UNIVERSITÉ DE LIMOGES

École Doctorale Science – Technologie – Santé

Faculté des Sciences et Techniques

Laboratoire XLIM Département – C^2S^2 UMR CNRS 6172

Thèse No [36-2008]

Thèse

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE LIMOGES

Discipline / Spécialité : Télécommunications hautes fréquences et optiques

présentée et soutenue par

Mohammad Reza ZAHABI

Le 29 Septembre 2008

Analog Approaches in Digital Receivers

Thèse dirigée par Mr. Vahid MEGHDADI et Mr. Jean-Pierre CANCES

Jury :

Président :	
Mr. Bruno BARELAUD	Professeur – Université de Limoges
Rapporteur	
Mr. Emmanuel BOUTILLON	Professeur – Université de Bretagne Sud
Co-rapporteurs :	
Mme. Catherinne DOUILLARD	Professeur – ENST Bretagne
Mr. Cyril LAHUEC	MCF – Télécom Bretagne
Examinateurs :	
Mr. Benoît GELLER	MCF HDR – ENSTA
Mr. Vahid MEGHDADI	MCF HDR – Université de Limoges
Mr. Jean-Pierre CANCES	Professeur – Université de Limoges
Invité :	
Mr. Jean-Michel DUMAS	Professeur – Université de Limoges

to my parents,

to my family :

Maryam and Amir Reza

Table of Contents

Acknowledgments

I hereby would like to acknowledge and extend my heartfelt gratitude to Dr. Vahid Meghdadi for serving as a co-advisor and for tremendous technical guidance, continuing support and insights.

I am deeply thankful to my professor and supervisor Jean-Pierre Cances for his helpful instructions, support and encouragement which has made the completion of this thesis possible.

I also would like to express my sincere thank to professor Jean-Michel Dumas for his support and providing an congenial ambience to work.

My special thanks to Professor Bruno Barelaud and Dr. Sébastien Darfeuille who shared with me their substantial knowledge on CAD and educated me to employ Cadence in my thesis.

I had the opportunity to work closely with my friends and colleagues in ENSIL, Amir Saemi and Hamid Meghdadi for the enlightening discussions that promoted the technical insight for all of us. I am also indebted to Hamid for his great help and contribution on CAD and simulations.

I would like to thank Dr. Cyril Lahuec, Pr. Catherine Douillard and Pr. Emmanuel Boutillon for serving on my defense committee and for their valuable comments.

Many thank goes to all of my friends in ENSIL for the exiting time we spent together.

I would also thank to university of Mazandaran who sponsored my stay in France for almost four years.

Last but not least, a special mention for Dr. Amir Hossien Rezaie who helped me all the time and made all this possible. I owe more than I can say!

This dissertation was provided, thanks to the open source softwares, Inkscape for most of the drawings and TeXnicCenter environment for $\LaTeX\,2_{\varepsilon}$.

Abbreviations and Acronyms

ADC	Analog to Digital Conversion (or Converter)
APP	A Posteriori Probabilities
AWGN	Additive White Gaussian Noise
BCJR	Bahl, Cocke, Jelinek and Raviv
BER	Bit Error Rate
BP	Beliefs Propagation (probabilities propagation)
BPSK	Binary Phase Shift Keying
CC	Convolutional Code
CMOS	Complementary Metal Oxide Semiconductor
DAC	Digital to Analog Conversion (or Converter)
DSP	Digital Signal Processing (or Processor)
FER	Frame Error Rate
FG	Factor Graph
FPGA	Field Programmable Gate-Array
FPOA	Field Programmable Object Array
IC	Integrated Circuit
IF	Intermediary Frequency
iid	Independent and identically distributed
LDPC	Low Density Parity Check
NG	Normal Graph
PSD	Power Spectral Density
QPSK	Quadrature Phase Shift Keying
RSC	Recursive Systematic Convolutional
SP	Sum-Product (as in the SP algorithm)
TVC	Time Varying Convolutional

Abstract

Modern digital receivers need computationally demanding processes that leads to prohibitive complexity and power consumption. The idea of lending analog blocks for realization of digital algorithms can sometimes relaxes the complexity and high power consumption in such systems. Analog approach in digital receivers is attractive since it offers parallel processing with very smaller transistor count and chip area comparing to their digital implementations. The issue of analog approaches in digital receivers is studied in this dissertation by concentrating on two areas; *analog decoding* and *front-end processing*.

Analog decoding emerged in 1998, concerns exploiting analog circuitry to realize channel decoders in digital receivers. For analog decoding, the realizations of some efficient decoders are presented along which our contribution in this area in conjunction with graph theory is proposed. Bit-error-rate estimations of the analog decoders have been performed by running time-consuming simulations on Cadence in order to achieve almost realistic results and to justify our designs.

In addition analog realization of Viterbi algorithm is considered. It is shown that there is a very nice solution for realization of Add-Compare-Select (ACS) that plays the central rule in Viterbi algorithm. This study leads to a simple ACS realization suitable for parallel and fast Viterbi decoder in nanosecond range. To prove the design, circuit level simulations are provided while the bit-error-rate estimation is restricted to behavioural simulation.

For front-end processing, a CMOS mixed-signal programmable filter is designed and investigated. The filter is suitable for high-rate communication systems. The proposed filter has analog input and analog sampled outputs. A good advantage of this design is the storage of filter taps in a digital memory. So like a digital filter they can be changed at any time. The filter is based on simple CMOS inverter and thus can be integrated efficiently with the digital parts of a system. Finally, a FIR cosine rolloff filter is designed and studied by simulation in time and frequency domains.

Chapter 1

Introduction

In the past decade digital communications dominates rapidly the way of accessing to information and transmitting data. Communication appliances are continuously developing and becoming more and more sophisticated with lower prices. Wireless digital communications is undoubtedly a main part of these developments. Power consumption is now a challenging issue in design of high-tech devices. The work of Claude Shannon in 1948 is the starting point of these evolutions by founding the mathematical theory of digital communications and changing the sight of scientists to this issue. Shannon band is still a benchmark for newly designed communication systems. His work faced the community with a new problem, "How to protect effectively information bits through transmissions?" This question developed the bases of channel coding. Despite several remarkable works after Shannon, limitations of technology halted the practical developments in this area. For example the work of Galager on LDPC codes remained at the theory level and were mostly ignored due to demanding computational effort.

In twenty century we are facing again the evolution with much more clear results influencing out lifestyle. For the first time Claude Berrou and Alain Glavieux proposed the so-called turbo-codes with the capacity approaching feature in 1993. At the same time the community paid attention to the Galager's work on LDPC codes presented since the early 1960s. This time the current platforms allowed handling heavy processes that yielded a comparative result with respect to turbo codes.

Development of graph theory and using it in coding was also important. This method allowed scientists a better look into coding theory by using graphical diagrams and revealed the close relationship between turbo codes and LDPC codes. Later some authors employed graph directly to the design of good codes. Graph theory introduced the important concept of iterative processing. Iterative algorithms now covers the synchronization and equalization processes and sometimes is applied at the same time with decoding process. The relationships between these three issues are often very complex and finding an iterative solution is not always possible. Besides, the Expectation-Maximization algorithm, known since many years with its strength mathematical bases, was proposed and successfully applied to beak such complex problems into several smaller modules and obtaining more simple iterative algorithms. In this dissertation we concentrate on decoding and implementation of some efficient decoders, notably analog decoders. Analog decoding is a new subject in the field of decoder design. We present our contribution in this subject. In addition we proposed an efficient structure for front-end part of a receiver, i.e. matched filtering by employing analog devices. we show that our design yield a good trade-off between power consumption and speed. Analog schemes are sometimes attractive since they offer parallel processing with small transistor count and chip area compared to the conventional digital implementations. There are very promising results in this area which have been reported, yet there are some limitation with respect to digital implementations that need more effort.

1.1 System Model

Assume that binary information with equal zero-one probability is passing via a noisy channel. In the case of non-equal probability the source coding can fulfill this requirement. Look at figure 1.1 for a general structure of digital communication systems. Digital source may be an intrinsic digital source such a storage data or it may be an analog source that is converted to digital stream. In forward-error correction (FEC) scheme some form of redundancy is added to the source in order to cope with unreliable communication media. This process is performed by channel encoder and decoder in transmitter and receiver respectively. The abstract information turns into physical signal in the modulation block and it is strongly dependent to the channel type. For some sort of media a baseband modulation is sufficient wherein the spectrum fit channel requirement. On the other hand for long-haul transmissions, bandpass modulation is required where baseband signal impressed upon a carrier wave.

1.2 Error Control Coding

Bit errors in digital transmission depend on the quality of channel often denoted by signal-to-noise ration (SNR). Forward error correcting (FEC) as an important aspect of error control coding, is a promising solution for reliable communication over unreliable channels. The principle of FEC is to add redundancy to information by inclusion of extra digit to source. According to information theory a nearly errorless transmission is possible by using FEC systems. The drawbacks, however is higher transmission delay, systems complexity and power consumption. A multitude of FEC systems have therefore been devised to suit various applications. Two major categories of FEC systems are block codes and convolutional codes. The former performs over a fixed number of information bits (block) while the later performs over the semi-infinite stream of data bits such as media broadcasting. In

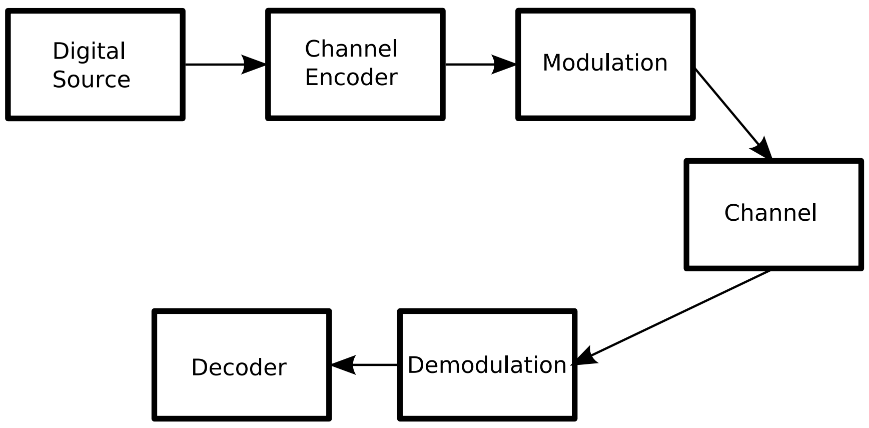


Figure 1.1: Outline of digital transmission system

fact, we can always partition data streams into blocks and apply block coding to them. So block codes sound to be more applicable than convolutional codes.

1.3 Modulation

Modulator associates analog signal (from a set of possible signals / constellation) to one or several adjacent bits at its input. Figure 1.2 illustrate some important examples of baseband and bandpass modulation. A very simple example of baseband modulation is pulse amplitude modulation (PAM) with rectangular pulses (figure 1.2(a)). Such a simple scheme does not often yield a good error performance owing to signal distortion in most practical cases. The presence of inter-symbol interference (ISI) and noise along with limited bandwidth demands more sophisticated signaling to mitigate bit errors in the regenerated signal at receiver side. This problem was known since 1924 by the work of Nyquist on telegraph signal. He proposed the so-called cosine rolloff pulses (figure 1.2(b)) to reduce the effect of ISI and to obtain additional good properties such as simple synchronization and being bandlimited to avoid spectral spillover for a given bandwidth (and hence to increase spectral efficiency). Sometimes this topic may be covered in communication texts under the name of "pulse shaping".

Long-haul digital transmission, on the other hand, usually requires continuous wave modulation to generate a bandpass signal suited to the transmission medium (figure 1.2(d-f)). There are many modulation methods with their own pros and cons. Two important parameters in this area is the spectral efficiency and power efficiency. Spectral efficiency is measured in `bit/sec/Hz` and expresses the amount of information that can be transmitted over a given bandwidth in a specific digital communication channel. According to Nyquist sampling theorem, the spectral efficiency is upper-bounded to $2\log_2(M)$ `bit/sec/Hz` for a M-ary signaling without ISI. For example QPSK modulation has two times more efficient as BPSK modulation. Except the two last methods, which are using quadrature constellation, the other ones are using simple constellation. By using quadrature constellation we obtain a bit rate of two times as much as simple constellation given a fixed bandwidth. Power efficiency of a spectrum, on the other hand, represents the percentage of power spill over to the side lobs. Modulations that employ smooth transition e.g. raised cosine have better power efficiency than those with fast transition such as NRZ pulse. In addition the complexity of transmitter and receiver is an important issue and is considered along with aforementioned aspects. We may have a much more complicated constellation in exchange of pert in overall performance owing to closeness of constellation points. Bear in mind that the general rule for the error performance of a digital communication system depends directly on Euclidean distance of the constellation points and inversely on noise power density, i.e.

$$P_{error} = Q\left(\frac{distance}{\sqrt{2N_0}}\right) \tag{1.1}$$

1.4 Trellis Coded Modulation

Sometimes we can not partition a communication system sharply, as it is for Trellis Coded Modulation (TCM). TCM is a modulation scheme which allows highly efficient transmission of information over band-limited channels such as telephone lines. TCM was invented by Gottfried Ungerboeck in 1982 [1] and played a key role in the Information Age by increasing the speed by a factor of two with the same error rate. Conventionally the coding is a digital function and modulation which is an analog function, are done separately and independently. In TCM, however the two functions are merged into a single function.

1.5 Communication Channel Models

Continuous-time AWGN channel is a random channel whose output is a real random process $y(t) = x(t) + n(t)$, where $x(t)$ is the input waveform, regarded as a real random process, and $n(t)$ is a real white Gaussian noise process with single-sided noise power density N_0 which is independent of $x(t)$. Moreover, the input $x(t)$ is assumed to be both power-limited and band-limited. The average input power of the input waveform $x(t)$ is limited to some constant value P. Despite of various modulation schemes, it is always possible to reduce a continuous-time AWGN channel model to an equivalent discrete-time AWGN channel model $y = x + n$. The SNR of this channel model is then:

$$\text{SNR} = \frac{P}{N_0 W}, \tag{1.2}$$

where $N_0 W$ is the total noise power in the band W. The two parameters W and SNR turn out to characterize the channel completely for digital communications

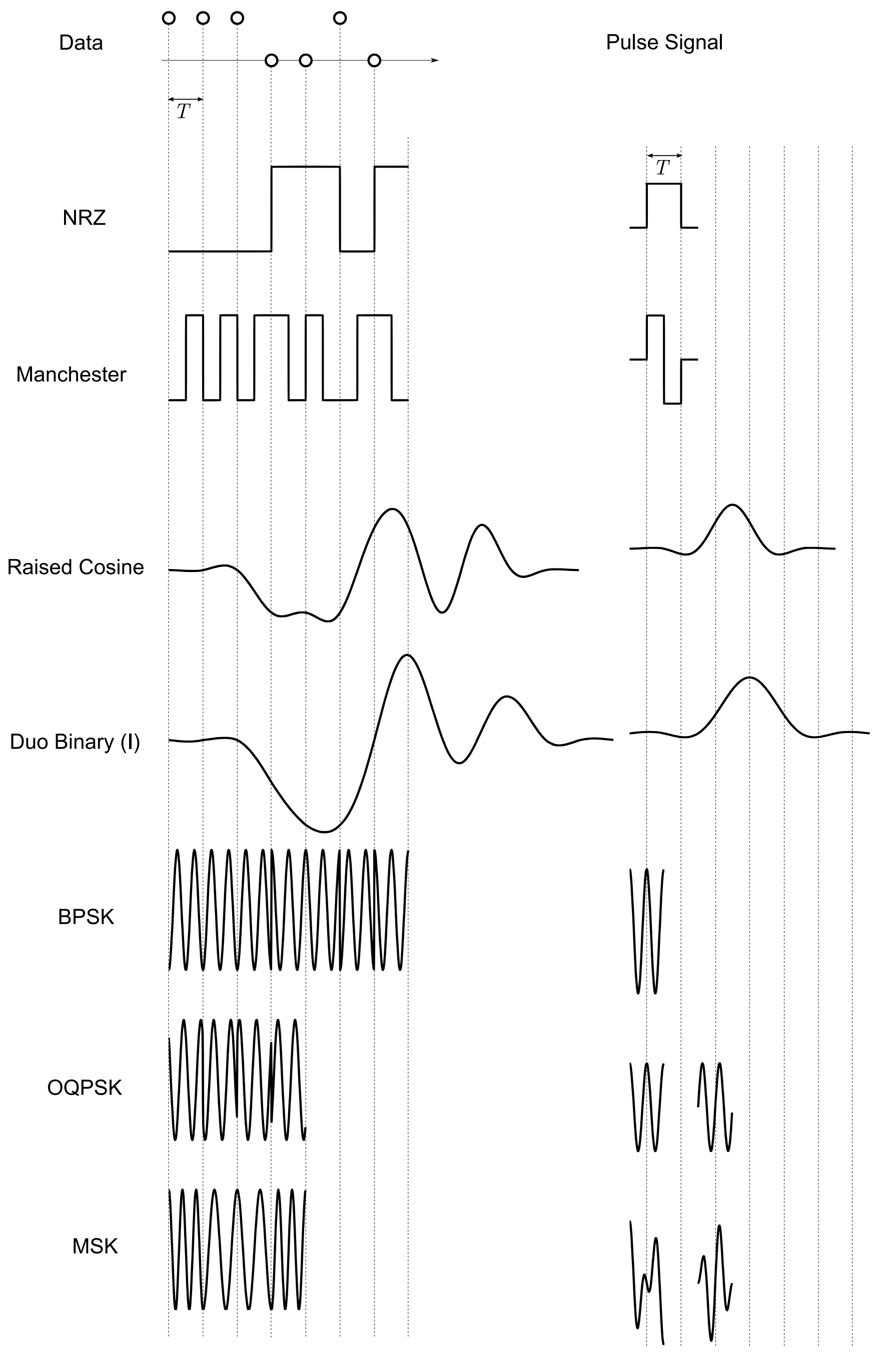


Figure 1.2: Several modulation techniques for baseband and bandpass digital transmission

purposes. If a particular digital communication system transmits a continuous bit stream over such a channel at the rate of R (`b/s`), then the spectral efficiency of the system is said to be $\rho = R/W$ (`b/s/Hz`). The Shannon limit on spectral efficiency is:

$$C = \frac{1}{2}\log_2(1+\mathrm{SNR}) \qquad \texttt{b/s/Hz} \tag{1.3}$$

Reliable transmission is possible when $\rho < C$, but not when $\rho > C$.

1.6 Demodulation and Optimum Receiver

This section outlines the concept of optimum detection in M-ary receivers. We will use the convenience of geometric representation of signal to facilitate the necessary calculations[2]. The transmitter sends a signal $s_i(t)$ from the set of M signals. The signal at the destination appears as $s(t)$ owing to corruption by AWGN, $n_w(t)$. We assume a one-sided spectrum of N_0 for the noise. The receiver task is to compare $s(t)$ by each of the M available signals in the signal set at the receiver (coherent system). A reasonable rule to define the optimality of receivers is to minimize the error probability that is, the i^{th} symbol is recognized by the receiver if it has the largest *a posteriori* probability.

$$P(s_i \mid s) > P(s_j \mid s) \Longrightarrow \hat{s} = s_i \qquad \forall \quad i \neq j \tag{1.4}$$

This rule is known as *maximum a posteriori* (MAP) rule. Suppose that the orthogonal basis $\phi_k, \{k \in 1,\ldots,K \mid K \leq M\}$, span the signal set in a subspace S_K. The received signal $s(t) = s_i(t) + n_w(t)$ dose not fall entirely within S_K. As illustrated in 1.3, decomposition of $n_w(t)$ into relevant noise $n(t)$ and irrelevant noise $n_\perp$ is needed for spanning the received signal in S_K. Also the relevant signal $x(t)$ is the projection of the received signal in S_K as depicted in figure 1.3. The *maximum Likelihood*(ML) detection is achieved by choosing the closest signal in the signal set to $x(t)$. So the receiver task is the calculation of d_i, the distance between $x(t)$ and $s_i(t)$ $\forall i \in \{1\ldots K\}$.

$$d_i = \|x - s_i\|^2 = \|x\|^2 - 2\langle x, s_i\rangle + \|s_i\|^2$$

The constant value of $\|x\|^2$ has no effect in comparison. The $\|s_i\|^2 = E_i$ is the energy of i^{th} symbol and in general is not constant in some sort of modulations. In addition, according to orthogonality of $n_\perp$ to S_K the mid term $\langle x, s_i\rangle$ is equal to $\langle s, s_i\rangle$. Two common ways to calculate $\langle s, s_i\rangle$ are by correlation and matched filtering as it is depicted in figure 1.4.

1.7 Error Probability

Average error probability of a M-ary systems can be formulated according to the preceding sections. Define P_j as the probability of erroneous detection of j^{th} sym-

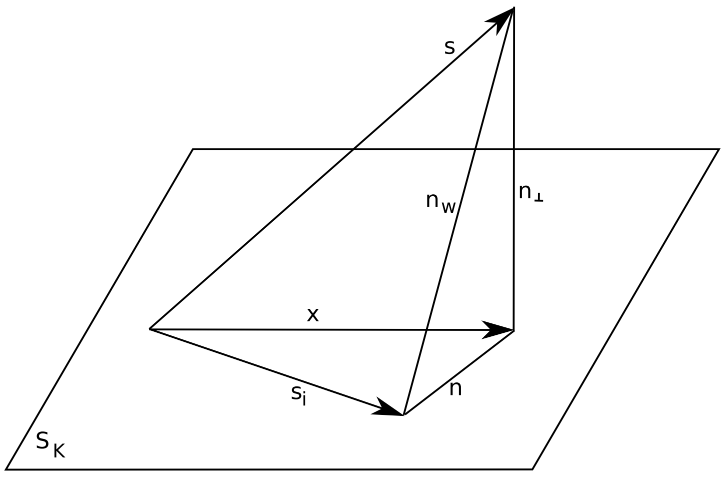


Figure 1.3: Geometric representation of received signal

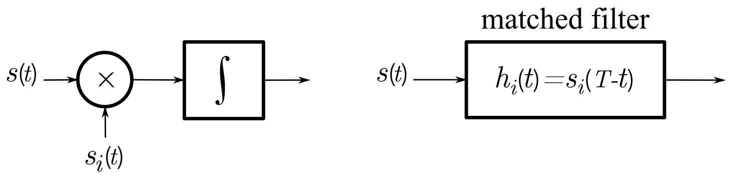


Figure 1.4: Correlation receiver(left) and matched filtered receiver (right)

bol that correspond to $s_j(t)$. The average error probability is then:

$$P_e = \frac{1}{M}\sum_{j=1}^{M} P_j \tag{1.5}$$

providing that the symbols are equiprobable. Now define P_{ij} the probability of s_j to be recognized as s_i. In fact if the noise component in $(s_i - s_j)$ direction exceeds the halfway between s_i and s_j an error will take place. Figure 1.5 shows the geometrical illustration of this phenomenon. Given a Gaussian distribution for the noise, the error probability can be written as follows:

$$P_{ij} = \int_{d_{ij}/2}^{\infty} p_n(n_k)dn_k = Q\left(\frac{d_{ij}}{\sqrt{2N_0}}\right), \tag{1.6}$$

where $d_{ij} = \| s_i - s_j \|$. Summing up P_{ij} over i and using equation 1.5 yields the upper bound for a general M-ary system.

$$P_e \leq \sum_{i \neq j} P_{ij} \tag{1.7}$$

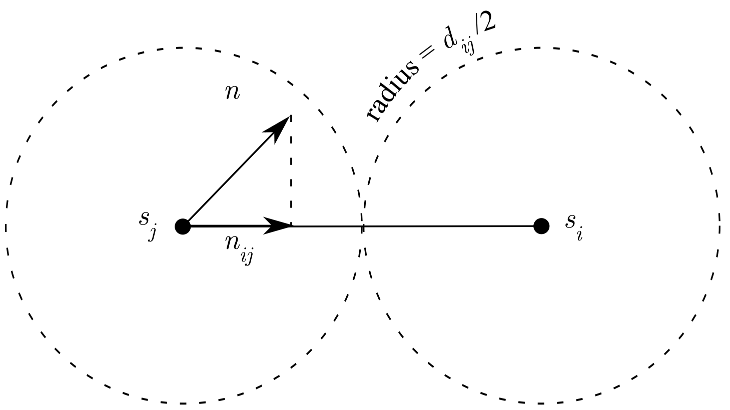


Figure 1.5: Geometrical illustration of signal to noise ratio

1.8 Front-end Processing in Digital Receivers

At least tree schemes are envisagable for digital receivers when they are accompanying with decoders and/or other processing blocks in digital domain. Figure 1.6 shows the outline for the receiver structures. The first structure is the conventional one for baseband receivers. The analog filter at the input performs matched filtering while its output is sampled and converted via ADC to digital stream. The post processing is then accomplished by DSP processors or newly FPGA or FPOA ICs to perform decoding and other necessary algorithms for say synchronization and equalization tasks.

In order to achieve more flexibility the scheme of *software defined radio receiver* was postulated that consists merely of an ADC converter as close as possible to the receiver front-end as depicted in figure 1.6(b). This block diagram is remains valid for both of baseband and bandpass signals. The principle of *direct down conversion* (DDC) discusses the theory of bandpass reception. Unlike the conventional bandpass receivers with their intermediate-frequency (IF) section, here sampling at low frequency imply intentional aliasing that down converts the bandpass signal into baseband samples. However DDC required a sophisticated ADC with large bandwidth and input dynamic range. The filtering is done over digital samples in digital domain. Since all processing are in digital domain, this structure provide a highest level of reconfigurability and receiver is able to capture signals of multitude of standards at different times. At the same time the power consumption and complexity may be prohibitive that put a limit to its applications.

The block diagram in figure 1.6(c) is a novel method based on analog decoders. In this power efficient scheme, ADC is no longer needed and the signal path remains in analog domain. The invention of analog decoder ushered in the development of very simple receiver structure with potentially very efficient speed-power-area trade-off. It is also possible to make more sophisticated structure, say turbo decoding, by using two analog decoders working in parallel. Note

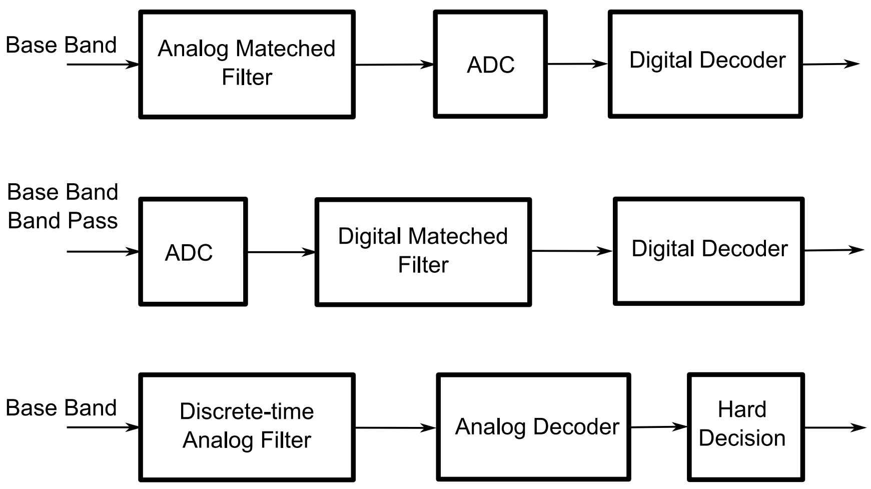


Figure 1.6: different schemes for a digital receiver

that there is only need for simple comparator at the decoder output to retrieve information bits. Promising successes in design of analog decoders have been reported so far[3, 4, 5, 6, 7, 8]

In order to gain the full advantage of analog decoders and to avoid DAC between the matched filter and the decoder blocks, we need an analog filter. But a traditional analog filter is not a good choice because it can not be integrated with the decoder and it is not enough flexible. Our contribution in this area was to devised a novel discrete-time analog filter featuring fully-CMOS and simple structure that can be integrated with analog decoder. Another important characteristic of this design is the ability to be reconfigured, like a digital filter, that embodying the concept of *Analogue Software Defined Radio Receiver*.

1.9 Dissertation Outline and Contributions

Chapter 2 commences by definition of convolutional codes and two general structures that realize encoders for a given code. Conversion between the two structures is presented by exploiting *cut-set retiming rule*. Using this method, we get a more clear view of encoder's *state* and besides one can use it to devise different structures for encoding of a code. Representing a code by its trellis is considered in this chapter as the classical and optimal decoding algorithm. Undoubtedly, the concept of trellis of a code extends to block codes that is mentioned briefly in this chapter. In addition some of the related issues such as tailbiting termination and time-varying convolutional codes for which we have had encountered during this thesis are explained.

Chapter 3 concerns graph theory and its application in definition of codes and decoding. The background about the encoders provided in chapter 2 and belief propagation algorithm illustrated in this chapter leads to novel approach. Our contributions in this case are presented in section 3.7. Briefly speaking, this section is targeted at decoding of tailbiting convolutional codes on graph. To this aim a unified method is devised to get simple graphs form tailbiting convolutional codes. This is done by reformulation of tailbiting constraint. An important aspect of this approach is the definition of a new vector. The study of algebraic properties of this vector reveals two important properties; first, the conditions for which tailbiting constraint remains valid and second, uncover the relationship between recursive systematic convolutional (RSC) codes and non-RSC codes. Last result is interesting in the sense that a graph dedicated for decoding of a RSC code can be employed to decode the related non-RSC code as well. In addition we notice the concept of *fatal loop*, a condition that should be avoided in order to belief propagation not to fail.

Chapter 4 provides an introduction on the history of analog decoding, our motivation for choosing it as platform of decoder realizations. The building blocks made of analog transistors in sub-threshold region are then presented in different level of schematic hierarchies. The designs in this chapter are based on the earlier works proposed in this context by different authors.

In chapter 5 we justify the idea of analog decoding by several decoding circuits at simulation level. Extended Hamming code as our first experience, (21,7) quasi-cyclic code and (16,8) tail-biting convolutional code are considered that demonstrate the feasibility of analog decoding. Among them, (16,8) tail-biting convolutional code is presented for the first time and is an example of a unified approach brought in section 3.7 for general tail-biting convolutional codes. The approach features a simple binary graph suitable for realization of analog decoder. For the quasi-cyclic code, we shows that how re-labeling of nodes in its graph leads to a better modularity of decoder system.

This chapter also contains several simulations to figure out some critical characteristics of the analog decoders; the effect of reference current on speed and performance of the decoder, the effect of input signal amplitude and the issue of symmetry between two inputs of our generic cell and its affect on the performance of decoders.

Chapter 6 is dedicated to design a new high-speed Viterbi decoder in nanosecond time range. The core circuitry in this design is winner-take-all (WTA), well-known in neural network context. Although the MOS transistors in the original design of WTAs work in sub-threshold region, we change the operating point well above strong inversion. In WTAs with their inherent positive feedback this is always possible and yields a far superior speed to the original design.

Beside many advantages of such analog scheme, there are also difficulties. The first problem with WTA arises from the fact that classical Viterbi algorithm is based on "Min" function while WTAs provides "Max" function. The second problem is lack of "argMax" functionality needed for trace-back operation in Viterbi algorithm.

To overcome the first problem, we introduce a mechanism of normalization by an extra WTA. In addition, regarding the "limited dynamic range of path metric" give rises to obtain even more simplicity. That is using only one WTA-N for an arbitrary number of states. This is very important for codes with large number of states, say 64 or more.

Second to add the functionality "argMax" we modify the WTA circuit in our design so that a logical output is added. As the simulations demonstrate, this output exhibits a steep variation and fast time response.

Finally the blocks are put together so that we are able to use it as Viterbi kernel in a decoder while gaining advantage of its simple, fast and CMOS-based structure.

Chapter 7 represents our contribution in the field of filtering. Inspired by a transconductor design in the context of continuous-time filter, we propose simple structure for FIR mixed-signal discrete-time filter. CMOS inverter gates features as a fixed-gain transconductor. Our motivation to overcome fixed-gain limitation leads to a mixed-signal MAC (multiply-accumulate) as a core block in FIR filters. The proposed structure has many advantages; remaining signal path analog and quantizing the filter taps instead, featuring wideband frequency range, being applicable efficiently with analog decoders, to name only some of them. A roll-off cosine filter is designed based on the proposed scheme and is verified by simulation in circuit level. The chapter is concluded by a comparison of designed filter with some of the other FIR filters.

Finally the last chapter concludes the dissertation and expands on the sequel to this work.

The contributions provided in this thesis can be equally found in the publications by the author [9, 10, 11, 12, 13, 14].

Chapter 2

Trellis and MAP Decoding

This chapter concerns the important issue of trellis structure of a code. The reason for its importance comes through the fact that for optimal decoding of a code we need its trellis definition. Traditionally, trellis had been employing for definition and decoding of convolutional codes while algebraic block codes were decoded by syndrome former. Later a sort of trellis were devised for block code where the number of states are not necessarily constant in different trellis section, i.e. time-varying trellis. According to this introduction we will commence with definition of convolutional codes and their trellis representation. Then the trellis diagram of block code will be illustrated briefly. Two important decoding algorithms over trellis of a code will be derived that include the forward-backward algorithm and forward-only algorithm. The subtle connection of these algorithms to the well-established Hidden Markov chain will then be explained.

2.1 Convolutional Codes in GF(2)

Defining the k-tuple binary input information at time i as:

$$x_t = \{x_t^{(1)}, \cdots, x_t^{(k)}\},$$

the whole input sequence **x** is represented by the sequence:

$$X = \cdots, x_{-1}, x_0, x_1, x_2, \cdots$$

The convolutional coded output is composed of n-tuple stream as:

$$Y = \cdots, y_{-1}, y_0, y_1, y_2, \cdots$$

where y_i is defined by n-tuple:

$$y_t = \{y_t^{(1)}, \cdots, y_t^{(n)}\}.$$

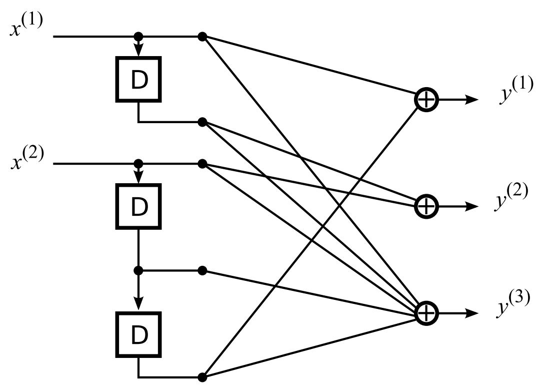


Figure 2.1: An example of rate $2/3$ convolutional encoder

Such a definition represents a rate k/n convolutional code assuming the following relationship is held:

$$Y(\mathcal{D}) = X(\mathcal{D})\mathbf{G}(\mathcal{D}) \tag{2.1}$$

where $\mathcal{D}$ is delay operator that convert a numerical vector to a polynomial. For example $X(\mathcal{D})$ is:

$$X(\mathcal{D}) = \cdots + x_{-1}\mathcal{D}^{-1} + x_0 + x_1\mathcal{D} + x_2\mathcal{D}^2 + \cdots$$

The $k \times n$ polynomial generator matrix[1], $\mathbf{G}$ is in general a rational function of $\mathcal{D}$. One important aspect of convolutional codes is the simple structure of the encoder that is realizable by only delay and summation (xor). For example figure 2.1 shows a rate $2/3$ convolutional encoder with polynomial generator matrix:

$$\mathbf{G}(\mathcal{D}) = \begin{pmatrix} 1 & \mathcal{D} & 1+\mathcal{D} \\ \mathcal{D}^2 & 1 & 1+\mathcal{D}+\mathcal{D}^2 \end{pmatrix}$$

2.2 Trellis of a Codes and Tailbiting Trellis

A code trellis is a graphical representation of a code, heavily used in coding theory originally for convolutional codes and later for block codes. Trellis is comprised of nodes and branches in which every path, a concatenation of nodes and branches, represents a codeword (or a code sequence for a convolutional code). This representation makes it possible to implement dynamic programing for decoding of a code with reduced complexity.

At each time index t, the possible states of the trellis are indicated in a vertical array of nodes, each node corresponds to one state value. It often represents the

[1]This is a transfer function and should be distinguished from generator matrix which is a large numerical matrix

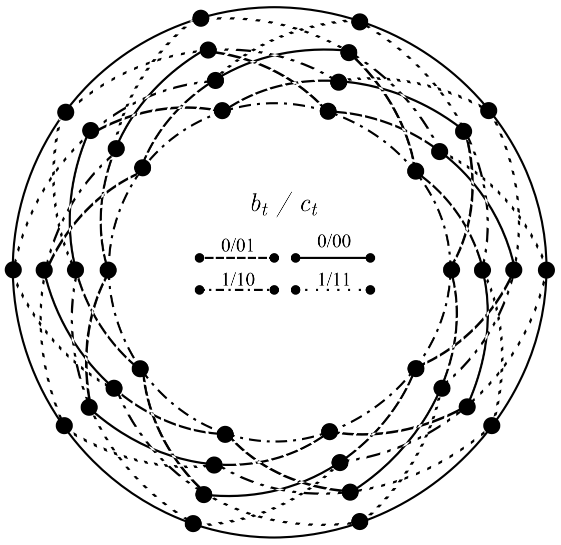


Figure 2.2: Tailbiting trellis of (7,5) RSC code

(binary) content of delays in corresponding encoder. A trellis branch connects two states and corresponds to a possible input. Each branch in trellis is labeled with the encoder output, c_t (or channel symbol) and the corresponding input b_t and may be noticed like b_t/c_t.

Time-invariant encoder description requires only a section of trellis say for times t and $t+1$, because the set of states and possible transitions does not change from time to time. Trellis of block codes are often time-varying and should be given completely to describe the corresponding code.

Another important issue in encoding (and consequently in decoding) is the initial and final states of encoder. It may be designed an encoder starts from zero and ends also to zero state. Another possibility is unknown state with equi- probable distribution among the possible states. Tailbiting is an alternative that imposes start state and end state to be equal. Tailbiting trellis thus have a circular shape, such a ones is shown in figure 2.2.

2.3 Encoder Realization

In this section we will consider two important realizations of encoder for convolutional codes among almost infinite possible realizations. Furthermore we will consider only convolutional codes with the polynomial generator matrix reduced to one row. This simplifies the notation without loss of generality because a linear combination of two or more such codes yields the complete code. Moreover, the classical convolutional codes are often described by single row generator polynomial matrices with rate $1/2, 1/3$ etc. The different encoder realizations for a code are input-output equivalent but the internal states of the encoders are can be differ-

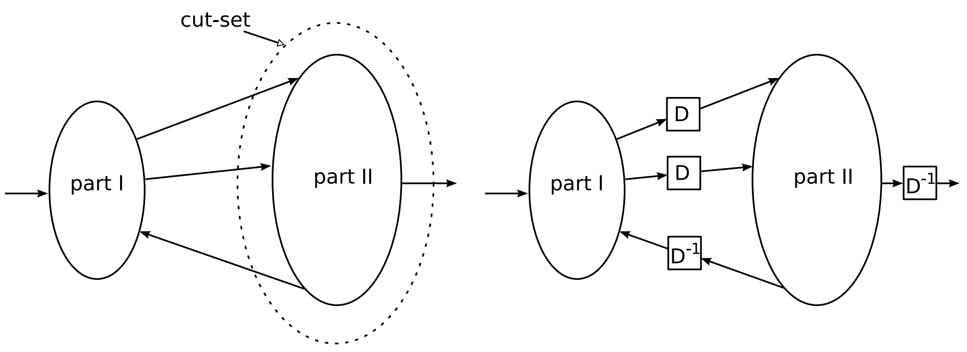


Figure 2.3: cut-set retiming for system conversion

ent. This fact can affect the trellis diagram of the code and the decoding process (c.f. section 3.7). Here we are only interested in minimal encoders. Such encoders have a minimum number of delay elements over all other encoders for the code. In addition we lend the concept of *cut-set retiming* to convert one structure to another.

cut-set retiming rule

The cut-set rule can be briefly stated as follows: if a part of a diagram can be separated from the other parts, one can apply the delay element $\mathcal{D}$ to inbound data lines and apply the advance elements $\mathcal{D}^{-1}$ to outbound data lines (or vise versa) as illustrated in figure 2.3. The new diagram have exactly the same characteristic as the first one and can be realize as far as any $\mathcal{D}^{-1}$ is absorbed by an already existing delay element, owing to the causality issue. As an illustrative example, consider the generator polynomial of $1/(1+\mathcal{D}+\mathcal{D}^3)$. The numerator is simply chosen to be 1 because the principle part of the state-space representation depends only on denominator.

The encoder shown in figure 2.4(a) is controller canonical form. We use cut-set rule to obtain finally the structure of 2.4(d). The procedure is illustrated in details in figure 2.4 wherein by using cut-set rule there is no need for any mathematical calculation.

Another advantage of cut-set method is that the relationship between state vectors in the two structures is obvious. For example suppose two encoders with direct form I and transpose direct form[2] II structures. If $\mathbf{x}$ and $\mathbf{y}$ are state vectors of these encoder respectively, then the relationship between $\mathbf{x}$ and $\mathbf{y}$ can be found easily as depicted in figure 2.5. This figure shows a general encoder with generator polynomial of:

$$\frac{1}{1+a_1\mathcal{D}+\cdots+a_m\mathcal{D}^m}$$

where the content of delay element represents the state for each structure. The relationship between state vectors $\mathbf{x}$ and $\mathbf{y}$ can be written directly by inspection as:

[2]Also referred as $\mathrm{II_t}$ in signal processing texts

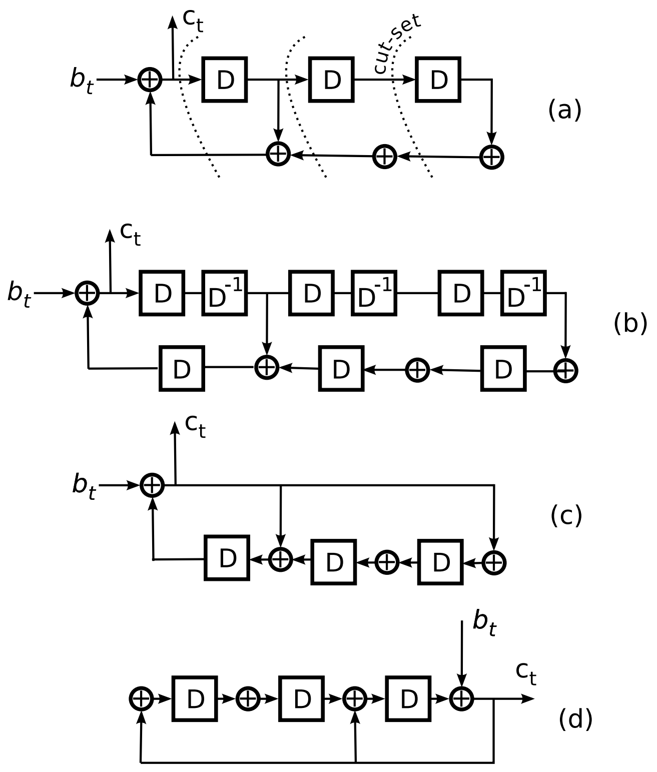


Figure 2.4: Evolution of encoder structure; from controller canonical form (Direct form I) to observer canonical form (Transpose direct form II) using the concept of cut-set retiming.

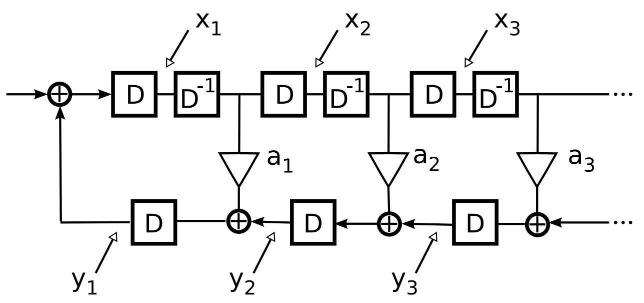


Figure 2.5: Relationship between two state vectors **x** and **y**

$$\mathbf{y} = \begin{pmatrix} a_1 & a_2\mathcal{D} & \cdots & a_m\mathcal{D}^{m-1} \\ 0 & a_2 & \cdots & a_m\mathcal{D}^{m-2} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_m \end{pmatrix} \mathbf{x} \tag{2.2}$$

State space equations, on the other hand, for direct form I and direct form $\mathrm{II_t}$ are respectively as follows. The relationship between two state space matrices is clearly the transposition, so accounts for the name *transpose* direct form II.

$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix}_{(t+1)} = \begin{pmatrix} a_1 & \cdots & a_{m-1} & a_m \\ 1 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix}_{(t)} + \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} u(t) \tag{2.3}$$

$$\begin{pmatrix} y_1 \\ \vdots \\ y_{m-1} \\ y_m \end{pmatrix}_{(t+1)} = \begin{pmatrix} a_1 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{m-1} & 0 & \cdots & 1 \\ a_m & 0 & \cdots & 0 \end{pmatrix} \begin{pmatrix} y_1 \\ \vdots \\ y_{m-1} \\ y_m \end{pmatrix}_{(t)} + \begin{pmatrix} a_1 \\ \vdots \\ a_{m-1} \\ a_m \end{pmatrix} u(t) \tag{2.4}$$

Choosing one of these structure, depends on the applications. For example in section 3.7 we need an structure with maximum redundancy in state vector. Comparing two above mentioned encoder, in direct form I the delay blocks acts as a shift register while in direct form $\mathrm{II_t}$, input adds to the content of delay. Thus as a result, direct form I is chosen to achieve a decoder reduced complexity.

2.4 Bit-wise Maximum A Posteriori Decoding

Nowadays Bit-wise MAP decoding dominates other algorithms due to its critical rule in turbo code. In decoding of turbo codes, there is need for calculation of bit's a posteriori and improving its certainty in a recursive manner.This is also critical to modern iteratively-decoded error-correcting codes including low-density parity-check codes to find the maximum a posteriori of a bit given a coded stream.

In this section we will briefly review two important approaches on MAP decoding algorithms. Considering MAP decoding at least by two points of view gives a better insight into the problem and provides motivation for further considerations. We assume that the trellis of the code is already known. Nevertheless for a concrete explanation, a sample encoder and its corresponding trellis diagram are shown in figure 2.6.

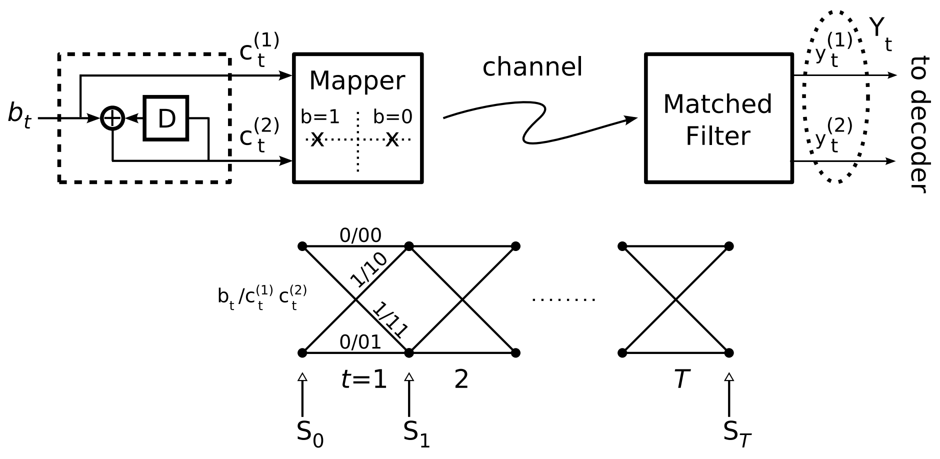


Figure 2.6: Definition of variables in a sample convolutional encoder and corresponding trellis diagram

Assuming information vector of length T is coded by a rate $1/n$ convolutional encoder and the coded stream mapped into constellation and transmitted through a channel where a receiver observes the channel output $Y_1^T = Y_1 \cdots Y_T$ as a $n \times T$ matrix. The content of this matrix may be $\{0,1\}$ for BSC or real number for AWGN channel model.

The goal of a MAP decoder is to find a posteriori probabilities for each bit and eventually to calculate log-likelihood ration (LLR) by incorporating all related information received from the channel:

$$LLR(b_t) = \ln \frac{P_r(b_t = 0|Y_1^T)}{P_r(b_t = 1|Y_1^T)}, \tag{2.5}$$

where b_t is the tth bit that correspond to tth trellis section of the code in a T-length trellis.

Several algorithms have been developed to find 2.5. Here we consider BCJR algorithm [15] and Forward-only algorithm [16]. BCJR algorithm is used for maximum a posteriori decoding of error correcting codes defined on trellises (principally convolutional codes). The algorithm is named after its inventers: Bahl, Cocke, Jelinek and Raviv. BCJR algorithm is the well-known forward-backward algorithm in the context of Hidden Markov Model (HMM) while the forward-only algorithm is a result of clever deviation from BCJR algorithm that eliminate the need for backward recursion in expense of more hardware usage.

In the following we will often encounter at the expression:

$$\gamma_t(i,j) = P(S_t = j, Y_t | S_{t-1} = i) \tag{2.6}$$

which is the joint probability of transition from state i to state j and observation at the time instance of t. This probability can be further factored into two terms as follows:

$$\gamma_t(i,j) = P_r(S_t = j|S_{t-1} = i)P(Y_t|S_t = j, S_{t-1} = i) \tag{2.7}$$

The first term is characterized by the a priori probability of the information bit b_t and the structure of code, i.e. the existence of a path between states i and j. The second term is characterized by the channel which is obviously independent from the first process. It is more convenient to denote the second term as $P(Y_t|C_t)$. For example for the trellis in figure 2.6 there are four such probabilities correspond to $C_t \in \{00, 01, 10, 11\}$.

2.4.1 BCJR Algorithm

Towards finding an efficient procedure to calculate the equation 2.5, it is more convenient to replace the nominator and denominator with $P(b_t = 0, Y_1^T)$ and $P(b_t = 1, Y_1^T)$ respectively that clearly don't change the final result. Then we can partition them as follow, say for the nominator:

$$P(b_t = 0, Y_1^T) = \sum_{\substack{\{i,j\} \\ b_t=0}} P(S_{t-1} = i, S_t = j, Y_1^{t-1}, Y_t, Y_{t+1}^T). \tag{2.8}$$

The expression under summation can be factorized using Bayes' rule:

$$\begin{aligned} &P(S_{t-1} = i, S_t = j, Y_1^{t-1}, Y_t, Y_{t+1}^T) \\ &\quad = P(S_{t-1} = i, Y_1^{t-1})P(S_t = j, Y_t, Y_{t+1}^T|S_{t-1} = i, Y_1^{t-1}) \\ &\quad = P(S_{t-1} = i, Y_1^{t-1})P(S_t = j, Y_t|S_{t-1} = i, Y_1^{t-1}) \\ &\qquad\qquad P(Y_{t+1}^T|S_{t-1} = i, Y_1^{t-1}, S_t = j, Y_t) \end{aligned}$$

The first term is called forward variable and commonly denoted by $\alpha_{t-1}(i)$. By definition forward variable is:

$$\alpha_t(i) = P(S_t = i, Y_1^t) \tag{2.9}$$

In the second term, we can ignore Y_1^{t-1} because state transition at time t is clearly independent of our preceding observation. According to 2.7 this term is $\gamma_t(i,j)$. Similarly in the last term we can ignore Y_1^{t-1}, S_{t-1} and Y_t because the observation after t depends only on the state at time t. This term is called backward variable and is commonly denoted by $\beta_t(j)$:

$$\beta_t(j) = P(Y_{t+1}^T|S_t = j) \tag{2.10}$$

Therefor equation (2.8) can be expressed by:

$$P(b_t = 0, Y_1^T) = \sum_{\substack{\{i,j\} \\ b_t=0}} \alpha_{t-1}(i)\gamma_t(i,j)\beta_t(j) \tag{2.11}$$

Obviously for $b_t = 1$ we obtain a similar equation as follows:

$$P(b_t = 1, Y_1^T) = \sum_{\substack{\{i,j\} \\ b_t=1}} \alpha_{t-1}(i)\gamma_t(i,j)\beta_t(j) \tag{2.12}$$

The advantage of BCJR algorithm comes from recursive and simple computation of the forward and backward variables α and β. To obtain the recursive equations, one can write a marginalize expression as follows:

$$\alpha_t(j) = P(S_t = j, Y_1^t) = \sum_i P(S_{t-1} = i, S_t = j, Y_1^{t-1}, Y_t)$$

The expression under summation can then be factorized as:

$$\begin{aligned} \alpha_t(j) &= \sum_i P(S_{t-1} = i, Y_1^{t-1}) P(S_t = j, Y_t | S_{t-1} = i, Y_1^{t-1}) \\ &= \sum_i \alpha_{t-1}(i)\gamma_t(i,j) \end{aligned} \tag{2.13}$$

Equivalently we can extend the probability for β to obtain a recursive formula. From equation (2.10):

$$\begin{aligned} \beta_t(j) &= P(Y_{t+1}^T | S_t = j) \\ &= \sum_k P(Y_{t+1}, Y_{t+2}^T, S_{t+1} = k | S_t = j) \\ &= \sum_k P(S_{t+1} = k, Y_{t+1} | S_t = j) P(Y_{t+2}^T | S_t = j, S_{t+1} = k, Y_{t+1}) \end{aligned} \tag{2.14}$$

The first term is $\gamma_{t+1}(j,k)$ and the second, after ignoring the irrelevant terms S_t and Y_{t+1} is equal to $\beta_{t+1}(k)$. Therefor the backward recursive formula is:

$$\beta_t(j) = \sum_k \beta_{t+1}(k)\gamma_{t+1}(j,k) \tag{2.15}$$

So the BCJR algorithm uses the forward recursion of 2.13 and backward recursion of 2.15 to calculate α and β for $t \in \{1, \cdots T\}$. Then applying 2.11 and 2.12 to calculate the nominator and denominator of 2.5 is the goal of the algorithm.

One important but implicit step in this procedure is the initialization of the algorithm. In order to employ 2.13 and 2.15 there is need for α and β at $t = 0$ and $t = T + 1$ respectively. This issue is addressed under the term *trellis termination* in literatures.

2.4.2 Forward-only MAP Algorithm

Here we try to find $P(b_t = 0, Y_1^\tau)$ and $P(b_t = 1, Y_1^\tau)$ for $1 \le t \le \tau$. It is obvious that for $\tau = T$ these two probabilities turn out to be the ones in 2.5. Because of similarity between the cases $b_t = 0$ and $b_t = 1$, we will only extend the equations for $b_t = 1$. We have:

$$P(b_t = 1, Y_1^\tau) = \sum_i P(b_t = 1, S_\tau = i, Y_1^\tau) \qquad 1 \le t \le \tau, \tag{2.16}$$

The expression in the summation plays a major rule in this algorithm and it is worthwhile to dedicate a notation for it, say:

$$\Lambda_i^\tau(b_t = 1) \overset{\triangle}{=} P(b_t = 1, S_\tau = i, Y_1^\tau) \qquad 1 \le t \le \tau, \tag{2.17}$$

In an attempt to find a recursive formula for Λ, one can use the marginalization and Bayes' rules to write:

$$\Lambda_j^{\tau+1}(b_t = 1) = \sum_i P(b_t = 1, S_{\tau+1} = j, Y_1^{\tau+1}, S_\tau = i) \quad 1 \le t \le \tau+1 \tag{2.18}$$

$$= \sum_i P(b_t = 1, S_\tau = i, Y_1^\tau) \, P(Y_{\tau+1}, S_{\tau+1} = j | S_\tau = i, b_t = 1, Y_1^\tau) \quad 1 \le t \le \tau+1,$$

If for a moment we exclude $\tau+1$ from the time range, then the first expression under summation becomes clearly $\Lambda_i^\tau(b_t = 1)$. Thus we first continue for $1 \le t \le \tau$ and later we will return to the case of $t = \tau+1$. Note that b_t and Y_1^τ in the second part have no effect on the probability and can be ignored. According to definition in (2.6) it is $\gamma_{t+1}(i,j)$. Finally we obtain the following recursive equation, known as *update* recursion:

$$\Lambda_j^{\tau+1}(b_t = 1) = \sum_i \Lambda_i^\tau(b_t = 1)\gamma_{t+1}(i,j) \qquad 1 \le t \le \tau \tag{2.19}$$

Note that Λ has two time variables, τ and $t = \{1, \cdots, \tau\}$. So calculation of $\Lambda^{\tau+1}$ implies recomputation of Λ for all $t = 1, 2, \cdots \tau+1$.

We now return to equation (2.18) and write it for $t = \tau+1$:

$$\Lambda_j^{\tau+1}(b_{\tau+1} = 1) = \sum_i P(b_{\tau+1} = 1, S_{\tau+1} = j, Y_1^{\tau+1}, S_\tau = i)$$

$$= \sum_i P(Y_1^\tau, S_\tau = i) P(b_{\tau+1} = 1, S_{\tau+1} = j, Y_{\tau+1} | S_\tau = i, Y_1^\tau)$$

The first term in this equation is obtained by marginalizing Λ_i^τ over b_t, i.e. $\Lambda_i^\tau(b_\tau = 1) + \Lambda_i^\tau(b_\tau = 0)$ and the second term in which we can ignore Y_1^τ is:

$$P(b_{\tau+1} = 1, S_{\tau+1} = j, Y_{\tau+1} | S_\tau = i, Y_1^\tau) = \begin{cases} \gamma_{\tau+1}(i,j) & b_{\tau+1} = 1 \\ 0 & \text{otherwise} \end{cases}$$

So we reach an equation which is called *extend* recursion:

$$\Lambda_j^{\tau+1}(b_{\tau+1}=1) = \sum_{\substack{i \\ b_{\tau+1}=1}} \big(\Lambda_i^{\tau}(b_\tau=1)+\Lambda_i^{\tau}(b_\tau=0)\big)\gamma_{\tau+1}(i,j) \tag{2.20}$$

The update and extend recursions for $b=0$ are trivially similar to 2.19 and 2.20 respectively. The algorithm start with initial value of Λ and applying *extend* and *update* recursion repeatedly for $\tau = 1,2,\cdots,T$. Given Λ_i^T, applying (2.16) for $\tau = T$ yield the result, i.e.:

$$P(b_t=1,Y_1^{\tau}) = \sum_i \Lambda_i^T(b_t=1) \tag{2.21}$$

2.5 Block-wise MAP decoding

The problem with bit-wise MAP decoding is that the decoded stream is not necessarily a valid codeword. On the other hand in some applications we need to assure about codeword, that yield a better Frame Error Rate (FER) performance in the decoding process. Theoretically block-wise MAP is defined as follows:

$$\hat{c} = \operatorname*{argmax}_{\forall c\in\mathcal{C}} P(c|Y) \tag{2.22}$$

where c is a codeword in the code space $\mathcal{C}$ and Y is the observation. Viterbi algorithm is the best example of block-wise decoding that maximize $P(Y|c)$, i.e. Maximum Likelihood (ML) decoding and is equivalent to MAP for equally likely symbols. Viterbi algorithm is a special case of dynamic programming, was conceived by Andrew Viterbi as an error-correction scheme for noisy digital communication links, finding universal application in decoding of convolutional codes. For memoryless channel, equation 2.22 is written in additive form where log function is used to ease the implementation step:

$$\hat{c} = \operatorname*{argmax}_{(c_1c_2\cdots)\in\mathcal{C}} \sum_t \log\left(P(y_t|c_t)\right) \tag{2.23}$$

2.6 Illustrative example of bit-wise and block-wise decoding and a little more

If we spend less than a page and give a small example that clearly demonstrates bit-wise and block-wise decoding side-by-side, why not do it now!

Refer to figure 2.6, and assume that we encode five bits, $(b_1\cdots b_5)$ by the decoder into $(C_1\cdots C_10)$, then transmit 0 by +1 and 1 by -1 through AWGN channel with noise spectrum N_0 (arbitrary chosen $N_0=2$) and we get $(Y_1\cdots Y_5)$ at receiver's filter output. Table 2.1 gives an instance of values calculated for this example. For each valid path in the trellis, a probability can be associated and calculated by multiplying the probabilities of corresponding edges. There are 16 valid paths where

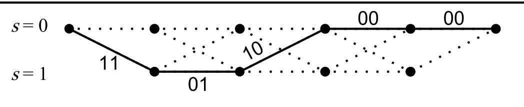
										Five-bit trellis of $[1\ 1/(1+\mathcal{D})]$ $0 \longrightarrow +1 \qquad 1 \longrightarrow -1$
-0.95	-1.95	0.87	0.60	-1.74	0.58	1.05	1.38	0.66	1.47	y (channel output)
0.13	0.02	0.85	0.77	0.03	0.76	0.89	0.94	0.79	0.95	$P(c{=}{+1}\|y) = \dfrac{p(y\|c{=}{+1})}{p(y\|c{=}{+1}) + p(y\|c{=}{-1})}$
(0.13)(0.02)(0.85)(0.77)(0.03)(0.76)(0.89)(0.94)(0.79)(0.95)										= 2.44 e -5 = $P(00000\|y)$
$(\overline{0.13})(\overline{0.02})(\overline{0.85})(0.77)(0.03)(0.76)(0.89)(0.94)(0.79)(0.95)$										= 1.4 e -3 = $P(11000\|y)$
(1-0.13) ⋮										⋮
$(\overline{0.13})(\overline{0.02})(0.85)(\overline{0.77})(\overline{0.03})(0.76)(0.89)(0.94)(0.79)(0.95)$										= 0.077 = $P(10100\|y)$
⋮										⋮
$(0.13)(0.02)(\overline{0.85})(\overline{0.77})(\overline{0.03})(0.76)(\overline{0.89})(\overline{0.94})(\overline{0.79})(\overline{0.95})$										= 8.7 e -8 = $P(01111\|y)$

Table 2.1: Bit and block wise decoding by an example

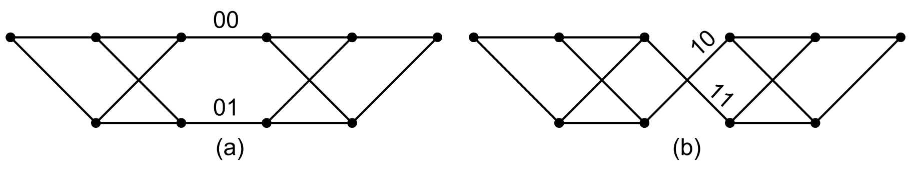


Figure 2.7: Paths that correspond to b_3 equal zeros (a) and one (b)

the one highlighted by solid line in the trellis has the largest probability of 0.077. This path is called *Viterbi Surviving path* which is claimed to represents the decoded output. This is an example of block-wise decoding because the information bits are taken out from a valid path in the trellis. Bit-wise MAP decoding considers all paths in the trellis, but divide them into two sets, one that correspond to $b_t = 0$ and the other for $b_t = 1$. For example consider b_3 in table 2.1 to be decoded. For the case $b_3 = 0$, MAP algorithm sum up the probabilities of 8 paths that has the edges as in figure 2.7(a). Similarly figure 2.7(b) shows other 8 paths that correspond to $b_3 = 1$. The final step is straightforward i.e. according to the larger probability, MAP decision is accomplished. MAP algorithm requires large memory and a large number of operations involving exponentials and multiplies. The algorithm is likely to be considered prohibitive for implementation in some communication systems. An important deviation from standard MAP to relax the complexity of computations is Max-Log-MAP algorithm. It may be applied to BCJR, Forward-only or other similar algorithms. The idea is to work with log of variables instead of variables themselves. For example α in BCJR algorithm is replace with $\overline{\alpha} = \ln(\alpha)$. This is also true for the β and γ. The principle approximation now can be applied

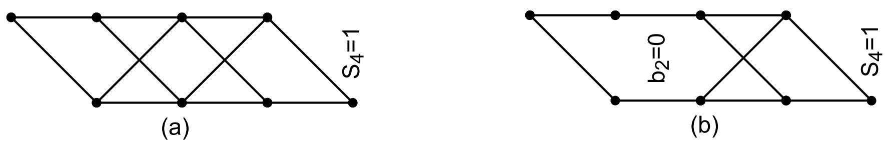


Figure 2.8: Graphical representation of $\alpha_4(1)$ in BCJR algorithm (a) and $\Lambda_1^4(b_2 = 0)$ defined in Forward-only algorithm (b)

to these log variables by the following rule:

$$\ln\left(\sum_k e^{\lambda_k}\right) \approx \max_k \lambda_k \tag{2.24}$$

where λ is dummy variable. Turning back to our example in table 2.1 and assuming again the decoding of b_3, Max-Log-MAP takes only the path with maximum probability among the 8 paths in figure 2.7(a) as an estimation for $\ln(P(b_3 = 0|Y))$. Similarly, $\ln(P(b_3 = 1|Y))$ is approximated by the path with maximum probability among the 8 paths in figure 2.7(b).

Before finishing this section, we would like to express graphically the meaning of α in BCJR algorithm and Λ in Forward-only algorithm. By definition $\alpha_t(i) = P(Y_1^t, S_t = i)$. This means summing up all paths of a sub-trellis with length t that ends at state i. For example to calculate $\alpha_4(1)$ we sum 8 paths shown in figure 2.8(a). In the other hand, Λ is defined by the equation (2.17). The value of Λ is equal to the sum of the probabilities of those paths that ends at i^{th} state at time τ and have the specified b_t. Figure 2.8(b) shows the paths for calculation of $\Lambda_{i=1}^{\tau=4}(b_2 = 0)$.

It should be remarked that the above discussion aims to give the concept of decoding algorithms and to compare them theoretically. The implementation issue is the other thing that should be considered separately which also depends on the type of platform to be employed.

2.7 Time-varying Convolutional Codes

Convolutional codes can be extended to be time-varying in order to achieve powerful codes with still tolerable complexity. Related issue in this context are LDPC convolutional codes and quasi cyclic codes. This issues first introduced in [17] and the class of quasi cyclic codes in [18, 19].

As a counterpart to equation 2.1, we may have a time-varying polynomial generator matrix $\mathbf{G}(t)$. A practical and realizable TVC code, however can be obtained if the time variation repeats with a constant period T. Definition of TVC codes can be best understood by the structure of its encoder. Figure 2.9 shows such an encoder with memory m and rate $1/2$. The extra devices with respect to a CC encoder are two multiplexers that give the feature of time-dependency to the encoder. At each clock cycle the multiplexers receive clock signal φ that control their function. They are reset every T clocks for a code with the period T.

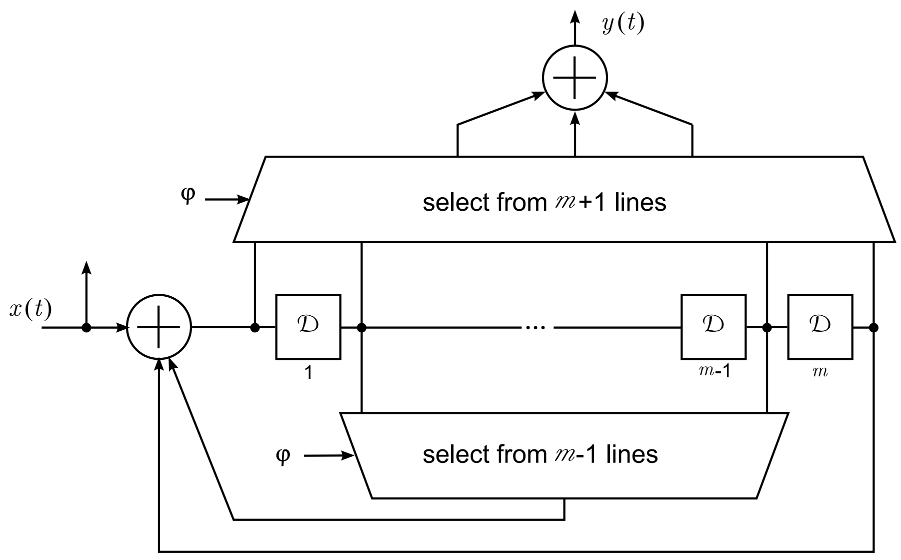


Figure 2.9: An instance of time-varying convolutional code

The order of rational polynomial of a TVC code with memory m is at most m and thus can be denoted by $m+1$ time-dependent coefficients: $\{h_0(t), \cdots h_m(t)\}$ each with dimension $n \times (n-k)$. The constraint at time t for input $x(t)$ is:

$$x_t h_0(t) + x_{t-1} h_1(t) + \cdots + x_{t-m} h_m(t) = 0. \tag{2.25}$$

Consequently the parity check matrix of a TVC codes that satisfies $XH' = 0$ is generally as follows:

$$H' = \begin{pmatrix} & \vdots & & & \\ h_0(t-m) & \cdots & h_m(t) & & \\ & & \vdots & & \\ & & h_0(t) & \cdots & h_m(t+m) \\ & & & \vdots & \end{pmatrix} \tag{2.26}$$

Since the coefficients have been assumed to be periodic, time variable in matrix (2.26) are modulo T and a primitive block can be extracted so that the parity check matrix is constructed based on.

$$P = \begin{pmatrix} h'_m(1) & h'_{m-1}(1) & \cdots & h'_0(1) & 0 & \cdots & 0 \\ 0 & h'_m(2) & \cdots & h'_1(2) & h'_0(2) & \cdots & 0 \\ \vdots & & & & & & \vdots \\ 0 & 0 & h'_m(T) & & \cdots & & h'_0(T) \end{pmatrix} \tag{2.27}$$

Paying attention to the size and structure of primitive matrix is essential in design of a modular decoder, i.e. partitioning of decoder into identical sub-blocks. This

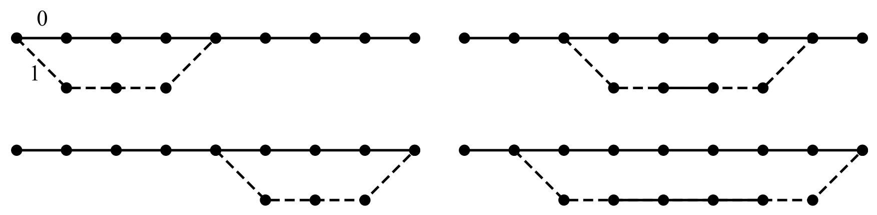


Figure 2.10: The sub-trellis associated to four rows of extended Hamming (8,4) generator matrix

issue is considered in design of analog decoder for a quasi cyclic code in section 5.3. Quasi cyclic codes are the systematic version of time-varying periodic LDPC convolutional codes and are similarly defined by code memory and code period.

2.8 Trellis Representation of Block codes

The issue of representing a given block code by a trellis returns originally to a paper in 1973 by Bahl, Cocke, Jelinek, and Raviv [15]. In this paper an important connection between convolutional codes and block codes has been revealed. After that there were lots of contribution to the problem targeted to create a better trellis. Figure of merits for construction of a trellis is different in the papers, some of them yield a trellis with minimum number of edges, other yield minimum number of states, etc. The methods have been well covered by two tutorial papers [20] and [21] and a recent thesis that covers the issues as well as construction of tail-biting trellis [22] for a block code. We will give a simple example of trellis construction now. We consider extended hamming (8,4) code defined by the following matrix. Note that in this case the parity check matrix is equal to the generator matrix.

$$H = G = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \tag{2.28}$$

Following the method explained in [20], the procedure for trellis construction begins with sub-trellis associated to each row in G. This corresponds to a $(n,1)$ sub-code. The sub-code corresponds to each row of the generator matrix G. The sub-trellis associated to sub-code clearly has 2 states at most, because one input bit can be sufficiently expressed by two states. The four sub-trellis associated to each row of the G are depicted in figure 2.10. The next and important step is to merge the sub-trellis to obtain the final trellis. To do this we need the concept of trellis product. Any transition in a trellis can be characterized by three-tuple $\{S_i, b, S_j\}$ that defines a branch with the value $b \in \{0,1\}$ emanates from S_i and goes to S_j. The product of two such tuples is defined as follows:

$$\{S_i, b_1, S_j\} \times \{S_m, b_2, S_n\} = \{S_iS_m, b_1 \oplus b_2, S_jS_n\} \tag{2.29}$$

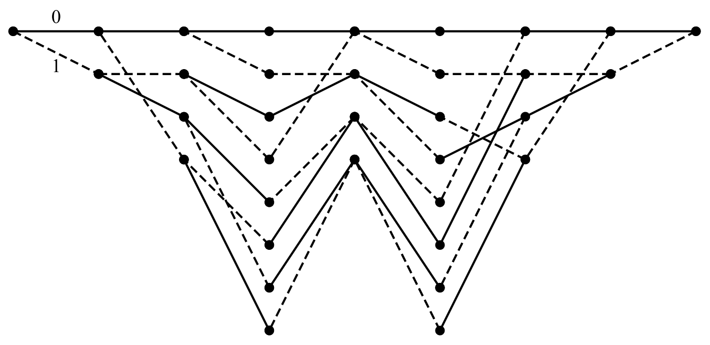


Figure 2.11: Trellis diagram for (8,4) Hamming code

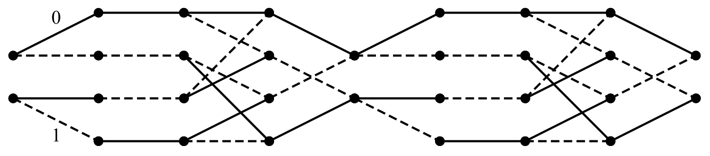


Figure 2.12: Tail-biting trellis diagram for (8,4) Hamming code

It is obvious that multiplication of two sub-trellis with state cardinality of two, yields a trellis with state cardinality of four. In general with a generator matrix of k rows, the upper bound for number of state will be 2^k. For the above example we obtain the figure 2.11.

In practice there is often need for a trellis with more regular shape such that the state cardinalities at different sections of trellis don't differ so much. Many researchers worked on this problem by introducing tail-biting concept. In a tail-biting trellis any path that starts and ends at the same point represents a valid codeword. In general, a tail-biting trellis can reduce the maximum state cardinality to the square root of the original conventional trellis. For example the Hamming (8,4) code have a tail-biting trellis as in figure 2.12.

Given a trellis for a block code, all of the decoding methods described so far are applicable to block codes. This provide an optimum decoding results and is useful in digital or analog decoding schemes.

2.9 HMM and Decoding

Hidden Markov model (HMM) have been finding a nice tool for speech processing. HMM provide a good model for representing temporal sequences of data, what is often claimed for speech signals. At the same time, coding processes can be regarded as a Markov chains, but the difference in notation may cause this connection obscure. In the following we try to like up these important concepts. For HMM we adopt the notation in the well-known tutorial by Rabiner [23].

HMM defined by two consecutive stochastic processes. The first process is the

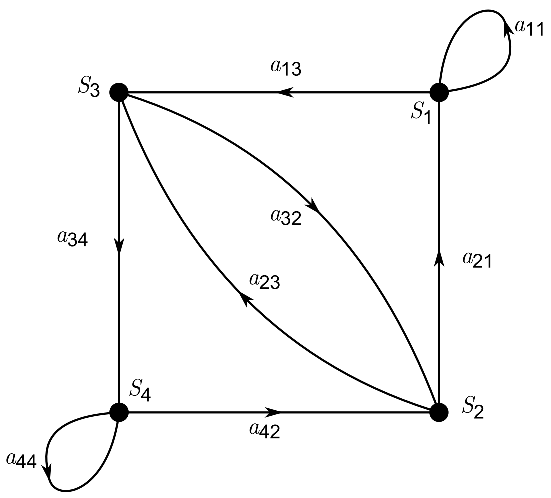


Figure 2.13: a Markov chain with four states

transitions between the states with different probabilities and the second process is to release output(s) according to the probability distribution of the states. The Markov property says that a current state of the model is affected by the previous state. The state transition probability matrix $A = \{a_{ij}\}$ is used to define the first process as:

$$a_{ij} = P(q_{t+1} = S_j | q_t = S_i) \qquad 1 \leq i, j \leq N, \tag{2.30}$$

where N states is assumed for the model and the discrete variable q_t is used to show the state at time t. It is convenient to represent A graphically. For example figure 2.13 shows a Markov chain with four states with state transition probability mentioned on each edge. It is clear that in any case the standard constraints on the probabilities hold:

$$a_{ij} > 0, \qquad \sum_{j=1}^{N} a_{ij} = 1.$$

Beside the the transition matrix, we need a vector to define starting probabilities of each state, that is:

$$\pi_i = P(q_1 = S_i), \qquad 1 \leq i \leq N.$$

In addition for the second stochastic process that relates to observation from each state, we may define a discrete or continuous density function. In general we can write:

$$b_j(y) = P(y|S_j). \tag{2.31}$$

To construct a direct relationship with coding process, the transitions between encoder states can be considered as the first process of HMM and the effect of

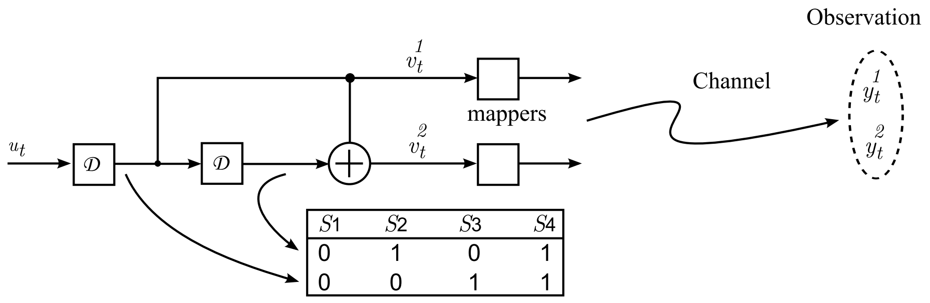


Figure 2.14: An instance of convolutional encoder and its relationship to HMM

channel on encoder output is considered as the second process. For instance, for BSC we have a discrete density function while for AWGN we have a continuous density function as the second process in HMM. There is still a small difference between coding and HMM defined as above. In definition of HMM, we have firstly a transition to a state and secondly an observation is obtained. On the other hand, in coding we have an observation as soon as a transition takes place. So in order to simulate HMM process in coding there is need to consider encoder input as a part of state. Equivalently we may add a delay (or set of delays for multi-input encoder) at encoder input and take the state as usual. Figure 2.14 represents this modification for a simple encoder. The diagram for state transitions is the one has been already shown in figure 2.13. Transition matrix in this case have values of 0.5 for all valid transition, providing that information source hase equiprobable 0 and 1. In other cases, say turbo iteration, these a priori values can change. So the matrix A is:

$$A = \begin{pmatrix} 0.5 & 0 & 0.5 & 0 \\ 0.5 & 0 & 0.5 & 0 \\ 0 & 0.5 & 0 & 0.5 \\ 0 & 0.5 & 0 & 0.5 \end{pmatrix}.$$

For the observation we have to use the probability distribution of channel that is trivial. Given such equivalent model, all of the algorithms in HMM context are applicable to coding. Apart from the historical point of view there are not any difference with what we are using in decoding context.

2.10 conclusions

This chapter has illustrated implicitly the importance of state as an intermediate variable, in the structure of encoders and decoders. Different encoder realizations for a given code is in fact a re-definition of state variables. Different decoding algorithms, say BCJR, forward-only and HMM approach which have been considered, are rooted in definition of state variables. The attention on this issue is crucial in our study and is related to graph theory that will be considered in chapter 3.

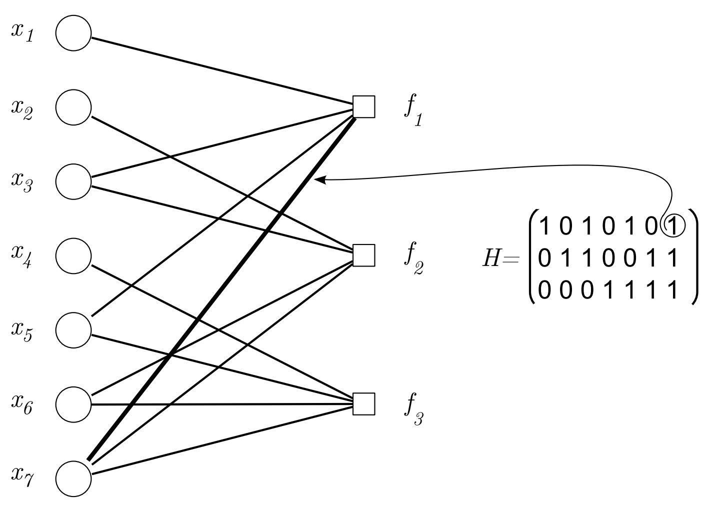
Chapter 3

Graph and Sum Product Algorithm

A graphical model is a family of probability distributions defined in terms of a directed or undirected graph. The nodes in the graph are identified with random variables, and joint probability distributions are defined by taking products over functions defined on connected subsets of nodes. The graph provides an appealing visual representation of a joint probability distribution, but it also provides a great deal more. By exploiting the graph-theoretic representation, the formalism provides general algorithms for computing marginal and conditional probabilities of interest. Moreover, the formalism provides control over the computational complexity associated with these operations.[24]

Historically, exploiting graphs to define codes and decoding returns science the invention of low-density parity-check (LDPC) codes of Gallager [25]. Gallager also invented what today is called the sum-product algorithm for a posteriori probability (APP) decoding. Some years later, Tanner [26] founded the general study of codes defined on graphs, introduced generalized constraints, and proved the optimality of the sum-product and min-sum algorithms for decoding codes defined on cycle-free graphs. MacKay [27] was the first one who employed the concept of Bayesian network and belief propagation into coding theory. Modern interest in this subject ushered in with the work of Wiberg, Loeliger, and Koetter [28, 29], who rediscovered Tanner's results and introduced states into Tanner graphs, thus allowing connections to trellises and to turbo codes. Connections to Bayesian networks and Markov random processes were then recognized [30, 31]. Later Forney took the original definition of factor graph and applied some modifications to allow direct connection of function nodes. He called this new graph *normal graph* [32].

When the messages in a graph are conditional probabilities, the sum-product algorithm takes the name of *belief propagation* (or more naturally probability propagation). Belief propagation algorithm has close relationship with Bays's rule and Markov chain. In fact the Bayes graphs were used by Pearl's works since 1988 to represent Bayes formula graphically.



x_1
x_2
x_3
x_4
x_5
x_6
x_7
f_1
f_2
f_3
$H=\begin{pmatrix}1&0&1&0&1&0&\textcircled{1}\\0&1&1&0&0&1&1\\0&0&0&1&1&1&1\end{pmatrix}$

Figure 3.1: Construction of Tanner graph from parity check matrix

Given any parity check matrix, H for a linear block code, its one-to-one equivalent Tanner graph can be drawn by the following straight-forward procedure. Assume the code is a $[n,k]$ ensemble and therefore H is a $n\times(n-k)$ matrix. The graph contains two types of node[1] , variable node[2] and check node[3]. we draw n variable nodes associated with every column of H and $n-k$ check node associated with every row of H. Then we draw an edge for every non-zero element in H that connects the corresponding variable node and check node. Figure 3.1 illustrates the procedure for a Hamming code.

3.1 Probability Conventions in graphs

Before starting the algorithm of belief propagation, we deals herein with the quantity of *belief* in a graph. Basically they are conditional probabilities such as $p(b|observation)$ were b is a single bit and *observation* is the information obtained from channel (directly or indirectly) for one or several received symbol(s). For simplicity we define:

$$\pi(b|observation) \overset{\triangle}{=} P(b=0|observation) \tag{3.1}$$

It may be computationally advantageous to use another quantity, say likelihood ratio defined as:

$$\lambda(b|O) = \frac{\pi(b|O)}{1-\pi(b|O)} \tag{3.2}$$

[1]called also vertex

[2]called also symbol node

[3]called also function node or constrain node

Another possibility is likelihood difference defined as:

$$\delta(b|O) = 2\pi(b|O) - 1 \tag{3.3}$$

Finally the log likelihood ratio (LLR) which is more interesting in BP algorithm is defined as:

$$\Lambda(b|O) = \ln \frac{\pi(b|O)}{1-\pi(b|O)} \tag{3.4}$$

The inverse of LLR is then:

$$\pi(b|O) = \frac{e^{\Lambda(b|O)}}{1+e^{\Lambda(b|O)}} \tag{3.5}$$

3.2 Probability propagation at Graph's Nodes

Before the expressing of BF algorithm, two following basic questions should be answered. Herein the random processes are assumed to be statistically independent.

First question: given $\pi(b_1|y_1)$ and $\pi(b_2|y_1)$, what is $\pi(b_1 \oplus b_2|y_1, y_2)$?

$$\begin{aligned}\pi(b_1 \oplus b_2|y_1,y_2) &= P(b_1=0, b_2=0 \cup b_1=1, b_2=1|y_1,y_2) \\ &= P(b_1=0,b_2=0|y_1,y_2) + P(b_1=1,b_2=1|y_1,y_2) \\ &= \pi(b_1|y_1)\pi(b_2|y_2) + [1-\pi(b_1|y_1)][1-\pi(b_2|y_2)] \end{aligned} \tag{3.6}$$

We can write expression 3.6 more compactly based on LLR as follows. From 3.5 and the identity relation $\tanh(x/2) = (e^x-1)/(e^x+1)$ we obtain:

$$\pi(b|O) = \frac{1+\tanh(\Lambda/2)}{2} \tag{3.7}$$

Applying 3.7 into 3.6 yeild the result:

$$\tanh\left(\frac{1}{2}\Lambda(b_1 \oplus b_2|y_1,y_2)\right) = \tanh\left(\frac{1}{2}\Lambda(b_1|y_1)\right)\tanh\left(\frac{1}{2}\Lambda(b_2|y_2)\right). \tag{3.8}$$

Hagenauer [33] introduced the compact notation $\boxplus$ for this operation and called it *boxplus*.

$$\Lambda(b_1 \oplus b_2|y_1,y_2) \stackrel{\triangle}{=} \Lambda(b_1|y_1) \boxplus \Lambda(b_2|y_2) \tag{3.9}$$

By induction one can generalize the above relationship for more than two operands. One can simply verify that the following properties hold in boxplus operation.

$$\Lambda(b) \boxplus \infty = \Lambda(b) \qquad \Lambda(b) \boxplus (-\infty) = -\Lambda(b) \qquad \Lambda(b) \boxplus 0 = 0 \tag{3.10}$$

A useful approximation to 3.8 is by replacing *tanh* by *min* operator.

$$\Lambda_1 \boxplus \Lambda_2 \approx sign(\Lambda_1) sign(\Lambda_1) \min(|\Lambda_1|, |\Lambda_2|) \tag{3.11}$$

where for the simplicity the indexes are replaced for the argument in LLR functions. The sign of LLR of a bit is regarded as the hard decision of the bit and its absolute value is regarded as the certainty of the bit.

Second question: Given extrinsic information $P_{ex}(b|y_1)$ and $P_{ex}(b|y_2)$, what is $\pi(b|y_1, y_2)$? The emphasis on *extrinsic* implies that it is a pure channel information and no a priori information is delivered by it. It is important because we don't want to take into account the a priori $\pi(b)$ several times. By definition the extrinsic probability is thus:

$$P_{ex}(b|y) = \frac{p(y|b)}{p(y|b=0) + p(y|b=1)} \tag{3.12}$$

Answer to this question is a basic Baye's problem and can be written as:

$$\begin{aligned}\pi(b|y_1, y_2) &= \frac{p(y_1, y_2|b=0)\pi(b)}{p(y_1, y_2)} \\ &= \frac{p(y_1|b=0)p(y_2|b=0)\pi(b)}{p(y_1|b=0)p(y_2|b=0)\pi(b) + p(y_1|b=1)p(y_2|b=1)(1-\pi(b))}\end{aligned}$$

Using the extrinsic probability definition of 3.12 and replacing for $p(y_1|b)$ and $p(y_2|b)$ yield:

$$\begin{aligned}&\pi(b|y_1, y_2) = \\ &\frac{\pi_{ex}(b|y_1)\pi_{ex}(b|y_2)\pi(b)}{\pi_{ex}(b|y_1)\pi_{ex}(b|y_2)\pi(b) + (1-\pi_{ex}(b|y_1))(1-\pi_{ex}(b|y_2))(1-\pi(b))}\end{aligned} \tag{3.13}$$

Note that $\pi(b)$ is the *a priori* information of the bit b and is only used when it is applicable, such as turbo decoding. This equation takes much simple shape when is expressed based on LLR. Applying 3.5 yields:

$$\Lambda(b|y_1, y_2) = \Lambda(b|y_1) + \Lambda(b|y_2) + \Lambda(b) \tag{3.14}$$

More interestingly, the extrinsic channel LLR, $\Lambda(b|y)$, is proportional to y which is desirable for its simple implementation in a decoder. To see that, we consider two special cases of BSC and Gaussian channels. Assume p_e as crossover probability in BSC and assume a BPSK modulation, then:

$$\begin{aligned}\Lambda(b|y) &= \ln \frac{\pi_{ex}(b|y)}{1-\pi_{ex}(b|y)} \\ &= \ln \frac{p(y|b=0)}{p(y|b=1)} \\ &= \begin{cases} \ln \frac{1-p_e}{p_e} & y=+1 \\ \ln \frac{p_e}{1-p_e} & y=-1 \end{cases} \\ &= \left(\ln \frac{1-p_e}{p_e}\right) y\end{aligned} \tag{3.15}$$

The approach for AWGN channels is similar except for incorporating normal distribution of Gaussian noise into the calculation.

$$\begin{aligned}\Lambda(b|y) &= \ln\frac{\pi_{ex}(b|y)}{1-\pi_{ex}(b|y)}\\ &= \ln\frac{p(y|b=0)}{p(y|b=1)}\\ &= \ln\frac{\exp(-\frac{1}{2\sigma_n^2}(y-\alpha E)^2)}{\exp(-\frac{1}{2\sigma_n^2}(y+\alpha E)^2)}\\ &= \left(\frac{2\alpha E}{\sigma_n^2}\right)y \qquad (3.16)\end{aligned}$$

where α denotes the fading coefficient for a fading channel and is set to one for AWGN channels. We have assumed that the signal is observed at the receiver's filter output where E denotes the magnitude of this signal at the sampling time. Also σ_n^2 is noise variance at the filter output. Assumption of using matched filter in receiver and ideal sampling time, maximize the ratio in equation (3.16). Under this assumptions, E becomes E_s, the energy of the transmitted pulse[4], and the value of σ_n^2 becomes $\frac{N_0}{2}$,the power spectrum of noise in the channel [5].

The factor of y in both BSC and AWGN channels is a function of channel and reflects the reliability of channel under consideration and will be shown herein with K_c.

3.3 Belief Propagation Algorithm

After the previous introductory sections, we are preparing to define the exchange of extrinsic information for which the concept of belief propagation will be emerged. we start with Tanner graph but some of the other graphical conventions will be discussed later. According to equations 3.9 and 3.14, there are only two type of operations in BP algorithm that leads to a bipartite graph with two type of nodes, *function* node and *variable* node. First, we will consider the binary graph where all messages are related to a binary variable. Other graph with non-binary variable i.e. trellis diagram of a code will be considered in the next chapter.

To better demonstrate the BP algorithm, we start by figure 3.2(a) where the process in a variable node is illustrated. Assume that LLR is chosen arbitrary for the message in this graph. The outgoing message from x_7 node is the LLR for this bit given other information received from the channel. These information include our direct observation of the bit, x_7, as well as two other ones which are depicted by two incoming arrows in x_7 node. By applying equation 3.14 and arbitrary ignoring $\pi(b)$ we have:

$$\Lambda(b_7|x_7, group1, group2) = \Lambda(b_7|x_7) + \Lambda(b_7|group1) + \Lambda(b_7|group2)$$

[4]with a unity factor to normalize the dimension

[5]This is the direct results of Wiener-Kinchine theorem and ergodicity of Gaussian process.

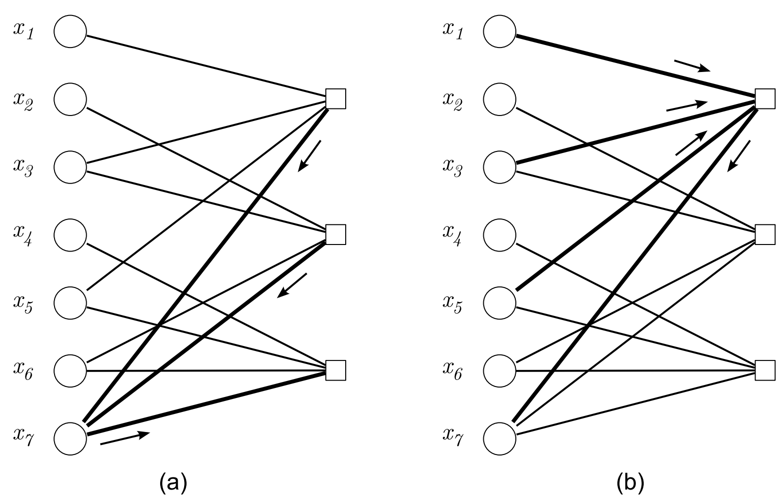


Figure 3.2: Illustration of message passing algorithm on a graph, variable-to-function message (a) and function-to-variable message (b)

Clearly, this process augments the certainty of b_7 by incorporating other information in addition to x_7 itself. *group*1 and *group*2 are used to simplify the notation and they are stand for $x_1 \oplus x_3 \oplus x_5$ and $x_2 \oplus x_3 \oplus x_6$ respectively. sometimes we may emphasis on the message direction. In such cases the above calculation is recognized as a variable-to-function (V2F) message. The second and last process in Tanner graph is calculation of messages in reverse direction, i.e. function-to-variable (F2V) message. Refer to figure 3.2(b), the outgoing message from the function node is defined and calculated as follows:

$$\Lambda(b_1 \oplus b_3 \oplus b_5 | x_1, x_3, x_5) = \Lambda(b_1|x_1) \boxplus \Lambda(b_3|x_3) \boxplus \Lambda(b_5|x_5)$$

where we use the equation 3.9 extended for three inputs.

Knowing the calculation of V2F and F2V messages, the belief propagation algorithm is briefly defined as recursive calculation of these messages in a graph.

3.4 Other graphical representations

After Tanner who treated the codes and decodings by his bipartite graphs [26], a group in ISIT'97 including Tanner, generalized the concept and introduced factor graph [34]. The main idea behind this generalization was a graphical model for coding (and beyond as well) by applying them to arbitrary functions. Almost at the same time Forney who was also in the group, introduced *normal graph* in a separate paper [32]. Fortunately all of the abovementioned graphical models differ only in their appearance at least when applied to coding theory. In order to better illustrate these different representations, we give some examples for them, supporting by their equivalent Tanner graph.

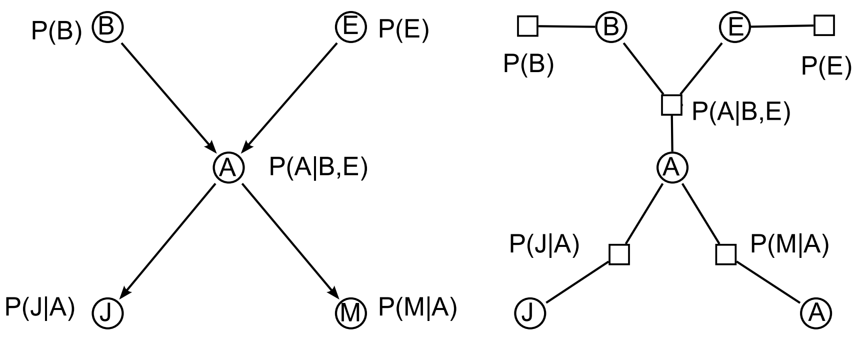


Figure 3.3: Procedure to convert a Bayes net to a factor graph

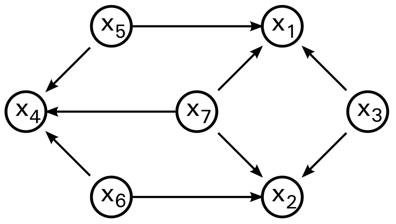


Figure 3.4: Baye's net fo Hamming (7,4) code

We first introduce Bayes net which is the oldest one, had been exploiting for BP algorithm. Figure 3.3(a) is an example of such graph. Note that it is a directed graph for which we can simply recognize the parent variable(s) and the child variable(s) at any point. For example in this figure B and E are parents of A where J and M are childes of A. Given a Bayes graph one can obtain corresponding factor graph easily. For doing this we create function node for each variable and simply copy the variable nodes. Then connect function node to its variable node and to the parent variable node(s). Finally associate a conditional probability to each variable node. This procedure results in a undirected graph as illustrated in figure 3.3(b).

As an another example, the hamming code depicted in figure 3.1 has the following constraints:

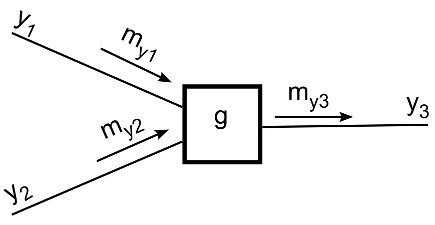
$$x_1 = x_3 \oplus x_5 \oplus x_7, \quad x_2 = x_3 \oplus x_6 \oplus x_7, \quad x_4 = x_5 \oplus x_6 \oplus x_7,$$

using these constraints, the global function can be factorized as follows:

$$\begin{aligned} P(x_1,\dots x_7) = P(x_1|x_3,x_5,x_7)&P(x_2|x_3,x_6,x_7) \\ &P(x_3)P(x_4|x_5,x_6,x_7)P(x_5)P(x_6)P(x_7), \quad (3.17) \end{aligned}$$

and its corresponding Baye's graph is shown in figure 3.4

Another popular graph is Factor Graph (FG) and like before contains two type of nodes, function node and variable node. Wiberg [] developed this definition by introducing an extra node to graph, named state node. The definition of Wiberg



$$m_{y3} = \sum_{y_1,y_2} g(y_1,y_2,y_3) m_{y1} m_{y2}$$

Figure 3.5: Messages in Normal graph

extents the application of graph to trellis and forward-backward algorithm, state space model, etc. On the other hand Forney preferred to made a modification to Wiberg graph and he called it Normal Graph (NG). In NG there is only the function node and the edges handles the variables in a graph. There are two possibilities: First, the observed variables are shown as half edge. Second, other variables are states and are shown as full edges. According to Forney this is a new representation in the filed of graph, but is well known in system theory and resembles the block diagram representation. Another effect of NG definition is the nominal difference in message passing algorithm. In NG there are only one equation for evaluation of messages as illustrated in figure 3.5. Although it seems to be more simple using NG, but the complexity of actual algorithm dose not change. For example in decoding process there are need for two function nodes: soft xor gate and soft equal gate ($\boxplus$ and $\boxed{=}$). The first node is a *boxplus* node and the second node lends the equation of variable node. Thus we have once again the known message passing algorithm in our graph.

Figure 3.6 shows the Tanner graph in figure 3.1 and its equivalent normal graph.

3.5 Optimality of BP Algorithm

Each iteration of belief propagation can be thought of as an operator $\mathfrak{F}$ that inputs a list of messages $m(t)$ and outputs a list of messages $m(t+1) = \mathfrak{F}\big(m(t)\big)$. Thus belief propagation can be thought of as an iterative way of finding a solution to the fixed point equations $m_\infty = \mathfrak{F}(m_\infty)$. McEliece et al. [35] have shown a simple example for which $\mathfrak{F}$ contains multiple fixed points and belief propagation finds only one. They also showed a simple example where a fixed-point exists but the iterations do not converge. So belief propagation will obviously not work for arbitrary networks and distributions. However when nodes in a graph are described by jointly Gaussian random variables, there are sufficient conditions for the convergence of BP algorithm that yields the correct posteriori [36].

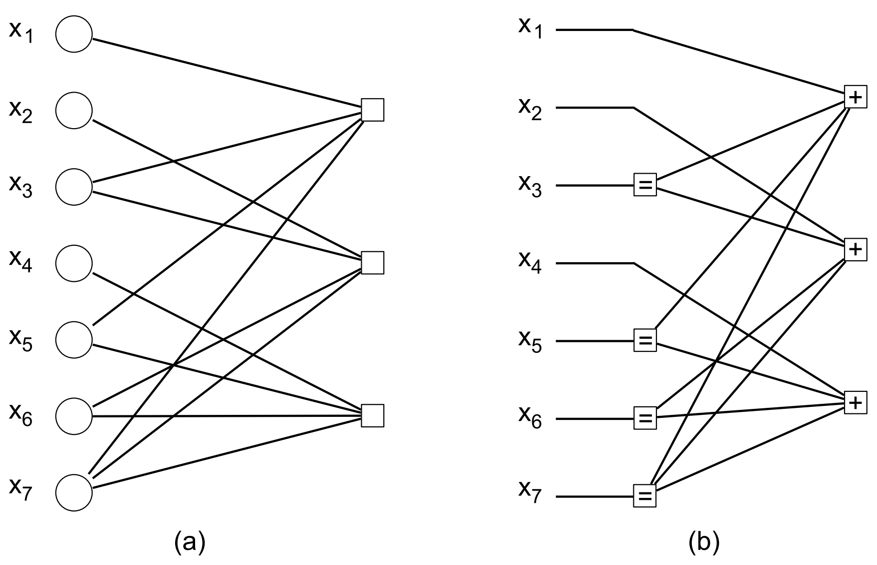


Figure 3.6: An instance of a factor graph and its equivalent normal graph

3.6 Scheduling the Messages in BP Algorithm

In the previous sections we considered the calculation of the messages in graph. But the details on ordering the messages and initialization was not given. This issue is considered under the title of *scheduling*. Several scheduling method have been devised since now, say Flooding, Probabilistic vertical and Horizontal shuffle, etc. [37]. For circuit speed consideration and the analog implementation that uses a parallel structure, flooding schedule is a good choice for simulation. In this schedule, once a new message is generated, it is then passed to its destination and once the result on an edge changes, a new message is generated on that edge. If multiple messages are generated at the same time on the graph, then they are all passed out at the same time. Also, the flooding schedule is a simple message passing schedule for simulation software since there is no order to the update [38]. Nevertheless in a real analog decoder there is no clock and this issue is not applicable any more.

3.7 Decoding of Tailbiting Convolutional Codes on Graphs

This section proposes a general method to develop Tanner graphs from tail-biting convolutional codes (CC). Recursive Systematic Convolutional (RSC) and non-RSC codes are considered consistently and it leads to a unique graph applicable for decoding of both RSC and non-RSC codes. In addition the graphical representation is extended to derive the condition for which the tail-biting termination is valid. Later in chapter 5 the graph will be realized by exploiting the analog decoding scheme and MOS transistors and simulation results will be provided.

Although it has been mentioned that decoding on trellis results in optimal so-

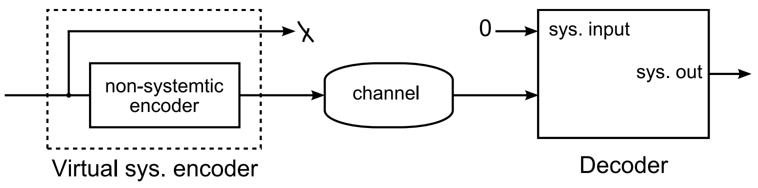


Figure 3.7: A method that incorporates systematic information into the codeword to obtain a graph with desired nodes.

lution, the complexity of the required circuit becomes prohibitive for large information vector length and/or high code complexity. In fact trellis representation introduces 2^m states where m is the number of delay elements in the associated minimal encoder. Handling of this non-binary variable demands a complex realization. In addition, if we change m, the design of circuit changes.

In contrast to trellis representation, this section concerns definition and decoding of tailbiting CC by bipartite binary graph (c.f. 3.3) and sum-product algorithm. As we saw earlier in such graphs there are only two simple operations of sum and boxplus. Such versatile binary graph are suitable for analog realization. We also consider tailbiting convolutional codes which is again suitable for our propose due to the inherent feedback in its graph.

3.7.1 Non-systematic tailbiting Convolutional Codes

In a feed-forward encoder, state at any time is a function of several recent information bit(s) and thus it is easy to obtain a tail-biting code by loading the encoder's registers with appropriate last information bits. This simple operation however introduces a latency equivalent to one frame of information.

The strategy to obtain the graph is simple in this case as outlined in Figure (3.7). We force the generator matrix to be systematic by incorporating the systematic information into the codeword. So the encoder output stream consists of information bits as well as the parity bits. But the information bits are not sent and they are actually discarded in the transmitter (this can be regarded as a systematic puncturing). At destination, we have a decoder with systematic input. Since the information bits are not sent, systematic inputs are initialized to null values at receiver input (e.g. zero in log-likelihood domain). Assume a non-systematic encoder defined by the $k \times n$ matrix G. Including information bits results in the larger systematic generator and parity matrices of $G_S = [I_k|G]$ and $H_S = [H|I_n]$ respectively. This approach makes the final parity check matrix preserve the form of the code. More importantly, the extrinsic values for information bits can now be obtained from the decoder. The later advantage is essential in a turbo decoder. Using H_S one can draw a graph to show the code graphically and to perform decoding on it. We postpone this issue upon to the next subsection, when the encoders with feedback have been considered.

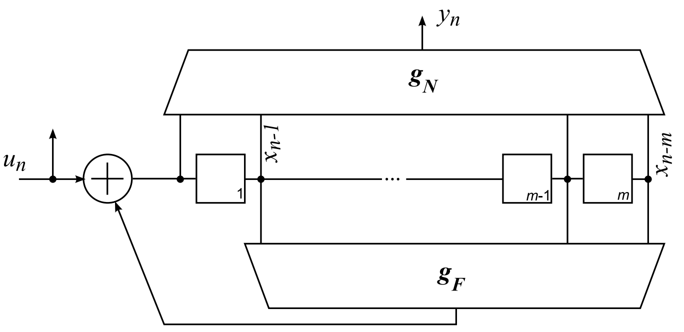


Figure 3.8: Controller canonical realization of recursive encoder

3.7.2 Encoder with Feedback

The encoders with feedback are commonly used for RSC codes. Generation of tailbiting RSC codes is more difficult than non-RSC codes because the last state of the encoder is a function of entire information vector. Solution to this problem is based on the state space equations of the encoder [39]. State space equations depend on the encoder structure. Among varieties of structures, the controller canonical form is more desirable for our work. This encoder diagram and its connection to observable canonical form has been illustrated in figure 2.4. A general controller canonical encoder diagram depicted again in figure 3.8 for convenience. Such minimal realization has the maximum redundancy for the elements of the state vector because except for the first state, all other ones perform only a shift right at each clock cycle and their contents don't change. The state space equations can be easily written by direct inspection or using equation 2.3.

$$x_{n+1} = Ax_n + Bu_n \tag{3.18}$$

where

$$A = \begin{pmatrix} g_1 & \cdots & g_{m-1} & g_m \\ 1 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 1 & 0 \end{pmatrix} \qquad B = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{3.19}$$

In A elements g_i are related to the polynomial $g_F(\mathcal{D}) = g_1 + g_2\mathcal{D} + \cdots + g_m\mathcal{D}^{m-1}$, and characteristic function of the encoder:

$$g_D(\mathcal{D}) = 1 + g_1\mathcal{D} + g_2\mathcal{D}^2 + \cdots + g_m\mathcal{D}^m. \tag{3.20}$$

Applying the constraint that the final state equals the initial state yields the following answer:

$$\left(A^N + I_m\right)x_0 = x_N^{\{zs\}} \tag{3.21}$$

where N is the length of information vector, I_m is identity matrix of size m, x_0 is the initial state to be calculated and $x_N^{\{zs\}}$ is the zero-state response of the encoder.

Accordingly the coding procedure to obtain a tailbiting systematic code from recursive encoder is as follows:

1. Apply information vector to the encoder with zero initial state. The final state will be $x_N^{\{zs\}}$.
2. Calculate x_0 from equation (3.21).
3. Run the encoder with x_0 as initial state and take the encoder output which is a tail-biting code

Recall that equation (3.21) may have no solution for some particular N and A. In such case a tail-biting code does not exist. We will return to this problem more accurately in the next subsection.

3.7.3 Algebraic properties of tailbiting RSC and non-RSC codes

In this subsection we will discuss the subtle relation between tailbiting non-RSC and RSC codes which leads to a unique graph for both codes.

First, consider the recursive encoder in figure (3.8) and recall the equality at the sum (xor) output:

$$u_n + x_n + g_1 x_{n-1} + \cdots + g_m x_{n-m} = 0 \tag{3.22}$$

Second, define X to be a vector of length N defined as follows:

$$X = (x_{-m} \cdots x_0 x_1 \cdots x_{N-m-1}) \tag{3.23}$$

Note that each m-tuple of adjacent elements in X is a state of encoder. For example $(x_{-m} \cdots x_{-1})$ is the initial state, $(x_{-m+1} \cdots x_0)$ is the state after first clock pulse and so on. Definition of X in 3.23 implicitly implies the concept of tailbiting because the vector is truncated at x_{N-m-1} after which the initial state of encoder will be generated. The properties of X can be best demonstrated by figure (3.9). Equation (3.22) hold for every m adjacent nodes in this figure. In addition we can conclude that X is cyclic, i.e. choosing an arbitrary starting point, we will return to it after N clocks. This property may be formulated as follows:

$$X^{[i]}(\mathcal{D}) \equiv \mathcal{D}^i X(\mathcal{D}) \pmod{1 + \mathcal{D}^N} \tag{3.24}$$

where $X(\mathcal{D})$ is polynomial representation of (3.23) and $[i]$ denotes i-times circular right shift. Using equations (3.22) and (3.24) we can obtain a useful relationship for tail-biting RSC codes:

$$g_D(\mathcal{D}) X(\mathcal{D}) \equiv \mathcal{D}^m u(\mathcal{D}) \pmod{1 + \mathcal{D}^N} \tag{3.25}$$

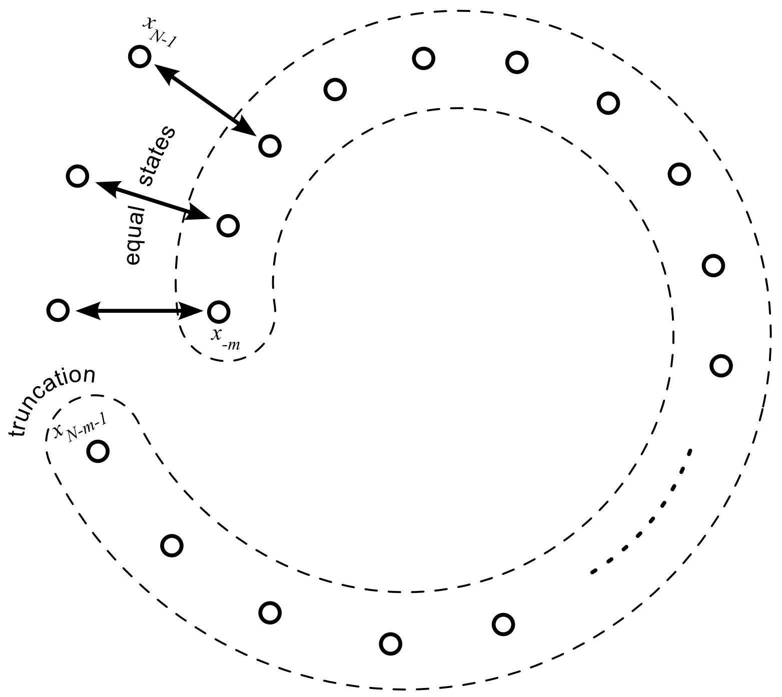


Figure 3.9: A global state vector comprised of local state vectors

where $g_D(\mathcal{D})$ is defined in (3.20) and $u(\mathcal{D})$ is a polynomial made up by the entire information vector of length N. From equation (3.25) we find that for an arbitrary $u(\mathcal{D})$, the existence of $X(\mathcal{D})$ depends on the existence of multiplicative inverse of $g_D(\mathcal{D})$. From the theories of finite field this inverse exists if and only if $g_D(\mathcal{D})$ is prime to $(1+\mathcal{D}^N)$.

A similar equation for the encoder's output y_n can be written as follows:

$$y(\mathcal{D}) \equiv g_N(\mathcal{D})X(\mathcal{D}) \pmod{1+\mathcal{D}^N} \tag{3.26}$$

Equations (3.25) and (3.26) completely define the encoder's systematic and parity outputs respectively. These equation may be combined into a single polynomial generator matrix form:

$$u(\mathcal{D})\begin{pmatrix} 1 & \frac{g_N(\mathcal{D})}{g_D(\mathcal{D})} \end{pmatrix} \equiv X(\mathcal{D})\begin{pmatrix} \mathcal{D}^{-m}g_D(\mathcal{D}) & g_N(\mathcal{D}) \end{pmatrix} \pmod{1+\mathcal{D}^N} \tag{3.27}$$

In order to show the relationship between the encoder with and without feedback, one can writes a similar equations for the equivalent feedforward encoder. Assume an encoder with polynomials $g_D(\mathcal{D})$ and $g_N(\mathcal{D})$ that generate its first and second outputs respectively. Then we have:

$$\begin{pmatrix} y^{\{1\}}(\mathcal{D}) & y^{\{2\}}(\mathcal{D}) \end{pmatrix} \equiv u(\mathcal{D})\begin{pmatrix} g_D(\mathcal{D}) & g_N(\mathcal{D}) \end{pmatrix} \pmod{1+\mathcal{D}^N} \tag{3.28}$$

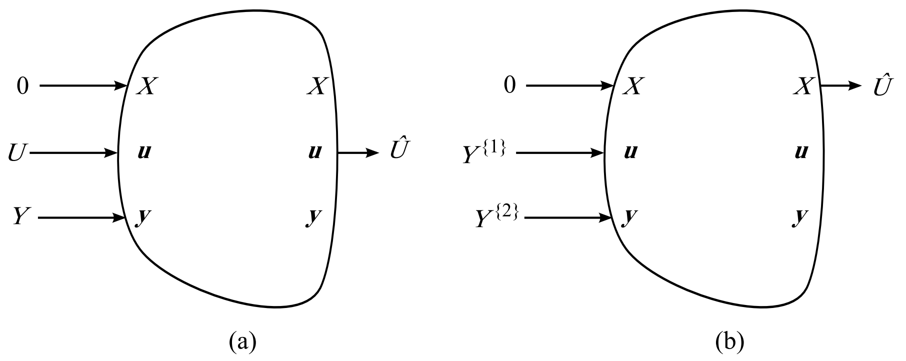


Figure 3.10: Two different connectivities for a unique decoder applicable to RSC and non-RSC tail-biting equivalent codes

Left side of equation (3.27) are the systematic and parity output of feedback encoder. Equivalently the left side of (3.28) gives two non-systematic outputs of feedforward encoder. Comparison the right sides of these equation reveals that $X(\mathcal{D})$ in feedback encoder plays the same rule as $u(\mathcal{D})$ in feedforward encoder. The result is unique block for two family codes as depicted in figure (3.10). Given a network dedicated for decoding of RSC codes as in figure (3.10)(a), one can employ it to decode non-RSC codes by exchanging u and X outputs (figure (3.10)(b)).

3.7.4 Binary Graph for Tailbiting Convolutional Codes

We can draw a graph from a parity check matrix and thus we rewrite the polynomial equations in matrix form to obtain the equivalent parity check matrix:

$$\begin{pmatrix} I_N \mid G_D \end{pmatrix} \begin{pmatrix} U \\ X \end{pmatrix} = 0 \tag{3.29}$$

$$\begin{pmatrix} I_N \mid G_N \end{pmatrix} \begin{pmatrix} Y \\ X \end{pmatrix} = 0 \tag{3.30}$$

where U and Y have length N and are comprised of entire information and output bits respectively. G_D is an $N \times N$ matrix comprised of the coefficients of $g_D(\mathcal{D})$ as a row vector in diagonal and zero elsewhere. Similarly G_N is constructed form $g_N(\mathcal{D})$ in this way.

We may eliminate X between equations (3.29) and (3.30), but the result will be a non-sparse matrix without a regular arrange of "one". Instead keeping X yields a sparse matrix (for large N). More importantly by keeping X the resulting graph is applicable for non-RSC codes as well as mentioned earlier. Thus we capture our graph according to the following parity check matrix:

$$\begin{pmatrix} I_N & 0 & G_D \\ 0 & I_N & G_N \end{pmatrix} \begin{pmatrix} U \\ Y \\ X \end{pmatrix} = 0 \tag{3.31}$$

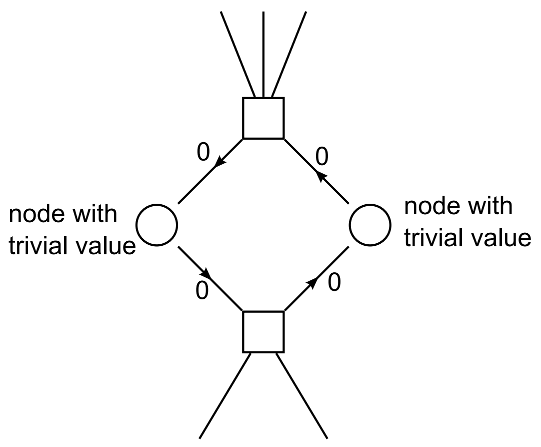


Figure 3.11: A fatal loop due to initial null values of variable nodes that need transformation to resolve the problem

Cycles in Graphs and Fatal Loop

Iterative decoding algorithms don't often converge to the optimal solution, but in a graph with adequate girth (minimum length of the cycles), the solution is completely acceptable. The challenging issue is that a graph is not unique for a given code and a better graph leads to a better result. The cycles in graph are more important in our case because a part of input values are initially zero. Figure 3.11 shows a fatal loop for which sum-product algorithm fails. The hidden nodes initially have null values (i.e. probability of 0.5). The received information from channel has nontrivial values and could update these values. But the nontrivial values combine with trivial ones in function nodes and yield again a trivial value.

Fortunately, we can easily sort it out owing to regular structure of the parity-check matrices. For example assume UMTS constituent code with generator polynomial of $(13{,}15)_{\text{oct}}$. The graph can be drawn according to equation(3.31), but it contains fatal loops. On the other hand the linear transformation of adding two rows of parity check matrix resolves the problem. A part of this graph is depicted in figure (3.12). Another graph depicted in figure (3.13) is for the codes $(1, \frac{1+\mathcal{D}2}{1+\mathcal{D}+\mathcal{D}^2})$. This graph is used in chapter 5 as case study for further simulations.

3.8 conclusions

The concept of iterative decoding on a graph has been discussed intuitively. It was shown that various labels and structures introduced in this context have no any serious difference with each others. A new global state vector was defined that leaded to a unified approach for decoding of taibiting convolutional codes. Application of this approach to the graph theory yielded interesting graph with low complexity suitable for analog realizations.

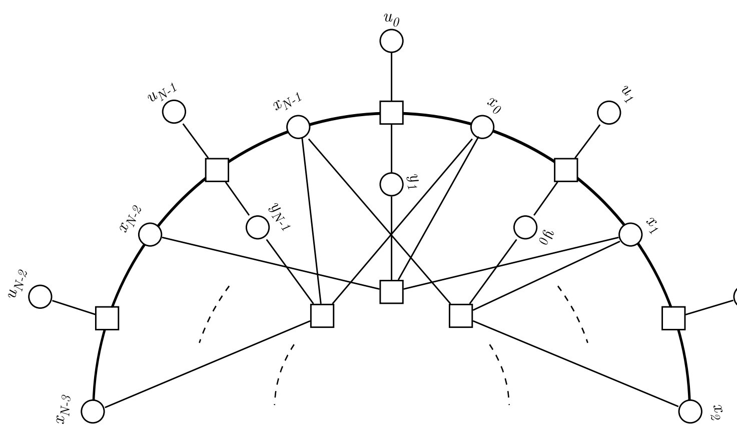


Figure 3.12: A part of binary graph of $(13{,}15)_{\text{oct}}$ tailbiting CC

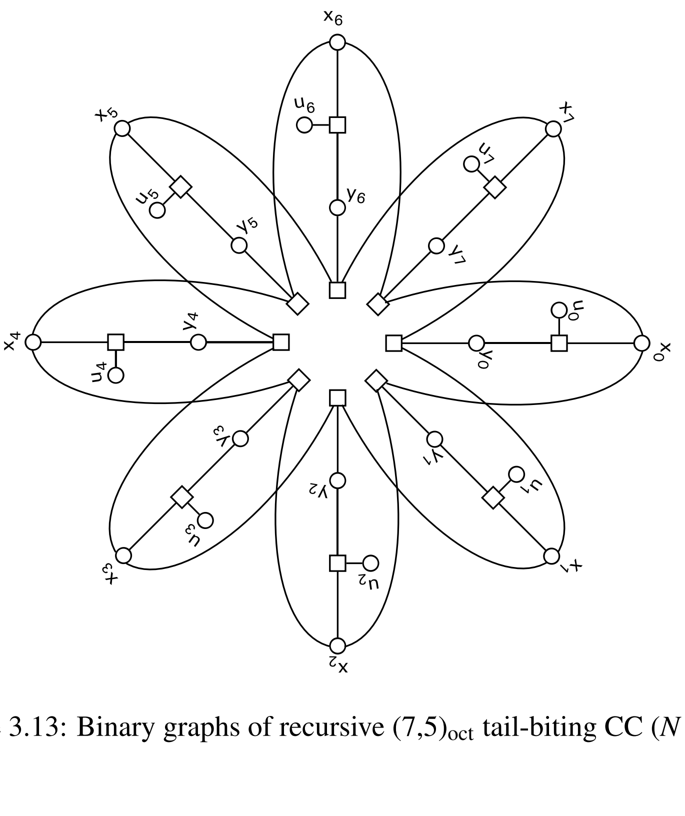


Figure 3.13: Binary graphs of recursive $(7{,}5)_{\text{oct}}$ tail-biting CC ($N = 8$).

Chapter 4

Analog decoding systems

The goal of analog decoding systems is to find a way to use an analog integrated circuit to solve the equations obtained by the decoding process. This chapter will provide some basic information about the analog implementation of decoding systems.

4.1 An introduction to analog decoding

Long before digital processors were invented, scientists had the vision to compute with analog electronic circuits, to build analog systems that could make calculations with high precision. Let us consider a complex analog circuit with lots of inputs to which we connect constant current or voltage sources. If the circuit is not unstable, it will *converge* to a stable state. When this steady state has been reached we can measure the voltage of each node or the current of each branch. In other word the circuit has *solved* a set of nonlinear equations. These equations are determined by the circuit topology and inputs. In this approach the measured currents and voltages represent the unknown parameters of the equations. The time needed for solving this set of nonlinear equations is the settling time of the circuit. The main problem is to find a circuit topology that leads to a set of equations similar to those obtained by the decoding process covered by previous chapter.

4.1.1 Motivations for analog decoding

There are a lot of interests in developing the analog decoders:

Power consumption: The decoding algorithms developed by the previous chapters are normally implemented by digital circuits like DSP or FPGA. The power consumption of these circuits is very high. A digital number of N bits needs N wires to be transmitted along the circuits. The ADC circuits are needed to interface the channel and the processor. On the other hand, analog circuits require much less power than digital processors, and they can manage numbers as analog voltages and currents using only one wire. The consuming ADC will be replaced with simple sample-and-hold circuits.

Speed: The digital implementation of decoders require a clock based process. Thousands of clock pulses are necessary to decode a single block of data. Also ADC is often a slow device that affects the total speed of the decoding algorithm. In the analog counterparts, The time needed for the solution is the converging time of the circuit, and the ADC is omitted. At the same time, because of the small circuit a very large scale parallelism is possible.

Surface: An analog decoder requires much less transistors than an FPGA or microprocessor. Depending on the desired calculation power of a processor some millions of transistors are needed, while an analog decoder require some hundreds to some thousands of transistors depending on the decoder complexity and parallelism degree.

It is important to note that there is a kind of compromising between these advantages, for example we achieve better speed at the cost of more parallelism inducing bigger circuit and more power consumption. However, analog network decoding confirms to have better combined power/speed performance than its digital counterparts (see for example [40] and the references inside).

4.1.2 History of analog decoding

The field of analog iterative decoders began with simultaneous proposals by Hagenauer and Loeliger following a suggestion by Wiberg [29]. These authors proposed using non-linear analog circuits to implement decoding operations [41, 42]. Since 1998, much effort has been spent towards turning these ideas into working chips. The first fully functional analog decoder of this type was built in 1998 by Lustenberger using discrete transistors [3]. Lustenberger then designed and manufactured first, a decoder for a four-state tail biting (18,9,5) convolutional code and later, a decoder for a (44,22,8) low density code, both in 0.8 μm BiCMOS technology, although both of them suffered some practical defects [43, 44]. Mörz et al. demonstrated a perfectly working decoder for a two state (16,8,3) tail-biting convolutional code in 0.25 μm BiCMOS technology [7].

Analog iterative decoders then emerged as a means of implementing Turbo codes and LDPC codes. The standard Viterbi algorithm is insufficient for these codes, because the decoder output is binary (i.e. hard information). A more complex algorithm, known as BCJR [45], is required for Turbo codes. The output of a BCJR decoder is soft, allowing it to be exchanged with a network of decoders for multiple rounds of decoding.

In 2001, Kschischang, Loeliger and Frey consolidated the theory of soft-information algorithms, including BCJR [34]. These algorithms were shown to be instances of a general algorithm, called the sum-product algorithm, which is described in terms of “factor graph”. It was further shown by Loeliger and Lustenberger that a large class of factor graphs could be mapped to analog circuits [42].

In 2003, Gaudet presented an analog decoder with a programmable interleaver for a (48,16) turbo code [8]. Winstead reported a decoder for a (16,11) Hamming code, as well as a decoder for a (16,11) product code in his thesis [46]. Recently, a successful implementation of a chip for decoding a (132,40) turbo code (according to the UMTS standard) was reported [5]. A more detailed history of analog decoders is given in [40] and [46].

4.2 Theoretical Aspects of Analog Decoding

In this section we are about to discuss the principal aspects leading to the analog approach for the error control decoding systems. we will start our study by a brief description of MOS transistor under the subthreshold region, then we will follow the section by introducing basic concepts of analog decoding systems such as LLR and boxplus operator. Finally we will introduce our circuitry for realizing analog decoders.

4.2.1 Subthreshold conduction

As we will see in the following sub sections, an analog decoder is made of transistors. For proper operation of the circuit, it is necessary that the transistors follow an exponential characteristics. According to basic electronic texts, BJT[1] transistors are devices with such characteristics. The problem with a BJT is that it has a non-negligible base current which will result in finite input resistance and therefore voltage drop over the input stage. Furthermore, a BJT transistor is much more expensive than a MOS transistor, because it needs more technological operations [47].

In analyzing CMOS devices, it is sometimes assumed that the device turns off abruptly as gate-source voltage V_{GS} drops below the threshold voltage V_{Th}. In reality, for $V_{GS} \approx V_{th}$ a weak inversion layer still exists and some current flows from drain to source. Even for $V_{GS} < V_{th}$, drain-source current I_D exists and it exhibits an *exponential* dependency on V_{GS}. Called *subthreshold conduction*, this effect can be formulated for V_{DS} greater than roughly 200 mV as [48]:

$$I_D = I_s \, e^{\frac{V_{GS}}{\zeta V_T}} \tag{4.1}$$

In this equation I_s denotes the transport saturation current, $\zeta > 1$ is a nonideality factor and $V_T = kT/q$ that is about 26mV in ambient temperature. We also say that the device operates in *weak inversion*. Equation 4.1 is similar to the exponential $I_C - V_{BE}$ relationship in a bipolar transistor.

Working under the subthreshold condition results in decreased transistor speed [49] [50]. As a result one can decide to increase the decoder speed at the cost

[1] Bipolar Junctional Transistor

of approaching or even entering the square root region of the MOS transistor. In this case the circuit speed increases, but on the other hand, the power consumption of the circuit will increase. As the MOS approaches the edge of the subthreshold region, bit error rate will increase slightly.

4.2.2 LLR and Boxplus operator

Refering to chapter 3, the marginal function implemented by function node in the decoding systems are of type of GF(2) sum (or binary XOR) operation. A function node gets the probability of all of its adjacent variable nodes but one, it assumes that the GF(2) sum of all of its neighbors must be 0, it calculates the *extrinsic* probability of its remaining member to be 0 or 1, and it passes this value to its remaining neighbor. Therefore, the basic operation of analog decoders is to calculate the probability of a statistical variable to be 0 or 1.

In analog decoding we do not send separate messages for a variable probability to be 0 or one. Instead a term *Log-Likelihood Ratio LLR* is used. Let us consider X to be a binary random variable which can take the values 0 or 1. The LLR value of X is noted $L(X)$ and is equal to:

$$L(X) = \ln \frac{P_X(x=0)}{P_X(x=1)} \tag{4.2}$$

It is simple to derive the following equations from equation 4.2 by noticing that $P_X(x=0) + P_X(x=1) = 1$:

$$P_X(x=0) = \frac{1}{1+e^{-L(X)}} \tag{4.3}$$

$$P_X(x=1) = \frac{e^{-L(X)}}{1+e^{-L(X)}} \tag{4.4}$$

We use the isomorphism $0 \leftrightarrow +1$ and $1 \leftrightarrow -1$. The expectation of the binary random variable X is defined as:

$$\begin{aligned} E[X] &= \lambda(X) = (+1)P_X(x=+1) + (-1)P_X(x=-1) \\ &= \frac{1-e^{-L(X)}}{1+e^{-L(X)}} = \tanh \frac{L(X)}{2} \end{aligned} \tag{4.5}$$

This expectation is referred to as the “soft bit” denoted by $\lambda(X)$ has a range of -1 to $+1$. Now let us consider three statistical variables U_1, U_2, and U_3. We will assume that U_1 and U_2 are two independent binary variables and $U_3 = U_2 \oplus U_1$, with $\oplus$ the GF(2) sum . Note that this is the marginal function used by decoders based on sum product algorithm. We can derive an expression for $L(U_3)$, the LLR value of U_3. Figure 4.1 shows that the numeric value of U_3 can be written as a real multiplication of U_1 and U_2. In fact in λ-domain, XOR can be written as a real multiplication because of the multiplication of the expectations:

$$\lambda(U_3) = \lambda(U_1) \cdot \lambda(U_2) \tag{4.6}$$

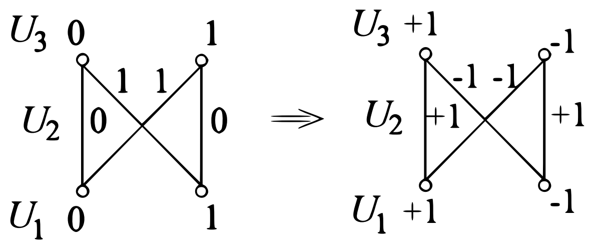
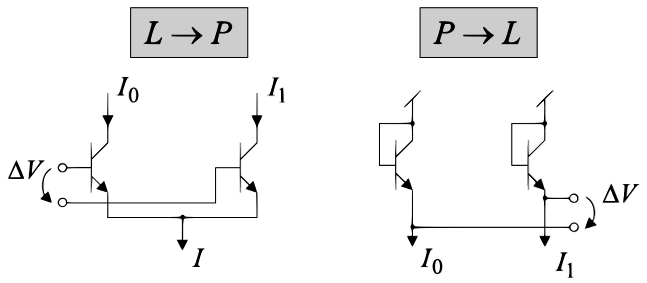


Figure 4.1: Binary XOR gate



$$\boxed{\frac{\Delta V}{V_T} = L(X) \qquad \frac{I_0}{I} = P_X(x=0) \qquad \frac{I_1}{I} = P_X(x=1)}$$

Figure 4.2: Conversion from Log-likelihoods into probabilities and vice versa

Using Equation 4.5 in Equation 4.6 we can derive :

$$\begin{aligned} L(U_3) &= 2\tanh^{-1}\left[\tanh\left(\frac{L(U_1)}{2}\right)\cdot\tanh\left(\frac{L(U_1)}{2}\right)\right] \\ &= L(U_1) \boxplus L(U_2) \end{aligned} \tag{4.7}$$

Here, $\boxplus$ denotes the *Boxplus* operator which is an abbreviation of the function implied by equation 4.7 [33]. The operation is also called *soft XOR gate* by Loeliger [51].

4.2.3 Boxplus elementary circuit

In the previous sub section we have introduced the boxplus operator. this operator is the main building block of each analog decoder. In this section we will introduce the first and simplest circuit to realize this function using analog devices.

First of all let us study a differential amplifier using bipolar transistors. Such a circuit is shown in figure 4.2 (left hand side). We know that the collector current I_C is given by equation 4.1. One can easily calculate the output currents I_0 and I_1 as follows:

$$I_0 = I\frac{1}{1+e^{-\frac{\Delta V}{V_T}}} \tag{4.8}$$

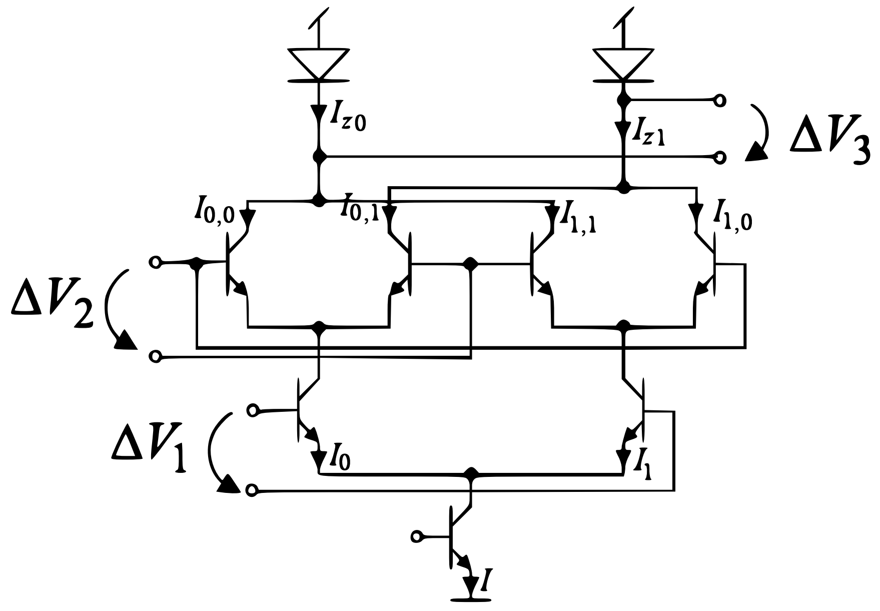


Figure 4.3: Simple boxplus circuit

$$I_1 = I\frac{e^{-\frac{\Delta V}{V_T}}}{1+e^{-\frac{\Delta V}{V_T}}} \tag{4.9}$$

Where ΔV is differential input voltage and $I = I_0 + I_1$. By comparing these equations with Equations 4.3 and 4.4 one can instantly find that I_0/I and I_1/I correspond to $P_X(x=0)$ and $P_X(x=1)$ respectively. Furthermore, $\Delta V/V_T$ equals $L(X)$. The differential output current $\Delta I = I_0 - I_1$ can be expressed from equations 4.8 and 4.9 as:

$$\Delta I = I_0 - I_1 = I\tanh\left(\frac{\Delta V}{2V_T}\right) \tag{4.10}$$

Again we can easily find the similarity between equation 4.10 and 4.5 by replacing $\Delta I/I$ by $\lambda(X)$. We can say that the circuit on the left hand side of Figure 4.2 converts the LLR value of a binary random variable to the probabilities of 0 and 1. Generally in the analog decoding systems we interpret the differential voltages as LLR's and the currents as probabilities.

The circuit on the right hand side of Figure 4.2 performs the inverse transformation from probabilities (currents) into a log-likelihood ratio (differential voltage).

Now we will turn our attention to realization of the boxplus operator. Let us analyze the circuit in Figure 4.3. Because the lower stage of the circuit is identical to the left hand side circuit of Figure 4.2, we can calculate the currents I_0 and I_1 from equations 4.8 and 4.9 just by replacing ΔV by ΔV_1. Note that the upper stage circuit contains two of such circuits, one with I_0 and other with I_1 as common (tail) currents. So we can write:

$$\begin{aligned}
I_{0,0} &= I_0 \frac{1}{1+e^{-\frac{\Delta V_2}{V_T}}} \\
&= I\left(\frac{1}{1+e^{-\frac{\Delta V_1}{V_T}}}\right)\left(\frac{1}{1+e^{-\frac{\Delta V_2}{V_T}}}\right) \qquad (4.11)\\
I_{0,1} &= I_0 \frac{e^{-\frac{\Delta V_2}{V_T}}}{1+e^{-\frac{\Delta V_2}{V_T}}} \\
&= I\left(\frac{1}{1+e^{-\frac{\Delta V_1}{V_T}}}\right)\left(\frac{e^{-\frac{\Delta V_2}{V_T}}}{1+e^{-\frac{\Delta V_2}{V_T}}}\right) \qquad (4.12)\\
I_{1,0} &= I_1 \frac{1}{1+e^{-\frac{\Delta V_2}{V_T}}} \\
&= I\left(\frac{e^{-\frac{\Delta V_1}{V_T}}}{1+e^{-\frac{\Delta V_1}{V_T}}}\right)\left(\frac{1}{1+e^{-\frac{\Delta V_2}{V_T}}}\right) \qquad (4.13)\\
I_{1,1} &= I_1 \frac{e^{-\frac{\Delta V_2}{V_T}}}{1+e^{-\frac{\Delta V_2}{V_T}}} \\
&= I\left(\frac{e^{-\frac{\Delta V_1}{V_T}}}{1+e^{-\frac{\Delta V_1}{V_T}}}\right)\left(\frac{e^{-\frac{\Delta V_2}{V_T}}}{1+e^{-\frac{\Delta V_2}{V_T}}}\right) \qquad (4.14)
\end{aligned}$$

Looking at Figure 4.3 we see that $I_{z0} = I_{0,0} + I_{1,1}$ and $I_{z1} = I_{1,0} + I_{0,1}$, therefore we can calculate I_{z0} and I_{z1}:

$$\begin{aligned}
I_{z0} &= I_{0,0} + I_{1,1} \\
&= I \frac{1+e^{-\frac{\Delta V_1 + \Delta V_2}{V_T}}}{\left(1+e^{-\frac{\Delta V_1}{V_T}}\right)\left(1+e^{-\frac{\Delta V_2}{V_T}}\right)} \qquad (4.15)\\
I_{z1} &= I_{0,1} + I_{1,0} \\
&= I \frac{e^{-\frac{\Delta V_1}{V_T}} + e^{-\frac{\Delta V_2}{V_T}}}{\left(1+e^{-\frac{\Delta V_1}{V_T}}\right)\left(1+e^{-\frac{\Delta V_2}{V_T}}\right)} \qquad (4.16)
\end{aligned}$$

The upper stage of the circuit is a probability-to-LLR converter, or current-to-voltage converter with the following equation:

$$\begin{aligned}\Delta V_3 &= V_T \ln(\frac{I_{z0}}{I_{z1}}) \\ &= V_T \ln(\frac{1+e^{-\frac{\Delta V_1+\Delta V_2}{V_T}}}{e^{-\frac{\Delta V_1}{V_T}}+e^{-\frac{\Delta V_2}{V_T}}}) \\ \Longrightarrow \frac{V_3}{V_T} &= \ln(\frac{1+e^{-\frac{\Delta V_1+\Delta V_2}{V_T}}}{e^{-\frac{\Delta V_1}{V_T}}+e^{-\frac{\Delta V_2}{V_T}}}) \qquad (4.17)\end{aligned}$$

A set of simple but long algebraic operations leads to the point that Equations 4.17 and 4.7 are identical. So the circuit in figure 4.3 is a realization for the boxplus operator as far as 4.1 is valid.

4.2.4 Boxplus circuit in our design

The boxplus circuit that was implemented in our design was the same circuit as the one in figure 4.3 with some modifications. The bipolar transistors are replaced by the MOS transistor to reduce power consumption of the circuit. The circuit is shown in figure 4.4. The differential input LLRs are applied to the $V_{x,0}$ and $V_{x,1}$ for U_1 and $V_{y,0}$ and $V_{y,1}$ for U_2. The value of $Vddd$ is chosen as to ensure $I_z = 10$nA to ensure that all of the transistors will stay in subthreshold zone, thus the MOS transistors have an exponential behavior and the circuit behaves like the circuit shown in Figure 4.3. V_n input is used to drive output common mode voltage to the level required by the transistors of next stage.

4.3 Realizing different building blocks

In this section we will explain the step by step approach followed to design different building blocs of our analog decoder. The technology in use is 0.35 μA CMOS. The approach discussed here is mainly from [38]. This also include the size of transistors in our preliminarily design. In section 5.5 the sizes become optimized in favor of getting a better performance (also c.f. to figure 5.20 that the tight result is due to the optimized dimensions for the transistors.)

4.3.1 Modularity

Another category of blocks for realizing analog decoders are *adders*. This adders are used in the variable nodes where the product of all adjacent function nodes are summed up to form the output message of these nodes. The adder circuit is much like a boxplus element. In fact we can realize an LLR adder just as we do for a boxplus element with some small changes in the manner in which we connect different blocs. Figure 4.5 shows this property. In order to obtain more modularity we have divided the box plus circuit into two parts [38]:

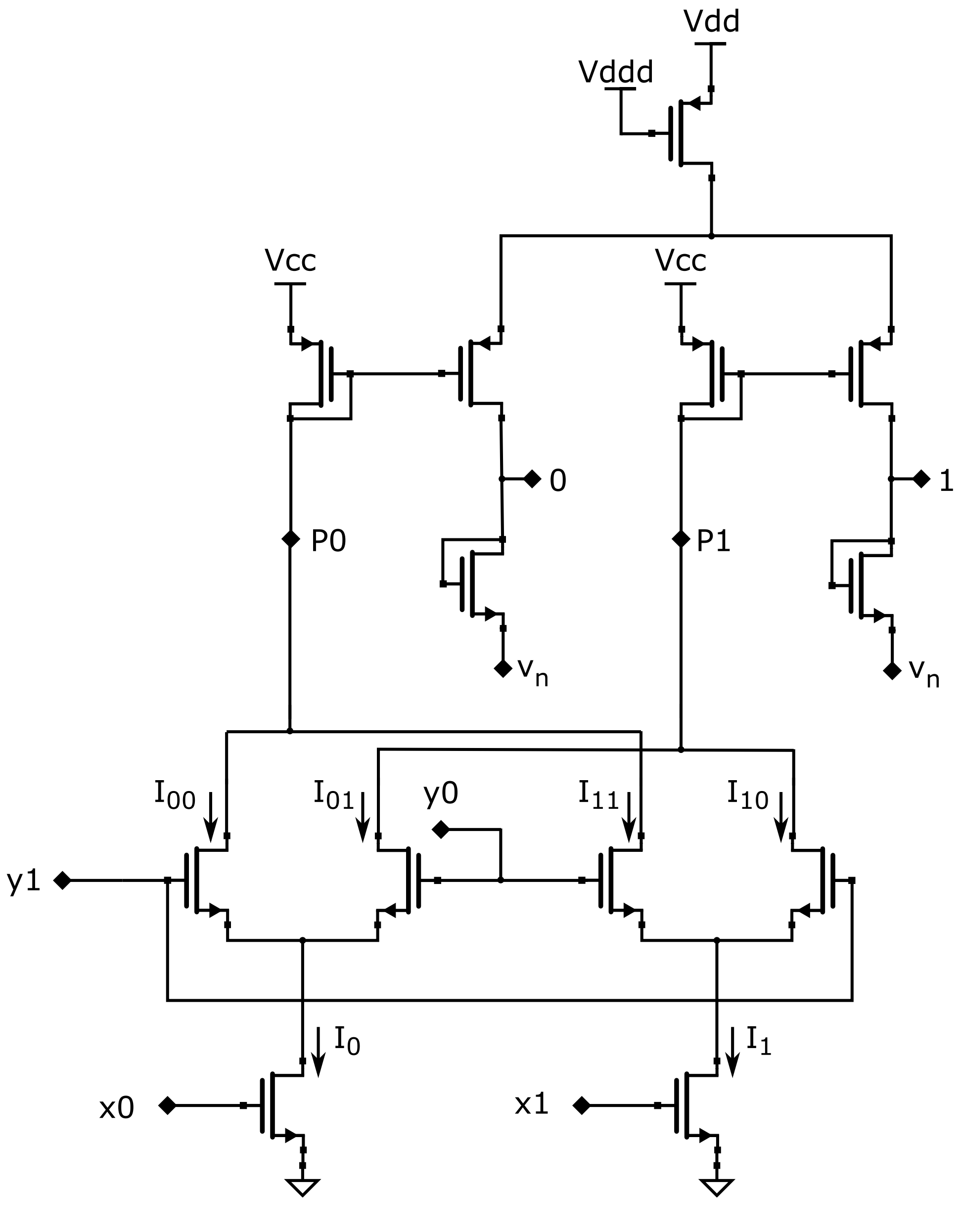


Figure 4.4: Box plus circuit using the MOS transistors

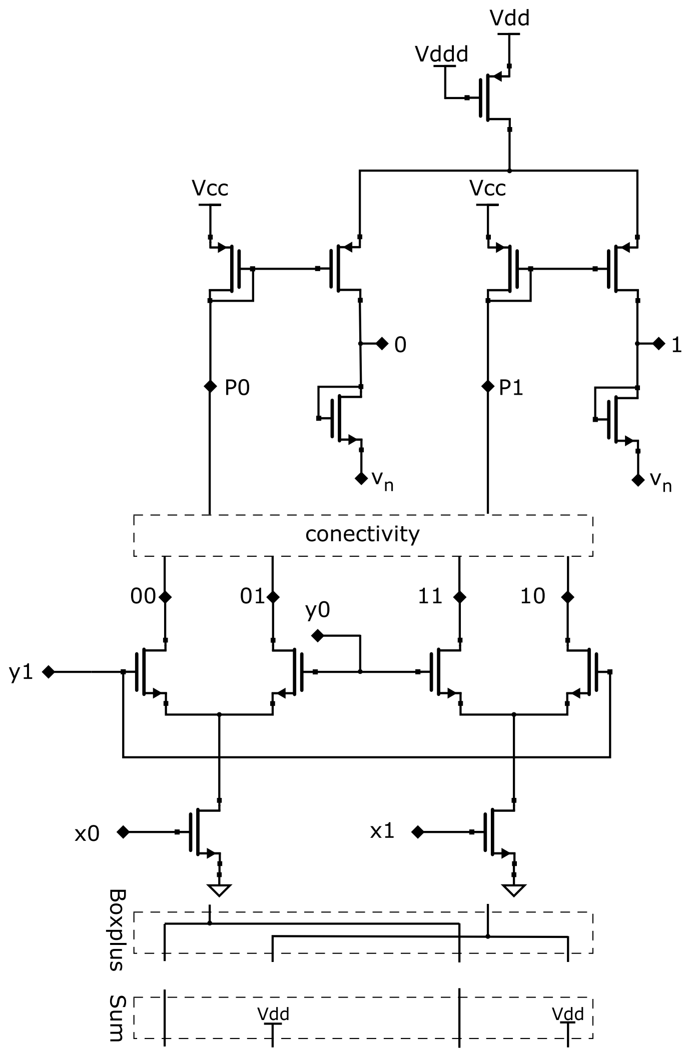


Figure 4.5: Realization of Boxplus and summing circuits

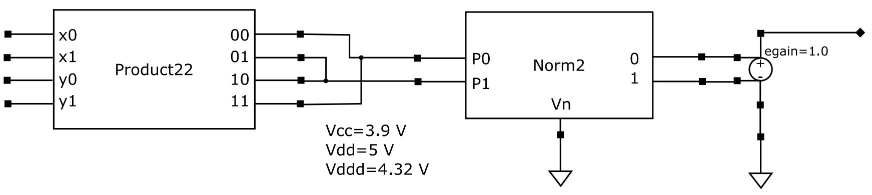


Figure 4.6: The Boxplus circuit schematic in CADENCE

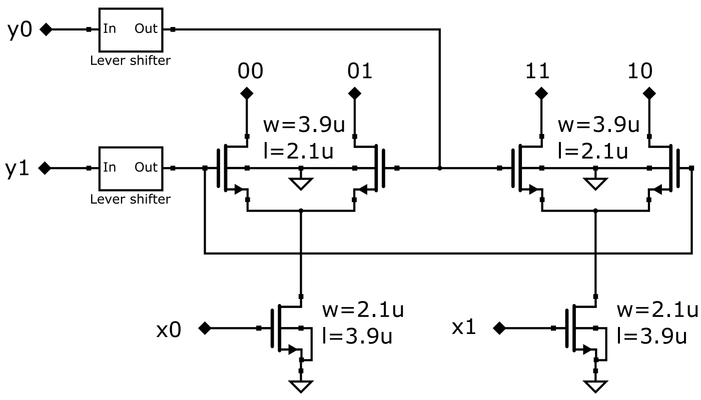


Figure 4.7: Product22 block as a part of Boxplus cell

Product22: This block is a classical multiplier, however in our application full range of signal is used for the required nonlinear characteristic.

Norm2: This block will normalize the outputs of the product22 block so that the output voltages of the block meet the common mode levels required by the next stages, and the output currents does not derive next stage transistors out of the subthreshold region.

4.3.2 Boxplus circuit in CADENCE

During this project, we have used the package CADENCE to design and simulate different blocks. The boxplus circuit implementation is shown in figures 4.6, 4.7 and 4.8. This circuit has been simulated to assure that it follows equation (4.7). The obtained results are shown in figures 4.9, 4.10 and 4.11. As we can see, the theoretical and simulated characteristics are proportional with a proportionality constant that will be calculated later.[2] This constant is ζV_T in equation 4.1.

[2]See section4.3.6

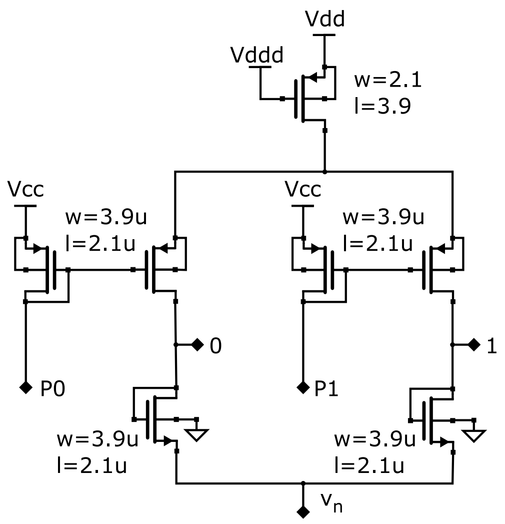


Figure 4.8: Norm2 block as a part of Boxplus cell

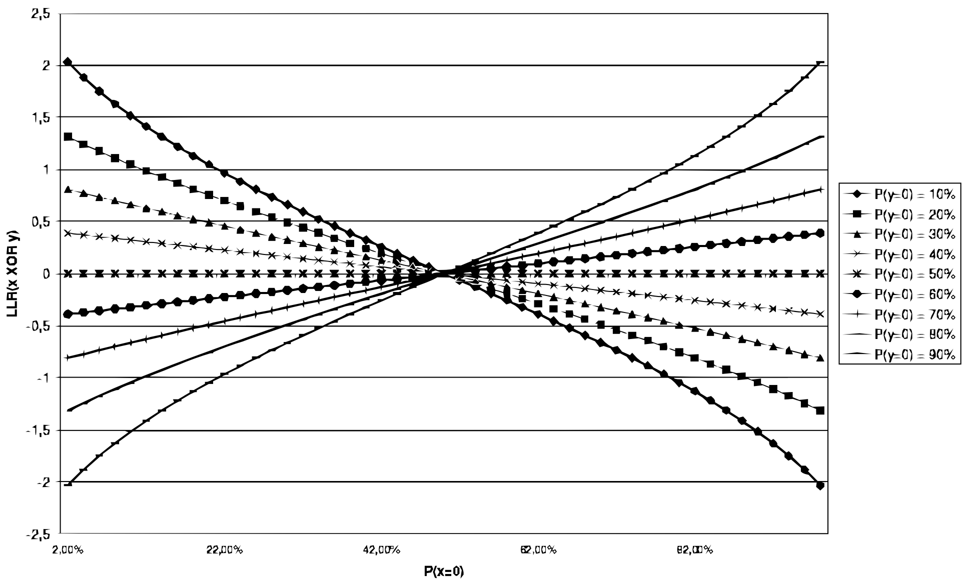


Figure 4.9: Characteristic curve of ideal Boxplus by equation 4.7

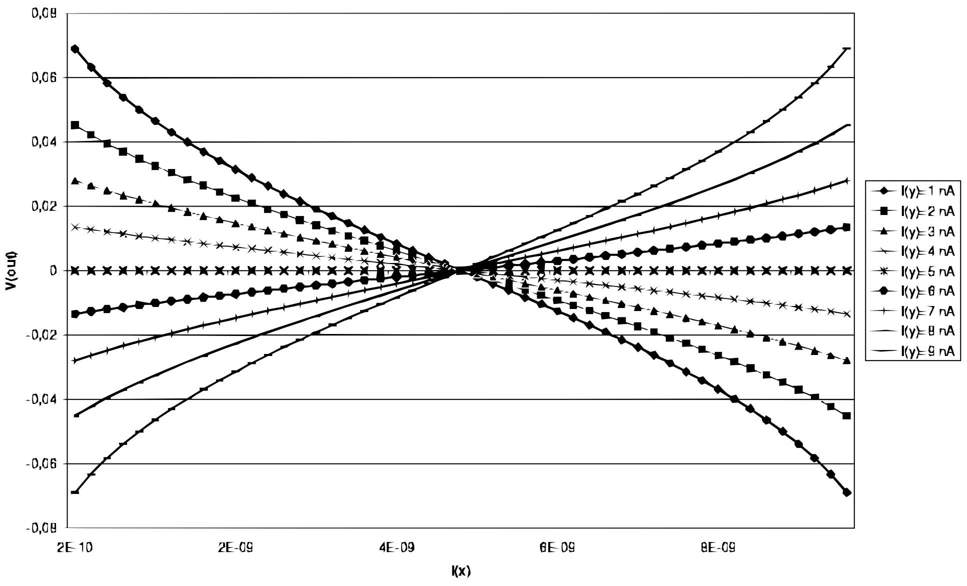


Figure 4.10: Characteristic curve of Boxplus by simulation of figure 4.6

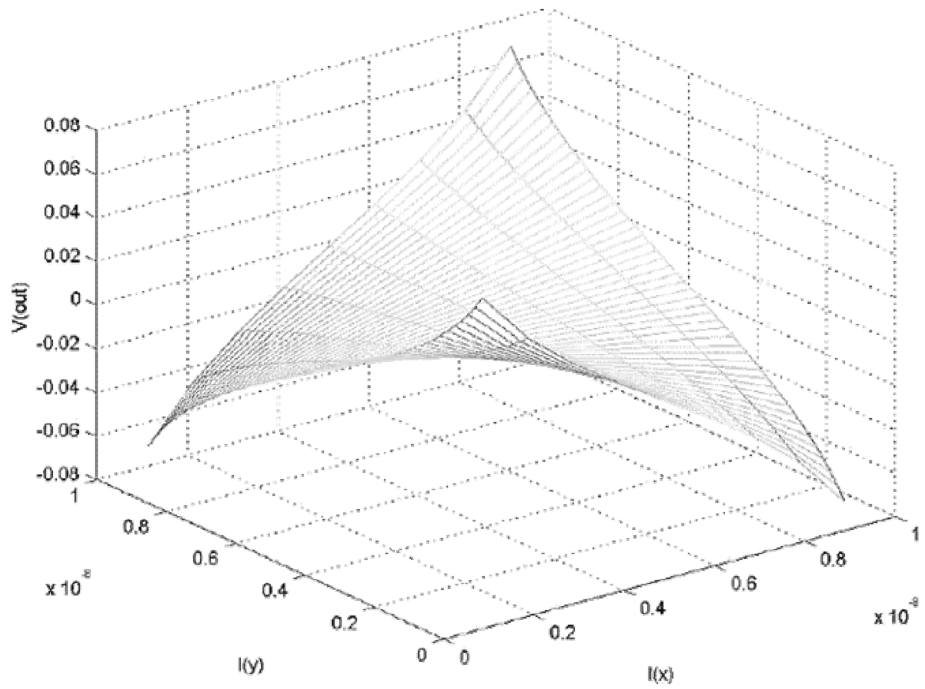


Figure 4.11: Three dimensional view of characteristic curve of the Boxplus simulation

4.3.3 Creating Three input function node

As shown in Figure 4.12, a three input box plus is comprised of 3 independent Boxplus circuits. This block will send to each of its ports the boxplus result of its two other ports. Figure 4.13 shows the design circuit in CADENCE. Note that this block is a three input function node, it takes the messages (differential voltages meaning LLR's) from two of its ports, and send the result as an LLR value (differential voltage) to its remaining input. Each port is defined by 4 wires, two pair of differential voltages, one of which serves as an input to the circuit while the other one sends out the output. This is because there are messages passing in two sides, from and to the circuit, while a line cannot act as an input and output at the same time.

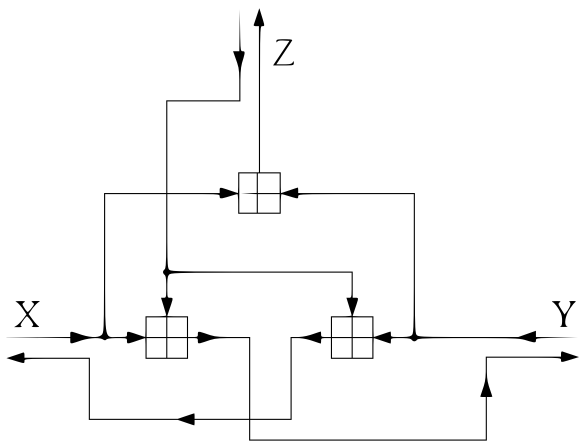


Figure 4.12: Three input boxplus

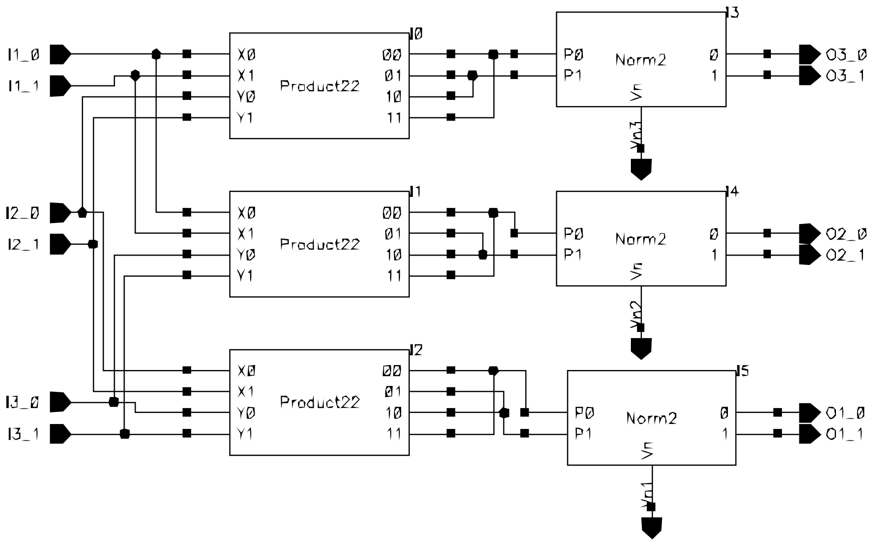


Figure 4.13: Schema of three-input boxplus element

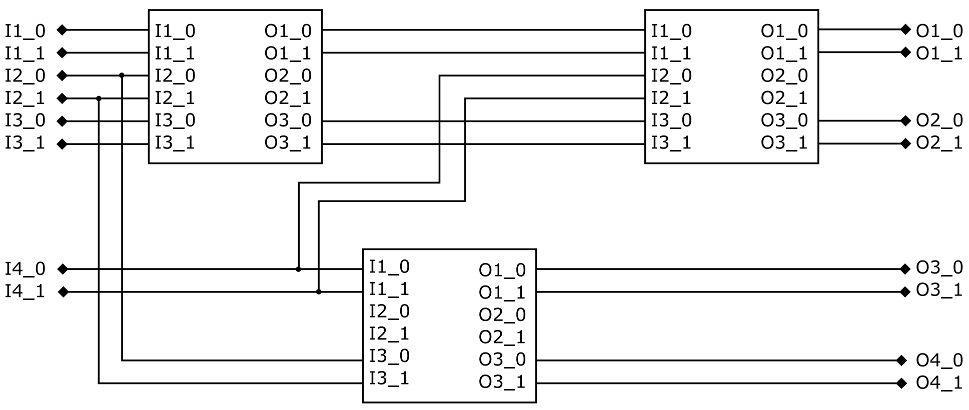


Figure 4.14: Four-input function node by using tree-input function node

4.3.4 Creating four-input function nodes

From Equation 4.7 it is easy to derive third order LLR operation:

$$\begin{aligned}(L(U_1) \boxplus L(U_2)) \boxplus L(U_3) &= L(U_1) \boxplus (L(U_2) \boxplus L(U_3)) \\ &= L(U_1) \boxplus L(U_2) \boxplus L(U_3) \qquad (4.18)\end{aligned}$$

As a result we can use three blocks of three-input boxplus to form a four-input buxplus or function node as shown in 4.14. Following the same method we can make function nodes with 5 and more inputs.

4.3.5 Creating variable nodes

A variable node is simply a number of adders. It passes to each of its ports the sum of the values passed by its other ports. A variable node, like a function node, has four lines to interface with external blocks. One pair of differential inputs, and one pair of differential outputs. In the decoding applications we can characterize a variable node by:

$$Out_i = \sum_{j \neq i} In_j \qquad (4.19)$$

The one-input, two-input and three-input variable nodes are shown in the figures 4.15, 4.16 and 4.17 respectively. In Figure 4.15, the PI input is LLR at the output of the channel. I1 is the dynamic input, which is the message that comes from adjacent function node. The O output is the dynamic output that goes to the adjacent function node. For the case where there is only one input, it simply reflects the channel information. Finally, FO is the final output which is the sum of the dynamic input and the channel information. The norm2 block assures that the channel information meet the circuit requirements. In Figure 4.16, the O1 dynamic output is the sum of I2 dynamic input and PI, the channel information. Of the same way

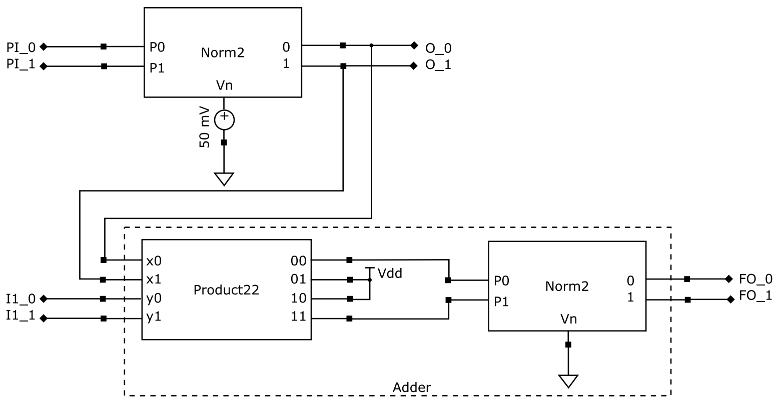


Figure 4.15: One-input variable node

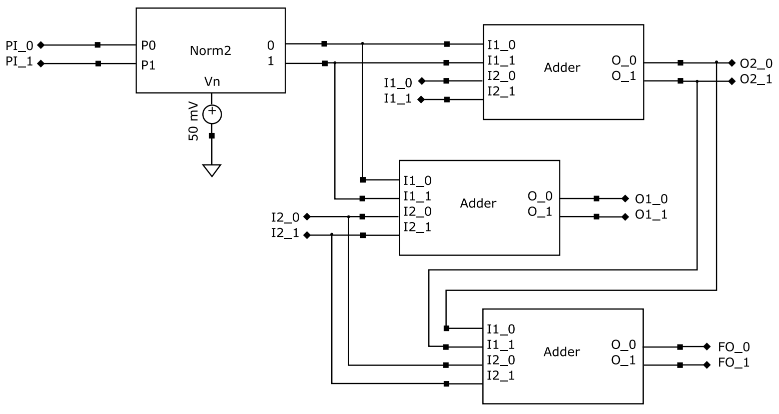


Figure 4.16: Two-input variable node

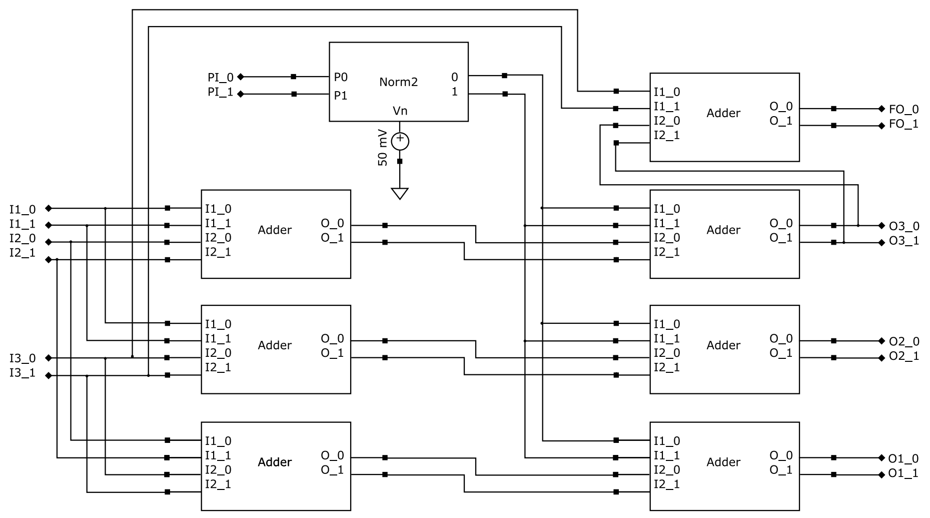


Figure 4.17: Three-input variable node

the O2 output is the I1 input plus channel information. FO will be used as the final decision criteria and is the sum of all dynamic inputs plus the channel information. We can generalize these circuits to obtain variable nodes with more ports.

4.3.6 Other elements and blocks

In this section we will introduce some circuit interfacing facilities that induce more simplicity as well as designing some necessary blocks to realizable the decoder.

Using Bus in schematic

All of the inputs and outputs used in the circuits are differential. At the same time, all of the connections between a function node and a variable node are bi-directional, which means that a pair of wires is necessary for each direction. Thus every edges in a factor graph need four wires to be implemented, very soon the circuit will be covered by wires which makes the debugging very difficult if not impossible.

CADENCE makes it possible to use buses or *wide wires* to represent a number of wires called *narrow wire*. We have changed the interface between the circuits from wires to buses containing each one four wires. In this way the schematics are much more understandable and they look like the original factor graph.

Connecting the channel to the circuit

In this subsection we will interface the output of a noisy AWGN channel to our circuit. We know that the output of channel is a random variable whose expectation is $+1$ or -1 depending on transmitted bit and whose variance is $\sigma_n^2 = N_0/2$. The

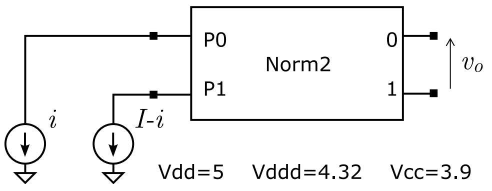


Figure 4.18: Circuitry to determine ζV_T

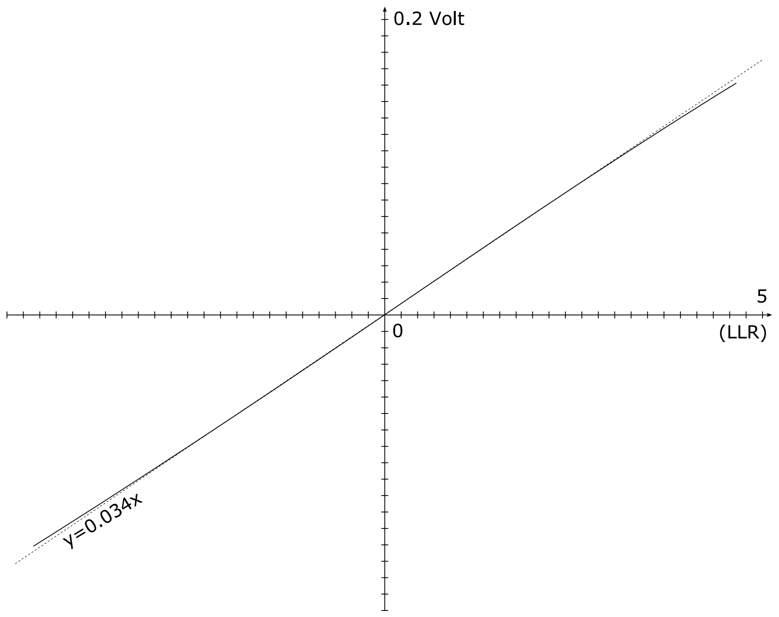


Figure 4.19: Determining the factor ζV_T by linear regression

circuit's input needs to be the LLR value multiplied by ζV_T. Let us calculate the LLR value of the random variable X at the output of an AWGN channel in terms of N_0. Using the equations of Gaussian noise probability density function and equation 4.2 we can write:

$$\begin{aligned} L(X) &= \ln\left(\frac{p(x=1)}{p(x=-1)}\right) = \frac{\frac{1}{\sqrt{2\pi\sigma_n^2}}e^{-\frac{(x-1)^2}{2\sigma_n^2}}}{\frac{1}{\sqrt{2\pi\sigma_n^2}}e^{-\frac{(x+1)^2}{2\sigma_n^2}}} \\ &= \ln(e^{\frac{x}{\sigma_n^2}}) = \frac{2x}{\sigma_n^2} = \frac{4x}{N_0} \end{aligned} \tag{4.20}$$

Equation 4.20 shows that knowing the one-sided power spectral density N_0, we can easily convert the channel output to the LLR value. We use the circuit in figure 4.18 to measure the factor ζV_T. The resulting curve is shown in figure 4.19. As we can see in this figure, the circuit operates quit linear in a wide range of LLR values, and the proportionality constant ζV_T is about 34 mV in our case.

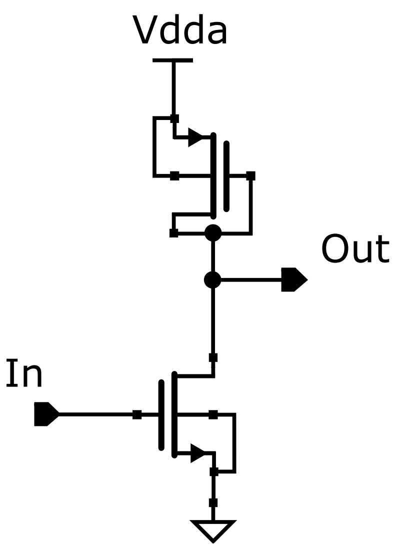


Figure 4.20: Level shifter circuit

Level shifter

The amplifiers of the two stages in the boxplus circuit need different common mode voltages. Some designers [38] have provided their norm2 block with two different set of outputs, the first one being suitable for the lower transistors and the second for the upper ones. The user will choose between these outputs, and selects the appropriate output according to the needs of the next block.

We have chosen another approach by adding a level shifter in the input of our product22 circuit to shift the input voltage level up to the level required by transistors. The level shifter block is shown in figure 4.20.

4.4 conclusions

It has been shown that a subtle change in a Gilbert's multiplier can easily realizes the necessary blocks in a graph providing that they are regarded as large signal circuits. In such approach, log-likelihood ratio (LLR) is represented by differential voltage and probability is represented by current in the circuit. Finally it is shown that given the primitive blocks, say third-order function and variable nodes, we can develop more complex nodes for realization of any given graph. This issue was discussed based on our approach in Cadence.

Chapter 5

Case Study and Simulations

In the previous chapter we have constructed different building blocks needed to implement analog decoders. In this chapter we will explore to design and simulate real decoders and verify the efficiency of our analog decoders.

5.1 Extended Hamming (8,4) decoder

As the first approach to analog decoding we have decided to implement a simple decoder, with a limited number of nodes and a straightforward theory. The (8,4) extended Hamming code has chosen for this propose. The corresponding Tanner graph is shown in figure 5.1.

5.1.1 Realizing extended Hamming (8,4) code

We can use our building blocks design in the previous chapter to implement the factor graph shown in Figure 5.1. As we can see in this figure we need the following blocks to realize the Tanner graph of extended Hamming (8,4) decoder:

- Four-input function nodes (f_1-f_4)
- One-input variable nodes (x_1, x_5)
- Two-input variable nodes (x_2, x_4, x_6, x_8)
- Three-input variable nodes (x_3,x_7)
- Eight inputs to inject channel LLR values to the circuit.
- Eight outputs to display a posterior LLR values

Figure 5.2 shows the schematic designed in CADENCE for this decoder.

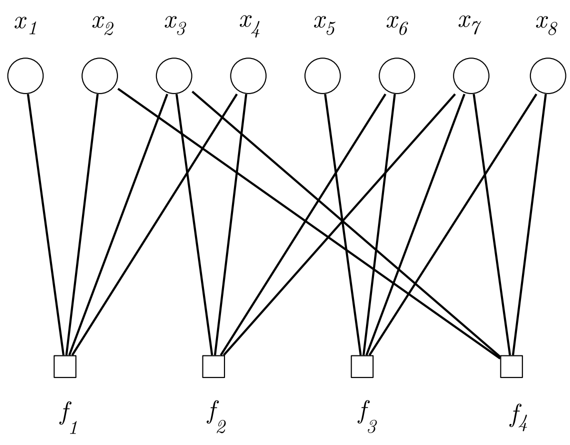


Figure 5.1: Tanner graph of (8,4) extended Hamming code

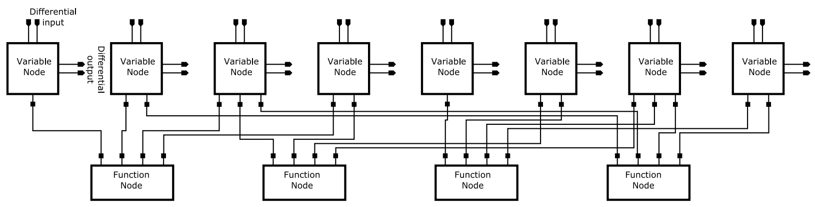


Figure 5.2: Hamming (8,4) decoder in CADENCE

5.1.2 Results of simulations

DC mode simulations

We have simulated the circuit with SPECTRE in DC mode to verify the schematic diagram and to test that the circuit does converge to the desired codeword. We know that each row of the parity-check matrix is a valid codeword, so we have injected the a prior probabilities according to one of the codewords and made sure that the a posterior probabilities converge to the selected codeword.

After assuring about the schematic, the next step was to verify that the Hamming code dose correct a single error. According to table 5.1 we have selected a codeword, changed one of the bits, and simulated the circuit in DC mode and observed that the error has been corrected. The input voltage of $\pm$75mV in our design corresponds to a certainty of 90%.

Transient mode analysis

It is interesting to perform a transient simulation and to see how the circuit reacts to a corrupted channel value. We have replaced the DC voltage sources by step voltage sources. In this case we have entered the invalid word "11111000" in lieu of valid codeword "11110000". The simulation result is depicted in figure 5.3. We can see that after about 5μs the decoder starts to correct the error (fifth bit) and after

Bit	**Selected word**	**Erred word**	**Input voltage(mV)**	**Output voltage(mV)**	**Hard decision**
1	0	0	+74.7	+233	0
2	0	0	+74.7	+181	0
3	1	1	-74.7	-218	1
4	**1**	**0**	+74.7	-115	**1**
5	0	0	+74.7	+226	0
6	1	1	-74.7	-190	1
7	1	1	-74.7	-138	1
8	0	0	+74.7	+194	0

Table 5.1: DC simulation results for extended Hamming decoder

25μs the outputs represent the expected valid codeword. In addition, before the fifth output converges to its correct level, the other outputs settled down to a lower voltages with respect to their final values. At about $t \approx 25\mu$s a second transient starts just when the fifth output polarity has changed. This make the other outputs to reach to a higher level of voltages. So we may argue that the inherent (nonlinear) feedback of the analog decoder circuit is destructive for invalid codewords and constructive for valid codewords.

5.2 Tail biting (7,5) convolutional decoder

Now we will turn our attention to the analog implementation of the tailbiting convolutional (7,5) code considered earlier in section 3.7 and represented by the graph in figure 3.13. The graph is repeated in figure 5.4 for convenience. Convolutional codes may be employed for coding of a large sequence of information bits. As a result the factor graph of such a code will be a very huge network and impossible to be implemented by analog circuits. Here the information sequence of size 8 is selected, thus the coded message will be of length 16. The result is a (16,8,5) code.

5.2.1 Circuit representation

We have considered the graph in figure 5.4 and designed the necessary CMOS circuit. The decoder is highly modular comprised of eight identical branches. Realization details of the branch is drawn in figure 5.5. For each branch there are two function nodes of order three, two variable nodes of order three (for x and y nodes) and a simple leaf node (for u). The complete decoder is realized hierarchically by simply replication of this branch as depicted in figure 5.6. As stated in section 3.7, this graph can decode RSC and non-RSC codes. We have run the simulations for RSC and non-RSC codes and we obtained exactly the similar results. It was demonstrated in figure 3.10 that decoding of two family codes differs only

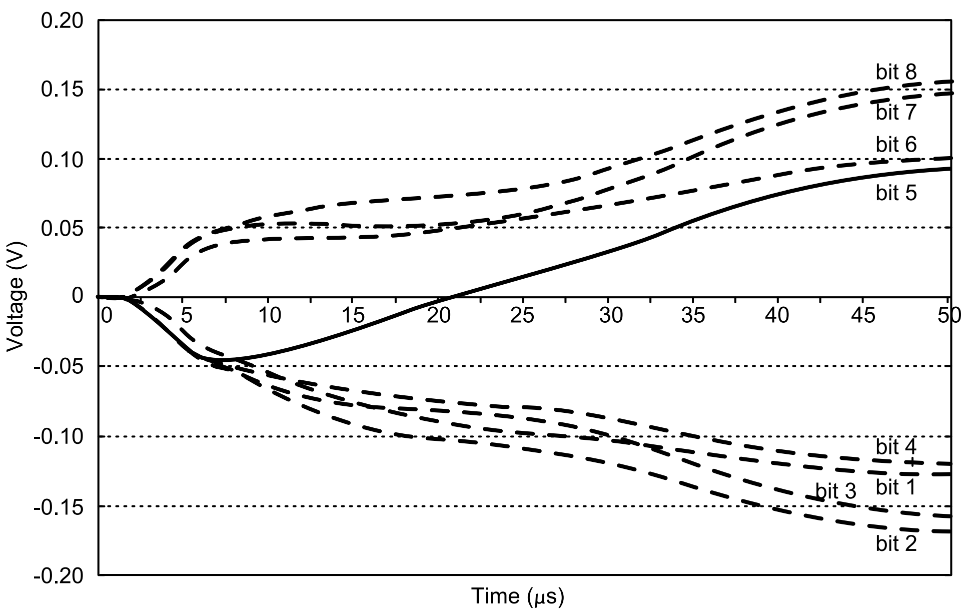


Figure 5.3: An instance of transient response of (8,4) extended Hamming code

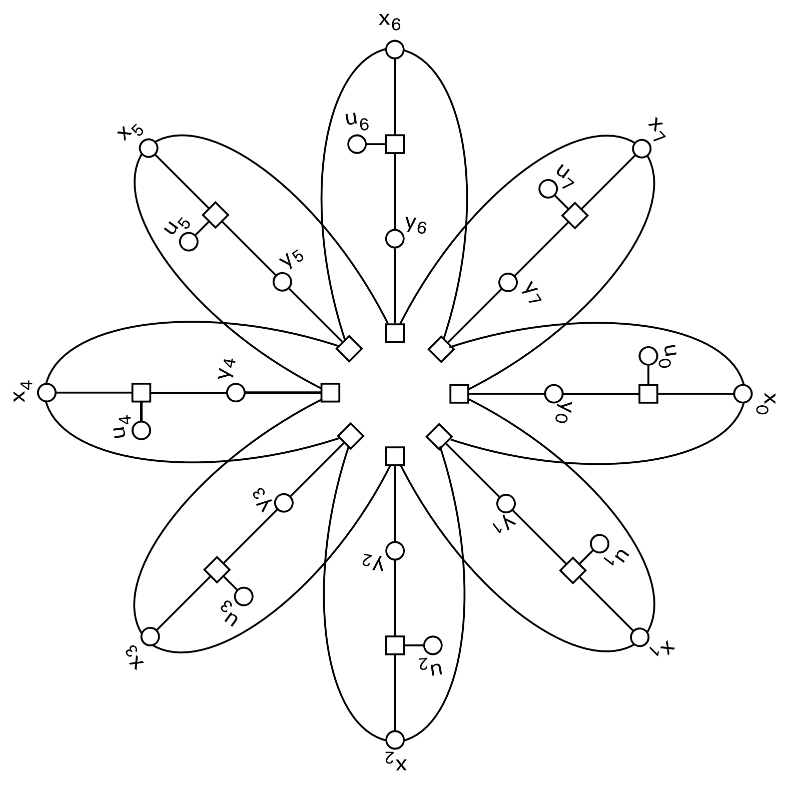


Figure 5.4: Graph representation of (7,5) tail-biting convolutional code for a frame length of eight

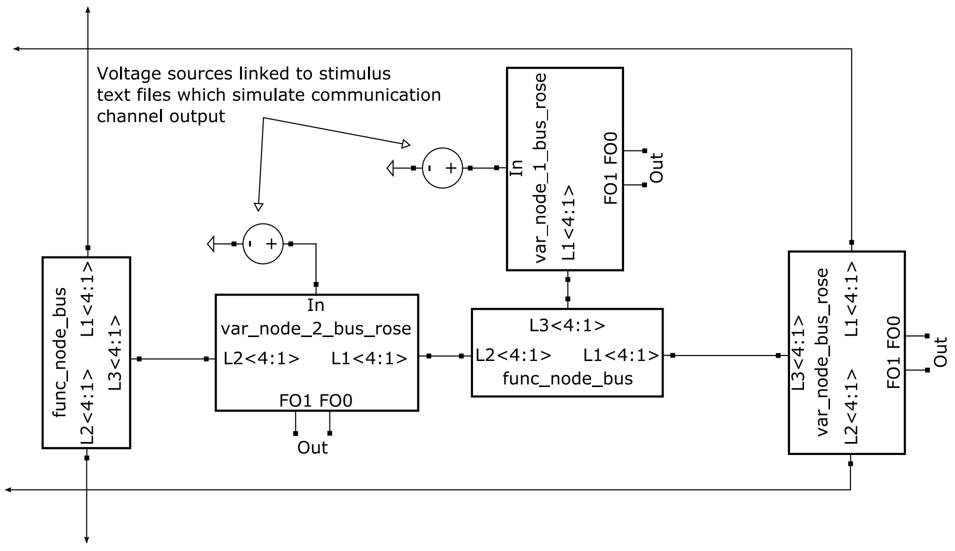


Figure 5.5: One branch of analog implementation of the (16,8,5) decoder

in input-output connectivities. Consequently, both of connectivities shows a similar performance because all node voltages of the graph either converge to a valid codeword or not and difference in labeling of nodes have no effect on the circuit.

Consider the graph in figure 5.4. For RSC code, u nodes are designated for systematic bits and y nodes for parities while the outputs are taken from u nodes. On the other hand for non-RSC code, u and y nodes are used for two parities while the outputs are taken from x nodes. In both cases the nodes x leave with zero input LLR.

Complete decoder consists of 16 differential inputs and 8 differential outputs. The decoder input signal was built off-line from a randomly generated stream of several thousands of information vectors. The encoded information bits were then corrupted by an AWGN source and scaled by the calculated proportionality constant ζV_T and then stored in a file. SPECTRE uses this file as stimulus to perform a nearly realistic BER estimation.

5.2.2 Transient responce

The circuit transient response is given by figures 5.7 in which a large time scale is selected to better represent the convergence trend. On the other hand figure 5.8 shows the details of this figure using a small time scale. Figure 5.8 shows that the erroneous fifth bit is detected after 2μs and the decoder started to correct it so that after 7μs it can be decoded correctly.

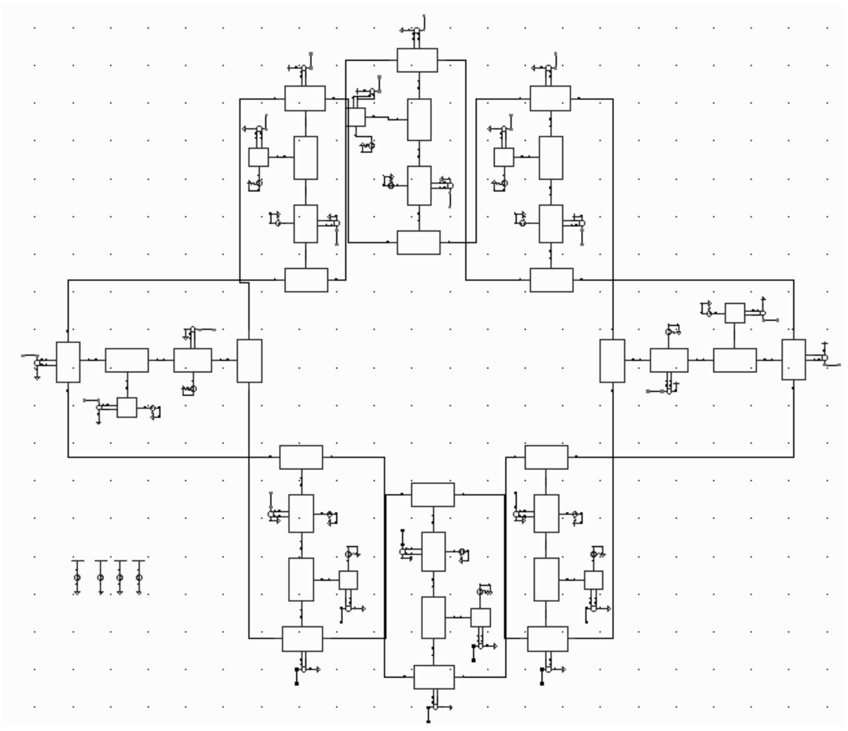


Figure 5.6: The analog (16,8,5) decoder

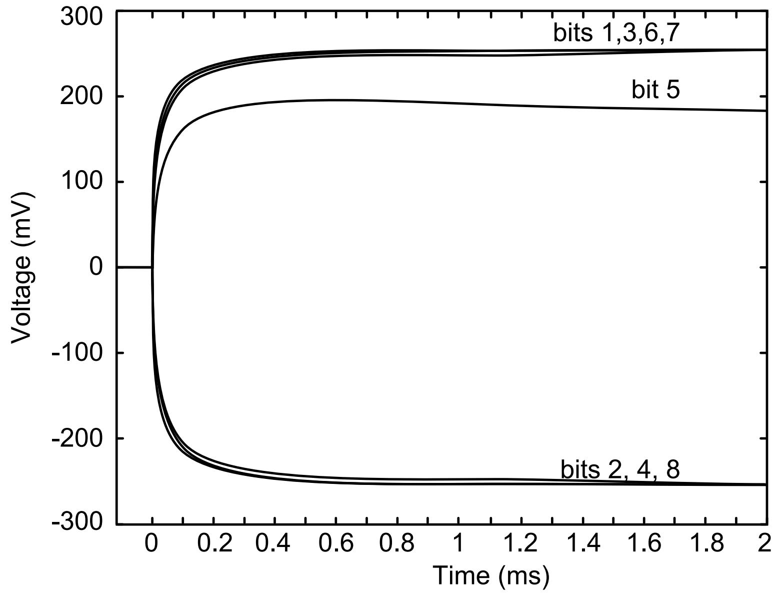


Figure 5.7: An example of transient response of the convolutional decoder

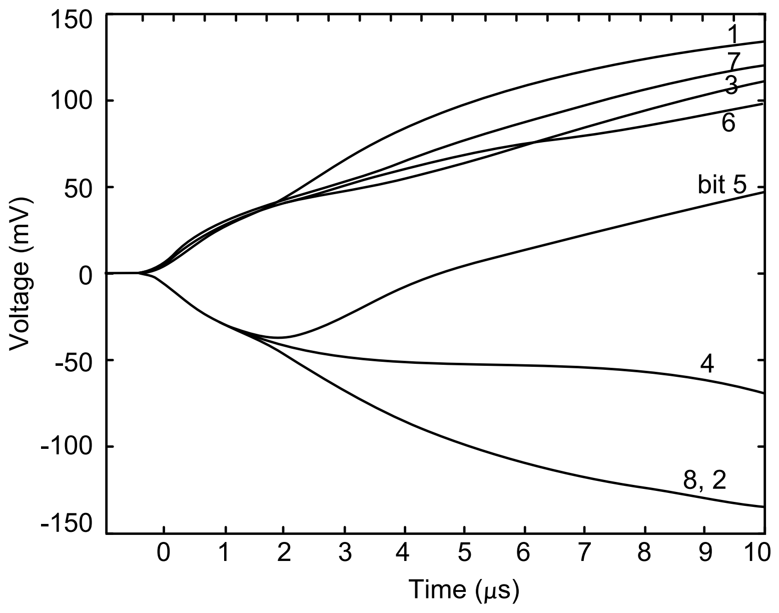


Figure 5.8: A portion of the transient response shown in figure 5.7

5.2.3 Decoder performance

We have used a stimulus input file in SPECTRE to analyze the bit error rate of our system. 16 bits are given to the decoder in parallel and after a fixed time, 8 decoder outputs are sampled and saved in an output file. Post processing of the output file is then performed to obtain the BER of the decoder. A short dead zone with zero voltages after each input allows decoder to be reset. This period discharges the parasite capacitors and lets the decoder to start from zero initial conditions for each information vector. The result is depicted in figure 5.9 for which a reference (tail) current of 10 nA has been assumed. Other curve obtained by classical BP methods are also included for comparison. Two curves follow each other tightly that demonstrates a reasonable good design. As we have mentioned, the decoder outputs are sampled after a fixed period. One important question is that how this time affect on the performance. Obviously a short sampling period degrades the performance because the decoder have a delay (but not constant) before settled down to a valid codeword. On the other hand a long sampling period sacrifices throughput and is not desirable. In order to show this effect and to compromise between them we have plotted the BER versus the sample time in figure 5.10. As we can see the BER remains almost constant after 30μs. Knowing that every codeword has 16 bits, we achieve a throughput of more than 500 kbit/sec. The parallel structure of decoder make it possible to increase this rate by increasing data frame length in exchange for more complexity.

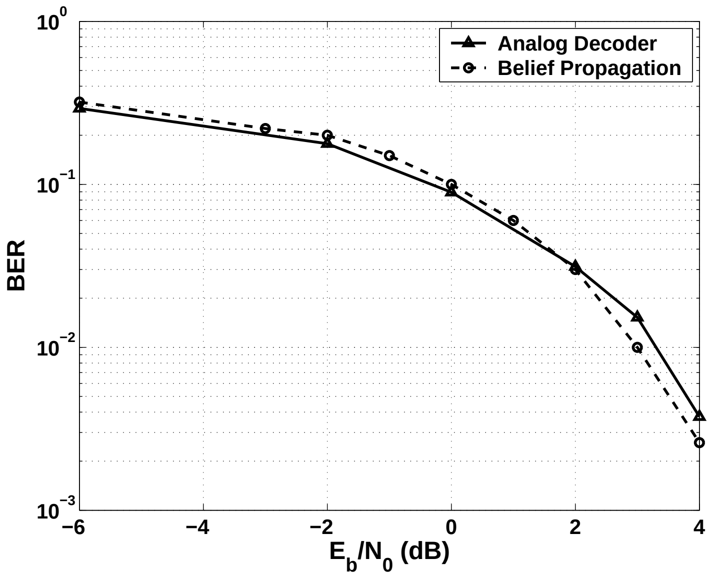


Figure 5.9: Decoder BER performance by circuit level simulation

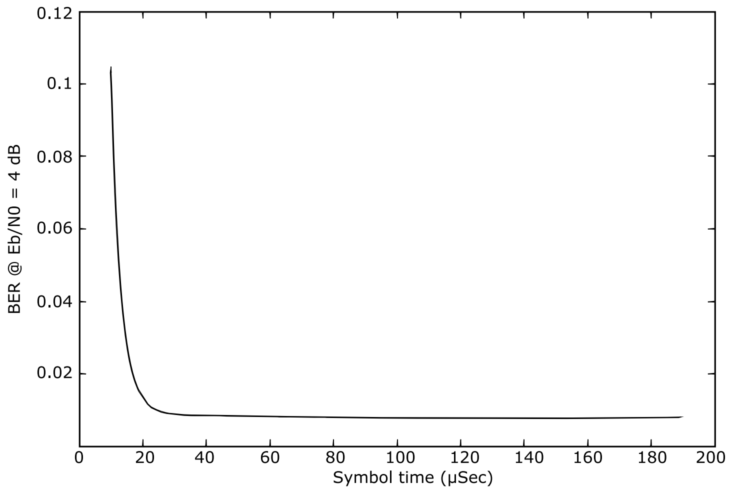


Figure 5.10: Effect of symbol time on the decoder performance

5.3 LDPC Quasi Cyclic Decoder

Tanner et al. have developed an algorithm for obtaining quasi cyclic convolutional codes using the circulant matrices [18]. Their approach provide a systematic algorithm for LDPC code construction and further to obtain LDPC convolutional codes. Here we review the gist of their algorithm and select a small code for our analog realization.

Given a prime m, the multiplicative order of nonzero element a modulo m is denoted by $O_m(a)$ and is defined as the smallest positive integer k so that:

$$a^k \equiv 1 \pmod{m} \tag{5.1}$$

The nonzero elements of GF(m)= $\{0,1,\cdots,m-1\}$ form a cyclic multiplicartive group. Given two nonzero elements a and b from GF(m) with multiplicative orders $O_m(a) = k$ and $O_m(b) = j$, a $j \times k$ matrix with element in GF(m) can be formed as follows:

$$P = \begin{pmatrix} 1 & a & a^2 & \cdots & a^{k-1} \\ b & ab & a^2b & \cdots & a^{k-1}b \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ b^{j-1} & ab^{j-1} & a^2b^{j-1} & \cdots & a^{k-1}b^{j-1} \end{pmatrix} \tag{5.2}$$

A Quasi Cyclic (QC) parity check matrix H is made of a $j \times k$ array of circulant sub-matrices as:

$$H = \begin{pmatrix} I_1 & I_a & I_{a^2} & \cdots & I_{a^{k-1}} \\ I_b & I_{ab} & I_{a^2b} & \cdots & I_{a^{k-1}b} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ I_{b^{j-1}} & I_{ab^{j-1}} & I_{a^2b^{j-1}} & \cdots & I_{a^{k-1}b^{j-1}} \end{pmatrix} \tag{5.3}$$

where I_x is an $m \times m$ identity matrix with rows cyclically shifted to left by $x-1$ positions. The resulting binary parity-check matrix is of size $jm \times km$, which means the associated code has a rate $R = 1 - \frac{j}{k}$. However the rate may be greater than this value due to linear dependence among the rows of H. By construction, every column of H contains j ones and every row contains k ones, and so H represents a (j,k) regular LDPC code. The codes constructed using this technique are quasi-cyclic with period k, i.e. cyclically shifting a codeword by k position results in another codeword. Some examples of LDPC quasi-cyclic codes constructed in this manner from a prime m are shown in table 5.2.

We have selected one of these codes for implementing in analog domain. For $m = 7$, $a = 2$ and $b = 6$ chosen from $GF(7)$, then $O_7(a) = 3$, $O_7(b) = 2$, and the parity-check matrix is:

$$H = \begin{pmatrix} I_1 & I_2 & I_4 \\ I_6 & I_5 & I_3 \end{pmatrix}_{14\times 21} \tag{5.4}$$

Replacing I_x in Equation 5.4 by 7×7 identity matrix shifted $x-1$ positions to left,

Block length	Parameters		Design rate	Actual rate	Circulant size
N	j	k	R_d	R	m
21	2	3	$1/3 = 0.3333$	0.3809	7
129	2	3	$1/3 = 0.3333$	0.3411	43
155	3	5	$2/5 = 0.4000$	0.4129	31
186	5	6	$1/6 = 0.1667$	0.1882	31
1477	3	7	$4/7 = 0.5714$	0.5727	211
3641	5	11	$6/11 = 0.5454$	0.5465	331

Table 5.2: Examples of codes constructed from prime circulant sizes

we obtain the following matrix.

$$H = \begin{pmatrix}
1&0&0&0&0&0&0&\;0&0&0&0&0&0&1&\;0&0&0&0&1&0&0\\
0&1&0&0&0&0&0&\;1&0&0&0&0&0&0&\;0&0&0&0&0&1&0\\
0&0&1&0&0&0&0&\;0&1&0&0&0&0&0&\;0&0&0&0&0&0&1\\
0&0&0&1&0&0&0&\;0&0&1&0&0&0&0&\;1&0&0&0&0&0&0\\
0&0&0&0&1&0&0&\;0&0&0&1&0&0&0&\;0&1&0&0&0&0&0\\
0&0&0&0&0&1&0&\;0&0&0&0&1&0&0&\;0&0&1&0&0&0&0\\
0&0&0&0&0&0&1&\;0&0&0&0&0&1&0&\;0&0&0&1&0&0&0\\
0&0&1&0&0&0&0&\;0&0&0&1&0&0&0&\;0&0&0&0&0&1&0\\
0&0&0&1&0&0&0&\;0&0&0&0&1&0&0&\;0&0&0&0&0&0&1\\
0&0&0&0&1&0&0&\;0&0&0&0&0&1&0&\;1&0&0&0&0&0&0\\
0&0&0&0&0&1&0&\;0&0&0&0&0&0&1&\;0&1&0&0&0&0&0\\
0&0&0&0&0&0&1&\;1&0&0&0&0&0&0&\;0&0&1&0&0&0&0\\
1&0&0&0&0&0&0&\;0&1&0&0&0&0&0&\;0&0&0&1&0&0&0\\
0&1&0&0&0&0&0&\;0&0&1&0&0&0&0&\;0&0&0&0&1&0&0
\end{pmatrix}_{14\times 21} \qquad (5.5)$$

The resulting Tanner graph has girth[1] twelve and is shown in figure 5.11. The associated code which is a (21,7) code, has minimum distance $d_{min} = 6$ and is a regular LDPC code. The best (21,8) linear code has $d_{min} = 8$. Another code, near to this characteristic, is extended Golay code (24,12) with $d_{min} = 8$. We can see that the graph has a repetitive structure. If the columns are regarded as blocks, two sort of inter-column connections are recognizable. The first category concerns the memory of the code while the second category related to cyclic properties of the code. In the graph depicted in figure 5.11 it is not easy to recognize them owing to short code length. We redraw this graph in figure 5.12 with increased code length to better show the inter-column connections. This figure contains only the connection related to code memory (i.e. those function nodes that are connected to the variable nodes in different column or time step). In this case a maximum memory 6 is obvious (e.g. f1 – v14)

Here, like our graph in figure 5.4 we are willing to have a modular structure, so that decoder expansion for larger codeword become easily possible by appending

[1]Girth is a term indicating the size of the smallest loop in the H

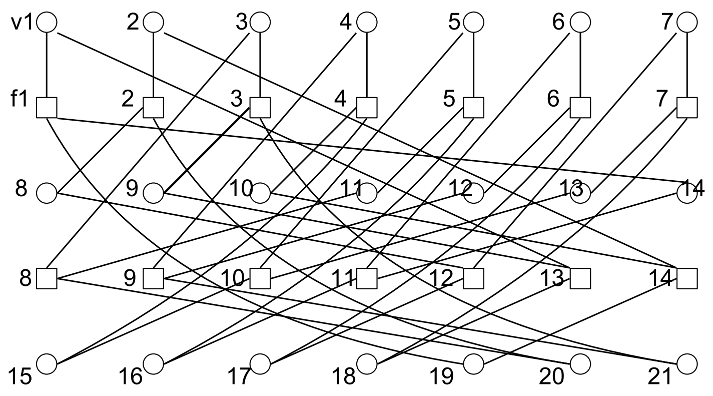


Figure 5.11: Tanner graph for a (21,8,6) QC code

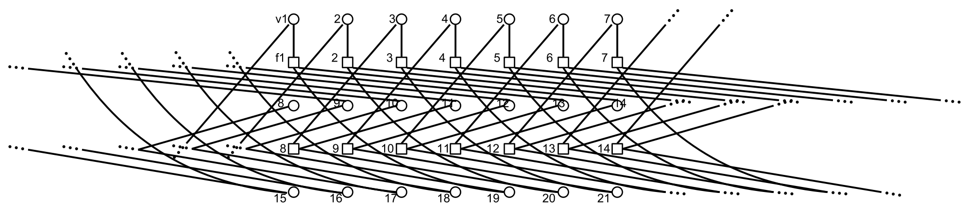


Figure 5.12: Extended version of the graph in figure 5.11 that exhibits inter-column connections

extra blocks. Toward this aim, we relabel the node in figure 5.11 in a suitable manner, so that the neighbor nodes arrange in a column. The results in the graph in figure 5.13 that corresponds to the following parity check matrix:

$$H = \begin{pmatrix}
1&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0\\
0&0&0&0&0&0&1&0&0&0&1&0&0&0&0&0&0&1&0&0&0\\
0&1&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&0&0&0&1\\
0&0&0&0&1&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&1\\
0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&0\\
0&0&1&0&0&0&0&1&0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&0&1&0\\
0&0&0&0&0&1&0&0&0&0&1&0&1&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0\\
0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&1&0&0&0&0&0\\
1&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&1&0&0\\
0&0&0&1&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0
\end{pmatrix}_{14\times 21} \tag{5.6}$$

In the following, we will see that this subtle modification, leads to a modular hierarchical structure for the decoder.

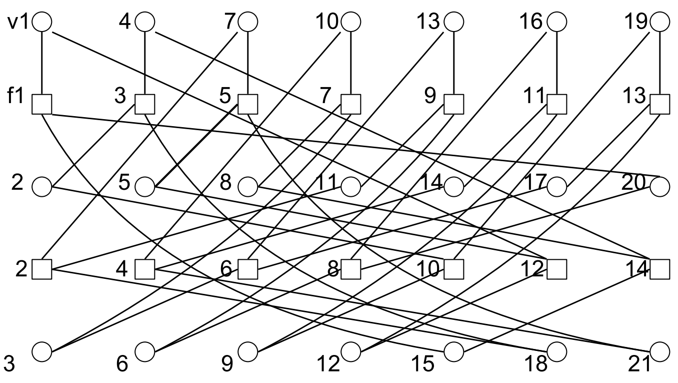


Figure 5.13: Tanner graph for the (21,8,6) QC code corresponding to matrix 5.6

Obtaining systematic full-rank code

In general, for a given H one can find many generator matrices that satisfy the code constraints. For H matrix in (5.6) we are interested in systematic code. To do this we will proceed using the method explained below.

First note that the matrix (5.6) is of rank 13, i.e. there is one parity-check equation that is linearly depended to the other equations. It means that one of the parity bits added to the information bits can be replaced by an independent information bit. So the actual rate will be $\frac{8}{21} = 0.3809$ instead of $\frac{7}{21} = 0.3333$.

1. Eliminate the last row of H in 5.6. It can be easily verified that the resulting matrix $H_{13\times21}$ has still rank 13.

2. Divide $H_{13\times21}$ in two parts H_1 and H_2 : $(H_{13\times21}) = \left(H_{1_{13\times8}} \mid H_{2_{13\times13}}\right)$

3. Write the parity-check equation for a received codeword, considering that the first 8 bits are information bits and the last 13 bits are parity bits:

$$H \cdot x = \left(H_{1_{13\times8}} \mid H_{2_{13\times13}}\right) \cdot \begin{pmatrix} u_1 \\ \vdots \\ u_8 \\ \hline v_1 \\ \vdots \\ v_{13} \end{pmatrix} = H_1 \cdot U + H_2 \cdot V = 0$$

   U and V denoting information bits and parity bits respectively.

4. Solve the above equation for V that is $V = H_2^{-1} \times H_1 \cdot U$

5. The systematic generator matrix G_{sys} is the matrix used to code information bits into codewords. It can be written as $G_{sys} = \left(I_{8\times8} \mid (H_2^{-1} \times H_1)^T\right)$

6. The G_{sys} matrix will be used to encode the information bits. When transmitted, codewords can be decoded using the same graph represented in figures 5.13.

Note that matrix H in (5.6) is not an standard systematic parity-check matrix, however, it is simple to verify that $HG_{syst}^{T} = 0$, i.e. data coded by G_{syst} can be decoded by this H that corresponds to the Tanner graph in figure 5.13.

5.3.1 Circuit representation

We can see that the Tanner graph in figure 5.13 is periodic with identical columns. Each column contains two function nodes and three variable nodes. All variable nodes have two adjacent function nodes and each of function nodes has three neighboring variable nodes. We have encapsulated one column in a distinct block to achieve better modularity for the decoder. The designed column block is shown in details in figures 5.14 and as a single block in figure 5.15. Inter-column connections are made possible by the in/out pins in this block.

Figure 5.15 shows the symbol view of the column, the upper pins are provided to inject the channel information into the circuit. The left pins are provided to establish the connections to the neighbor function nodes. The pins on right will be used to connect the circuit to the neighboring variable nodes. The bottom pins of the circuit are used to output the final LLRs. Seven blocks of shown in figure 5.15 can be merged to form a block of eight bits decoder as shown in figure 5.16. This schematic corresponds to the Tanner graph in figure 5.11. One advantage of this design is that it can be used for different message lengths. If the information message contains n bytes, the decoder will contain n such blocks, which are connected in series. Each block is connected to the next block, and the last block is connected to the first one. This configuration for the case of $n = 1$ and $n = 2$ are shown in figures 5.17 and 5.18 respectively. These codes all have the same rate and performance.

5.3.2 Transient response

The code in figure 5.17 has been simulated here. 21 channel LLRs are given to the circuit using the 21 voltages sources indicated on the top the figure. The waveform of these voltage sources are provided in text files. The final LLRs can be obtained from the voltage of 21 outputs at the bottom of the figure. Only the first 8 bits, corresponding to the information bits in our case. An instance of a transient response of the decoder is given in figure 5.19. The time axis in this figure is represented in logarithmic scale to allow a better visualization of the voltage variations. Thicker lines are the erroneous bits. These bits are corrected later by the decoder due to interaction with other parity bits. We can see that the circuit overall speed is roughly ten microseconds. Time needed for decoder to settle down depends on noise level and transistor current. We will consider this issue later in this chapter.

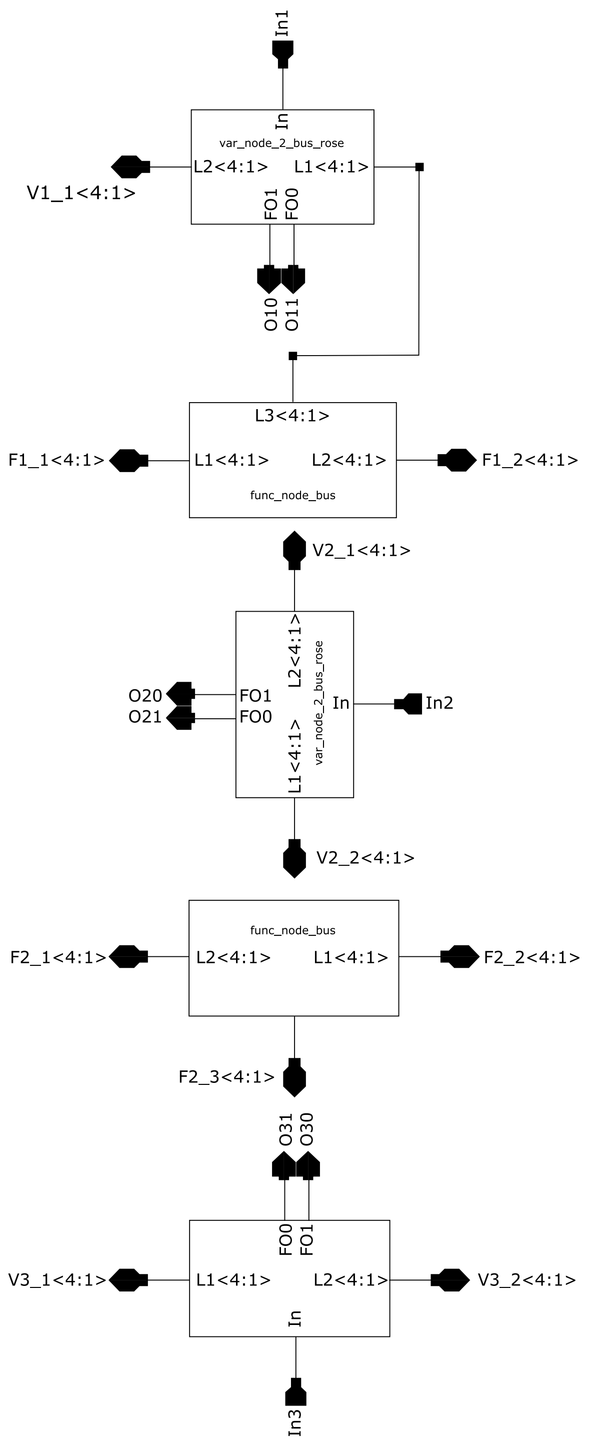


Figure 5.14: CADENCE schematic to realize one column of graph in figure 5.13. The input line are singlr ended while outputs are differential. Four-line bus connections are shown by the notation: <4:1>

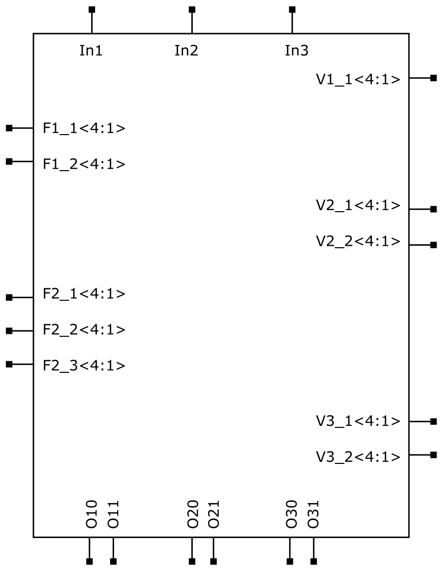


Figure 5.15: CADENCE symbol that encapsulates the schematic in figure 5.14 into a single block

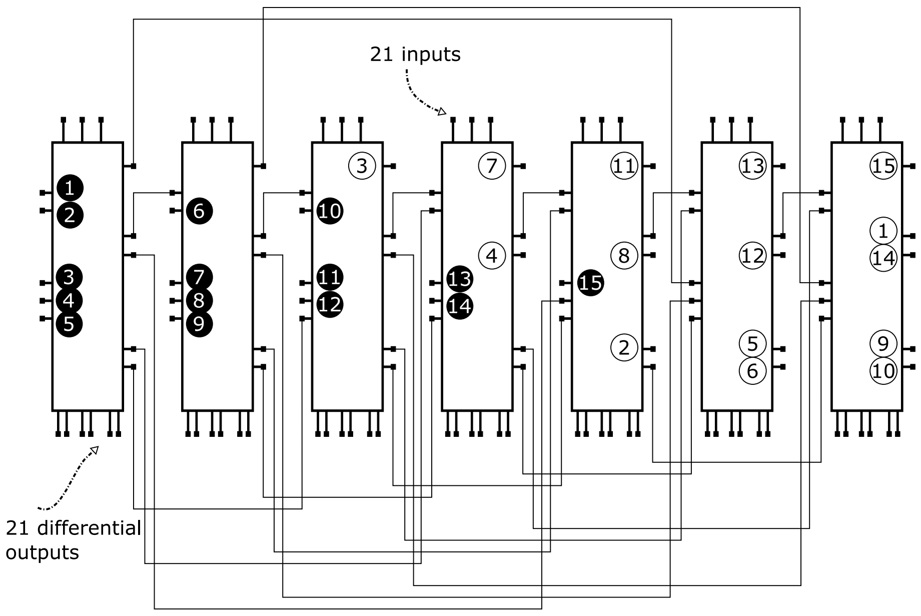


Figure 5.16: A block of LDPC decoder according to the graph in figure 5.13. Filled and empty circuls are used to better show the inter block connections (see figures 5.17 and 5.18)

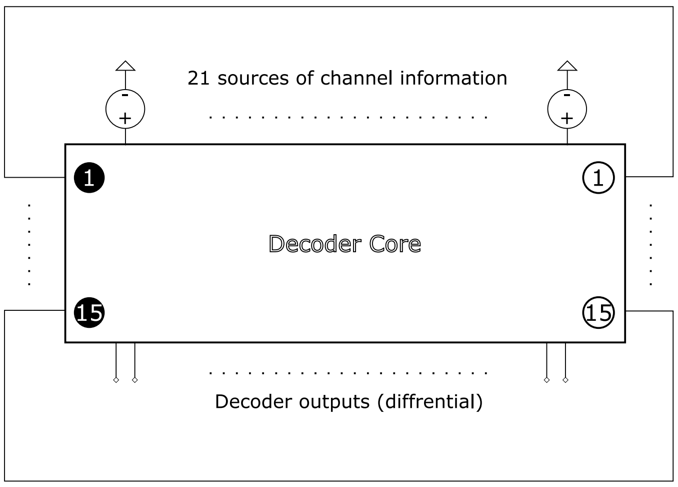

21 sources of channel information
Decoder Core
1
1
15
15
Decoder outputs (diffrential)

Figure 5.17: Top level hierarchy of the schematic digram for realization of the graph in figure 5.13

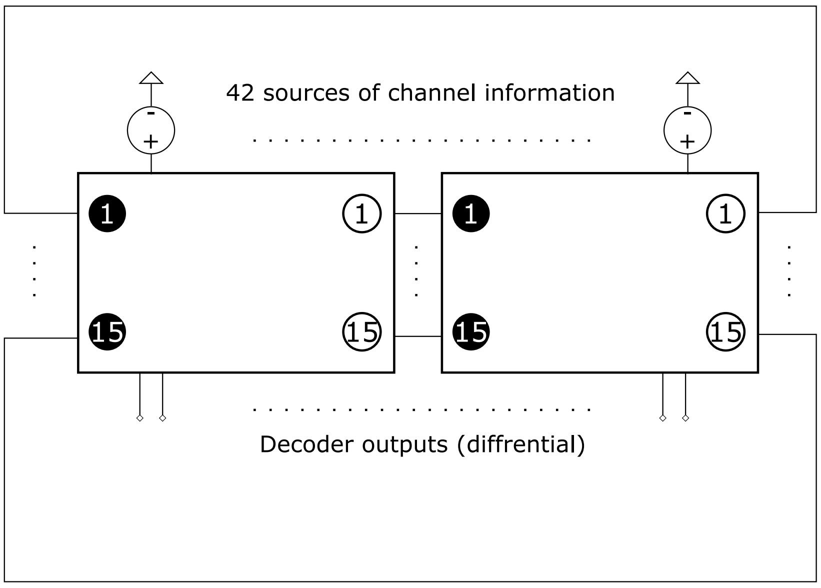

42 sources of channel information
1
1
1
1
15
15
15
15
Decoder outputs (diffrential)

Figure 5.18: Doubling the code length by cascading an extra block

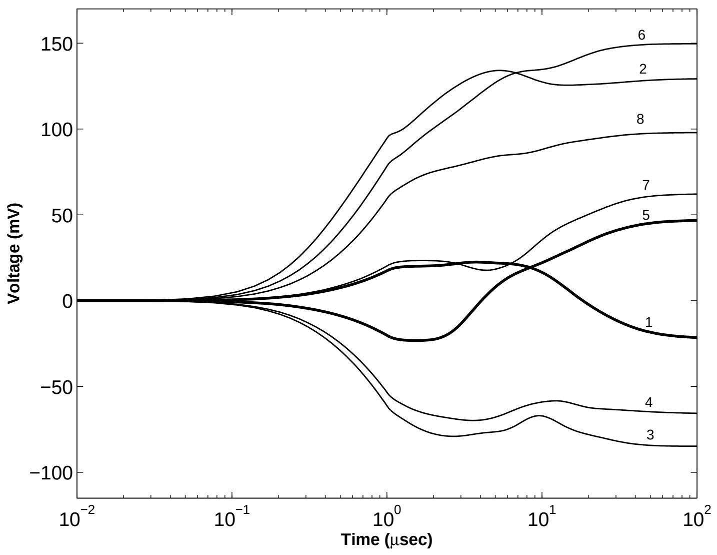


Figure 5.19: QC code transient response

5.3.3 Code performance

The BER analysis is done in three phases:

1. **Data generation(MATLAB):** In this phase, MATLAB generates a sequence of random bits, then it encodes the sequence and generates 21 sequence of codewords. Noise with desired levels is then added, and the values are scaled to the input requirement of circuit. The resulting waveforms are saved into ASCII files, each file storing the waveform of one bit of codewords (21 files here).

2. **Simulation(SPECTRE):** In this phase circuit is simulated in CADENCE environment. The software reads 21 ASCII files, and associates each file with an independent voltage source. Simulation can take several hours depending on the number of bits in the sequence. When simulation is finished, the 8 final voltages corresponding to the 8 a posteriori LLR values of information bits are stored in an ASCII text file.

3. **Post processing(MATLAB):** The file generated during second phase is read by MATLAB and compared to the original information sequence. The number of bit changes between decoder output and original sequence (i.e. number of errors) is counted and the BER is calculated.

Figure 5.20 shows BER curve for (21,8,6) QC code. The figure also contains the results obtained by belief propagation algorithm. Two curves are very close in this figure that justifies the analog decoder performance. It must be mentioned that we are not obtained such perfect result in our previous simulation of tailbiting (7,5)

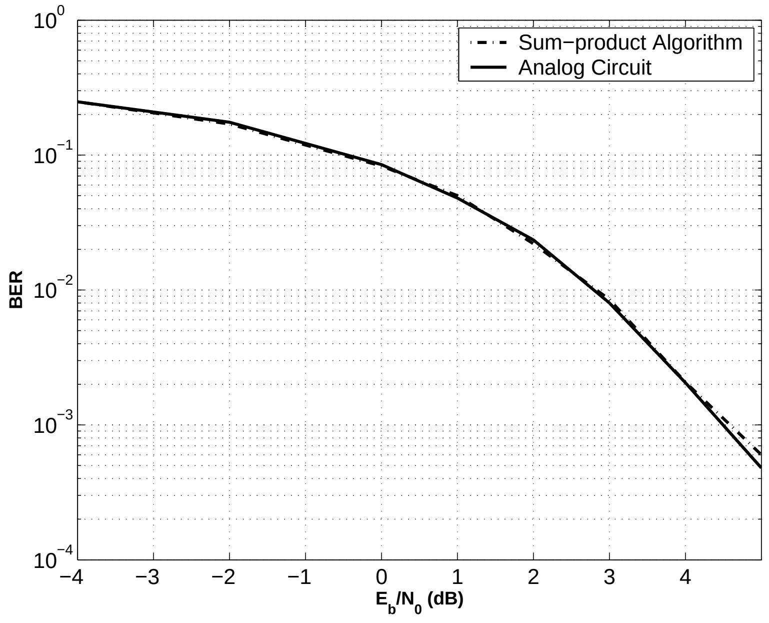


Figure 5.20: BER performance of the QC LDPC

CC. depicted in figure 5.9. The reason is a supplementary optimization step that was only applied to QC decoder. The aim of optimization step is to design good function and variable nodes by modification of design parameters on an ad hoc basis. The next section gives more details on this issue.

5.4 Optimization

The circuit presented so far is a prototype to ensure proper operation of analog decoders. There is a large variety of parameters that a designer can change in order to improve performance of the decoder. In this section we are going to introduce some of these parameters and analyze the way that these parameters can affect the performance of analog decoders. Both internal parameters (e.g. $\frac{W}{L}$, reference voltages,...) and external parameters (e.g. input voltage levels, symbol time,..) are considered.

5.5 Transistor’s Aspect Ratio

Analog decoder are made of primitives such as the product cell in figure 5.21. This simple circuit along with Norm circuit provides the necessary hyperbolic function for operation of analog decoder. Any deviation from the ideal characteristic can degrades the overall performance of decoding. We have found that one important source of error is the lack of good symmetry between two differential inputs in

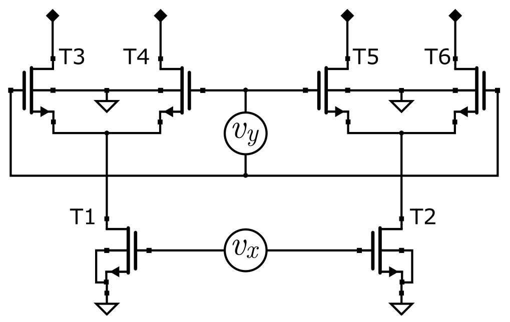


Figure 5.21: Product22 circuit schematic with optimized transistor size as are mentioned in table 5.3

Reference current	$T_3,\ldots,T_6$		T_1,T_2	
I_{ref}	W (μm)	L (μm)	W (μm)	L (μm)
10 nA	1.7	2.1	3.1	3.9
100 nA	1.7	2.1	3.1	3.9
1 μA	2.1	2.1	4	3.9

Table 5.3: W and L of transistors for different reference currents

Boxplus. In the other words, the cell output must be the same whether the inputs are applied to V_x and V_y or V_y and V_x.

We have changed W/L using trial and error method to get a symmetric hyperbolic tangent curve. Optimal W/L depends on the choice of reference current. We have used the currents of 10 nA, 100 nA, and 1 μA as reference currents in the cell and we obtained the results shown in table 5.3.

5.6 Reference Current

We have stated in section 4.2.1 that the reference current affects the circuit speed and performance, however we did not specify how do speed and performance change in responding to different reference currents. Figure 5.22 shows BER of tailbiting (7,5) decoder versus sampling time for different signal to noise ratios. we can see that smaller values of reference current result in less errors in detection (even for $E_b/N_0 = 4$dB the performance of $I_{ref} = 10$ nA is better than $I_{ref} = 100$ nA, if sampling time is later than 100 μs). We can see that BER is slightly different for $I_{ref} = 10$ nA and $I_{ref} = 100$ nA, but much worse for $I_{ref} = 1$ μA. The response is almost immediate for $I_{ref} = 1$ μA, about 5 μs for $I_{ref} = 100$ nA, and roughly 40 μs for $I_{ref} = 10$ nA.

The choice for the I_{ref} depends on design requirements, however for a usual application we can say that as for $I_{ref} = 100$ nA we gain one order of magnitude in speed and do not lose significantly in BER compared to $I_{ref} = 10$ nA.

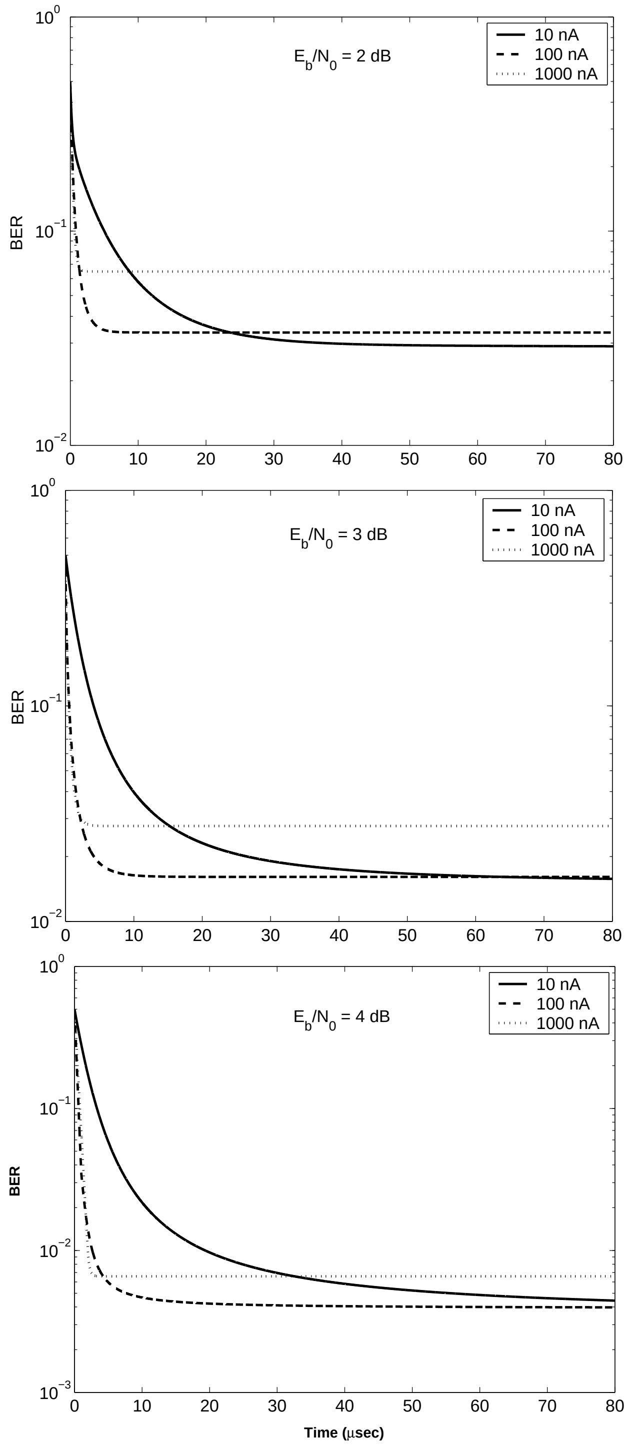


Figure 5.22: Circuit speed and performance for different currents and $\frac{E_b}{N_0}$

5.7 Symbol Rate

As we can see on figure 5.22 that the later we make our decision on the decoded bits, the lower BER we have. However the BER tends to a constant value when sampling time tends to infinity. On the other hand, if we use smaller sampling times, we gain on the speed, and consequently the amount of information transmitted by the channel. It is logic to let the sampling time equal to the value for which the BER arrives its final value $\pm 5\%$, i.e. about 2 μs for $I_{ref} = 1\ \mu A$, about 10 μs for $I_{ref} = 100\ nA$ and about 80 μs for $I_{ref} = 10\ nA$.

5.8 Input average magnitude

In section 4.3.6 we saw that the theoretical proportionality constant between normalized channel output (± 1) and voltages at the circuit inputs was:

$$V = \zeta V_T L(Y) = \frac{4\zeta V_T}{N_0} y = k_c y \tag{5.7}$$

where y is channel output. This equation does not account for the transistor saturation and noise figure. According to equation 5.7, for large signal to noise ratios, the output of channel must be amplified before being injected into the circuit. However, in order to prevent transistor saturation, input voltages should not be very large. On the other hand, equation 5.7 indicated that for small values of SNR, channel outputs must be attenuated before giving to analog decoder. But very small voltages cannot invoke any significant reaction in the circuit. The question is that how the optimum average input level varies around the theoretical value $\frac{V}{k_c y}$.

We have simulated the decoder response to a fixed input sequence for different channel SNRs. Figure 5.23 shows the results. In this figure, the horizontal axis is the average input level normalized to its theoretical value. We can see that for large SNRs the optimum input level tends to decrease from its theoretical value while for small SNRs it tends to increase from its theoretical value. However, for the signal to noise ratios examined in this repport the optimum average input varies slightly around theoretical value.

5.9 Operating Temperature

We know that the temperature may cause an effect on a circuit performance and degrades performance. In order to anticipate this effect, the earlier BER simulation was repeated for two extra limiting temperatures. Note that this type of simulation does not incorporate the sophisticated effects such as thermal gradients in IC die that also need appropriate model for transistors. As a result the following simple simulation may be far from a realistic situation. Nevertheless for such a low-power application, one might argue that the self-heating in the decoder circuit is negligible and it does not affect the overall performance substantially.

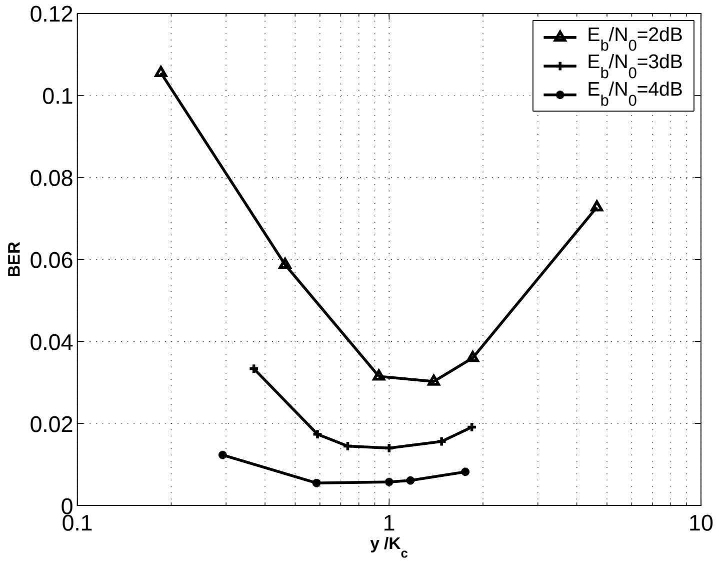


Figure 5.23: Effect of input average magnitude on the performance

We have simulated our analog decoder by a fixed information sequence on different ambient temperatures ($T = 15°$ C, $T = 27°$ C, $T = 40°$ C). Figure 5.24 shows the results. In figure 5.24, we can see that the temperature does not affect the circuit performance significantly. The reason is that analog decoder finally converge to a saturated (and stable) state. The temperature may have, and has, an effect on the intermediate (transient) behavior of the response, but it has not a significant effect on the final values.

5.10 conclusion

Several decoders have been considered and implemented by analog approach. Fortunately, in all cases we had promising results that validate the approach. For the graph proposed in section 3.7, we have a tight accordance of BER result with the corresponding theoretical curve. Monitoring the convergence of decoder circuit shows that a latency of about 30 μS is sufficient for reliable decoding of input vector at the condition of $E_b/N_0 = 4$dB. This is obtained for reference (tail) current of only 10 nA in the cells.

A quasi cyclic code is also studied by its decoder implementation. For this decoder an ultimate modularity has been obtained during relabeling of nodes in its graph.

Mismatch is an important issue in analog decoder and has been considered by many authors. Instead of giving a complex formula based on tolerances of design parameters, we have faced this issue rather differently . Here the asymmetry of basic cell due to systematic and non-systematic defects has been recognized as the

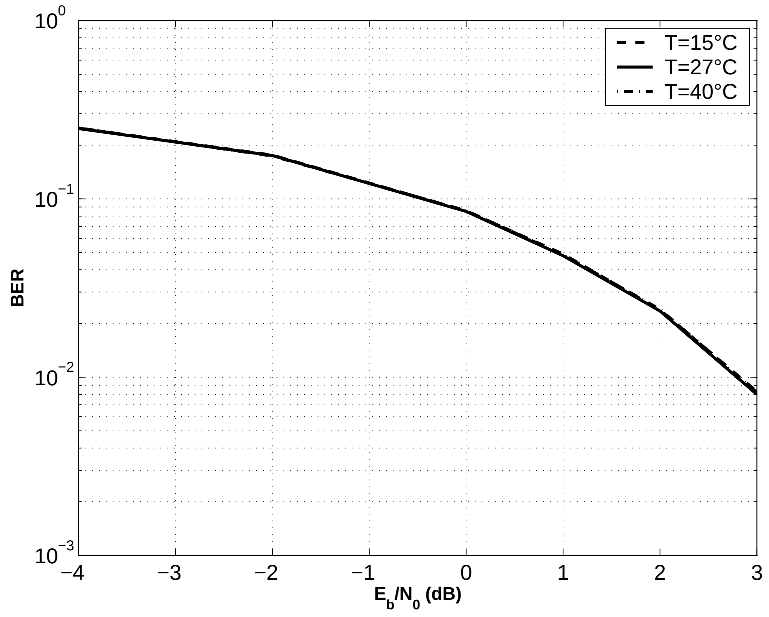


Figure 5.24: Temperature effect on circuit performance

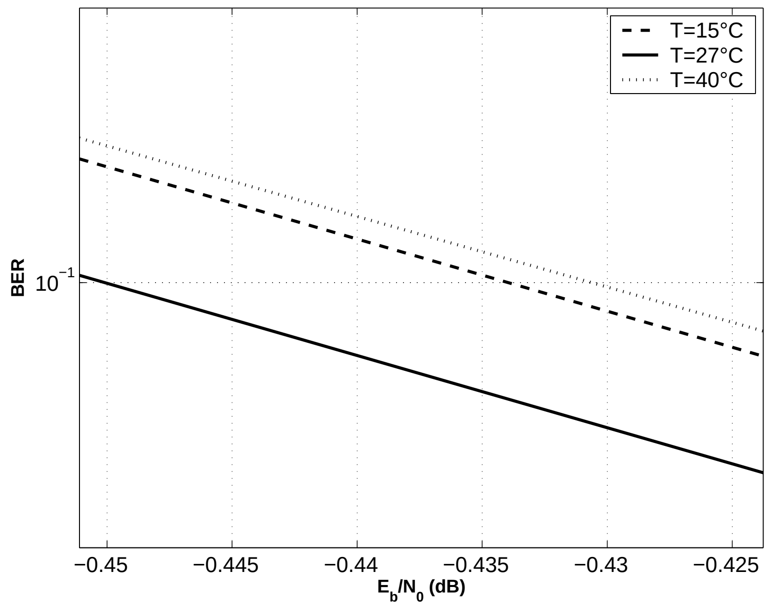


Figure 5.25: Magnified version of figure 5.24

main source of performance degradation in a decoder. Systematic defects refer to the design issue in a circuit that leads to (anticipated) imperfections. In our case, for example, the body effect of MOS transistors is a source of systematic defect while transistor's mismatch falls in non-systematic defect category. Controlling the asymmetry in the basic cell yield decoders that response very close with respect to the theory.

Chapter 6

Fast Viterbi Decoder in Analog CMOS

The implementation of Viterbi algorithm (VA) to obtain high throughput has been always of great interest due to its application in convolutional decoders, trellis code demodulation, turbo-codes, etc. Many digital architectures, adapted to FPGA or digital VLSI circuits, have been proposed to improve the bit rate [52, 53, 54]. Recently, the higher throughput with less power consumption seems to be possible using analog VLSI circuit [55, 56, 57, 41]. The idea is to replace the digital add-compare-select (ACS) with its analog equivalent. Implementation of ACS which is the kernel of digital realization of the algorithm is still challenging because its effect on overall speed and power consumption of decoder.

Besides, several ideas have been proposed for conditioning of analog signals, for example, finding the maximum (minimum) of two or more currents(voltages) [58]. It is possible to employ these ideas for realization of VA. The advantages are omitting analog to digital converter, saving quantization error as well as a better speed/ power trade-off. Current mode circuits are often preferred because summation of two signal is equivalent to tying their corresponding wires. Nevertheless replicating a current needs current mirror. New MOS technology features very short channel length and hence good speed. On the other hand effect of channel modulation makes them far from being a good current source. The situation is still worse in p-channel MOS assuming a typical n-Well process. So care must be applied in the design of such circuits.

In this work two Winner-Take-All (WTA) circuits are combined so that it can realize the principal part of VA. The output current of a WTA is the maximum of its input currents. A subtle change in standard VA is done to adapt the real circuit requirements.

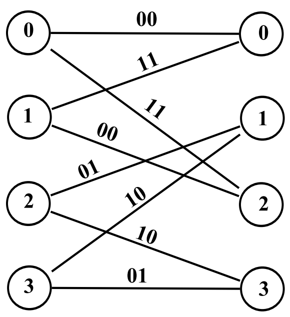


Figure 6.1: A part of trellis for generator $[1,(1+D+D^2)/(1+D^2)]$

6.1 ML Decoding of Convolutional Codes

Briefly, using some tail bits, a convolutional encoder can be considered as a block code where a sequence of k information bits is transformed to a new sequence of length n using a finite state machine with N_S states. The coded sequence $c_t = (c_{1,t}, c_{2,t}, \cdots, c_{n,t})$ then is applied to a modulator (mapper) with "-1" as null symbol and is sent through communication channel. After mapping, namely BPSK, the input channel symbols are denoted by x_t and the channel output by $y_t = x_t + n_t$ assuming a memoryless additive Gaussian noise channel. Generalization to the case of flat fading channel is trivial. The task of the receiver is to find the most likely coded sequence $\hat{C} = \{c_t | t = 1,2,3,\cdots\}$ from observations $Y = \{y_t | t = 1,2,3\cdots\}$ which can be formally written as follows for a memoryless channel:

$$\hat{C} = \underset{C}{\operatorname{argmax}} \sum_t \ln\left(P(y_t|c_t)\right) \tag{6.1}$$

Since all sequences of size n are not valid code sequences, the VA presents an efficient method to find only valid code sequences. Furthermore, using VA, the exhaustive search is replaced by much more efficient iterative method. Using trellis representation, $\ln(P(y_t|c_t))$ is called branch metric and when it accumulates over a specific path, it is called path metric of the path. Figure 6.1 shows some details of VA for a systematic recursive (7,5) convolutional code that is kept as a generic code for our simulations. The branch metrics d_t are calculated based on observation y_t and corresponding c_t for each edge in the trellis diagram. Assuming Gaussian noise, maximizing $\ln(P(y_t|c_t))$ is equivalent to minimizing $\|y_t - c_t\|$. At each state, branch metrics are added to path metrics of the corresponding previous stage based on the structure of the trellis, following by a comparison and selecting the smallest competitors. The new path metrics are passed to following stages. Denoting path metric of state k at time t as $D_t(k)$, This operation is summarized as follow:

$$D_t(k) = \min \left\{ \begin{array}{c} D_{t-1}(i) + d_t(i,k) \\ D_{t-1}(j) + d_t(j,k) \end{array} \right\} \tag{6.2}$$

Furthermore, for decoding of information bits, the surviving path should be stored for each state and used in trace back operation:

$$v_t(k) = \text{argmin} \left\{ \begin{array}{c} D_{t-1}(i) + d_t(i,k) \\ D_{t-1}(j) + d_t(j,k) \end{array} \right\} \tag{6.3}$$

6.2 Conditioning of Input Signal

Several circuits were considered in order to obtain a Max or Min function with good speed and low complexity. The later feature is more important in the system with a large number of states N_S. Among plenty of such circuits, it seems that the WTA in [58] still outperforms other ones in the sense of robustness, speed and complexity. Assuming a current mode WTA with Max functionality, we define a new variable in order to replace Min in equation (6.2) with Max as well as converting bipolar (voltage) variables to unipolar current variables as:

$$\lambda_t(i) = I_{ref} - \alpha f\big(d_t(i)\big) \tag{6.4}$$

Here $d_t(i)$ represents the distance between observation y_t and x_t for a particular $c_t = i$. The fact that adding (subtracting) a constant to all path metrics has no effect on the algorithm, we can expand $d_t(i)$ and eliminate the constant terms in order to achieve the linear equations as follows:

$$\begin{array}{l} d_t(0) \equiv d_t(0,0) = d_t(1,2) = \|y_t - (-1,-1)\| \equiv y_{1,t} + y_{2,t} \\ d_t(1) \equiv d_t(2,1) = d_t(3,3) = \|y_t - (-1,1)\| \equiv y_{1,t} - y_{2,t} \\ d_t(2) \equiv d_t(2,3) = d_t(3,1) = \|y_t - (1,-1)\| \equiv -y_{1,t} + y_{2,t} \\ d_t(4) \equiv d_t(1,0) = d_t(0,2) = \|y_t - (1,1)\| \equiv -y_{1,t} - y_{2,t} \end{array} \tag{6.5}$$

Thus $d_t(i)$ is a bipolar (voltage) variables. $f(\cdot)$ in equation (6.4) is a limiting function defined by:

$$f(x) = \begin{cases} LB & x < LB \\ x & LB < x < UB \\ UB & x > UB \end{cases} \tag{6.6}$$

where LB and UB are lower band and upper band respectively. The bands must be set properly so that overall Bit-Error-Rate (BER) remain almost unchanged. Recall that in a typical competing path, most branch metrics are minimal, we can argue that the bands must span over -2 (volt) with sufficient margin that depend directly on the noise variance. α is a constant with dimension of (Ω^{-1}). Finally I_{ref} is chosen so that λ remains always positive. It must be mentioned that equation 6.4 can be almost realize by only one MOS as a transconductance converter and a current conveyor [59].

6.3 Adapted Algorithm

In the algorithm there are three parameters, I_P , I_N and I_{th} , that should be set properly as will be described bellow.

1. initialization
   Set I_P , I_N and I_{th} and initialize the path metric as $M_0 = (I_P, 0, \cdots, 0)$.

2. Set $t = 1$ and calculate branch the metrics corresponding to each edge in trellis diagram according to equation 6.4, i.e. calculate:

$$\lambda_t(i,j) \quad \forall i,j \in \{0,1,\cdots,N_{s-1}\}$$

3. For each t (and all k) update the path metrics by:

$$M_t(k) = \begin{cases} G & M_{t-1}(i) < I_{th} \\ G - I_N & \text{otherwise} \end{cases} \tag{6.7}$$

   where the intermediate variable G is:

$$G = \begin{Bmatrix} M_{t-1}(i) + \lambda_t(i,k) \\ M_{t-1}(j) + \lambda_t(j,k) \end{Bmatrix} \tag{6.8}$$

Knowing that $0 < \lambda < \lambda_{max} = \alpha(UB - LB)$ and selecting $I_P > \lambda_{max}$ ensure that starting state is zero. In fact I_P propagates through the algorithm and determines the current that the circuit works around it. So it must be chosen so that the circuit has the best trade-off between speed and precision. In our design it is found by simulation to be $50\mu A$.

It is clear that path metrics continue to increase in each iteration and so rescaling of the metric is inevitable. This is done by subtracting I_N form all metrics at the specified condition as given in equation 6.7. Delay of rescaling does not accumulate on the delay of metric calculation because two WTAs work in parallel. This is why M_{t-1} is sensed instead of M_t. It is expected that metrics have variation between $I_{th} - I_N$ and I_{th} . Again they are set to the best operating range of the circuit. While practically possible, there is no need to check all path metrics to perform the rescaling. Dynamic range of path metrics is bounded in magnitude by a fixed quantity, Δ_{max}, due to pruning of surviving paths [60]. We have the following equation:

$$\Delta_{max} < \lambda_{max} \log_2(N_S) \tag{6.9}$$

So it will be sufficient to check out only one metric (say, the path metric of state "zero") and set $I_N < \Delta_{max}$ to ensure all other path metrics remain positive. A CMOS circuit that mimics the procedure tn the above algorithm was designed and simulated in the following section.

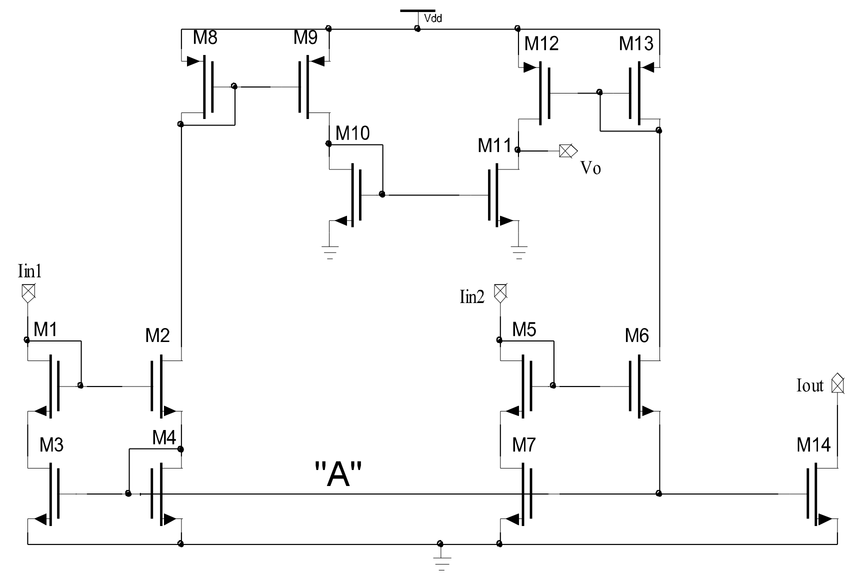
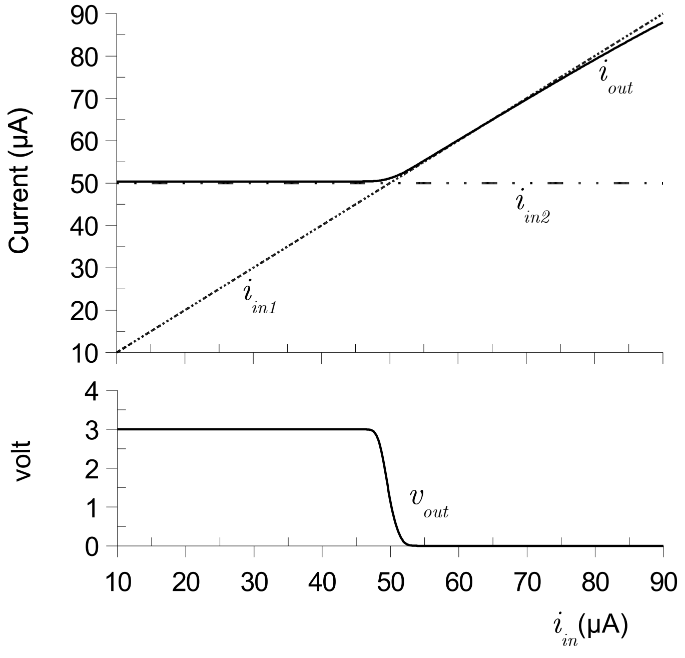


Figure 6.2: WTA-MAX circuit for realization of MAX and ARGMAX function.

6.4 Viterbi Decoder Kernel in Analog CMOS

In this section we present a CMOS realization of the algorithm proposed in section 6.3 and its functionality by means of transient and DC responses. The simulations are based on AMS0.35 process with BSIM3v3 level 53 model for MOS which were performed in Cadence$^{\mathrm{TM}}$. This analog design that realize the core of Viterbi decoder can be used in lieu of Add-Compare-Select (ACS) unit in conventional implementation of the algorithm. Nevertheless other part of a Viterbi decoder, say trace back, still remains unchanged. It is clear that with this analog circuit we don't need ADC at the input. The only things we need, are current switches (S/H) to hold the output currents and to feed it back to inputs.

Figures (6.2) shows a Winner-Take-All (WTA) circuits that is a main part of this design. There are two sort of outputs in this circuit, I_{out} which is an analog current and V_o as a logic output. I_{out} take the maximum values of two input currents, I_{in1} and I_{in2} with a high degree of precision despite the transistor characteristics. On the other hand, V_o goes to GND for $I_{in1} > I_{in2}$ and V_{dd} for $I_{in1} < I_{in2}$. So the circuit performs both MAX and ARGMAX functions and we call it WTA-MAX. Figure (6.3) shows the response of WTA-MAX during static and dynamic characteristics respectively. From figure (6.3) it can be recognized that the circuit settles down completely after 1.8nS and with referring to DC response, static error is negligible. Maximum error take place when two inputs are the same. In this case, figure (6.3) estimates an error of about 2%. At higher current, output curve bends due to effective mobility reduction. Fortunately it is far from the operating point of our



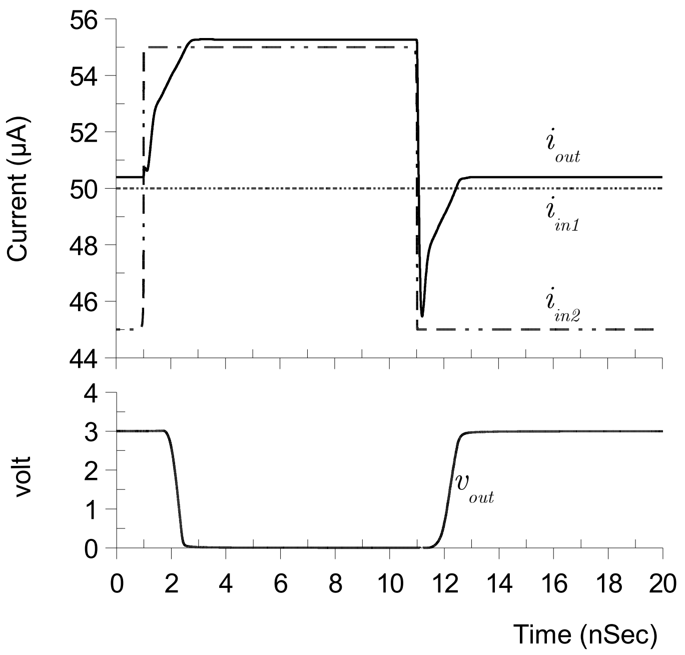


Figure 6.3: (top) DC characteristic of WTA-MAX in figure (6.2) , (down) A typical transient response

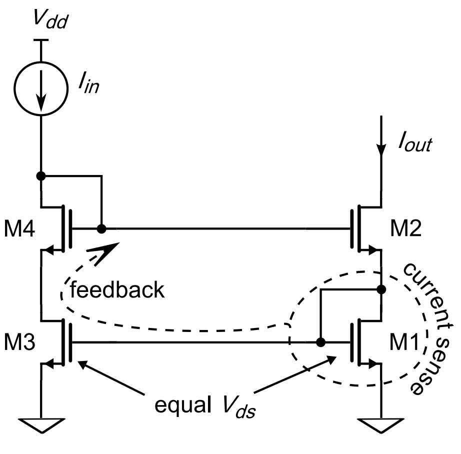


Figure 6.4: Wilson current mirror

design. Also V_o represents a very sharp variation. This logic output is neecessary for trace back in Viterbi decoding and will be connected to the circuitry of surviving path memory (or state register in register exchange decoding mode).

Here is a brief explanation of how it works and the reason for its speed and precision. One can consider this circuit from different point of view. We thought that Wilson current source is a good starting point for an intuitive analysis.

Figure 6.4 shows a Wilson current mirror, in which $I_{out} = I_{in}$. The precision of current mirror is pretty good for a wide range of current and for different transistor's characteristics. It is basically due to a strong inherent feedback in the circuit. M3 and M1 are feedback mechanism with a unit gain (complete feedback) while M2 provide the forward gain. In addition, thanks to M4 V_{ds} for M1 and M3 are almost equal that guarantees the unit gain of feedback network.

The WTA-MAX in figure 6.2 can be regarded as two competitive Wilson current mirrors. M1–M3 along with M4 can form a Wilson current mirror. Another possibility is M5–M7 and M4. So M4 will belong to that group of transistors with higher current level. This competitive behavior is a result of positive feedback in the circuit. The common node "A" in the circuit is responsible for this positive feedback. Suppose that I_{in1} is larger than I_{in2} for instance. The common node voltage "A" defined by "$I-V$" curve of M3. But this is also the gate voltage of M7 and this transistor conveys lower current by assumption. This make M7 saturated and pulls down the drain voltage of M7 and gate voltage of M6. Finally M6 is driven toward cut-off region. Without M6, we have M1–M4 that is a Wilson current source. M14 is added to simply duplicates the winning current. In addition, according to above comments, M2 drain current is either zero (almost) or equal to the winner current. This is also true for M6 drain current. It is straightforward to see that V_o is zero or V_{dd} for two competitive states of the circuit.

Rescaling of path metrics, as stated in the algorithm in section 6.3 requires a current comparator. Fortunately the original WTA discussed in [58] have a desired

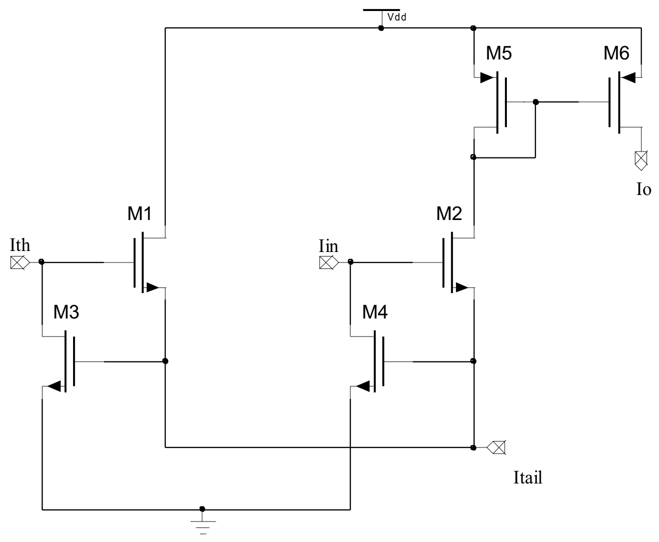


Figure 6.5: WTA-N circuits which realizes a current comparator

WTA-MAX	M8-M11	$3.6\mu m/0.35\mu m$
WTA-N	M5,M6	$1.2\mu m/1\mu m$
CSR	M1-M12	$1.2\mu m/1\mu m$
otherwise		$1.2\mu m/0.35\mu m$

Table 6.1: Transistor dimensions (W/L)

characteristic. The circuit (WTA-N) and its response are shown in figures (6.5) and (6.6) respectively. In obtaining the responses, a current source of $I_N = 20\mu A$ was connected to the port I_{tail} in figure (6.5). So when current I_{in} become greater than I_{th}, output current changes from zero to I_N. Figure (6.7) shows complete circuit as kernel of Viterbi decoder. It consist of WTA-MAX, WTA-N and some current mirror for current duplication necessity. Channel length is augmented for p-channel transistors in this figure to mitigate the effect of channel length modulation. The drawback is a little speed loss which is inevitable in CMOS technology. Transistor dimentions are indicated in table (6.1). Note also that for a code with N_S states we need the equal number of WTA-MAX, but only one WTA-N will be sufficient as discussed theoretically in section (6.3).

Moreover in figure (6.7), transistor M1 is just half of a current mirror. Depending on the code, there is need for replication of its current by sharing V_GS with the other transistors. For example for the code defined in figure (6.1), two edges leave each state so we need two copies of M1's current.

Figure (6.8) shows the effect of connecting WTA-N to WTA-MAX in the complete circuit of figure (6.7). Here a step change of $20\mu A$ was expected in WTA-MAX output, but it turns back to $45\mu A$, because I_N had been set to $15\mu A$. For a

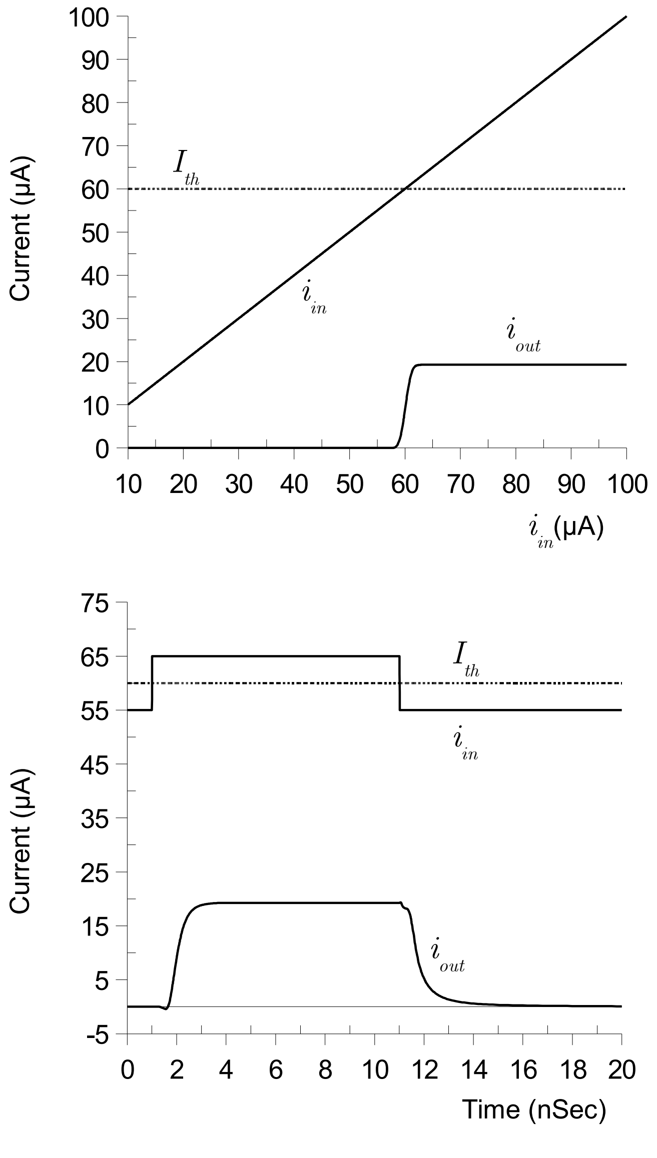


Figure 6.6: Typical dynamic and static response of WTA-N

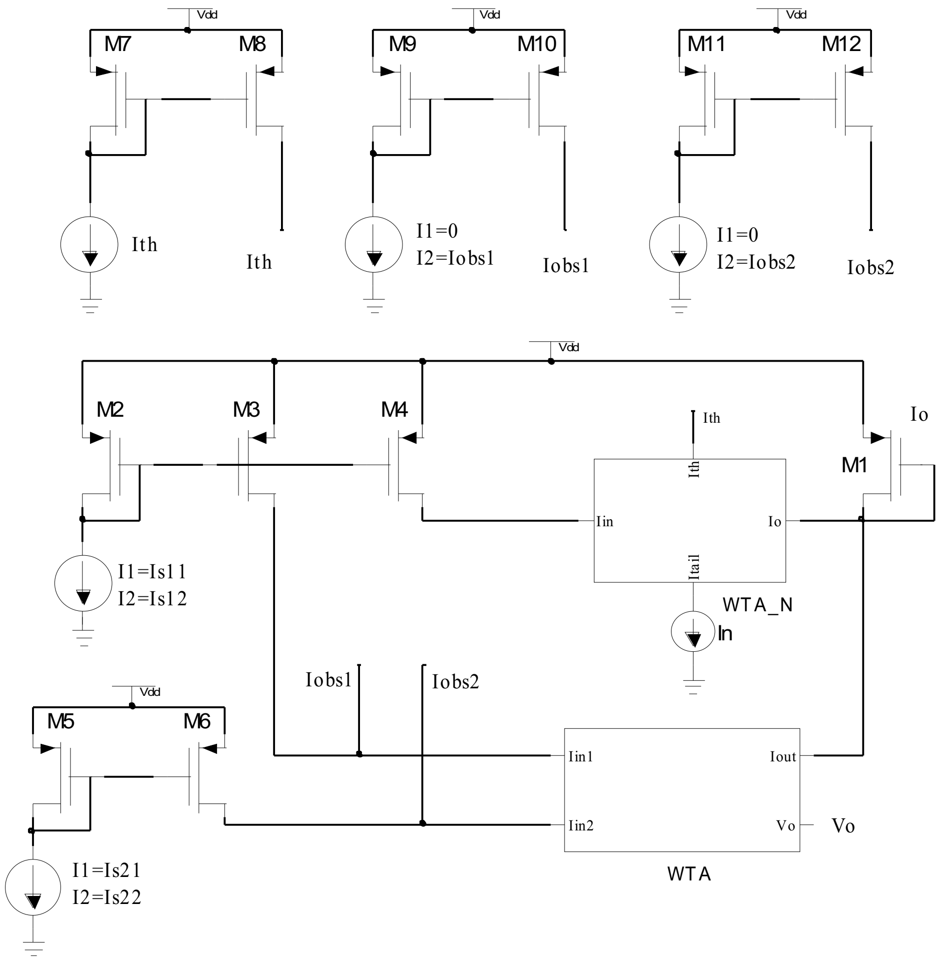


Figure 6.7: Complete circuit for Compare, Select and Rescaling (CSR) of two metrics

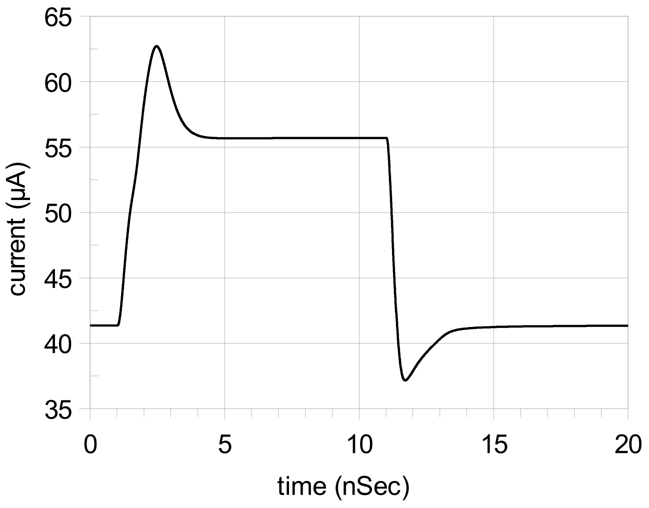


Figure 6.8: Typical transient response of complete circuit that illustrates the effect rescaling

complete characterizing the CSR[1] circuit, refer to figure (6.9), in which a family curves are produced by parametric sweep on I_{in1} and I_{in2}. The response is similar to figure (6.3) except an $I_N = 20\mu A$ abrupt change at the threshold current of $I_{th} = 50\mu A$.

6.5 Behavioral Simulation Results

In order to estimate the bit error rate of the proposed decoder, among several source of errors we consider two secondary effects which may impact overall performance of the analog Viterbi decoder. Conditioning of branch metrics is the first one that reduces the dynamic range of metrics. Second effect is the non ideal Max function due to its soft transition as depicted in figure (6.3). First issue is simply modeled by imposing equation (6.6) in VA. Figure (6.10) shows that a tighter range results in a worst performance and vise versa. To include the non ideal Max function we add a nonlinear function $g(\cdot)$ to the ideal Max function:

$$max^*(x,y) = max(x,y) + g(x-y)$$

with $g(\cdot)$ as:

$$g(x) = ae^{-b|x|} - \lambda|x|$$

The constants a, b and λ are obtained from figure (6.3). For our case, a least squares fitting results in $a = 0.65$, $b = 0.9$ and $\lambda = \frac{1}{30}$. Figure (6.11) shows effect of this model on bit error rate in which a dynamic range of UB-LB=4 (volt) has been

[1]Compare-Select-Rescaling

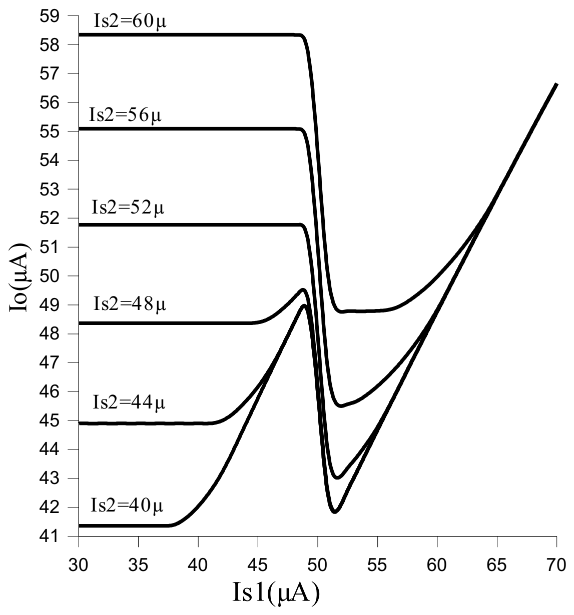


Figure 6.9: DC characteristic of CSR circuit

assumed. This figure justifies that non ideal issues in CSR design are tolerable and has minor effect on the overall performance of Viterbi decoder.

6.6 conclusions

We have shown that ACS block in a Viterbi decoder is realizable by CMOS circuit. The design here is based on WTA tailored for this purpose. In contrast to LDPC analog decoders, transistors carrying high current in order to obtain a desirable speed. The design is based on current-mode circuit, so branch and path metrics are represented by current. A solution for rescaling path metric has been proposed that dose not degrad the overall speed. Circuit level simulations revealed a latency of 3 nanosecond for the ACS + rescaling. With its very simple structure, a complete parallel architecture for a Viterbi decoder with large number of states, say 64 or 128, will be still tolerable.

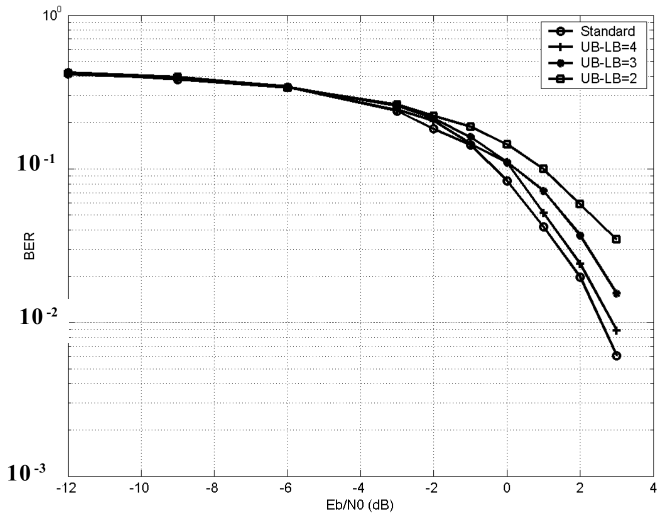


Figure 6.10: Effect of metric's range on bit error rate.

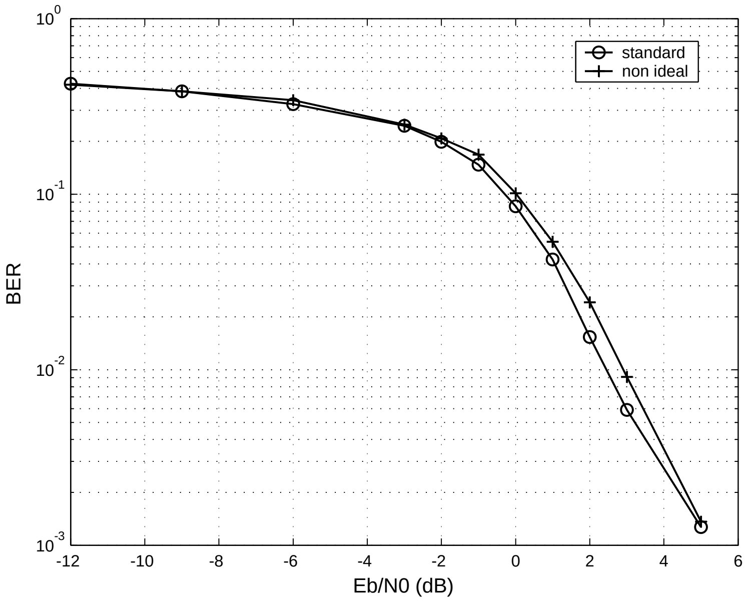


Figure 6.11: comparison of standard decoding with the one with non ideal Max function as well as limiting the dynamic range for the metrics.

Chapter 7

Analog FIR Filtering

7.1 Introduction

Despite the fact that digital signal processing is widespread in today's technology, existence of analog parts especially in front-end sections of communication systems is unavoidable. This kind of circuits are widely used in mixed-signal SoC (System on Chip) applications where the integration of analog and digital parts on the same chip is necessary [61]. It may be both economic and power efficient to pre-process analog signals at the analog/digital interfaces. The location of matched-filters in digital communication systems is a good example of such interfaces. By analog filter in this paper, we mean the sampled-data or equivalently discrete-time analog filters.

Figure 7.1 shows some important schemes used for digital receivers. In figure 7.1(a), the analog matched filter is used and its output is applied to an Analog-to-Digital Convertor (ADC). The digital stream from ADC block are then processed by a digital core (DSP, FPGA or FPOA)[1]. The significant potentials for analog filters with respect to their digital counterparts are lower power and smaller chip area. Two major realizations for analog sampled-data filters are the well known switched-capacitor (SC) and switched-current (SI) networks [62, 63, 64]. Low tolerance of capacitance ratio in monolithic fabrications makes precise control over SC filter's characteristics. However, the existence of operational amplifier in such networks limits the maximum attainable bandwidth. This problem limits the use of SC filters in high-rate digital receivers. Another problem is the use of floating capacitors (poly-poly caps) that makes such network incompatible with CMOS fabrication technology. SI networks, on the other hand, has better frequency response compared to SC networks and are compatible with CMOS fabrication technology. Several attempts have been made to increase the clock rate of SI networks using special active elements such as GaAs MESFET [65] in exchange for increase in costs.

Another challenge in analog domain arises from realization of reconfigurable

[1]Digital Signal Processor, Field Programmable Gate Arrays, Field Programmable Object Arrays

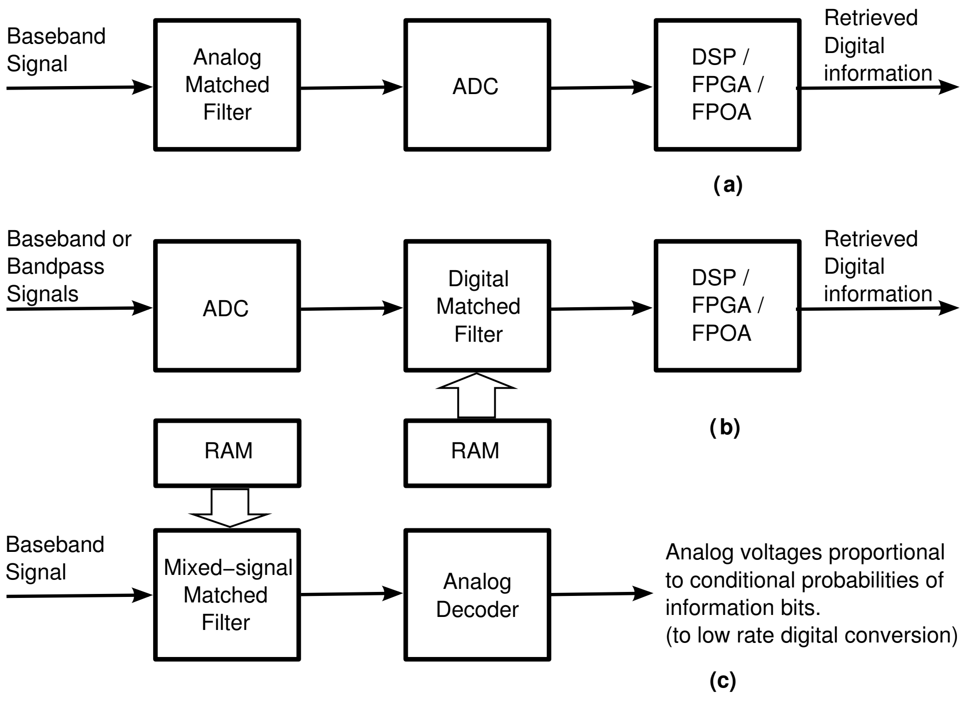


Figure 7.1: Three possible structures for digital receivers (a) Conventional receiver with analog matched-filter (b) Software radio inspired structure (c) Exploiting analog decoder with mixed-signal matched filter yield a simple and power efficient solution for digital receivers

filters. Reconfigurable filters are widely used where the filter taps may be changed in real time such as adaptive channel equalizer, multi-rate data transmission, etc. To take advantage of full programmability, reconfigurable filters are usually implemented in digital domain. Several attempts have been made to gain this feature in analog filters. For example the concept of floating gate was exploited to realize a reconfigurable FIR filter in audio frequency range [66]. In [67] a reconfigurable SI filter based on a CMOS ladder circuit [68] is proposed which is able to multiply a digital word by an analog current. As another example, a reconfigurable FIR filter based on CCD is presented in [69] in which the input quantity is electrical charge. These examples demonstrate the application of mixed-signal designs in the sense that the input quantity is analog while the multiplicand is a digital binary number.

Figure 7.1(b) depicts a structure based on software radio principle. According to this principle, ADC block is located as close as possible to analog front-end and any further processes are accomplished in digital domain. A sophisticated ADC is required to convert analog signals to digital at very high rate. The main advantage of this structure is the full programmability that yields a versatile system. Advances in digital platforms such as new evaluation boards are another attraction toward the use of this structure for research and prototyping. However, for specified applications where power consumption, complexity and costs are the main factors, this structure is not preferred and analog implementations can be a more reasonable choice.

Figure 7.1(c) demonstrates a new scheme for digital receivers in which the digital section is replaced with the analog decoder. Analog decoding was proposed at the same time by J. Hagenauer, et al. and H. A. Loeliger, et al. in 1998 [70, 42]. Briefly speaking, analog decoding is an alternative to digital decoders in which decoding algorithm is performed by exploiting nonlinear analog circuits. In comparison with conventional digital decoders, analog decoders have potentially less power consumption, higher speed and lower chip area. Note that the immediate advantage of such configuration is the elimination of ADC that means a great energy/area saving. Also, in this figure our proposed mixed-signal filter is placed before the analog decoder to handle high-rate symbol transmission.

In this chapter we aim to design a mixed-signal FIR filter suitable for front-end processing in high-rate digital receivers. In the proposed architecture, the signal path remains in analog domain while the filter taps are stored digitally, assuring reconfigurability as desired. The filter is based on simple CMOS inverters and exhibits very high bandwidth as a direct result of a design with no internal nodes. In other words, up to circuit diagram analysis, no pole introduces by the design and the bandwidth is infinite in theory. The practical bandwidth is only limited by secondary effects such as carrier transit time, path resistance, etc. In the application such as wireless communication systems our proposed structure is a good solution due to its simple and low complexity structure, low power consumption and fully CMOS compatible fabrication. Furthermore, the high-speed capability and its analog input-output can produce a consistent integrated circuit when is accompanied by the analog decoders.

7.2 Approximation to Continuous-time Convolution

Suppose $h(t)$ is a time limited impulse response of a causal filter with duration D to be realized, and $x(t)$ is the input signal. The analog filter output is given by convolutional integral:

$$y(t) = \int_{t-D}^{t} x(\tau)h(t-\tau)\,\mathrm{d}\tau \tag{7.1}$$

In digital communication systems, we are normally interested in the filter's output at some instants nT where T is the symbol duration. In practical case, the duration D is an integer multiple of symbol period T, i.e. $D/T = N$ so (7.1) can be written as follows:

$$y(nT) = \int_{(n-N)T}^{nT} x(\tau)h(nT-\tau)\,\mathrm{d}\tau \tag{7.2}$$

The integral can be approximated by dividing the integration interval into L subsections. Defining the k^{th} integration gain as:

$$g_k = \hat{h}\left((L-1-k)\Delta T\right) \tag{7.3}$$

where $\hat{h}$ is the quantized h, $\Delta T = \frac{D}{L}$ and $k = 0, \cdots, L-1$. The filter output can be expressed approximately as:

$$y(nT) = \sum_{k=0}^{L-1} g_k \int_{nT-(L-k)\Delta t}^{nT-(L-k-1)\Delta t} x(\tau)\,\mathrm{d}\tau \tag{7.4}$$

In order to realize (7.4), there is need for an integrator with the gain proportional to g_k at the instant $(L-1-k)\Delta t$. The above operations introduce three deviations from an ideal convolution. The first deviation is the truncation of filter's impulse response which is equivalent to convolution of the filter frequency response with $D\,\mathrm{sinc}(fD)\exp(-j\pi fD)$. The second deviation is due to (7.3) which is equivalent to the flat-top sampling of $h(t)$. It affects the overall frequency response via multiplication to $\Delta t\,\mathrm{sinc}(f\Delta t)\exp(-j\pi f\Delta t)$ and as Δt becomes small, this effect becomes negligible. The third deviation is the quantization of the filter taps, g_k, that can degrade the frequency response of the filter. One can minimize this effect by incorporating the quantization into the design as opposed to quantizing the coefficients after the filter has been designed [71]. The phase response remains linear providing that the symmetry of the filter taps is conserved.

7.3 Mixed-signal MAC

Realization of convolution in conventional digital systems is based on MAC unit. Equivalently, we introduce mixed-signal MAC which has analog input while the multiplicands are digitally-stored numbers. According to (7.4), the filter output can be approximately obtained by successively multiplication and integration of the analog input signal. Typically, the input is a voltage signal and the integration can be performed by using a voltage-to-current converter (V/I) followed by a capacitor. We first review the employed V/I circuit which is based on the transconductor presented in [72]. Then we discuss the architecture of mixed-signal MAC which is based on a bank of V/I converter. Some non-ideal issues in the circuit are also discussed at the following.

The following notations are used in the sequel. Variables in capital letter (e.g. V_{DD}) show DC or static values. Lower case variables with the subscribes in capital show the total values (e.g. v_I) and the variables all in lower case are small signal components (e.g. v_i). For example we may write $v_I = V_I + v_i$.

7.3.1 CMOS Inverter as V/I Converter

The principal operation of a CMOS inverter as V/I converter is depicted in figure 7.2. Starting with the classical square law formula for the transistors and applying Kirchhoff current law (KCL) at the output node of the inverter, the following equations are obtained:

$$i_N = \tfrac{1}{2}k_n(v_I - V_{THn})^2, \qquad i_P = \tfrac{1}{2}k_p(V_{DD} - v_I - V_{THp})^2,$$
$$i_O = i_N - i_P.$$

In these equations, k_n , V_{THn} , k_p and V_{THp} are the corresponding parameters for n-channel and p-channel transistors respectively. V_{DD} is the supply voltage and v_I and i_O are the input voltage and the output current respectively, as illustrated in figure 7.2. After some simple manipulations the output current versus input voltage can be expressed as follows:

$$i_O = \frac{1}{2}(k_n - k_p)(v_I - V_C)^2 + g_m(v_I - V_C) \tag{7.5}$$

where

$$g_m = \sqrt{k_n k_p}\,(V_{DD} - V_{THn} - |V_{THp}|) \tag{7.6}$$

and

$$V_C = \frac{V_{DD} - V_{THn} - |V_{THp}|}{1 + \sqrt{k_n/k_p}} + V_{THn} \tag{7.7}$$

With the same p-channel and n-channel transistors, i.e. $k_n = k_p$, the output current would be linear with respect to input voltage except an offset voltage equal to V_C. This offset is given by (7.7) and is approximately equal to $\frac{V_{DD}}{2}$. In practice, however, the first term in (7.5) exists owing to transistor mismatch which is very undesirable. To solve this problem a balance structure as shown in figure 7.3 can be used that effectively removes second order terms as well as taking advantages of a differential structure.

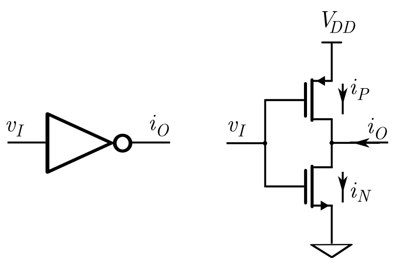


Figure 7.2: A CMOS inverter and its internal circuit which can be used as a voltage-to-current converter

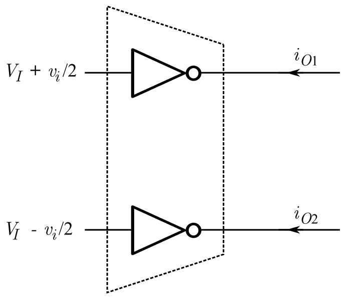


Figure 7.3: A balance structure yeild a good linear transconductor

Defining the differential output current $i_o = i_{O1} - i_{O2}$, the following relationship is obtained:

$$i_o = i_{O1} - i_{O2} = (k_n - k_p)(V_I - V_C)v_i + g_m v_i \tag{7.8}$$

where V_I is the common mode input voltage and g_m and V_C are defined in (7.6) and (7.7) respectively. The aspect ratios of the transistors are chosen so that $k_n \approx k_p$ and also V_I is kept close to V_C. Therefore the output current, i_o, will be almost equal to the second term in (7.8). In addition, the desired simple structure of the transconductor in figure 7.3, results in a very wideband circuit. In fact, there are only two input and output nodes with parasitic capacitors connected to AC ground. As a result no internal node exists and the circuit can operate up to very high frequency range.

7.3.2 Mixed-signal MAC Architecture

The nature of the output quantity in the V/I converter is obviously current. This makes it possible to tie the outputs together to perform current addition or subtraction. This simple principle leads to the MAC architecture in figure 7.4. There are M transconductors of gain g_m, connected in parallel. At the input side, a resistive ladder network provides binary-weighted attenuation of input signal, i.e. $\{v_i, v_i/2, \cdots, v_i/2^{M-1}\}$. These signals are connected to the transconductors via the switch boxes. By applying zero or V_{DD} voltage to the digital control lines b_0

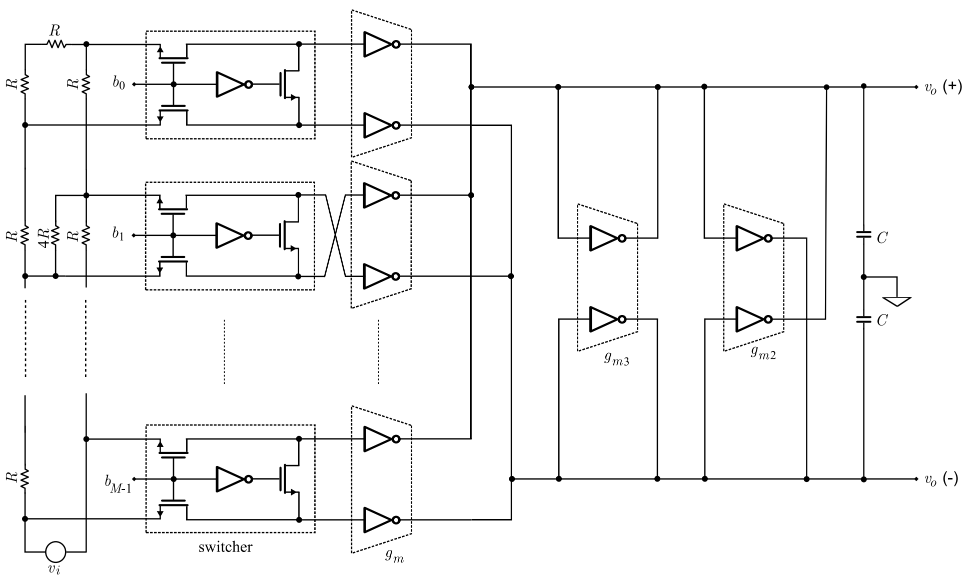


Figure 7.4: Mixed-signal MAC architecture comprised of resistors R and 4R, switchers, main transconductors, g_m, compensating transconductors g_{m2} and g_{m3} and grounded capacitors.

through b_{M-1}, each transconductor can be connected to or disconnected from the ladder network. This architecture is equivalent to a transconductor with the effective gain g_{me} ranging from zero to $(2\text{-}a_{\text{LSB}})g_m$ with the step a_{LSB} where $a_{\text{LSB}} = 1/2^{M-1}$. With the capacitors C at the output, g_{me}/C corresponds to the integral gain, g_k, in (7.4).

It is worthwhile to mention two important points about this design. First, the resistive network introduce internal node to the design and have negative effect on the overall bandwidth of the filter. Fortunately in the case of communication systems, input impedance is typically low (say, 50 Ω) and the resistors in the ladder are easily realizably in CMOS fabrication technology. The corresponding pole frequencies of this low impedance and parasitic input capacitances of switch box is very high and therefore tolerable (c.f. section 7.5).

Second, filters have often positive and negative tap values which in many case are not symmetric about zero. For example, the impulse response of a typical raised cosine filter is not symmetric about zero and ranges over $[-0.2\cdots 1]$[2]. It can be found that the optimal[3] four-bit quantized range should be proportional to $[-2\cdots 13]$. The conventional 2's complement binary number dose not fit this range well and the quantization error increase. Recall that a 2's complement four-bit binary number have the weights $\{1, 2, 4, \text{-}8\}$ that can span the range $[-8\cdots 7]$. Using other combination of binary weight may decrease quantization error. For example

[2]the exact value depends on rolloff factor

[3]MSE measure

in the case of raised cosine signal the binary weights {1, -2, 4, 8} minimize quantization error by spanning the proper range $[-2\cdots13]$. Fortunately this requirement can be accomplished in the proposed structure with no extra complexity as follows. We twist the inputs (or outputs) connection of those transconductors with negative weight and at the same time invert the corresponding command logic in the digital memory. For example, in figure 7.4 the input to the second transconductor has to be twisted and consequently b_1 becomes inverted (c.f. table 7.2) to obtain the desired span range $[-2\cdots13]$.

7.3.3 Secondary Effects

Finite output resistance is a common problem in analog circuits. In general, CMOS transistors suffer from channel length modulation effect which can be modeled by a finite output resistance. In figure 7.4, each transconductor has such an output resistance which appear in parallel at MAC output. The drawback with limited output resistance is introducing a low-frequency pole that limits the useful bandwidth of the filter at low-frequency. Fortunately, the so-called negative resistance compensation is easily applicable in this configuration with a minor extra complexity. As can be shown in figure 7.4, two transconductors of the gains g_{m2} and g_{m3} are connected at the output. It is easy to show that this is equivalent to a resistor with the value $1/(g_{m3}-g_{m2})$. Proper setting of g_{m2} and g_{m3} results in a negative resistance that will cancel out the output resistance of the main transconductors and yield a theoretical infinite output resistance. Based on (7.6) g_m is proportional to the aspect ratio (W/L) of the transistors (through k_n and k_p). Slight reduction of aspect ratio for the transistors in g_{m3} with respect to g_{m2} yields the required negative resistance.

Another important issue in practical implementations is the transistor mismatch. In order to estimate the amount of variation on transconductor gain, we target the dominant (second) term in (7.8) and calculate the variation of g_m. Using (7.6) and applying mathematical analysis to the variables subjected to mismatch, we obtain:

$$\sigma\left(\frac{\Delta g_m}{g_m}\right) = \sqrt{\frac{1}{4}\sigma\left(\frac{\Delta k_n}{k_n}\right)^2 + \frac{1}{4}\sigma\left(\frac{\Delta k_p}{k_p}\right)^2 + \frac{\sigma(\Delta V_{THn})^2 + \sigma(\Delta V_{THp})^2}{(V_{DD} - V_{THn} - |V_{THp}|)^2}} \tag{7.9}$$

where $\sigma(\cdot)$ stands for standard deviation of the corresponding parameter and is given by the foundry [73]. Table 7.1 shows the necessary parameters for calculation of (7.9). Taking into account transistor's dimensions given in section 7.5, we obtain from (7.9) that $\sigma = 6.54\times10^{-3}$ which is about 19 times less than a_{LSB} in our design and thus the mismatch effect is negligible for four-bit resolution.

Similarly, table 7.1 is applicable to temperature dependency calculation. Transconductance in (7.6) is subjected to temperature variation via mobility and threshold voltage of transistors. Taking derivation with respect to temperature and after some

Parameter	NMOS	PMOS	Unit	Comment
A_VT	9.5	14.5	mV μm	$\sigma(V_{TH}) = \frac{A\_VT}{\sqrt{WL}}$
A_K	0.7	1.0	% μm	$\sigma\left(\frac{\Delta K}{K} \times 100\right) = \frac{A\_K}{\sqrt{WL}}$
VT	0.50	-0.65	V	Threshold Voltage
TCVT	-1.1	1.8	mV/°K	Temp. coeff. of VT
BEX	-1.8	-1.3	–	Mobility exponent i.e. $\frac{\mathrm{d}(\ln\mu)}{\mathrm{d}(\ln T)}$

Table 7.1: Mismatch and process parameters for NMOS and PMOS [73]

manipulation, one obtains:

$$\frac{\Delta g_m}{g_m} = \frac{c_n + c_p}{2}\frac{\Delta T}{T} - \left(\frac{\delta V_{THn}}{\delta T} - \frac{\delta V_{THp}}{\delta T}\right)\frac{\Delta T}{V_{DD} - V_{THn} - |V_{THp}|} \tag{7.10}$$

In this equation c_n and c_p are mobility exponents which are named BEX in table 7.1. $\delta V_{TH}/\delta T$ is the temperature coefficient of threshold voltage which is given in table 7.1 for n-channel and p-channel devices under the name of TCVT. Replacing all parameters in (7.10) by their values yield the value of -3.8×10^{-3} at 300°K. For example a 10°C increase in temperature will lower g_m by about 3.8% which is only about one-third of a_{LSB}. Despite the temperature and mismatch uncertainties, the critical filter parameter remain unchanged because they are related to digital clock frequency and stored filter taps in memory. The slow variation of transconductance affects only the output magnitude. Suppose that the filter in connected to a decoder, it can be shown that the variation in magnitude degrades the performance of decoder very slightly [9].

Finally, in short channel devices, the classical square-law model is no longer valid. In this work, except switch transistors which have minimum channel length, the analog transistors designed to have a length of $1\mu m$ in order to achieve a suitable characteristic. Since the transistor count in this design is considerably low, a slight augmentation of device size can be tolerable. A good model for analytical analysis in such channel length is the so called alpha-power law model [74]. It can be shown that the differential structure still yields a good linearity despite the deviation from ideal square-law model but here it is only justified by simulation.

7.4 Filter Structure

Based on the timing digram in figure 7.5, the MAC output is valid after the integration period D. Integration period for the next MAC starts after T_o second where $1/T_o$ is the desired sampling rate at the filter output and in general we have the inequality $\Delta t \leq T_o \leq T$. Therefore the integration periods of the MAC units overlaps but all of them perform the same things. It means that $\lceil D/T_o \rceil$ identical MAC units are needed to retrieve all output samples[4] . In general, $9\log_2(M) + 10$ transis-

[4] $\lceil \cdot \rceil$ is ceiling function

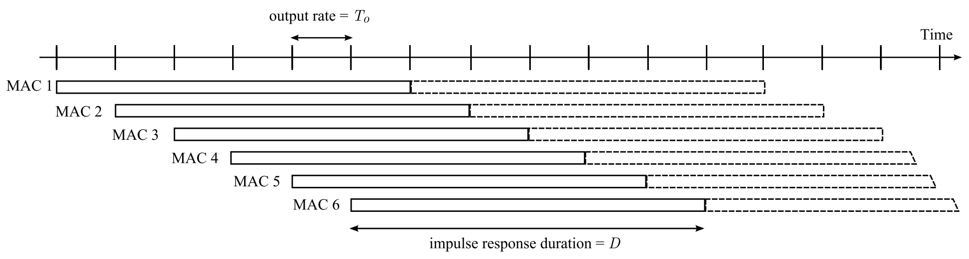


Figure 7.5: Timing diagram of several MAC units to continuously filter incoming signal

tors are needed for each MAC unit and we need N such units for a complete filter realization.

The digital commands for the MAC units are the same but a constant delay between them is needed. The necessary hardware to generate the delayed sequence is several (digital) latches as depicted in figure 7.6. The memory and the latches are clocked at the over sampling rate, i.e. reciprocal of Δt. Note that each single latch provides Δt delay and in general $T_o/\Delta t$ latches are needed to put between the MAC units in order to provide the necessary delay.

7.5 Design Details and Simulation Results

MAC unit is the main part of the proposed filter and thus it is important to be investigated more precisely. A MAC is implemented in Cadence^{TM} using AMS035 models. The aspect ratio for n-channel transistors is $1.2\mu m/1\mu m$. For p-channel transistors, the aspect ratio of $4.3\mu m/1\mu m$ is used in the main transconductor and g_{m2}. The aspect ratio of the p-channel transistors in g_{m3} is designed to be slightly lower i.e. $4\mu m/1\mu m$ to generate the required negative resistance. The number of quantization levels is arbitrary chosen to be $M = 16$ (4 bits) and the binary basis is again arbitrary selected to be {-1, 2, -4, 8} which is equivalent to the dynamic range of $[-5 \cdots 10]$. Using a supply voltage of 3.3 volt, the AC simulation for three different gains is shown in figure 7.7.

As expected, we obtain -20dB per decade curve with a $-\pi/2$ of phase that represents a precise integrator with a very large bandwidth. The low-frequency pole due to the output resistance (depends on the precision of negative resistance) can be as low as 10kHz and the second undesirable pole is located at about 4GHz. This pole is mainly due to the switching transistors that introduce internal nodes to the circuit. We can observe the linearity of the circuit using different gain values as depicted in figure 7.7 for the typical gains of 1, -5 and 10. It also shows the perfect phase behavior of the circuit for the negative gain.

Next simulation concerns a typical cosine rolloff filter design with sampling frequency = 160MHz, symbol rate = 40MHz and rolloff factor = 0.3. The filter taps were obtains from standard relationship of raised cosine formula. Then, they

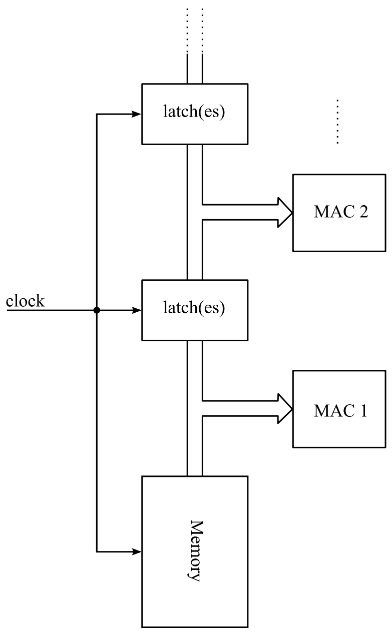


Figure 7.6: Digital circuitry in a complete filter serves tap values stored in digital memory and the required delay between MAC units by digital latch stack

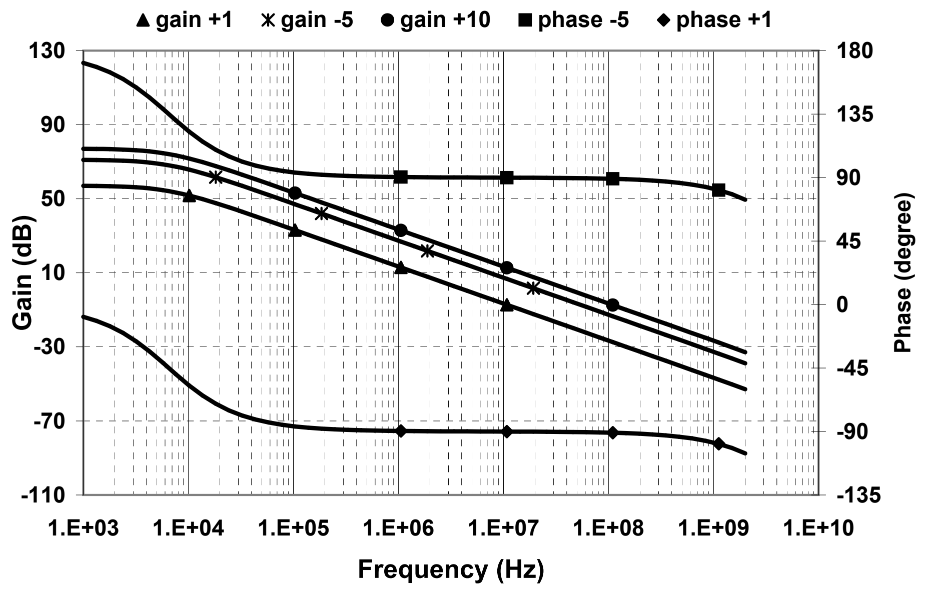


Figure 7.7: Frequency response of the MAC unit for three gain values

tap#	tap value	Code word	tap#	tap value	code word
1	0	0000	14	0.9398	0011
2	0.0783	1000	15	0.6265	0001
3	0.0783	1000	16	0.3133	0010
4	0.0783	1000	17	0	0011
5	0	0000	18	-0.1566	0100
6	-0.0783	1100	19	-0.1566	0100
7	-0.1566	0100	20	-0.0783	1100
8	-0.1566	0100	21	0	0000
9	0	0	22	0.0783	1000
10	0.3133	0010	23	0.0783	1000
11	0.6265	0001	24	0.0783	1000
12	0.9398	0011	25	0	0000
13	1.0181	1011	code word: $b_{LSB} - b_{MSB}$		

Table 7.2: Designed filter taps and the corresponding code words

were quantized by a 16-level uniform quantizer and the result mapped to binary number using the appropriate weights $\{1, -2, 4, 8\}$. The quantized tap values and the corresponding code words is depicted in table 7.2.

The filter's impulse response is truncated to $D = 6$ and output sampling rate is $T_o = 1$ (normalized time value). Therefore the MAC unit should be replicated six times in this design. Moreover, the integrating capacitors are discharged using a switching transistor at the beginning of each new integration interval.

Figure 7.8 shows the frequency response for this filter as well as a corresponding theoretical filter with the same number of quantization levels. It can be seen that the channel noise injection from the side lobs does not differ considerably for the practical and the theoretical filters; thereby the overall performances of a receiver with the proposed filter and theoretical filter are expected to be close to each other. The phase response is also quiet linear in pass band which is important in digital receivers.

Figure 7.9 shows the effect of quantization levels on Bit-Error-Rate (BER) of a receiver with our quantized-tap matched filter and compares it with theory, i.e. $P_{Error} = 0.5\,\text{erfc}\left(\sqrt{\text{SNR}/2}\right)$. This figure is drawn versus the SNR referred to the matched filter output. This proves that the proposed four-bit matched filter is quiet satisfactory for communication applications.

Another simulation is the time response of the practical filter and comparison with the response of an ideal filter with no quantized tap values. Figure 7.10 demonstrates the two curves obtain from simulation in Matlab and Cadence by setting a simulation as follows. An input Barker-13 sequence was shaped at the rate of 40MHz using a root raised cosine filter and was given to the theoretical and practical filters. The response of theoretical filter follows tightly the practical one. This shows that distortion of the proposed filter with only 4 bits of resolution is quiet acceptable.

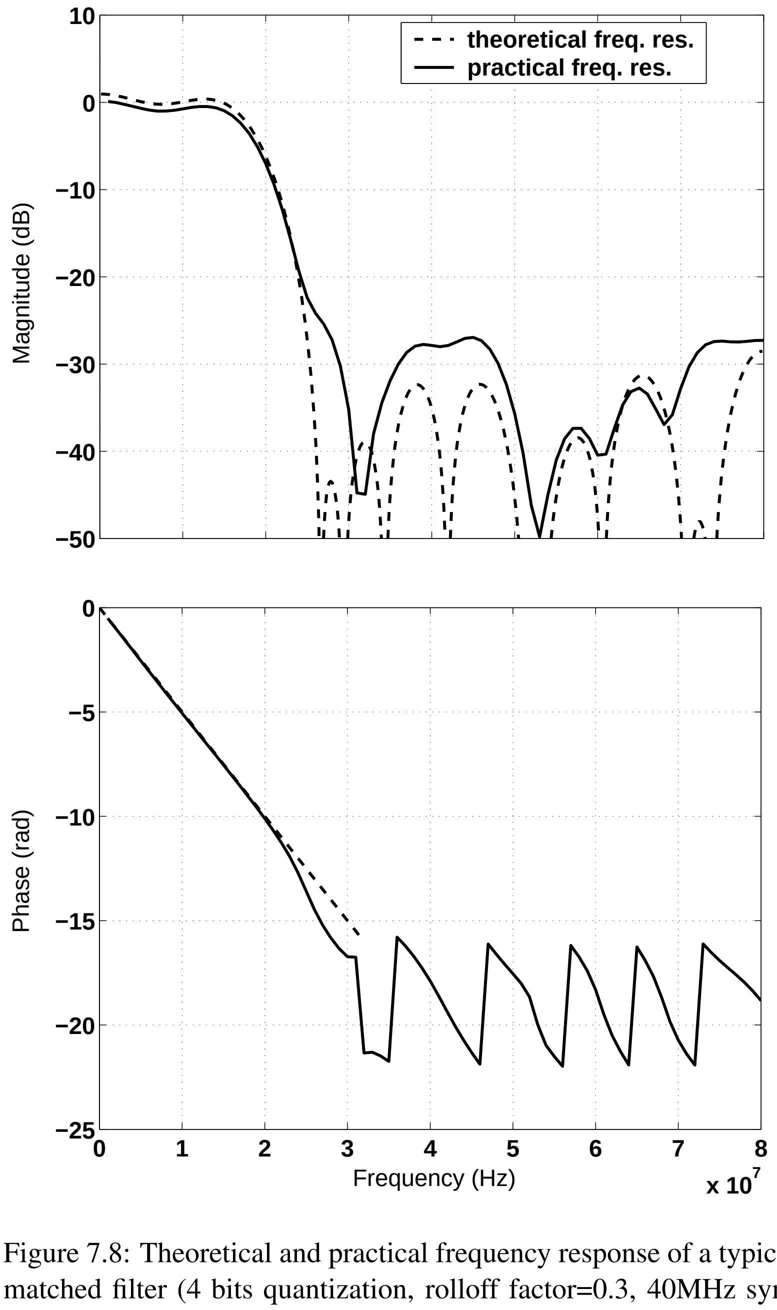


Figure 7.8: Theoretical and practical frequency response of a typical cosine rolloff matched filter (4 bits quantization, rolloff factor=0.3, 40MHz symbol rate, sampling frequency=160 MHz).

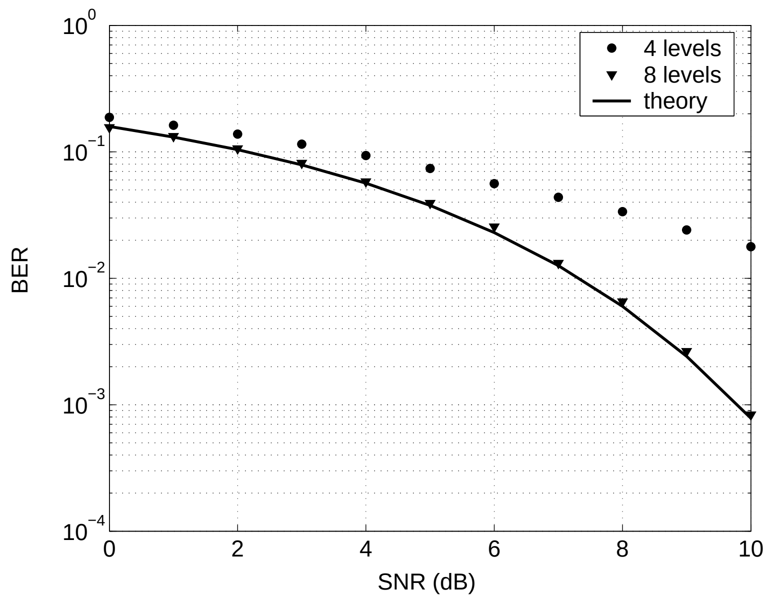


Figure 7.9: Bit error rate versus quantization levels (M) of a receiver with the proposed matched filter.

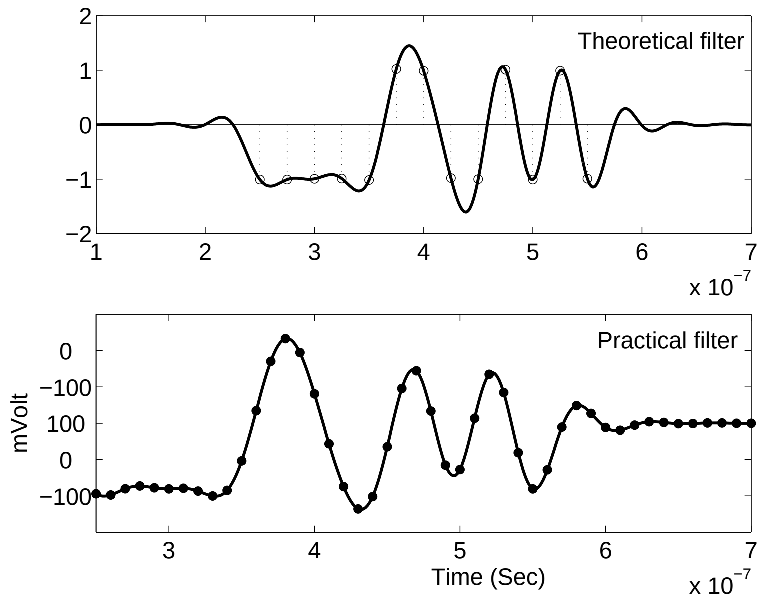


Figure 7.10: Time response simulation for an ideal root raised cosine filter (up) and a practical filter with 4-bit tap values. The circles in the upper figure shows Barker's stream and the filled circles in the lower figure are the output samples obtained from mixed-signal filter. They are interpolated for a better visualization. The residual norm (after normalization) of the figures is 0.117 in this case

Parameter	No. of Taps	Sampling Frequency	Static power consumption	Supply voltage	Technology
Analog [62]	8	20 MHz	13 mW	5 V	0.5 CMOS
ASIC [75]	24	20 MHz	328 mW	3 V	0.35 CMOS
TI [76]	32	20 MHz	400 mW	3.3 V	0.5 CMOS
FPGA [77]	8	20 MHz	210 mW	NA	NA
SI [78]	11	10 MHz	100 mW	5 V	2.0 CMOS
This work	25	40 MHz	17 mW	3.3 V	0.35 CMOS

Table 7.3: Comparison of the proposed filter with some other approaches

In our simulation each 4-bit MAC draws $858\mu A$ using a supply voltage of 3.3 volts. For $N = 6$ MAC units, the total power is about 17mW which is quite satisfying. Note that the power consumption has the logarithmic relationship, $\log_2(M)$, with the quantization levels M. Thus increase of M changes this power slightly. Table 7.3 summarizes the results of several recent works as well as this work. In comparison with commercial and digital implementations, e.g. ASIC, DSP (by Texas Instruments) and FPGA, great improvement in power consumption is achieved with the proposed filter. The analog and SI realizations consume lower power than the digital realizations. However, the power consumption of our proposed filter is considerably lower than SI approach. In comparison with the analog filter our mixed-signal structure is more interesting by considering its more number of taps, lower supply voltage and higher operating frequency as well as having no S/H in signal path.

7.6 conclusions

In this novel design of FIR filter, we have gained at a same time the flexibility of a digital filter, the simplicity of a CMOS circuit and low power consumption. The design is totally based on CMOS inverter, exploited differentially to achieve a good transconductor without any internal node. Signal path in the design remains analog while filter taps are quantized and store digitally. With a known filter impulse response, one can do far beyond a simple quantization task. For example a non-uniform and optimal quantization is possible. We had proposed a new binary system that effectively reduces the quantization error still with a uniform quantization.

No internal node means that there is not any pole in the circuit. This guarantees a very large bandwidth for the circuit with conventional CMOS technology.

Circuit level simulation of the designed filter and comparing the results with theory proved its performance and revealed an interesting value for its power consumption with respect to some recently reported filters.

Chapter 8

Conclusions and Perspective

In the context of convolutional codes, a class of versatile decoders based on Tanner graph for tail-biting constraint has been addressed. Also the condition for which tail-biting is valid was mentioned. Analog realization and simulation of corresponding decoder have been considered and the performance of decoder was compared with classical benchmark.

The effect of optimization of design parameters and its effect on the overall performance of our decoders have been noticed by a comparison between BER simulations. It was shown that optimization step is crucial in design of analog decoders in order to achieve a result near to the performance theoretical algorithm.

Analog Viterbi decoders has been studied and an analog circuit has been proposed to be replaced with Add-Compare-Select block. It has been shown that the proposed circuit is rather simple to implement while the achieved speed is in the range of nanoseconds. Practical limitations and non ideal behaviors have been considered during a behavioral simulations. Like other analog approaches in coding domains, this design also unrestricts decoders to be equipped with analog-to-digital converters.

The last chapter proposed a new CMOS based simple structure for the mixed-signal realization of FIR filters with high-frequency characteristics and suitable for digital receivers. A sample cosine rolloff filter was designed and implemented in Cadence using the proposed scheme. Frequency response, time domain and BER simulation have been performed and the result was compared to the theoretical benchmark. The results show the versatility of the proposed filter as a matched filter in digital receivers.

The results obtained so far, gives motivation for studying the issue of analog approaches in a broader range. For example in time synchronization algorithms, analog schemes are expected to be useful. There are also ideas about LLR extraction in a M-ary signaling and MIMO systems that need more effort to extend these concepts.

According to the best of our knowledge, innovations at the circuit level are restricted to two or three ideas since the invention of analog decoding. After the main idea proposed by Hagenauer [41] and Loeliger etal. [42], the design of Chris Winstead [46] for its contribution on low voltage design and the idea introduced by Frey [40] for a compact design of graph's node is remarkable. Also the idea of Hemati [79] who for the first time employed current-mode circuitry for this propose is interesting. In order to have a compact and low power decoder for still lager codes, much effort at circuit level design is required.

Setting up a more complete system by putting together the blocks based on the idea of mixed-signal filtering and analog decoding as suggested in figure 7.1c is a big deal toward a complete system.

Making the ideas become operational by implementing them on a chip and getting realistic measurement data is also necessary.

Incorporating other algorithms that better match with the proposed analog Viterbi kernel, say, modified feedback decoding algorithm (MFDA) [80] can relax the hardware complexity needed for trace back operation.

Extending the analog scheme to a more complex situation in which there is a channel with memory, in a sense, something like the work in [81].

Bibliography

[1] G. Ungerboeck, “Channel coding with multilevel/phase signals,” *IEEE Trans. Inform. Theory*, 1982.

[2] A. B. Carlson, *COMMUNICATION SYSTEMS*, 3rd ed. McGraw-Hill, 1986.

[3] F. Lustenberger, “On the design of analog VLSI iterative decoders,” Ph.D. dissertation, ETH Zürich, 2000.

[4] C. Winstead, J. Die, S. Little, C. Myers, and C. Schlegel, “Analog MAP decoder for (8, 4) hamming code in subthreshold CMOS,” Mar 2001. [Online]. Available: citeseer.ist.psu.edu/article/winstead01analog.html

[5] A. G. I. AMAT, S. BENEDETTO, G. MONTORSI, A. NEVIANI, A. GEROSA, and D. VOGRIG, “Design, simulation, and testing of a CMOS analog decoder for the block length-40 UMTS turbo code,” *IEEE transactions on communications*, vol. 54, no. 11, pp. 1973 – 1982, nov 2006.

[6] M. ARZEL, C. LAHUEC, F. SEGUIN, M. JEZEQUEL, and D. GNAEDIG, “Semi-iterative analog turbo decoding,” *IEEE transactions on circuits and systems I - regular papers*, vol. 54, no. 6, pp. 1305 – 1316, june 2007.

[7] M. Mörz, T. Gabara, R. Yan, and J. Hagenauer, “An analog 0.25 μm BiCMOS tail-biting MAP decoder,” in *Proc. IEEE Int. Solid-State Circuits Conf.*, Feb 2000, pp. 356–357.

[8] V. C. Gaudet and P. G. Gulak, “A 13.3 Mb/s 0.35 μm CMOS analog turbo decoder IC with a configurable interleaver,” *IEEE J. Solid-State Circuits*, vol. 38, pp. 2010–2015, Nov 2003.

[9] M. R. Zahabi, V. Meghdadi, H. Meghdadi, and J. P. Cances, “Versatile graphs for tail-biting convolutional codes,” in *IEEE International Symposium on Circuits and Systems*, May 2008, pp. 216–219.

[10] M. R. Zahabi, V. Meghdadia, and J.-P. Cances, “Analog decoding of tail-biting convolutional codes based on Tanner graph,” *IET Electronic Letters*, vol. 42, pp. 1167–1168, Sep 2006.

[11] M. R. Zahabi, V. Meghdadi, J. P. Cances, and A. Saemi, “A mixed-signal matched-filter design and simulation,” in *15th International Conference on Digital Signal Processing*, Jul 2007, pp. 272–275.

[12] M. R. Zahabi, V. Meghdadia, J.-P. Cances, and A. Saemi, “A mixed-signal matched-filter for high rate communication systems,” *IET Signal Processing Journal*, submitted for publication.

[13] M. R. Zahabi, V. Meghdadi, J. P. Cances, and A. Saemi, “Mixed analog and digital matched-filter design for high rate WLAN,” in *Global Telecommunications Conference*, Nov 2007, pp. 310 – 314.

[14] M. R. Zahabi, V. Meghdadi, J. P. Cances, A. Saemi, J. Dumas, and B. Barelaud, “Analog CMOS kernel for ML decoding of convolutional codes,” in *International conference on electrical engineering, ICEE*, Tehran, IRAN, Mar 2006.

[15] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, “Optimal decoding of linear codes for minimizing symbol error rate,” *IEEE Trans. Inform. Theory*, vol. IT-20, pp. 284–287, 1974.

[16] M. Xiao and A. Kavcic, “Path partitions and forward-only trellis algorithms,” *IEEE Trans on Information Theory*, vol. 49, pp. 38–52, 2003.

[17] A. J. Felström and K. S. Zigangirov, “Time-varying periodic convolutional codes with low-density parity-check matrix,” *IEEE Trans. Inform. Theory*, 1999.

[18] R. M. Tanner, D. Sridhara, A. Sridhara, T. Fuja, and D. J. Costello, “LDPC block and convolutional codes based on circulant matrices,” *IEEE Trans on Information Theory*, vol. 50, pp. 2966–2984, 2004.

[19] A. Sridharan, “Design and analysis of LDPC convolutional codes,” Ph.D. dissertation, University of Notre Dame, February 2005.

[20] F. R. Kschischang and V. Sorokine, “On the trellis structure of block codes,” *IEEE Transaction on Information Theory*, vol. 41, no. 6, pp. 1924–1937, nov 1995. [Online]. Available: www.comm.utoronto.ca/frank/

[21] R. J. McEliece, “On the BCJR trellis for linear block codes,” *IEEE TRANSACTIONS ON INFORMATION THEORY*, vol. 42, no. 4, jul 1996.

[22] A. V. Nori, “Unifying views of tail-biting trellises for linear block codes,” Ph.D. dissertation, Indian Institute of Science, sep 2005.

[23] L. R. Rabiner, “A tutorial on hidden Markov models and selected applications in speech recognition,” *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[24] M. I. Jordan, “Graphical models,” *Berkeley 94720*, 2003.

[25] R. G. Gallager, “Low-density parity-check codes,” *IRE Trans. Information Theory*, 1962.

[26] R. M. Tanner, “A recursive approach to low complexity codes,” *IEEE Trans. Inform. Theory*, 1981.

[27] D. J. C. MacKay and R. M. Neal, “Good codes based on very sparse matrices,” in *Cryptography and Coding*, 1999.

[28] N.Wiberg, “Codes and decoding on general graphs,” Ph.D. dissertation, Univ. Linköping, Linköping, Sweden, 1996.

[29] N. Wiberg, H. A. Loeliger, and R. Kötter, “Codes and iterative decoding on general graphs,” *Euro. Trans. Telecomm.*, 1995.

[30] R. J. McEliece, D. J. C. MacKay, and J. F. Cheng, “Turbo decoding as an instance of Pearl’s belief propagation algorithm,” *IEEE J. Select. Areas Commun.*, 1998.

[31] F. R. Kschischang and B. J. Frey, “Iterative decoding of compound codes by probability propagation in graphical models,” *IEEE J. Select. Areas Commun.*, 1998.

[32] G. Forney, “Codes on graphs: normal realizations,” *IEEE Transactions on Information Theory*, 2001.

[33] J. Hagenauer, E. Offer, and L. Papke, “Iterative decoding of binary block and convolutional codes,” *IEEE Trans. on Information Theory*, vol. 42, pp. 429–445, March 1996.

[34] F. R. Kschischang, B. Frey, and H.-A. Loeliger, “Factor graphs and the sum-product algorithm,” *IEEE Trans on Information Theory*, vol. 51, pp. 498–515, February 2001.

[35] R. McEliece, E. Rodemich, and J. Cheng, “The turbo decision algorithm,” in *Proc. 33rd Allerton Conference on Communications, Control and Computing*, Monticello, IL, 1995, pp. 366–379.

[36] Y. Weiss and W. T. Freeman, “Correctness of belief propagation in gaussian graphical models of arbitrary topology,” 1999.

[37] F. Guilloud, “Generic architecture for LDPC codes decoding,” Ph.D. dissertation, Télécom Paris, July 2004.

[38] J. Dai, “Design methodology for analog VLSI implementations of error control decoders,” Ph.D. dissertation, The University of Utah, December 2002.

[39] C. Weiss, C. bettstetter, S. Riedel, and D. J. Costello, “Turbo decoding with tail-biting trellises,” in *URSI Int. sym. Sig. Sys. and Elec.*, 1998.

[40] M. Frey, “On analog decoders and digitally corrected converters,” Ph.D. dissertation, Swiss Federal Institute of Technology, April 2006.

[41] J. Hagenauer, “Decoding of binary codes with analog networks,” in *Proc. Info. Theory Wksp.*, San Diego, CA, Feb 1998, pp. 13–14.

[42] H.-A. Loeliger, F. Lustenberger, M. Helfenstein, and F.Tarkoy, “Probability propagation and decoding in analog VLSI,” in *Proc. Int. Symp. Information Theory*. Cambridge, MA, 1998, p. 146.

[43] M. Frey, H.-A. Loeliger, F. Lustenberger, P. Merkli, and P. Strebel, “Analog-decoder experiments with subthreshold CMOS soft-gates,” in *Proc. IEEE Conf. On Circuits and Systems*, 2003, pp. 85–88.

[44] F. Lustenberger, M. Helfenstein, H.-A. Loeliger, F. Tarköy, and G. S. Moschytz, “All analog decoder for a binary (18,9,5) tail-biting trellis code,” in *European Solid-State Circuits Conference*, 1999, pp. 362–365.

[45] F. J. L. R. Bahl, J. Cocke and J. Raviv, “Optimal decoding of linear codes for minimizing symbol error rate,” *IEEE Trans on Information Theory*, vol. 20, pp. 284–287, March 1974.

[46] C. Winstead, “Analog iterative error control decoders,” Ph.D. dissertation, University of Alberta, 2005.

[47] B. Streetman, *Solis state electronic devices*. Prentice Hall, 2000.

[48] B. Razavi, *Design of analog CMOS integrated circuits*. McGraw-Hill, 2001.

[49] S. Aghtar, J. Haslett, and F. N. Trofimenkoff, “Subthreshold analysis of an MOS analog switch,” *IEEE Transactions on Electron Devices*, vol. 44, pp. 86–91, 1997.

[50] R. M. Swanson and J. D. Meindl, “Ion-implanted complementary MOS transistor in low-voltage circuits,” *IEEE J Solid-State Circuits*, vol. SSC-7, pp. 146–153, 1972.

[51] B. Frey and F. Loeliger, “Factor graphs and algorithms,” in *35th Alteron Conf on Communications, Contol, and Computing*, September 1997, pp. 666–680.

[52] P. J. Black and T. H.-Y. Meng, “A 1-Gb/s, four-state, sliding block Viterbi decoder,” *IEEE Journal of Solid-State Circuits*, vol. 32, no. 6, pp. 797–805, 1997.

[53] J. Ou and M. Prasanna, “Time and energy efficient Viterbi decoding using FPGAs,” in *IEEE International Conference on Acoustics, Speech, and Signal Processing vol. 5, pp. V/33-v/36, March 2005.*, vol. 5, 2005, pp. v/33–v/36.

[54] J. Tang and K. Parhi, “Viterbi decoder for high-speed ultra-wideband communication systems,” in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 5, March 2005, pp. v/37–v/40.

[55] S.-W. Choi and S.-S. Choi, “200 Mbps Viterbi decoder for UWB,” in *The 7th International Conference on Advanced Communication Technology*, 2005.

[56] H.Kim, H. Son, T. Roska, and L. Chua, “High-performance Viterbi decoder with circularly connected 2-D CNN unilateral cell array,” *IEEE Transactions on Circuits and Systems*, vol. 52, no. 10, pp. 2208–2218, Oct 2005.

[57] L. Dong, S. Wentao, L. Xingzhao, L. Hanwen, X. Youyun, and Z. Wenjun, “Neural networks based parallel Viterbi decoder by hybrid design,” in *Fifth World Congress on Intelligent Control and Automation*, vol. 3, Jun 2004, pp. 1923–1926.

[58] J. Lazzaro, S.Ryckebusch, M. Mahowald, and C. A. Mead, “Winner-take-all networks of order N complexity,” in *Proc. IEEE Conf. On Neural Information Processing - Natural and Synthetic*, Denver, 1988.

[59] E. A. Vittoz, “Analog VLSI signal processing:why, where and how,” *Journal of VLSI Signal Processing*, vol. 8, Oct 1994.

[60] A. P. Hekstra, “An alternative to metric rescaling in Viterbi decoders,” *IEEE Transactions on Communications*, vol. 37, no. 11, pp. 1220–1222, Nov 1989.

[61] L.-R. Zheng, X. Duo, M. Shen, W. Michielsen, and H. Tenhunen, “Cost and performance tradeoff analysis in radio and mixed-signal system-on-package design,” *IEEE Trans. on Advanced Packaging*, vol. 27, no. 2, pp. 364–375, May 2004.

[62] V. Srinivasan, G. Rosen, and P. Hasler, “Low-power realization of FIR filters using current-mode analog design techniques,” in *Thirty-Eighth Asilomar Conference on Signals, Systems and Computers*, vol. 2, Nov 2004, pp. 2223–2227.

[63] J. M. Rocha-Perez and J. Silva-Martinez, “SC implementation of FIR filters for digital communication systems,” in *Third International Workshop on Design of Mixed-Mode Integrated Circuits and Applications*, Jul 1999, pp. 179–182.

[64] N. Battersby and C. Toumazou, “Class AB switched-current memory for analog sampled data systems,” *Electronics Letters*, vol. 27, pp. 873–875, May 1991.

[65] C. Toumazou, N. Battersby, and M. Punwani, “GaAs switched-current techniques for front-end analogue signal processing applications,” in *Proceedings of the 35th Midwest Symposium on Circuits and Systems*, vol. 1, Aug 1992, pp. 44–47.

[66] E. Ozalevli, W. Huang, P. Hasler, and D. Anderson, “VLSI implementation of a reconfigurable mixed-signal finite impulse response filter,” in *IEEE International Symposium on Circuits and Systems*, May 2007, pp. 2168–2171.

[67] Y. L. Cheung and A. Buchwald, “A sampled-data switched-current analog 16-tap FIR filter with digitally programmable coefficients in 0.8μ CMOS,” in *IEEE International Solid-State Circuits Conference*, 1997, pp. 54–55.

[68] K. Bult and G. Geelen, “An inherently linear and compact most-only current division technique,” *IEEE Journal of Solid-State Circuits*, vol. 27, pp. 1730–1735, Dec 1992.

[69] A. M. Chiang, “Low-power adaptive filter,” in *IEEE International Solid-State Circuits Conference*, Feb 1994, pp. 90–91.

[70] J. Hagenauer and M. Winkelhofer, “The analog decoder,” in *Proc. Int. Symp. on Information Theory*. Cambridge, MA, 1998, p. 145.

[71] R. Storn, “Designing nonstandard filter with differential evolution,” *IEEE Signal Processing Magazine*, vol. 22, no. 1, pp. 103–106, Jan 2005.

[72] B. Nauta, “A CMOS transconductance-C filter technique for very high frequencies,” *IEEE Journal of Solid-State Circuits*, vol. 27, no. 2, pp. 142–153, Feb 1992.

[73] *ENG-228,* $0.35\mu m$ *C35 CMOS Matching Parameters and ENG-182,* $0.35\mu m$ *C35 CMOS Process Parameters*, Austriamicrosystems, rev. 2.0.

[74] K. A. Bowman, B. L. Austin, J. C. Eble, X. Tang, and J. D. Meindl, “A physical alpha-power law MOSFET model,” *IEEE Journal of Solid-State Circuits*, vol. 34, no. 10, pp. 1410–1414, Oct 1999.

[75] A. Erdogan, E. Zwyssig, and T. Arslan, “Architectural trade-offs in the design of low power FIR filtering cores,” in *IEE Proceedings on Circuits, Devices and Systems*, vol. 151, Feb 2004., pp. 10–17.

[76] *GC2011A-3.3V digital chip*, Texas Instruments Datasheet.

[77] G. Cardarilli, A. Re, A. Nannarelli, and M. Re, “Power characterization of digital filters implemented on FPGA,” in *Proceedings of the International Symposium on Circuits and Systems*, vol. 5, May 2002, pp. V.801–V.804.

[78] G. Liang and D. Allstot, “FIR filtering using CMOS switched-current techniques,” in *Proceedings of the International Symposium on Circuits and Systems*, vol. 3, May 1990, pp. 2291–2293.

[79] S. Hemati and A. H. Banihashemi, “Convergence speed and throughput of analog decoders,” *IEEE Transactions on Communications*, vol. 55, no. 5, pp. 833–836, 2007.

[80] A. Demosthenous and B. V. Tomatsopoulos, "Effects of decoding depth on the performance of the modified feedback decoding algorithm for convolutional codes," in *London Communication Symposium*, London, UK, 2003.

[81] C. Lahuec, G. L. Mestre, M. Arzel, F. Seguin, and M. Jézéquel, "Design and test of a 0.25-μm BiCMOS doublebinary analog APP decoder," in *Proceeding of Analog Decoding Workshop*, Jun 2006, pp. 35–38.

List of Figures

List of Tables

ANALOG APPROACHES IN DIGITAL RECEIVERS

Abstract: Modern digital receivers need computationally demanding processes that leads to prohibitive complexity and power consumption. The idea of lending analog blocks for realization of digital algorithms can sometimes relaxes the complexity and high power consumption of digital receivers. The issue of analog approaches in digital receivers is studied in this dissertation by concentrating on two areas; analog decoding and front-end processing.

For analog decoding, the realizations of some efficient decoders are presented along which our contribution in this area in conjunction with graph theory is proposed. In addition, analog realization of a fast Viterbi decoder is considered. It is shown that there is a very nice analog solution for realization of Add-Compare-Select that plays the central rule in Viterbi algorithm. In order to justify the proposed analog decoders, Cadence package is used.

For front-end processing, a novel mixed-signal programmable filter is designed and investigated. The filter is suitable for high-rate communication systems. The proposed filter has analog input and analog sampled outputs. The filter is based on simple CMOS inverter and thus can be integrated efficiently with digital parts.

Keywords: Analog Decoder, Graph, Sum Product Algorithm, Belief Propagation, MAP, Trellis, Tailbiting, Analog Viterbi, FIR Filter, Mixed Signal

APPROCHES ANALOGIQUES DANS LES RECEPTEURS NUMERIQUES

Résumé : Cette thèse propose d'utiliser des circuits analogiques pour réaliser des algorithmes numériques. Le but étant de diminuer la complexité et la puissance consommée et augmenter la vitesse. Deux applications gourmandes en temps de calcul ont été considérées dans cette thèse : le décodeur et le filtre RIF.

On propose une structure analogique CMOS très efficace pour un décodeur Viterbi et pour un décodeur sur les graphes de Tanner. Les structures proposées ont été implantées et testées sous l'outil Cadence et démontre la validité de notre démarche.

Quant au traitement de signal à l'entrée de décodeurs, un filtre RIF programmable utilisant la technologie CMOS a été étudié, conçu et implanté. La structure proposée est bien adapté aux systèmes de communications haut-débits. Le filtre possède une entrée analogique et une sortie échantillonnée, basée sur un simple inverseur CMOS et peut donc être intégré de manière efficace avec les parties numériques sur une seule puce.

Mots Clés : Décodeur Analogique, Graphe, Propagations de Croyance, Algorithme de Viterbi, Filtrage RIF, Implantation Mixte