-bit	Power Loss
Obstructed in factory	3	0.05	91.20	405.51 μ W	414.95 μ W	9.44 μ W
Obstructed in building	4	0.05	108.2	20.23 mW	20.70 mW	0.47 mW
	5	0.05	125.2	1.012 W	1.035 W	23 mW
	6	0.05	142.2	50.58 W	51.76 W	1.18 W

A typical WLAN device may deliver up to a transmitter power of 1 Watt. The WLAN link shows transmitter powers requirements that fall below this limit. From Table 4.2, at the WLAN data rate, switching ADC resolution from 6-bits to 4 bits would result in a power savings

of 1.22 mW Up to a path exponent of 4, the savings in ADC power consumption overshadows the increase in transmitter power. This implies that an overall power savings would be enjoyed by implementing a variable rate ADC in the decoder.

On the other hand, environments with path loss exponents greater than 4 would require transmit power increases that surpass the power savings in the ADC. In these cases, implementing a variable ADC in the decoder would lead to power loss in the overall system. But, a variable resolution ADC can still be used to increase battery life in mobile units, even though the central data port would need to transmit more power.

4.3.5 A Local-Multipoint distribution system (LMDS) network

In the US, LMDS networks are to operate between the 27.5-31.3 GHz frequency band recently auctioned by the FCC. With an operating bandwidth of 1300 MHz, LMDS systems have the ability to support communication at extremely high data rates. By the same token, the extremely high operational frequency delivers a link that is highly susceptible to vegetative dispersion and rain fading. Because of the high-directivity of LMDS signals, subscribers exist typically as fixed-site stations, within a 5-kilometer range.

Since LMDS systems are typically designed to handle efficient voice, video, and data communication, it would be practical to demand a bit error rate (P_b) $< 10^{-6}$ and a data rate (R_b) > 115 Mbps. Also, due to the line-of-sight design of LMDS networks, a path loss exponent of 2 was used, along with the provision of sufficient fade margins for rain and vegetative fading [31].

Table 4.8: LMDS link specifications

Frequency (f)	~ 29 GHz
Bit Rate (R_b)	115 Mbps
Receiver Noise Figure (N)	6 dB
Receiver Noise Temperature (T_s)	30.02 dBk
Transmit Antenna Gain (G_T)	16 dB
Receive Antenna Gain (G_R)	36 dB
Rain Fade Margin	8 dB (1 km) / 40 dB (5 km)
Vegetative Fade Margin	15 dB (1 km) / 15.7 (5 km)

Table 4.9: Transmitter powers for LMDS link

Propagation Condition	d (km)	PL (dB)	P_T		
			6-bit resolution	4-bit resolution	Power Loss
Dry air	1	121.7	605.3 μ W	619.0 μ W	13.7 μW
	5	135.7	17.86 mW	18.28 mW	0.47 mW
Rain	1	129.3	3.819 mW	3.908 mW	89 μW
	5	161.3	6.053 W	6.194 W	141 mW

Except for the situation with rain at a distance of 5 kilometers, all transmitter power values in table 4.9 are practical LMDS power values. This would mean that, under rainy conditions, a 5-kilometer link capable of delivering the required data rate and BER performance cannot be maintained. But, for all the other situations, the maximum increase in transmitter power is 0.47 mW. From Table 4.8, at the LMDS data rate, switching ADC resolution from 6-bits to 4 bits would result in a power savings of 3.41 mW. This implies that an overall power savings would be enjoyed by implementing a variable rate ADC in the decoder.

4.4 Optimizing Decoding Iterations for further Power Conservation

Turbo decoding is typically practiced in a fixed-iterative fashion, where the same number of decoding iterations are repeated for every received frame. When transmitting through an AWGN channel, different frames are transmitted with different degrees of noise corruption. The number of decoding iterations, M , is usually picked to allow sufficient iteration for the worst noise corrupted frame to converge. Most transmitted frames typically require less than M decoding iterations before convergence. The process of decoding turbo codes can be improved to reduce the average decoding computation, and thus achieving even better power efficiency, if the number of decoding iterations is optimized for every transmitted frame.

Different stopping criteria have been introduced to assist in optimizing decoding iterations for turbo codes, with each one having its own edge over the others. In this thesis, we employed the Sign Difference Ratio (SDR) stopping criterion that is introduced in [2]. The SDR criterion works off the principle that the cumulative sign difference between the LLRs of two consecutive iterations will converge to zero about 0.5-1 iterations after the number of bit errors reaches zero. While “good frames” (frames that are easy to decode), will achieve this convergence at different rates as the number of iterations increases, “bad frames” never converge.

By checking the cumulative sign changes in consecutive LLRs, the decoder can be optimized to stop unnecessary iterations for good frames. The SDR criterion delivers comparable performance to other top stopping criteria, and is as computationally efficient as the best previous known technique, the Sign Change Ratio (SCR), with the added advantage of demanding less storage space [2].

Figure 4.5 below shows the BER curves of using fixed iterative decoding versus SDR iterative decoding.

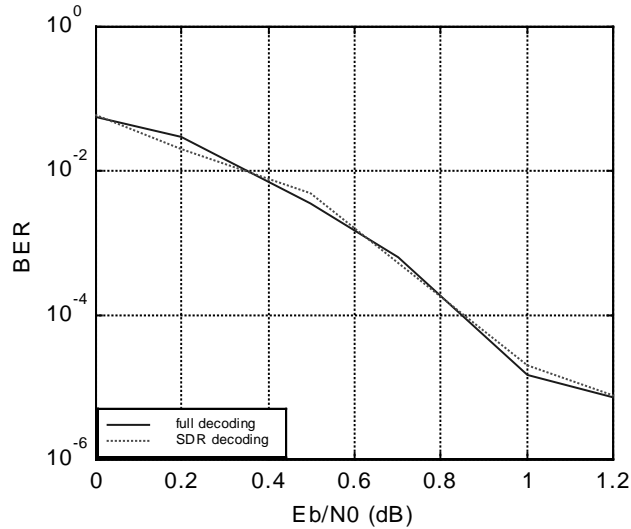


Figure 4.5: BER performance of fixed decoding versus SDR decoding

It can be observed from Figure 4.5 that using the SDR stopping criterion does not result in any obvious deterioration in the BER performance of turbo codes. Figure 4.6 shows the average number of decoding iterations completed when obtaining the results of figure 4.5. It can be observed that the number of decoding iterations drops to somewhere in the order of 4 in the low BER region ($BER < 10^{-3}$). Since this is the most common region of operation for turbo codes, the overall computational savings may lead to significant power savings in a communication system.

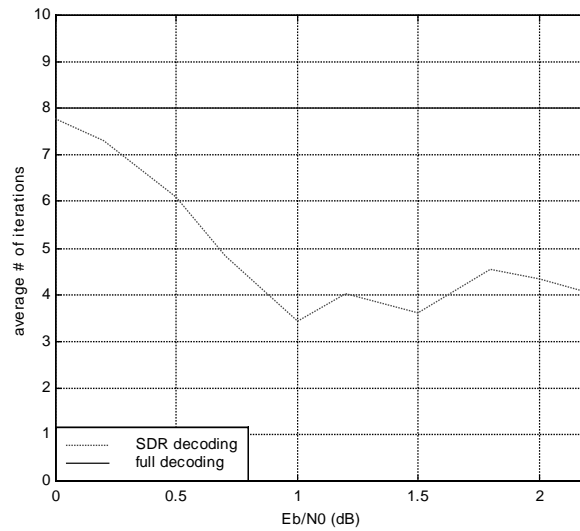


Figure 4.6: Average number of iterations for fixed and SDR decoding

The only problem with the above results, and the results in [yufei] is that the simulations were performed without the implementation of analog-to-digital conversion at the receiver. In order to make investigate how the SDR criterion would affect a more practical system, the SDR simulations were repeated using 3-bit, 4-bit and 6-bit ADC resolutions at the decoder. The BER performances of these simulations were compared to the results of Section 4.1. This is illustrated in Figure 4.7

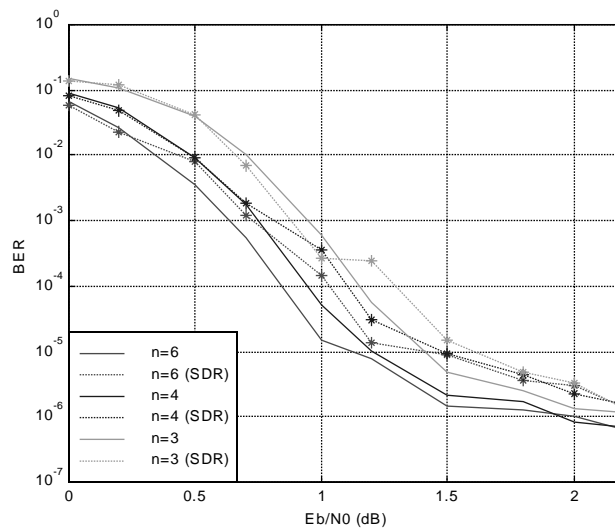


Figure 4.5: BER performance using different ADC resolution with and without SDR stopping criteria

The simulation results illustrated in Figure 4.7 show a significant difference between the BER performance of turbo codes when decoded with a fixed-iterative algorithm, as opposed to an SDR-iterative algorithm. In fact, as we move lower on the BER scale, the performance of all resolutions of the SDR-iteration decoder begin to converge at a slightly higher level than those of the fixed-iterative decoder. On that note, in order to maintain the same BER performance with the SDR decoder (and the same ADC resolution), transmitter power would again need to be increased. The question now is whether we save enough power in the decoder to warrant a transmitter power increase.

In [6], simulations were performed using the Log-MAP decoding algorithm to determine power consumption characteristics of a turbo decoder. These simulations were based upon a 1023 frame size, a code rate equal to 1/3, and a data rate of 125 kbps. The analysis in [6] is centered on the claim that 85 to 90 % of total power dissipated in a CMOS device is due to capacitance switching (dynamic power dissipation). Power dissipation measurements were taken under three different modes of operation: two modes with one active decoder and one idle decoder; and one mode with two decoders idle. It was found in [6] that in an 8 millisecond time frame, each decoding iteration dissipates 47.3 mW for a total of 1.638 milliseconds, with 11.1 mW dissipated for the remainder of the time slot(the idle state). These power values would vary slightly from decoder to decoder based upon the efficiency with which the decoding algorithms are written. Based upon the work in [6], values of energy consumption per decoding iteration are displayed in table 4.10 below.

Table 4.10: Energy consumption per decoding iteration

Iteration	Energy Consumed (μJ)
1	148
2	207
3	267
4	326
5	385
6	445
7	504
8	563

On this token, Table 4.11 displays the minimum power savings achieved when implementing SDR-iterative decoding over a full-iterative decoding.

Table 4.11: Power savings of the SDR turbo decoder

E_b/N₀	Average # iterations when using SDR				Average energy saved (μJ)
	6-bit ADC	4-bit ADC	3-bit ADC	Average	
0	7.5714	7.7778	8	7.7807	13
0.2	6.5833	7.3418	8	7.3084	41
0.5	5.0789	5.5703	7.6250	6.0914	113.2
0.7	4.0118	4.3198	6.2174	4.8497	186.8
1.0	3.1290	3.2814	3.9259	3.4454	270.1
1.2	3.8787	3.9208	4.2705	4.0233	235.8
1.5	3.4093	3.4776	3.9333	3.6067	260.5
1.8	4.3039	4.4225	4.9329	4.5531	204.4
2.0	4.1491	4.3928	4.4712	4.3377	217.2
2.2	3.9287	3.9706	4.2822	4.0605	233.6

Most communication systems today are designed to deliver a BER of 10^{-5} or better. According to the encoder and decoder structures implemented in this thesis, this would imply operation above an E_b/N_0 value of 1.2 dB. From Table 4.10 above, it is evident that using the SDR-decoding algorithm would help achieve energy savings between 200 and 280 μJ, and that is without considering the power savings in the FPGA.

In maintaining the same BER performance, Figure 4.6 shows that the transmitter power would have to be increased by about 0.5 dB. Considering that all the communication systems analyzed in section 4.3 operate with transmitter powers ranging between 10 mW and 1 W, table 4.12 illustrates the power savings comparisons of an SDR-decoder operating under different transmitter power conditions.

Table 4.12: Energy considerations for SDR-decoding

Order of P_T (mW)	P_T increase (mW)	Energy loss from P_T increase (μJ)	Energy savings from SDR-decoding (μJ)	Comments
10	1.2	9.76	200 - 280	✓ SDR-decoding
100	12.2	97.76	200 - 280	✓ SDR-decoding
1000	122.2	977.76	200 - 280	- SDR-decoding

The results displayed in figure 4.12 illustrate that; an SDR-decoder will deliver considerable energy savings in applications where the order of the transmitter power falls below 100 mW. But in applications with transmitter powers in the order of 1Watt, it will be more economical and convenient to implement full-iterative decoding. In conclusion, there are many applications

where the use of an SDR-iterative decoder can help achieve better power efficiency, while maintaining the required BER performance.

CHAPTER 5

CONCLUSION AND FUTURE WORK

In the course of this thesis, we showed the relationship between the resolution of the ADC in a turbo decoder, the power dissipation, and the BER performance of a turbo coded system. As expected (with certain limitations), increasing the ADC resolution delivers better BER performance, but at the expense of increasing power dissipation in the decoder. We then investigated the technical feasibility of increasing transmitter power to allow the dynamic switching of ADC resolution in a turbo decoder in order to reduce overall power consumption. Through the aid of link budgets, we showed that, under different conditions, a variable resolution ADC decoder could deliver better power management for different existing communication systems.

We also investigated the effect of using the sign-difference-ratio (SDR) iterative stopping criterion on the performance of turbo codes. We showed that without quantization in the receiver, the SDR decoder will perform as well as a full-iterative decoder, but in simulating a more practical system with an ADC in the decoder, the SDR decoder depreciates the BER performance of turbo codes. In order to maintain the same BER with the SDR decoder, the transmitter power must be increased. We then proceeded to quantify the amount of power savings delivered by implementing the SDR as opposed to a full-iterative decoder, and showed that the power savings in the decoder would overshadow the transmitter power increase.

Putting all our research findings together, it can be inferred that a smart turbo decoder that adaptively switches ADC resolution and iteration per frame can deliver better power efficiency while delivering similar BER performance as a traditional fixed-iterative, fixed-resolution decoder.

5.1 Future Work

All the trade-off issues analyzed in this thesis were discussed from results of simulations that were independently performed. The relationships between BER and ADC resolution, between ADC resolution and power dissipation, and between decoding iteration and power dissipation, were all independently investigated, then combined to give the findings of this thesis. In order to obtain a more comprehensive/valid understanding of all the trade-off issues related to our smart

decoder, it would be very advantageous to develop a software/hardware simulation that will incorporate all these variables into one simulation platform.

The resolution of the variable resolution ADC is adjusted externally with an RF pre-processing circuit that runs off a control signal. After the digitization of one analog value, a new resolution can be loaded onto the pre-processing circuitry by feeding in the control signal. In the simulations in [Carrie], the control signal was simply the engineer picking a resolution on the simulation device, and the nature of the required circuitry or a practical control signal is not known. In other words, the variable resolution ADC simulated is not self adaptive. For a variable resolution turbo decoder the control signal to the pre-processing circuit needs to be self-adaptive. This would require the design of a pre-processing logic circuit that runs off the power levels of the received signal. Because the nature and operational conditions of this RF circuit are not known, the received analog signal may also have to be conditioned before being useful as a control signal to the pre-processing circuit. The smart decoder will then use the SDR algorithm to adapt the number of decoding iterations per frame. Simulating this system along side a traditional decoder will give a more comprehensive understanding of all existing trade-offs. Figure 4.4 below illustrates this idea.

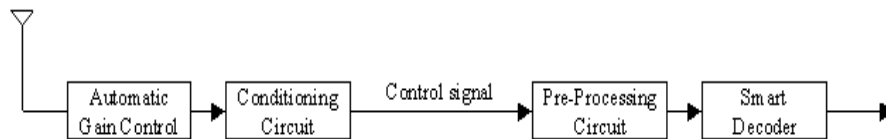


Figure 5.1: Smart Turbo Decoder

Furthermore, the link budgets displayed in Chapter 4 are based upon second generation wireless systems. In these cases, increasing transmitter power by a little bit does not have adverse effects on the entire system. But since turbo codes are being specified for use in third generation systems, it would be informative to investigate the feasibility of our smart decoder in one of these systems. Power control is a significant issue in third generation systems, and a slight increase in transmitter power could adversely affect the nature of interference in one of these systems. It would thus be informative to investigate how much interference would be introduced into the propagation environment by increasing transmitter power to facilitate the use of our smart decoder.

Bibliography

- [1] Yufei Wu and Brian D. Woerner, "The Influence of Quantization and Fixed Point Arithmetic Upon the BER Performance of Turbo Codes," Proc. IEEE Veh. Tech. Conf., (Houston, Tx), May 1999.
- [2] Yufei Wu and Brian D. Woerner, "A Simple Stopping Criterion for Turbo Decoding," IEEE Comm. Letters, to appear.
- [3] Jeffery A. Wepman, "Analog-to-Digital Converters and Their Applications in Radio Receivers," IEEE Communications Magazine, pp. 39-45, May 1995.
- [4] Mathew C. Valenti, "Iterative Detection and Decoding of Wireless Communications," PHD Dissertation, Virginia Polytechnic and State University, July 1999.
- [5] Carrie Aust, "A Low-Power, Variable-Resolution Analog-to-Digital Converter," M.S. Thesis, Virginia Polytechnic and State University, June 2000.
- [6] Jia Fei, "On a Turbo Decoder Design for low Power Dissipation," M.S. Thesis, Virginia Polytechnic and State University, July 2000.
- [7] Jan M. Rabaey and Massoud Pedram, Low Power Design Methodologies. Norwell, MA: Kluwer Academic Publishers, 1996.
- [8] John Gardiner and Barry West, Personal Communication Systems and technologies. Norwood, MA: Artech House Inc., 1995.
- [9] Theodore S. Rappaport, Wireless Communications, Principles and Practice. Upper Saddle River, NJ: Prentice Hall, 1996.
- [10] G.B. Clayton, Data Converters. New York, NY: John Wiley & Sons Inc., 1982.
- [11] Michael J. Demler, High-Speed Analog-To-Digital Conversion. San Diego CA: Academic Press Inc., 1991.

- [12] C. E. Shannon, "A Mathematic Theory of Communication" Bell Sys. Tech. J., vol. 27, pp. 379-423 and 623-654, 1948.
- [13] C. Berou, A. Glavieux, and P. Thahitimasjshima, "Near Shannon Limit Error Correction Coding and Decoding: Turbo-Codes(1)," Proc. IEEE Int. Conf. On COMM. (Geneva, Switzerland), pp. 1064-1070, May 1993.
- [14] A. Wepman, J. R. Hoffman, and J. E. Shroeder, "An initial study of RF and IF digitization in radioreceivers," NTIA Report 95-xxx, 1995.
- [15] G. D. Forney, "The Viterbi Algorithm," Poc. IEEE, vol. 61, pp. 268-278, March 1973.
- [16] L. R. Bahl, J. Cocke, F. Jelinek and J. Raviv, "Optimal Decoding of Linear Codes for Minimizing Symbol-Error-Rate," IEEE Trans. Inf. Theory, vol. 20, pp. 284-287, Mar. 1974.
- [17] R. M. Lober, "A DSP-Based Approach to HF Receiver Design: Higher Performance at a Lower Cost," RF Design, vol. 16, no. 8, 1993, pp. 92-100.
- [18] M. Amarandos and S. Andrezyk, "Considerations in the Development of A Low Cost, High Performance Receiver Based on DSP Techniques," DSP Applications, vol. 2, no. 12, Dec., 1993, pp. 1-14.
- [19] N. W. Spencer, "Comparison and Stae-of-the-Art Analog-to-Digital Converters," Massachusetts Institute of Technology, Licoln Laboratory, Lexington, MA, Project Report AST-4, March, 1998.
- [20] F.H. Irons and T. A. Rebold, "Characterization of High-Frequency Analog-to-Digital Converters for Spectral Analysis Applications," Massachusetts Institute of Technology, Licoln Laboratory, Lexington, MA, Project Report AST-2, Nov., 1982.
- [21] N. T. Tao and M. Vetterli, "Optimal MSE Signal Reconstruction in Oversampled A/D Conversion Using Convexity," Proc. ICASSP '92. 1992, vol. 4, pp. 165-168.

- [22] R. W. Lucky, *Silicon Dreams: Information, Man, and Machine*. New York, NY: St. Martin's Press, 1989.
- [23] P. Elias, "Coding of Noisy Channels," *IRE Conv. Record*, vol. 4, pp. 37-47, 1955.
- [24] S. B. Wicker, *Error Control Systems for Digital Communication and Storage*. Upper Saddle River, NJ: Prentice Hall Inc., 1995.
- [25] M. C. Reed and S. S. Pietrobon, "Turbo-code termination schemes and a novel alternative for short frames," in *Proc., IEEE PIMRC*, pp. 3554-3558, 1996.
- [26] Yufei Wu, "Implementation of Serial and Parallel Concatenated Convolutional Codes," PHD Dissertation, Virginia Polytechnic and State University, May 2000.
- [27] Kevin M. Daugherty, *Analog-to-Digital Conversion: A Practical Approach*. New York, NY: McGraw-Hill Inc., 1995.
- [28] L. W. Couch, *Digital and Analog Communication Systems*. Upper Saddle River, NJ: Prentice Hall Inc., fifth ed., 1997.
- [29] The Wireless LAN Association, <http://www.wlana.com/intro/standard/index.html>.
- [30] G. Ungerboeck, "Trellis-Coded Modulation with Redundant Signal Sets, Part I: Introduction," *IEEE Communications Magazine*, vol. 25, pp.5-11, Feb. 1987.
- [31] C. W. Bostian and E.P. Manning, "LMDS Propagation Workshop," Virginia Polytechnic Institute and State University.

Vita

Abayomi Jemibewon was born in Ibadan, Nigeria on June 3rd 1978. He received his Bachelor of Science degree in Electrical Engineering from Virginia Polytechnic and State University in May 1998. During the summer of May 1998, he worked at National Instruments, Austin Texas. While at National Instruments, he worked as a quality control design engineer, modifying product circuitry to ensure successful completion of product quality certification tests. He started in the M.S. program at Virginia Tech in the fall of 98, and joined the MPRG in the fall of 99. His research interests include wireless communication systems and channel coding. In August 2000, he will join Engineer Rotational Program at the Motorola plant in Boynton Beach Florida.