UNIVERSITÀ DEGLI STUDI DI PADOVA
DIPARTIMENTO DI INGEGNERIA
DELL'INFORMAZIONE

DOTTORATO DI RICERCA IN INGEGNERIA
ELETTRONICA E DELLE TELECOMUNICAZIONI

XXI CICLO

Design of low-power analog circuits for analog decoding and wireless sensors nodes

DOTTORANDO:

Silvia Solda'

SUPERVISORE:

Prof. Andrea Neviani

COORDINATORE:

Ch.mo Prof. Matteo Bertocco

Padova, 31 maggio 2009

Posta elettronica : solda@dei.unipd.it

Numero telefonico : +39 049 827 7664

Indirizzo :

Dipartimento di Ingegneria dell’Informazione
Università degli Studi di Padova
via Gradenigo 6/B
35131 Padova
Italia

Sommario

La prima parte di questo lavoro di tesi e' dedicata alla decodifica analogica e presenta la progettazione di un'interfaccia di I/O per un decodificatore iterattivo completamente analogico per un codice convoluzionale concatenato in serie e di un decoder analogico per Trellis Coded Modulation (TCM) per la correzione degli errori in memorie Flash multilivello. Il decodificatore iterattivo rappresenta un grosso passo avanti nell'evoluzione dei decodificatori analogici in quanto e' possibile riconfigurarne sia la lunghezza di blocco che il rate del codice. Per di piu', con un'efficienza di 2.1nJ/bit, migliora fino a 50 volte le prestazioni in termini di efficienza dei decodificatori digitali con la stessa lunghezza di blocco. Le potenziali prestazioni e le limitazioni dell'approccio analogico per un decodificatore per TCM sono state investigate considerando due diversi decodificatori, uno a 4 stati ed uno ad 8 stati, entrambi sviluppati in un processo CMOS standard con una lunghezza di canale di 0.18 μm.

Nella seconda parte della tesi viene presentato il design di un transciver per una radio ad impulsi a banda larga (UWB-IR), con particolare enfasi sulla progettazione del trasmettitore. Il trasmettitore utilizza una nuova combinazione di mixer e amplificatore di potenza per generare un impulso gaussiano con una larghezza di banda di 1.25GHz ed una frequenza centrale di 7.875GHz. Il nuovo circuito, inoltre, include un trasformatore monolitico in modo tale da generare una tensione di uscita di $3.2V_{pp}$, necessaria per garantire la distanza di connessione richiesta di almeno 10 metri. Il trasformatore e' stato progettato in modo da massimizzare l'efficienza in termini di potenza e, allo stesso tempo, realizzare un filtro ladder del quarto ordine al fine di ridurre le emissioni fuori banda del trasmettitore stesso. Confrontando l'efficienza di questo design con trasmettori per UWB-IR allo stato dell'arte si e' visto come la soluzione da noi proposta porti ad un miglioramento dell'efficienza del trasmettitore di un fattore pari a 10.

Abstract

The first part of this work concerns analog decoding. It presents the design of the I/O interface for a fully analog iterative decoder for a serially concatenated convolutional code and of a fully analog Trellis Coded Modulation (TCM) decoder for error correction in multi-level (ML) flash memories. The iterative decoder represents a significant step ahead in the evolution of analog decoders due to its reconfigurability in both block length and code rate. Moreover, with an efficiency of 2.1nJ/bit, it outperforms digital decoders with the same block length of a factor up to 50. The potential performance and limitations of the analog approach for a TCM decoder have been investigated considering a 4-state and an 8-state decoder, both developed in a 0.18 μm standard CMOS process.

In the second part of the thesis, the design of a low-power transceiver chipset for ultra wideband impulse radio (UWB-IR) is presented, with particular emphasis on the transmitter design. In particular, the transmitter uses a novel combined mixer and power amplifier to generate a Gaussian pulse with 1.25GHz bandwith and center frequency of 7.875GHz. The combined MRX-PA includes a monolithic transformer to reach a maximum output voltage swing of $3.2V_{pp}$, necessary to ensure the required link distance of 10 meters. The transformer has been designed in order to maximize the power efficiency and at the same time to realize a fourth-order ladder filter, so as to reduce the transmitter out-of band emissions. The efficiency of this design has been compared with state-of-the-art UWB-IR transmitters, showing how the proposed solution leads to an improvement in the transmitter efficiency of a factor of almost 10.

Introduction

The density and speed of integrated circuit computing elements has increased roughly exponentially for a period of several decades, following a trend described by Moore's Law. While it is generally accepted that this exponential improvement trend will end, it is unclear exactly how dense and fast integrated circuits will get by the time this point is reached. Working devices have been demonstrated that were fabricated with a MOSFET transistor channel length of 6.3 nanometers using conventional semiconductor materials, and devices have been built that used carbon nanotubes as MOSFET gates, giving a channel length of approximately one nanometer.

The density and computing power of integrated circuits are limited primarily by power dissipation concerns. Even if several techniques have been developed to reduce it, however, if current trends continue, "Energy costs, now about 10% of the average IT budget, could rise to 50% ... by 2010" [1].

In this thesis, we cope with the problem of reducing power consumption in two different key application fields: the design of decoders for both Turbo codes and multilevel Flash memories error correcting codes and the realization of low-power transceivers for ultra-wideband (UWB) impulse radio (IR).

Turbo codes, first proposed by Berroux and Glavieux in 1993 [2], have become extremely popular in the last few years till the point to be adopted as standard codes for a wide range of telecommunication applications, such as 3G/UMTS cellular phones [3] and satellite digital video broadcasting [4]. At the same time, Flash memory market growth has been explosive in the past decade, driven by cellular phones and other types of electronic portable equipment, such as palm top, portable PC, mp3 audio player, digital camera and so on. Thus, reducing the decoders power consumption for such applications is already crucial and will become more and more decisive in the next few years.

UWB-IR have become an active research area with proliferation of portable electronics, as it promises unprecedent data rates for short-range commercial radios, combined with precise locationing and high energy efficiency. These benefits stem from the use of wide bandwidths and impulse signaling, implying high channel capacity and precise time resolution [5]. A critical specification for energy efficient short-range radio is, of course, the energy per bit, that is the energy spent to transmit and receive a single information bit.

This thesis consists of two parts: in the first we address the decoding problem, while in the second the design of a chipset for UWB-IR is presented.

The pioneering work of Loeliger's [6] and Hagenauer's [7] groups led to the first successful implementations of analog iterative decoders [8–10] in BiCMOS technology, and demonstrated the potential advantages of this approach with respect to the digital one both in terms of decoding speed and power efficiency. However, the limitations of the analog implementation, due to the fully parallel decoding process, have soon become clear. The only way to make the analog decoder circuitry complexity independent on the codeword length is to reduce the fully parallelism of data processing by introducing the concept of sliding window decoding [11, 12]. Following this decoding strategy, an *hybrid* analog decoder for Turbo codes has been designed, which combines the advantages offered by the analog approach in terms of power consumption together with those typical of the digital implementations, such as a reduced area occupation and a greater versatility.

Using the sliding window decoding approach, the decoder interface circuitry becomes the bottleneck in terms of power consumption ad area occupation, as we will see in Chapter 1. In fact, in order to store the channel data and to exchange information during the iterative decoding process, two large power hungry analog memories are needed.

Thus, as a significant effort in the design of complex analog decoder is spent in the realization of the interface circuitry, analog decoders could be successfully used in all those applications where a memory is already implemented, such as within stand-alone Flash memory chips. Following this consideration, a novel TCM analog decoder for next generation multilevel Flash memories has been designed, with encouraging performance both in terms of area occupation and power consumption with respect to state-of-the-art digital decoders.

In Chapter 1, the architecture and main features of an iterative fully analog decoder for a serially concatenated convolutional code, which implements the sliding window concept are presented. In particular, the specifications for the decoder input interface circuitry, which consists of an analog memory, a voltage to probability converter and a digital to analog converter, are drawn and its design and optimization are then analyzed in details.

Chapter 2, after a brief introduction to multilevel Flash memories, analyzes the advantages and drawbacks of commonly used error correction codes in order to derive a novel ECC scheme based on Trellis Coded Modulation.

In Chapter 3, the complete design of an analog decoder for the TCM ECC proposed

in Chapter 2 is presented. In particular, the potential performance and limitations of the analog approach are investigated considering a 4-state and an 8-state analog TCM decoder, both designed for an effective storage density of 3 information bits/cell. Transistor-level simulations of the overall decoders, including the circuit interface between the Flash memory cells and the decoder core, show how the proposed approach can achieve a decoding speed comparable with the state-of-the-art linear block codes occupying a small area, with almost no loss in terms of BER with respect to the ideal decoding algorithm.

In order not to lengthen the thesis too much, some basic concepts of coding theory are introduced in the Appendix A. The idea is to give the reader only the background information necessary to understand the projects described in this work. Thus, particular emphases is placed on the analog decoding approach for Trellis Coded Modulation and Turbo codes. Interested readers can refer to the bibliography for a more detailed discussion on coding theory.

The second part of the thesis is devoted to the design of a low-power transceiver chipset for UWB-IR.

Chapter 4 briefly introduces UWB signalling systems together with some legal aspects due to the Federal Communications Commission restriction on the transmitted power spectral density. Then the system analysis of our chipset for UWB-IR sensor networks is carried out in order to draw the specifications for both receiver and transmitter. For this purpose, a behavioral model of the receiver implemented in UMC 0.13-μm RFCMOS process has been developed, which also allowed to successfully test a synchronization algorithm.

The design of the transmitter is then presented in details in Chapter 5, where the comparison with state-of-the-art UWB-IR transmitters shows how the proposed solution leads to an improvement in the transmitter efficiency of a factor of almost 10.

Contents

List of Tables

List of Figures

1

Input Interface for Analog Decoders

Although analog decoders demonstrated large improvements in terms of power consumption, chip area and throughput with respect to their digital counterparts [8, 9, 13, 14], they are still far from being used in realistic applications due mainly to the fact that their circuitry complexity increases linearly with the codeword length. The only possible solution to this limitation is to reduce the fully parallelism of data processing by introducing the concept of sliding window decoding [11, 12].

In this chapter, the architecture and main features of an iterative fully analog decoder for a serially concatenated convolutional code, which implements the sliding window concept, are presented. In particular, the specifications for the decoder input interface circuitry, which consists of an analog memory, a voltage to probability converter and a digital to analog converter, are drawn and then its design and optimization are analyzed in details.

1.1 Hybrid Turbo Decoder

The input interface circuitry whose design will be described in Sec.1.2.3, is part of an *hybrid* Turbo decoder which combines the advantages of the analog approach, by implementing a fully analog decoding core, together with some features typical of the digital implementations, such as the iterative use of the same computational hardware unit and an increased flexibility both in terms of code rate and frame length.

1.1.1 Serially Concatenated Code Structure

The hybrid Turbo decoder is designed to decode in the analog domain a novel serially concatenated convolutional code recently proposed in literature for satellite applications

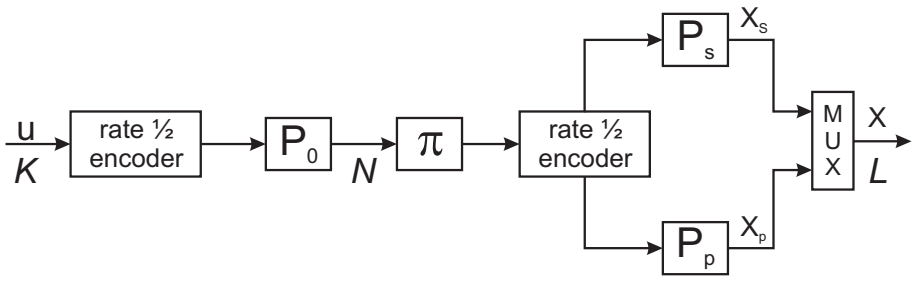


Figure 1.1: Block diagram of the serially constituent code scheme

[15] due to its high performance, simplicity and versatility.

The code block diagram is depicted in Fig.1.1. It consists of the concatenation of two identical rate-1/2 4-state systematic recursive convolutional codes with generator polynomials:

$$G(D) = \left[D^2+1, D^2+D+1\right] \tag{1.1}$$

The two encoders scheme, together with a code trellis section, are shown in Fig.1.2.

A way to increase any code rate is offered by "puncturization". A punctured code is a higher rate code formed by discarding or puncturing specific codeword symbols of the output of a low-rate code. Given the original low-rate code, the resulting punctured code depends on the pattern of codeword symbols being discarded. This pattern is called the perforation pattern of the punctured code and it can be conveniently defined by a perforation matrix P with elements

$$p_{i,j} = \begin{cases} 0 & \text{if symbol } i \text{ of the branch } j \text{ is punctured} \\ 1 & \text{if symbol } i \text{ of the branch } j \text{ is retained} \end{cases}$$

In our case, the outer code is punctured to rate 2/3 through the puncturing matrix

$$P_o = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix} \tag{1.2}$$

while the inner code systematic bits and parity bits are punctured separately through puncturers P_s and P_p, as described in detail in [15]. By changing the perforation patter of the two puncturers P_s and P_p in a rate-compatible fashion, the overall code rate can be varied between 1/3 and (virtually) 1.

Let's define K, N and L the information block size, the interleaver size and the codeword size respectively. The binary information data, collected in a vector $\mathbf{u}$ of length

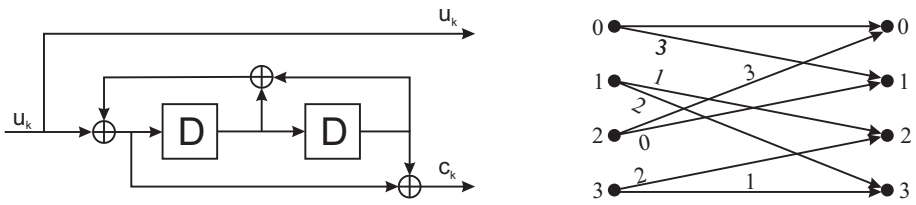


Figure 1.2: Constituent encoder scheme of the SCCC Turbo code

K, is first encoded by the outer encoder, then punctured by the fixed puncturer P_o and interleaved before being forwarded to the inner encoder.

The output of the inner encoder is then punctured by the puncturers P_s and P_p, so as to generate the two output streams $\mathbf{c}_s$ and $\mathbf{c}_p$. These vectors are finally multiplexed to form the L-bit codeword sequence $\mathbf{c}$.

The coded sequence $\mathbf{c}$ is sent to the channel as the signal sequence $\mathbf{x}$ obtained at the output of a binary pulse-amplitude modulation PAM or, equivalently, a BPSK modulator. The relationship between the modulated signal and the coded bits is:

$$x_k = \sqrt{E_S}(2c_k - 1) \tag{1.3}$$

having denoted by E_S the signal energy. The channel is assumed an additive white Gaussian noise (AWGN) channel with two-sided noise power spectral density $N_0/2$.

Since the rate of the outer code punctured through P_o is 2/3, the interleaver length is given by $N = 3K/2$. We assume the interleaver length to be a multiple of a base interleaver size N', which, as described in [15], is fixed to 300 bit. Thus, for larger interleaver lengths, 4 multiples of N' are considered, that is interleaver size of 300, 600, 1200 and 2400 bits. The corresponding information block sizes can be deduced as:

$$K = \frac{2}{3}N - 2 = \frac{2}{3}M \cdot N' - 2 \tag{1.4}$$

where $M = 1,2,3,4$ and the two termination bits, added to reset the encoder memory to the original state, are also taken into account.

The codeword length L can be calculated by recalling that two parity bits $c_{k,1}$ and $c_{k,2}$ are generated for each information bit u_k which, together with the two termination bits, gives:

$$L = M \cdot (N' + 2) \tag{1.5}$$

which leads to $L = 604, 1208, 2416, 4832$ for the 4 interleaver sizes considered.

The channel date-rate is fixed at 100Mb/s.

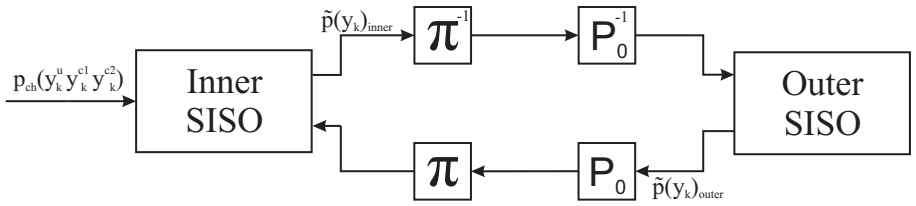


Figure 1.3: SCCC Turbo code decoding scheme

1.1.2 Analog Decoding Procedure

The fully parallel analog Turbo decoding procedure is illustrated in Fig.1.3.

The two SISO modules, each corresponding to one of the two encoders reported in Fig.1.2, perform the decoding by exchanging extrinsic *soft-information*. These *soft-information* can be regarded as an index of the confidence level reached by the *a posteriori* estimate of the original information bits u_k. It is worth to highlight that the two SISO units, the inner and the outer SISO, which correspond to the inner and outer encoder respectively, work on input data streams with different length and typology.

The inner SISO soft-inputs are:

- the channel output symbols y_k^s transition probabilities $p_{ch}(y_k^s) = p(y_k^s|x_k^s)$ with $s = u, c_1, c_2$;
- the additional information on u_k and c_1 given by the outer SISO as extrinsic probability $\tilde{p}(y_k)_{outer}$.

The outer SISO receives from the inner one the bits u_k and c_1 extrinsic information $\tilde{p}(y_k)_{inner}$ and produces two outputs:

- an *a posteriori* $\hat{u}_k$ estimate of the user bits u_k;
- new extrinsic information on the u_k and c_1 symbols which constitute the input for the inner SISO.

1.1.3 Hybrid Decoding Procedure

A fully-parallel analog implementation of the decoding algorithm described in Sec.1.1.2 would require two SISO modules with length 2400 and 1600 respectively, so as to handle the maximum codeword input length of 4800 bits. This would lead to a prohibitively large chip area.

To reduce the decoder complexity, a non fully-parallel approach is adopted. The basic idea is to implement only one single SISO unit, which is used to decode both the outer and the inner code. In addition, the SISO unit implements only a window of the overall code trellis of N' trellis sections. Thus, for larger block lengths the module is reused several times to decode each of the constituent codes.

The choice of the window dimension is a tradeoff between circuit complexity and speed. Increasing N' increases the circuit complexity but, at the same time, allows higher decoding parallelism. As the time required by the SISO to converge to stable values is roughly independent of the block size, as proved by simulations, the decoding speed increases linearly with the size of the window. As a good tradeoff between circuit speed and complexity, we fixed $N' = 300$ trellis sections, which corresponds to the shortest codeword considered with length $L = 604$ bits.

To explain the hybrid decoding process, an interleaver size of 2400 bits, corresponding to 8 subblocks of minimum length $N' = 300$, is assumed. The decoding procedure can be summarized in the following few steps:

1. the decoder loads the channel information relative to the intere frame onto an analog input memory;

2. during the first half iteration, the SISO unit works as *inner* SISO. For each subblock of $N' = 300$ bits the SISO is fed with the corresponding channel transition probabilities $p_{ch}(y_k^s)$ and the extrinsic information generated by the "*outer* SISO" during the previous half iteration on the inner code input bits $\tilde{p}(y_k)_{outer}$. At its output it generates the extrinsic information $\tilde{p}(y_k)_{inner}$ on the 300 input bits which are then stored following the natural order in the first row of an extrinsic memory. This process is repeated 8 times, one for each subblock, until all the 8 rows of the extrinsic memory are loaded with the corresponding extrinsic values.

3. during the second half iteration, the SISO unit works as *outer* SISO. Thus, for each subblock, it is fed with the extrinsic information $\tilde{p}(y_k)_{inner}$ on the outer code output bits, properly deinterleaved and punctured, generated during the previous half iteration by the "*inner* SISO" and with the extrinsic information on the outer code input bits stored in the extrinsic memory. At its output the SISO unit generates the a posteriori probabilities of the user bits on which decisions on the transmitted symbols are taken. The SISO also computes the extrinsic information $\tilde{p}(y_k)_{outer}$ on the outer code output bits which will be stored, properly interleaved and punctured,

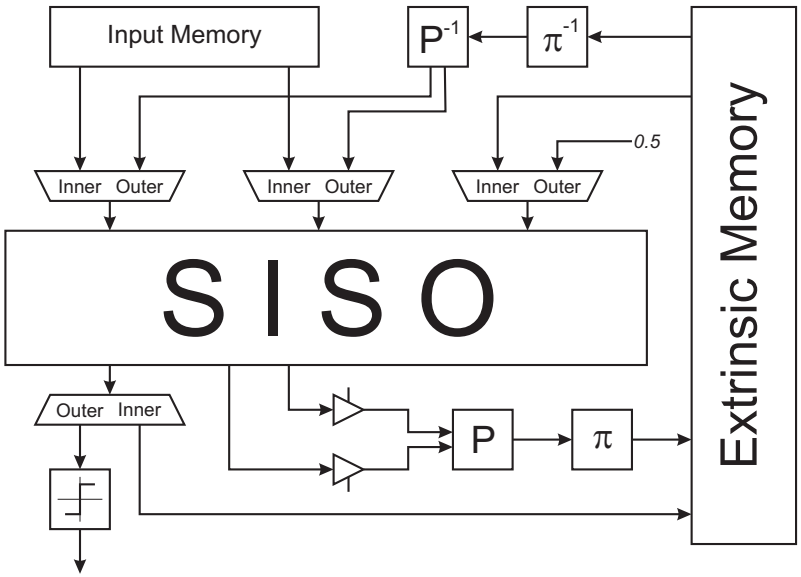


Figure 1.4: Analog decoder block diagram

in the extrinsic analog memory. This process is repeated for each subblock until the extrinsic values for the entire block are stored in the memory.

4. Step 2 and 3 are repeated as many times as the maximum iterations number $N_t = 10$. At the last iteration, decisions on the transmitted bits are taken.

1.1.4 Analog Decoder Architecture

A block diagram of the analog decoder implementing the hybrid decoding algorithm described in Sec.1.1.3 is shown in Fig.1.4. As already pointed out, it consists of a single SISO unit, which has to match both the inner and the outer code.

More in details, the SISO analog network has been designed following the approach described by Loeliger in [6] and successfully adopted in several analog decoder prototypes [8, 13, 14, 16]. It consists of several sum-product cells which operate on two current input vectors I_x and I_y representing the probability distributions $p(x)$ and $p(y)$ of discrete random variables and yield one output current vector I_z, according to (A.53). Thus, each component of the output vector is the sum of products of input vectors component pairs.

As already pointed out, the iterative nature of the proposed decoder requires the SISO to match both the outer and the inner code. Since the trellis length is different in the two cases, as it consists of 200 sections for the outer code which become 300 for the inner code, an appropriate combination of switches is added to the basic scheme as reported

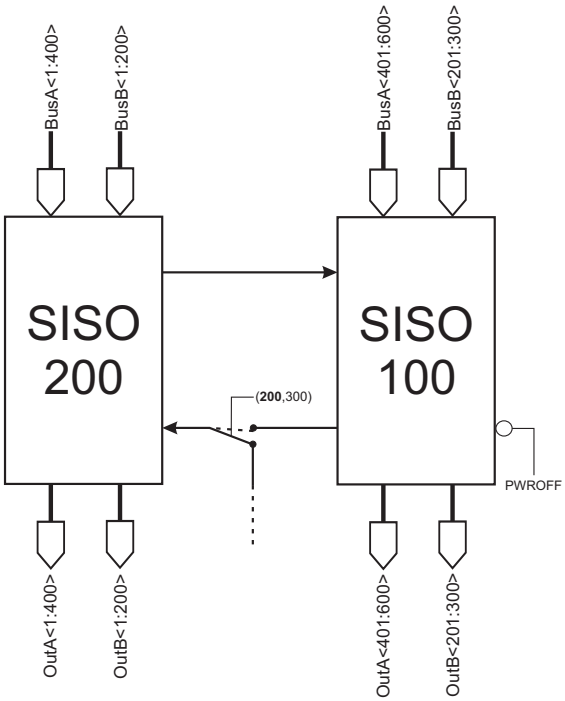


Figure 1.5: SISO simplified diagram with switch between *inner* and *outer* configuration

in Fig.1.5 in order to move the SISO termination in the proper position when the shorter length is needed.

Furthermore, to allow the exchange of the extrinsic information during the iterative decoding process, few multiplexers and an analog extrinsic memory are added. This memory must be able to store up to 2400 couples of soft values, corresponding to the extrinsic probabilities $\tilde{p}(0)$ and $\tilde{p}(1)$. The extrinsic memory is organized as a bank of 8 rows, that is the maximum subblocks number, and 300 columns, which correspond to the subblock length.

The input memory is needed to store the channel information during the entire decoding process. Its sizing, design and optimization will be described in Sec.1.2.3.

1.1.5 CMOS Technology

The hybrid analog decoder has been designed in the UMC 0.18-μm CMOS process. The key features of this technology are:

- minimum channel length: 0.18-μm;
- dual supply voltage: 1.8V and 3.3V;

- P-substrate;
- single poly, six metal layers (1P6M);
- Twin-Well to realize nMOS with isolated substrate;
- high performance mixed-mode signal capabilities;
- radio frequency MOS transistors.

In order to optimize the decoder power consumption, we used only devices working at the lower supply voltage, that is $V_{dd} = 1.8$V.

1.2 Input Interface Circuitry

The input interface, whose block diagram is shown in Fig.1.6, has to store a frame of channel output symbols y_k on the input analog memory and then to convert each of them into a pairs of currents representing the corresponding channel transition probabilities $p_{ch}(y_k) = p(y_k|x_k)$. In order to ease the testing phase, the channel output is fed to the memory in a 7-bit digital representation and it is converted to a differential voltage by a digital-to-analog converter.

1.2.1 Specifications

The input interface specifications have been drawn in order to guarantee the proper functioning of the overall analog decoder and also to make the whole system performance competitive with respect to the corresponding digital implementations. In particular, the memory write and access time together with its accuracy have been set so as to ensure a correct decoding process, while the specifications regarding the power consumption and area occupation are deduced from performance comparison with digital decoders.

Write and access time As the channel data rate is equal to 100Mb/s, the demodulator output is sampled at a frequency of 100MHz, which leads to a maximum memory write time of 10ns. The access time specification can be set by recalling that every decoding iteration lasts for 300ns and that, to speed up the decoding process, the correct channel transition probabilities values must be available at the SISO input in the shortest time. Thus, a reasonable value for the access time would be in the range of $10 \div 30$ns, that is at least one order of magnitude smaller than the decoding time.

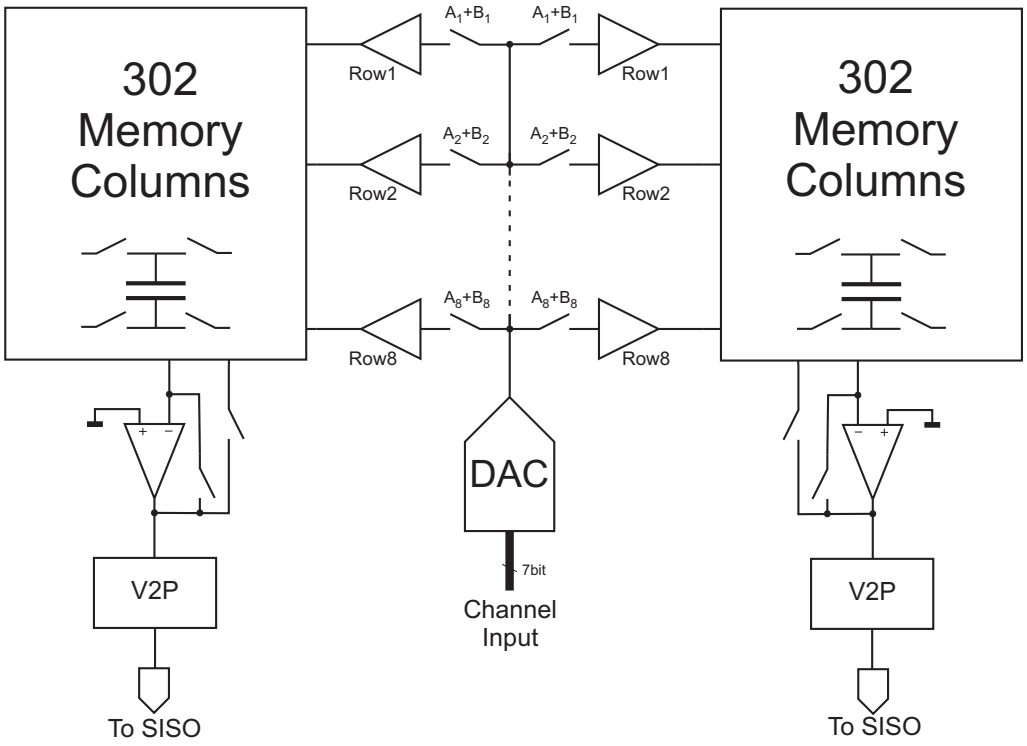


Figure 1.6: Input interface block diagram

Precision The accuracy specification has been drawn from the precision requirements for fixed point implementations of iterative decoders for concatenated codes with interleaver [17]. In particular, given the total number of bits n_b used for the log-likelihood ratio LLR numeric representation, where the LLR for the binary alphabet $\{0,1\}$ is defined as:

$$\lambda_k = \log \frac{p(y_k|c_k = 1)}{p(y_k|c_k = 0)} \tag{1.6}$$

the required number of bits of precision depends on n_b. For realistic applications, two cases have been considered, that is $n_b = 5$ and $n_b = 4$, leading to the following design hints:

- for $n_b = 4$, the best choice is (4,1) using one bit of precision and three for the dynamic. It yields performance less than 0.1dB worse than the ideal ones for low E_b/N_0;

- for $n_b = 5$, the best choice is (5,2) using two bits of precision and three for the dynamic. It yields performance almost identical to the ideal one.

Power consumption One of the key element to judge the performance of the hybrid decoder is definitely its efficiency, that is measured in terms of energy per decoded bit. This latter is defined as:

$$Energy/dec = \frac{Power\ consumption}{Input\ bit\ rate \times Code\ rate} = \frac{Power\ consumption}{Output\ bit\ rate} \tag{1.7}$$

In the UMTS Turbo decoder presented in [14], the energy per decoded bit has a minimum value of 12.6nJ/bit, that is 10 times smaller than the equivalent digital implementations.

In order to further improve this efficiency, our target is to reach for the hybrid decoder an energy per decoded bit of 2nJ/bit. This leads, according to (1.7), to a maximum power consumption for the overall analog decoder of 60mW.

As the extrinsic memory and the SISO power consumption has been estimated around 6.5mW and 13mW respectively, and 10mW should be enough for digital control unit, this leaves a power budget of 30mW for the input memory.

Area occupation Another important element in evaluating the goodness of the hybrid solution we propose, is the area occupation. As already stressed out, one of the main limitations of analog decoder is represented by the fact that their chip size increases linearly with the codeword length. The hybrid decoder has been introduced to overtake this very limitation.

Due to the large number of memory locations required for the analog memory, this block could be the most demanding in terms of area occupation, especially if we decide to use a capacitor as storage element.

As the die area for the analog decoder implementation is fixed at 5mm $\times$ 10mm and the room required for I/O pads and ESD protections as also to be taken into account, we assume the area available for the decoder circuitry to be equal to 4mm $\times$ 9mm. It is reasonable to allocate for the input memory up to a quarter of this area.

1.2.2 Voltage-to-Probability Converter

The interface between the memory array and the decoder core has the task to compute the channel conditional probabilities $p_{ch}(y_k) = p(y_k|x_k) = p(y_k|c_k)$ required by the SISO. These probabilities are defined as:

$$p(y_k|x_k = x_j) = \frac{1}{\sqrt{2\pi}\sigma_n} exp(-\frac{(y_k - x_j)^2}{2\sigma_n^2}) \tag{1.8}$$

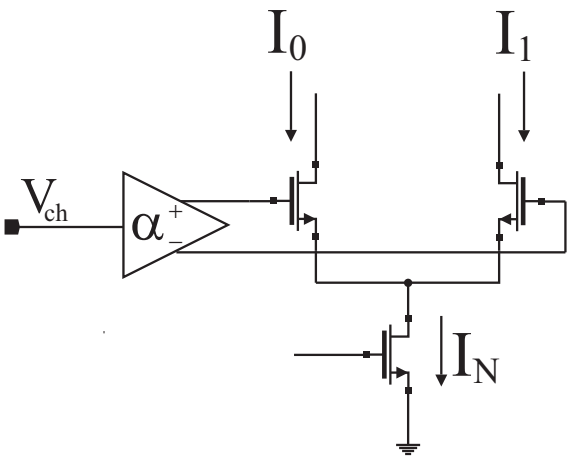


Figure 1.7: Voltage to probability converter for a binary alphabet

where σ_n^2 is AWGN channel noise variance, y_k is the channel output, x_k is the transmitted signal and $\{x_j|0 \leq j \leq 1\}$ denotes the set of all possible signal amplitudes. In our case, as we use a binary alphabet $\{0,1\}$ with a BPSK modulation $\{-a,+a\}$, the (1.8) can be implemented by a simple differential pair, as described in [18]. A simplified schematic of the voltage to probability converter is shown in Fig.1.7.

The differential pair output currents are proportional to the conditional probabilities according to:

$$\begin{aligned} I_{D1} &= I_N \frac{p[y_k|x_k=-a]}{p[y_k|x_k=-a]+p[y_k|x_k=+a]} \\ I_{D2} &= I_N \frac{p[y_k|x_k=+a]}{p[y_k|x_k=-a]+p[y_k|x_k=+a]} \end{aligned} \tag{1.9}$$

This follows from the fact that, if the nMOS transistor are working under weak inversion, their drain currents are equal to:

$$\begin{aligned} I_0 &= \frac{I_N}{1+exp(\frac{-2\alpha V_{ch}}{nU_T})} \\ I_1 &= \frac{I_N}{1+exp(\frac{2\alpha V_{ch}}{nU_T})} \end{aligned} \tag{1.10}$$

where V_{ch} is the voltage received from the channel, α is the amplifier gain and nU_T is the

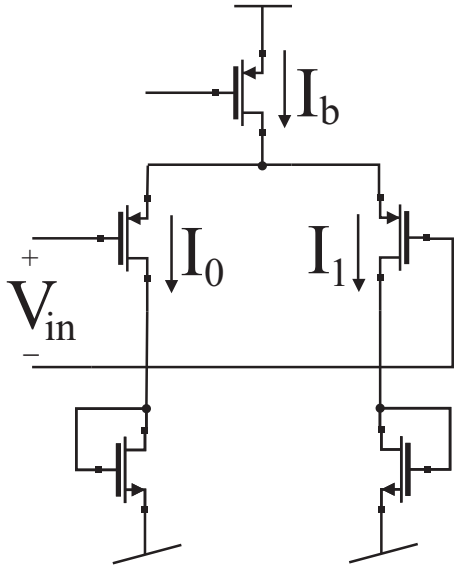


Figure 1.8: Voltage to probability schematic

temperature equivalent voltage. The two conditional probabilities (1.9) can be written as:

$$\begin{aligned} I_N \cdot \frac{p[y_k|x_k=-a]}{p[y_k|x_k=-a]+p[y_k|x_k=+a]} &= I_N \cdot \frac{e^{-\frac{(V_{ch}+a)^2}{2\sigma_n^2}}}{e^{-\frac{(V_{ch}+a)^2}{2\sigma_n^2}}+e^{-\frac{(V_{ch}-a)^2}{2\sigma_n^2}}} \\ &= I_N \cdot \frac{1}{1+e^{-\frac{(V_{ch}-a)^2-(V_{ch}+a)^2}{2\sigma_n^2}}} \\ &= I_N \cdot \frac{1}{1+e^{\frac{2a\cdot V_{ch}}{2\sigma_n^2}}} \\ I_N \cdot \frac{p[y_k|x_k=+a]}{p[y_k|x_k=-a]+p[y_k|x_k=+a]} &= I_N \cdot \frac{1}{1+e^{-\frac{2a\cdot V_{ch}}{2\sigma_n^2}}} \end{aligned} \tag{1.11}$$

Thus, if the gain α is set equal to:

$$\alpha = \frac{nU_T a}{\sigma_n^2} \tag{1.12}$$

it is immediate to deduce the (1.9) from (1.11).

Implementation The voltage to probability converter V-to-P has been implemented by means of a pMOS differential pair, as shown in Fig.1.8, where the two nMOS transistors constitute the SISO module input stage.

The two output currents I_0 and I_1, proportional to the conditional probabilities according to (1.9), are given by:

$$\begin{aligned} I_0 &= \frac{I_b}{1+e^{\left(\frac{V_{in}}{n_p U_T}\right)}} \\ I_1 &= \frac{I_b}{1+e^{\left(-\frac{V_{in}}{n_p U_T}\right)}} \end{aligned} \tag{1.13}$$

The module bias current I_b has been set equal to the bias current of the sum-product cells of the SISO unit, that is $I_b = 1\mu$A. In this way, we have no loss of performance due to the fact that the sum of the input currents of the sum-product module is less than its bias current.

As the voltage to probability conversion relies on the exponential voltage to current characteristic of the input pair pMOSs, they have to be sized so as to work under weak inversion for currents up to 1μA, that is the maximum current flowing through one pMOS.

In order to define the transistors dimensions, precision issues must also be carefully considered. The precision requirement for the voltage to probability module can be deduced by considering the equivalence between the log-likelyhood ratio defined by equation (1.6) and the normalized conditional probabilities. For example, let's consider the probability represented by the V-to-P current I_0. In terms of log-likelyhood ratio, this can be written as:

$$\begin{aligned} \frac{p[y_k|c_k=0]}{p[y_k|c_k=0]+p[y_k|c_k=1]} &= \frac{e^{-\frac{y_k^2}{2\sigma_n^2}}}{e^{-\frac{y_k^2}{2\sigma_n^2}}+e^{-\frac{(y_k-1)^2}{2\sigma_n^2}}} \\ &= \frac{1}{1+e^{\left[-\frac{(y_k-1)^2}{2\sigma_n^2}+\frac{y_k^2}{2\sigma_n^2}\right]}} \\ &= \frac{1}{1+e^{\frac{2y_k-1}{2\sigma_n^2}}} \\ &= \frac{1}{1+e^{\lambda_k}} \end{aligned} \tag{1.14}$$

By comparing this last expression with equation (1.13), we have:

$$\lambda_k = \frac{V_{in}}{n_p U_T} \tag{1.15}$$

From the precision requirements for the numeric representation of the log-likelyhood ratio λ_k in the digital domain reported in Sec.1.2.1, we can derive the accuracy required

for V_{in}. In particular, if we chose to represent the LLR with $n_b = 4$ bits, the best results are obtained with (4,1). The use of 1 bit for the precision leads to:

$$\frac{V_{in}}{n_p U_T} = 1/2^1 = 0.5 \tag{1.16}$$

in the analog domain, while the 3 bits for the dynamic become:

$$\frac{V_{in}}{n_p U_T} = 2^3 = 8 \tag{1.17}$$

n_p is a technology dependent paramiter, which has been estimated for the UMC 0.18-μm process through simulations, giving $n_p = 1.56$ for pMOS transistors and $n_n = 1.31$ for nMOS. This leads to a precision requirement for V_{in} of 20mV with a dynamic range of $\pm 162mV = 324mV_{pp}$.

As the input interface contains three main blocks, the DAC, the input memory and the voltage to probability converter, the error introduced by each of these blocks can be defined as:

$$\tilde{g}(x) = g(x) \cdot (1 + \varepsilon) \tag{1.18}$$

where $g(x)$ describes the ideal input-output relation for each block and ε represents the error introduced by the block non-idealities. We assume the error statistic of each block to be a second order Gaussian distribution, that is the error can be fully described by means of its standard deviation σ_ε and its mean m_ε.

As the input memory contains the memory cells plus several additional buffers, we divided the error introduced by this block between these two components. Thus the overall input chain contains four sources of error $\varepsilon_{DAC}, \varepsilon_{BUF}, \varepsilon_{OTA}, \varepsilon_{V2P}$, each described by a Gaussian distribution. Even if it is a simplification, we assume the four error sources to be independent.

The sum of independent Gaussian variables is still a Gaussian variable whose mean and standard deviation are given by:

$$\begin{aligned} m_{\varepsilon_t} &= \sum_{i=1}^{4} m_{\varepsilon_i} \\ \sigma_{\varepsilon_t} &= \sqrt{\sum_{i=1}^{4} \sigma_{\varepsilon_i}} \end{aligned} \tag{1.19}$$

If we assume the contribution of each block to be equal, equation (1.19) leads to $m_{\varepsilon_t} = 4m_\varepsilon$ and $\sigma_{\varepsilon_t} = 2\sigma_\varepsilon$. Thus, in order to fullfill the precision requirement given by

Parameter	A_{Vth}	C_o
Unit	$[mV \cdot \mu m]$	$[mV]$
nMOS	4.787	0.5328
pMOS	4.6899	0.1894

Table 1.1: Matching parameters

V-to-P	
M_p	$\frac{20}{0.4}$
M_n	$\frac{4}{0.3}$
M_b	$\frac{2}{4}$

Table 1.2: Voltage to probability transistor size in μm/μm

(1.16), we need $m_{\varepsilon_t} \pm 3\sigma_{\varepsilon_t} \leq \pm 20$mV. This equation can be satisfied if we impose $4m_\varepsilon + 3 \cdot \sigma_\varepsilon \leq 20$mV which, for each input interface block, becomes:

$$m_\varepsilon + 3\sigma_\varepsilon \leq \frac{20}{4}\text{mV} = 5\text{mV} \tag{1.20}$$

For the voltage to probability module, equation (1.20) imposes some constraints on the pMOS transistors dimensions. In particular, as the standard deviation of the threshold voltage error can be modeled by equation:

$$\sigma(\Delta V_{th}) = \frac{A_{Vth}}{\sqrt{W \cdot L}} + C_o \tag{1.21}$$

where A_{V_T} and C_o are technology dependent parameters whose values for the UMC 0.18-μm CMOS process are given in Table 1.1, the pMOS width and length have to be chosen so as to keep $\sigma(\Delta V_{th}) \leq 1.8$mV. At the same time, the transistor dimensions have to ensure they work in their exponential region for currents up to 1μA.

The transistors size for the voltage to probability converter are reported in Table 1.2, where the dimensions of the SISO input stage nMOS are given too.

With this choice, $\sigma(\Delta V_{th}) \simeq 1.85$mV.

1.2.3 Input Memory

The input memory is needed to store the channel output symbols y_k during the entire decoding process. In order to maximaze the decoder throughput, the input memory consists

of two identical banks, so that one can be loaded with the channel information while the other is fed to the decoder.

As the maximum codeword length to be handled by the decoder is equal to $L = 4832$, each memory bank consists of 4832 pseudo-differential memory cells. Thus, the overall memory counts 4832×2 memory locations organized in 604 columns and 8 rows. When the minimum codeword is selected, just the first memory row is used, while the whole memory is needed with codewords of maximum length.

In order to meet the need for low power consumption and high density, we chose an architecture based on switched-capacitor circuits to implement the input memory [19], as shown in the simplified memory scheme of Fig.1.9. The main advantage of this switched-capacitor based solution with respect to conventional buffered sample-and-hold circuit topology is that it requires only one active element per column, instead of one per capacitor, with a significant power saving.

During write operation, the input signal path is fed to 8 buffers, one for each row, in order to cope with the large parasitic capacitance of the long interconnections. The control signals W_a, W_b, R_a and R_b, with $W_a = \overline{W}_b$ and $R_a = \overline{R}_b$, select which bank is being loaded, the other one being fed to the decoder core.

The memory is read one row at a time by the SISO while working in the *inner mode* by placing the capacitors in feedback configuration across the corresponding OTAs. The writing operation is sequential, so one capacitor couple is accessed per clock cycle.

Memory cell Each memory cell is pseudo-differential and consists of two replicas of the same basic structure formed by five switches, a capacitor and an operational transconductance amplifier, as described in [20]. A single OTA is used for all the cells belonging to the same memory column and is shared between both memory banks A and B. Thus, considering the pseudo-differential nature of the memory, a total amount of 604×2 OTA are needed. A simplified schematic of the memory cell is shown in Fig.1.10. The top plate of the sampling capacitor can be shorted to the memory input or output by means of the two switches S_w and S_r, while the bottom plate is connected to the reference voltage V_{ref} or to the OTA input by the two switches S_{wa} and S_{ra} respectively. The additional switch S_{reset} serves to configure the operational amplifier as a voltage follower in order to force the input and output node to the bias voltage $V_b \simeq V_{ref}$ between two consecutive read phases.

The operation of the circuit can be described by dividing the data acquisition process

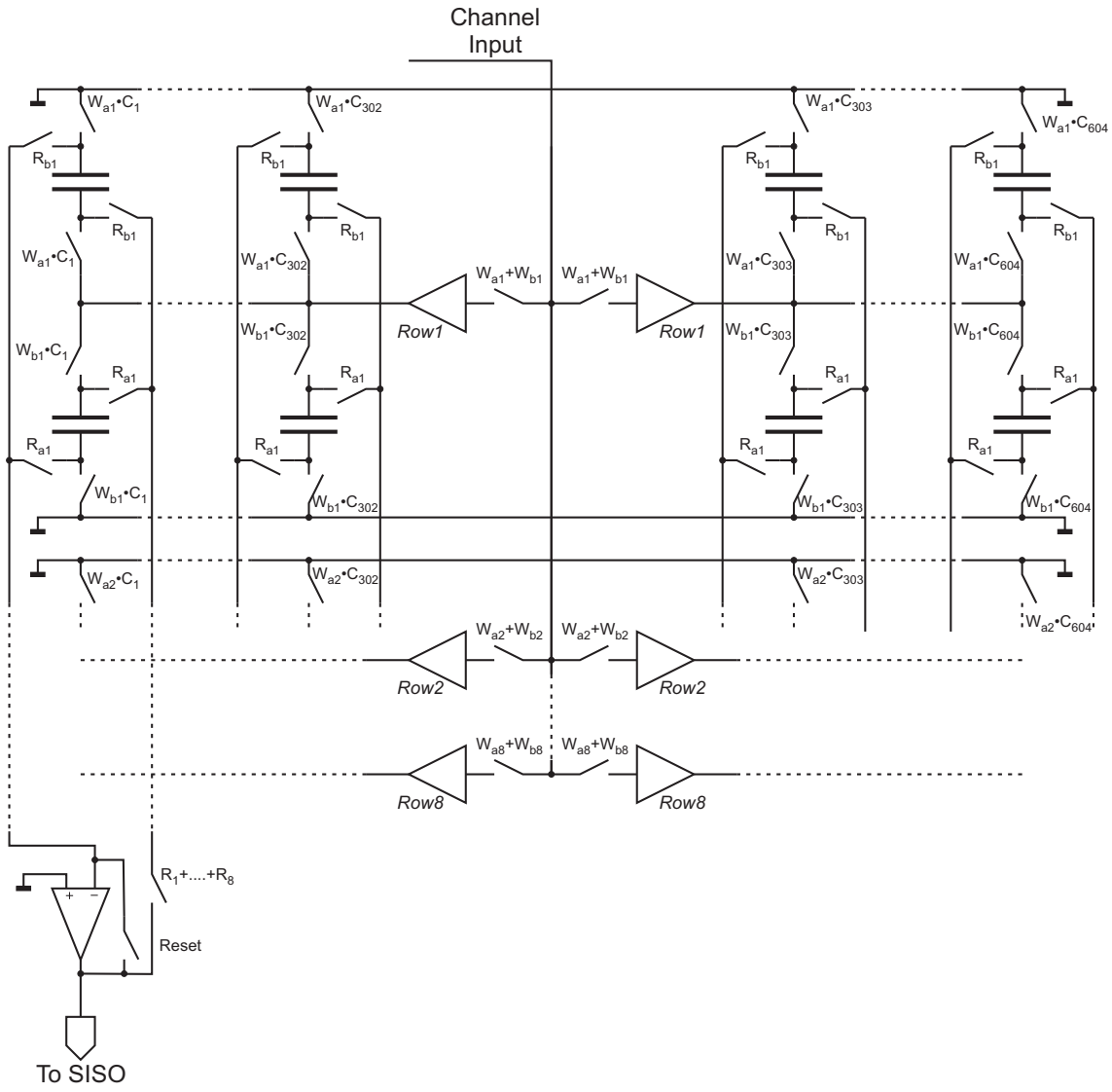


Figure 1.9: Input memory block diagram

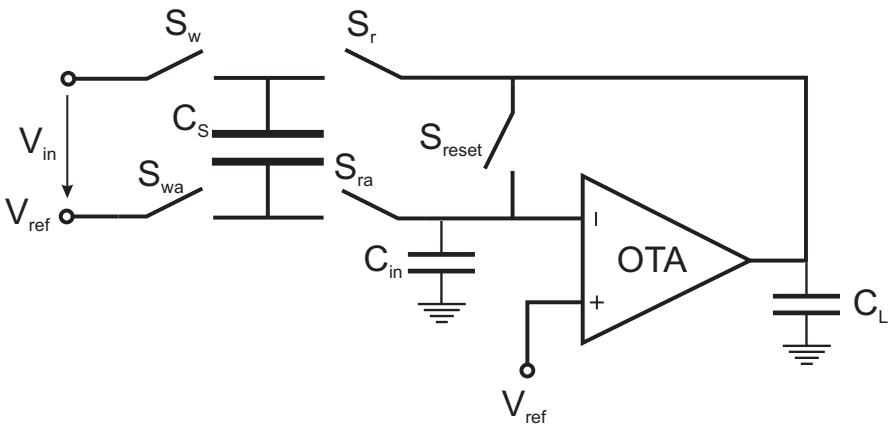


Figure 1.10: Memory cell structure

into write and read cycles. While a memory bank is being written, the data previously stored in the other one are fed to the decoder core. A simplified timing diagram for the memory read and write phase is shown in Fig.1.11, where T indicates the nominal clock period of 10ns.

During the write phase, switches S_w and S_{wa} are closed, while S_r, S_{ra} and S_{reset} are open, as the OTA is busy reading the data stored in the other memory bank. The input voltage V_{in} applied to the memory input is sampled and the voltage $V_{in} - V_b$ is stored across the capacitor C_S. During the read phase, switches S_r, S_{ra} are closed while S_w, S_{wa} and S_{reset} are open, placing the OTA in feedback configuration. Due to the amplifier high gain, the voltage at node X, V_x is held constant at a value almost equal to V_b. Thus no charge can be injected into the capacitor C_S leading $V_{out} = V_{in}$.

Switched-capacitors circuits suffer from charge injection errors. The bottom-plate sampling principle [21], can greatly help to reduce this error source. This means to adopt the switching sequence $S_{wa} \rightarrow S_w \rightarrow S_{ra} \rightarrow S_r$. However, using this switching sequence, the circuit precision can be further increased by:

- increasing the sampling capacitance C_S value. This, however, reduces the sampling bandwidth proportionally;
- making the charge Q_{wa} injected by switch S_{wa} equal to $-Q_{ra}$, where $-Q_{ra}$ is the charge injected by S_{ra}. This is only possible if both switches S_{wa} and S_{ra} are of the same type, nMOS or pMOS;
- minimizing both Q_{wa} and Q_{ra}. Therefore, the area of both switches S_{wa} and S_{ra}

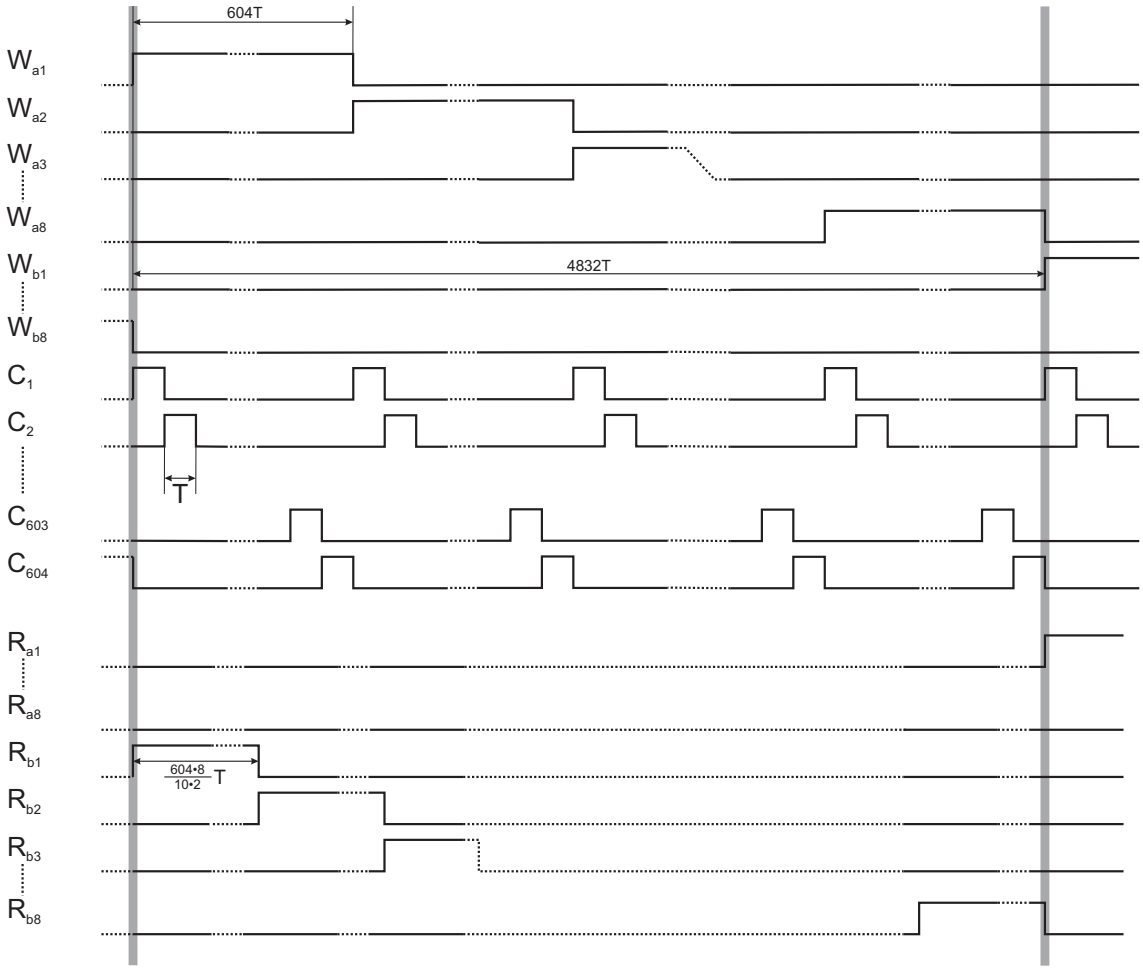


Figure 1.11: Read and write phase timing diagram

should be as small as possible. However, decreasing the width of the switches reduces the sampling frequency bandwidth accordingly.

Another source of error in the proposed circuit is due to the OTA non-idealities. In particular, the OTA finite gain causes the voltage at node X to differ from V_b while the finite input capacitance C_{in} at the same node determines a charge sharing phenomena between C_{in} and C_S during the read phase. The circuit analysis gives:

$$V_{out} = \frac{V_{in}}{1+\frac{1}{A_v}\left(1+\frac{C_{in}}{C_S}\right)} \simeq V_{in}\left[1-\frac{1}{A_v}\left(1+\frac{C_{in}}{C_S}\right)\right] = V_{in}\left[1-\frac{1}{A_v\beta}\right] \tag{1.22}$$

where $\beta = C_S/(C_{in}+C_S)$ indicates the feedback factor.

Moreover, as each memory cell has to be read several times, that is once per Turbo iteration, before being overwritten by a new channel information value, each read opera-

tion adds a data-dependent error on the value stored in the memory cell, whose magnitude is given by:

$$\varepsilon = -\frac{V_{in}}{A_v}\left(1+\frac{C_{in}}{C_S}\right) = -\frac{V_{in}}{A_v\beta} \tag{1.23}$$

As a results, in order to keep the error on the output voltage V_{out} within acceptable bounds, we need to carefully design the OTA so as to ensure an high low frequency gain A_v. The input capacitance C_{in} should also be minimized, even if the speed is not critical, so as to ensure a high feedback factor β.

The circuit speed during the write phase depends on the time constant:

$$\tau_{write} = (R_{on,w} + R_{on,wa})C_S \tag{1.24}$$

where $R_{on,w}$ and $R_{on,wa}$ are the on-resistance of switches S_w and S_{wa} respectively. Thus, the write time increases linearly with the sampling capacitance C_S and the switches on-resistance.

During the read phase, the circuit analysis leads to:

$$\tau_{read} = \frac{C_L C_{in} + C_{in}C_S + C_S C_L}{G_m C_S} = \frac{1}{\beta}\frac{C_L + (1-\beta)C_S}{G_m} \tag{1.25}$$

where G_m is the amplifier transconductance. The read time decreases with the amplifier transconductance while its relation with the feedback factor depends on the capacitance values. If C_L is dominant with respect to C_S, an increase of the feedback factor β leads to a reduction of the read time τ_{read}.

The slewing behavior of the circuit has also to be taken into account when designing the OTA. Upon entering the amplification mode, the circuit may experience a large step at the inverting input. As the input capacitance C_{in} is usually small, the voltage at the output node and at node X does not change immediately, but at the beginning of the read phase $V_x = -V_{in}$. This can force the amplifier into a slewing condition. The slewing factor depends on the chosen amplifier topology. However, for a common source amplifier it is approximately equal to I_B/C_L, where I_B is the common source bias current.

OTA design As already pointed out, the OTA has to be carefully designed so as to satisfy the input memory precision and speed requirements. First of all, as we need a high low frequency gain to keep the error on the read voltage within acceptable bounds, we decided to use a regulated cascode topology. The schematic of the OTA is depicted in Fig.1.12. The amplifier voltage gain A_v is given by:

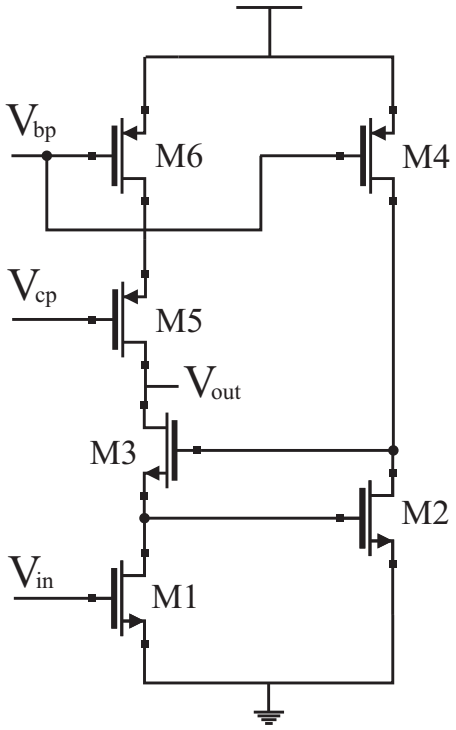


Figure 1.12: OTA schematic

$$A_v = g_{m1} R_{out} \tag{1.26}$$

where g_{m1} is transistor M_1 transconductance and the output resistance R_{out} is given by:

$$R_{out} = [(g_{m2} r_{ds2} r_{ds1}) g_{m3} r_{ds3}] // (g_{m6} r_{ds6} r_{ds4}) \tag{1.27}$$

while for the unit gain frequency we have:

$$\omega_u = g_{m1} C_{out} \tag{1.28}$$

as we assume the dominant pole to be at the output node. The relationship between the closed loop bandwidth $\omega_{CL} = \frac{g_m \beta}{C_L + (1-\beta) C_S}$ and the feedback factor $\beta = C_S/(C_{in} + C_S)$ depends on the values of the load capacitance C_L and the input capacitance C_{in}. This latter is given by the sum of the long interconnection parasitic capacitance, which has been estimated around 30fF, and the gate capacitance C_{g1} of M_1. Due to the tight precision requirements, M_1 will not be vary small, thus the contribute given by C_{g1} can not be neglected. We assume $C_{in} = C_w + C_{g1} = 60$fF. The contribution of the wire parasitic capacitance can be neglect in the evaluation of C_L, which is given by the input capacitance of the voltage to probability module, leading to $C_L = 50$fF.

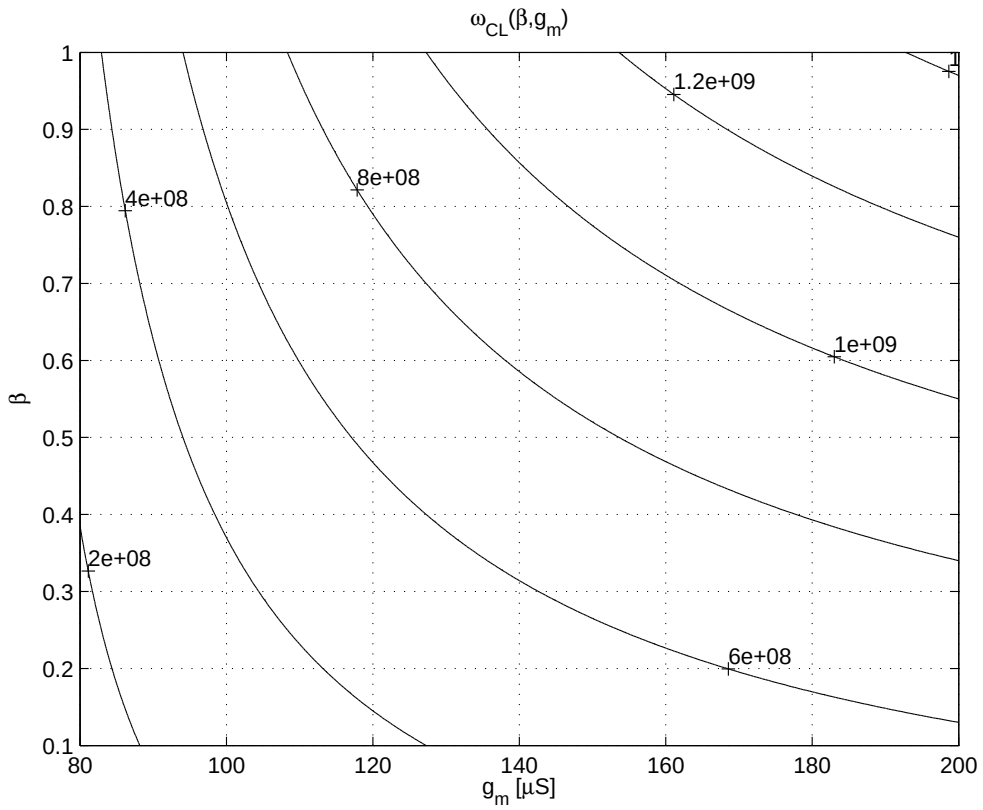


Figure 1.13: Countour lines for $\omega_{CL}(\beta, g_m)$

The contour lines for ω_{CL} as a function of β and g_m are shown in Fig.1.13. The specification for the read time of $\tau_{read} = 10$ns given in Sec.1.2.1 leads to $\omega_{CL} = 600$Mrad/s. As the contour lines do not vary linearly with β and g_m, the optimal region for each curve is the central one. Moving towards the sides, a small save in terms of area requires a large increase in power consumption and vice versa.

β as a function of C_S is plotted in Fig.1.14 while the contour lines for ω_{CL} as a function of C_S and g_m are shown in Fig.1.15.

Thus, in order to optimize our design, the feedback factor β has to vary between 0.3 and 0.7, which leads to a sampling capacitance C_S between 40fF and 150fF.

The choice of the sampling capacitance C_S depends also on other factors, such as:

- the write time specification;
- the OTA precision requirements;
- the charge injection phenomena;

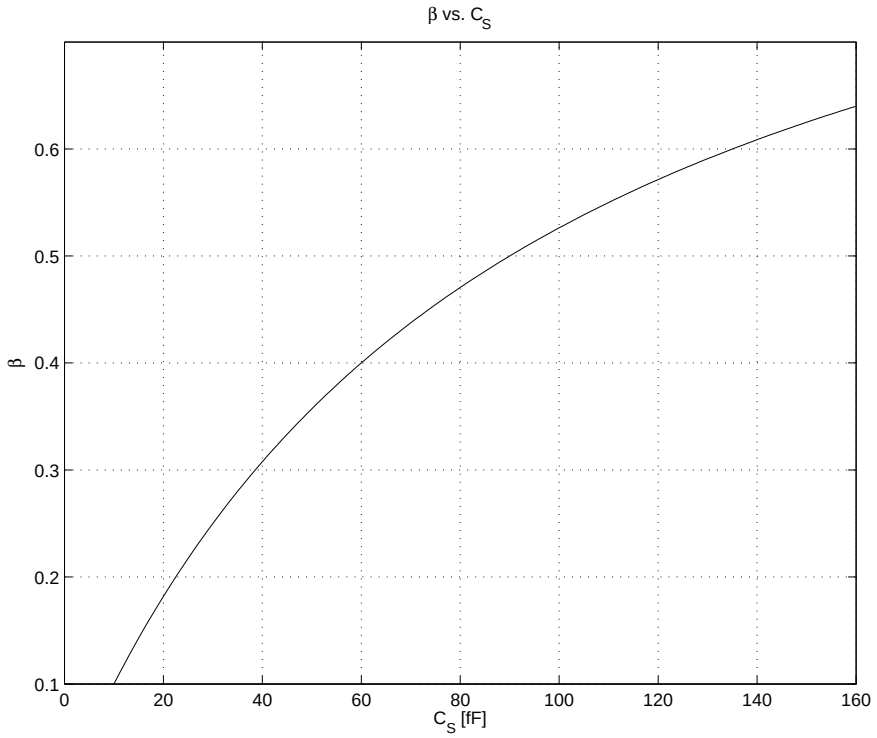


Figure 1.14: β as a function of C_S

- the leakage currents;
- the area occupation, as the memory will count up to 4832×2 capacitors.

In particular, for what it concerns the accuracy requirements, from (1.20) and (1.23) and recalling that the maximum input voltage is equal to 162mV and that each memory location is read up to 10 times, we can derive the open loop gain specification for the OTA as:

$$A_v\beta \geq \frac{10 \cdot 162 \cdot 10^{-3}}{1.8 \cdot 10^{-3}} \tag{1.29}$$

which leads to an open loop gain of at least 60dB.

As we used the bottom-sampling switching sequence, the error induced by charge injection is minimized and thus does not influence the choice of the sampling capacitance value.

The leakage current may be a non neglectable source of error, due to the long hold time, especially for the first data written in the memory.

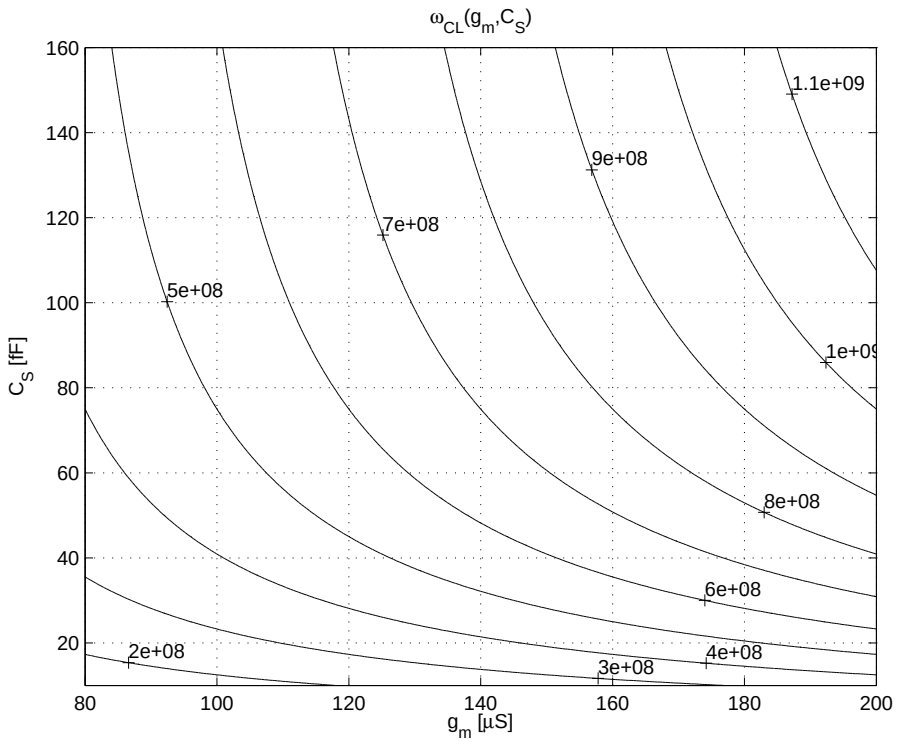


Figure 1.15: Countour lines for $\omega_{CL}(C_S, g_m)$

In order to estimate this error, we considered the worst case hold time of $48\mu s + 48\mu s$, where the first $48\mu s$ are the time it takes to store a maximum length codeword into the input memory, while the additional $48\mu s$ are the time required to decode it.

Worst case simulations at a temperature of 80^{O}C with a minimum size switch give a leakage current of $\simeq 0.34$pA, which leads to a voltage loss of less than 1mV for a sampling capacitance of 40fF. Moreover, as the data are stored in differential form and the leakage current is almost independent from the input voltage, the error induced by the leakage is mainly a commom mode error, as will be proved by overall memory simulations. As the memory has to allocate 4832×2 capacitors, the area occupation is the most critical factor in the choice of the sampling capacitance value C_S. The memory is organized in 8 rows, each containing 1208 capacitors, and it has to be fitted into a maximum area of 9mm $\times$ 200μm. Thus, the area available for each memory cell, which includes the sampling capacitor, the five switches and the additional room for the wiring, can be estimated in $\sim$ $7.5\mu\text{m} \times 25\mu$m. As the capacitance per unit area for the UMC 0.18-μm CMOS technology is 1fF/μm^2, we can consider a maximum capacitance value around 75fF.

Switches	
S_w	$\frac{0.86}{0.18}$
S_{wa}	$\frac{0.28}{0.18}$
S_r	$\frac{0.24}{0.18}$
S_{ra}	$\frac{0.24}{0.18}$
S_{reset}	$\frac{0.24}{0.18}$

Table 1.3: Switches size in μm/μm

OTA	
M_1	$\frac{10}{1}$
M_2	$\frac{5}{0.25}$
M_3	$\frac{2}{0.18}$
M_4	$\frac{10}{2}$
M_5	$\frac{0.5}{2}$
M_6	$\frac{6}{0.18}$

Table 1.4: OTA transistor size in μm/μm

Following these considerations, the final value for the sampling capacitance C_S has been set to 37.8fF, which corresponds to an area of $\sim 6\mu\text{m} \times 6\mu\text{m}$.

The switches S_w and S_{wa} have thus been sized so as to satisfy the write time specifications. Recalling that the on resistance of a MOS transistor is given by:

$$R_{on} = \frac{1}{\mu C_{ox} \frac{W}{L}(V_{DD} - V_{th} - V_{in})} \tag{1.30}$$

and that the write specification requires a $\tau_{write} \leq 1.5$ns, minimum size transistors can not be used. The memory cell switches dimensions are reported in Table 1.3. All switches are implemented with nMOS transistors.

Once the capacitor sampling value has been set, the OTA design follows straightforward. Its transistors size are reported in Table 1.4, while in Fig.1.16 the frequency response of the OTA in typical conditions is shown. We can see how the OTA exhibits an open loop frequency gain $A_v\beta \simeq 76$dB with a unit gain frequency $f_u \simeq 100$MHz, which corresponds to $\omega_u \simeq 630$Mrad/s. The OTA bias current I_B has been set to 10μA while the cascode bias voltage $V_{cp} = 600$mV.

The corner analysis gives as worst result an open loop frequency gain $A_v\beta \simeq 64.5$dB, an unit gain frequency $f_u \simeq 89.8$MHz with a phase margin of 75.5 deg, which is still well within specifications.

Input buffer To drive the long interconnection parasitic capacitance, a buffer employing a cascode amplifier in unit feedback configuration has been inserted for each memory row at the DAC output, as depicted in Fig.1.9. The wiring capacitance has been estimated by means of the well-known formula:

$$C_w = (C_a W + 2C_f + 2C_c)L \tag{1.31}$$

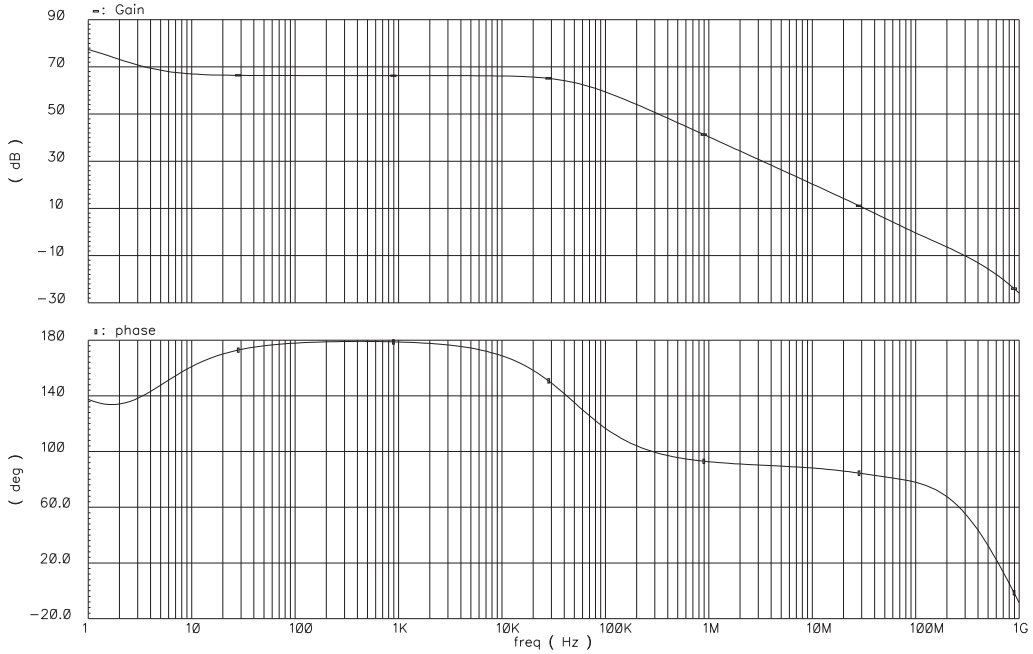


Figure 1.16: OTA frequency responce

where W and L are the interconnection wire width and length respectively, C_a is the *area capacitance* between the metal wire and the substrate, C_f the *fringing capacitance* and C_c the *coupling capacitance* with adjacent metal lines. Substituting the UMC technology paramiter into (1.31), we obtain $C_w = 850$fF. The capacitance due to the memory cell switches, $C_{sw} = 1.5$pF, has also to be taken into account.

The wire resistance is equal to $R_w = 1\text{k}\Omega$.

Due to the large parasitic capacitance and resistance, we decide to split the memory into two symmetrical parts and to use a pair of buffers to drive the cells of each memory row. Thus, each buffer has to be sized so as to drive a capacitance of $(C_w + C_{sw})/2 \simeq 1.2$pF with a time constant given by $\tau = R_w(C_w + C_{sw})/8 = 0.3$ns instead of $\tau = R_w(C_w + C_{sw})/2 = 1.175$ns, where a distributed RC network has been used to model the interconnection wire [22].

The buffer specifications can be easily deduced from the input memory ones given in Sec.1.2.1. In particular, a settling time $t_s \leq 9$ns is required in order to meet the write time specification while a low frequency gain $A_v \geq 50$dB meets the precision requirements.

The schematic of the cascode amplifier is shown in Fig.1.17, while its transistors size are reported in Table 1.5.

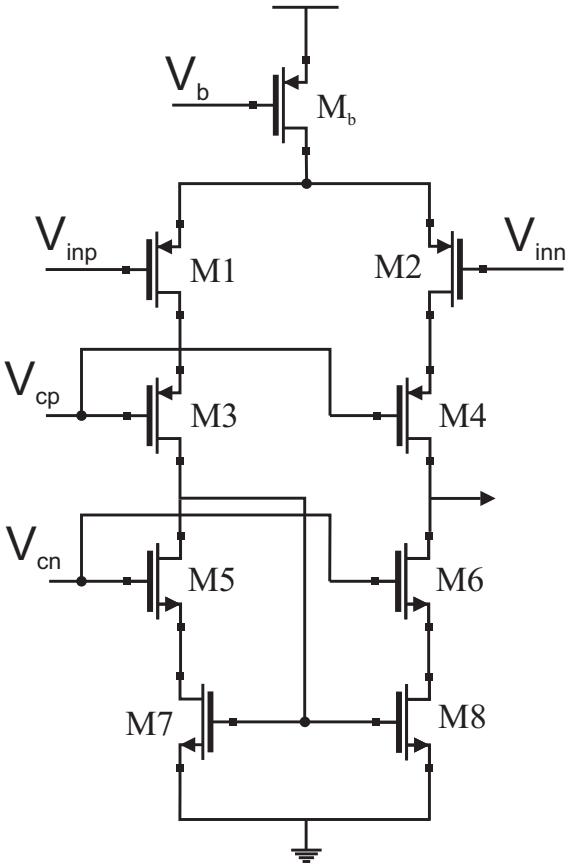


Figure 1.17: Input buffer schematic

The low frequency gain is given by:

$$A_v = g_{m1} R_{out} \tag{1.32}$$

where

$$R_{out} = (g_{m4} r_{ds4} r_{ds2}) // (g_{m7} r_{ds7} r_{ds6}) \tag{1.33}$$

Due to the large load capacitance, we expect the dominant pole to be at the output node. Thus the unit gain frequency can be written as:

$$\omega_u = 2 g_{m1} / (C_w + C_{sw}) \tag{1.34}$$

The buffer bias current I_B has been set to $I_B = 260\mu$A while $V_{cp} = 600$mV and $V_{cn} = 1.2$V are the pMOS and nMOS cascode voltage references respectively. As the pMOS of the input differential pair are biased so as to work between strong and moderate inversion in

Buffer	
M_1	$\frac{272}{0.25}$
M_3	$\frac{40}{0.18}$
M_5	$\frac{20}{0.25}$
M_7	$\frac{4}{1}$
M_b	$\frac{260}{2}$

Table 1.5: Buffer transistor size in μm/μm

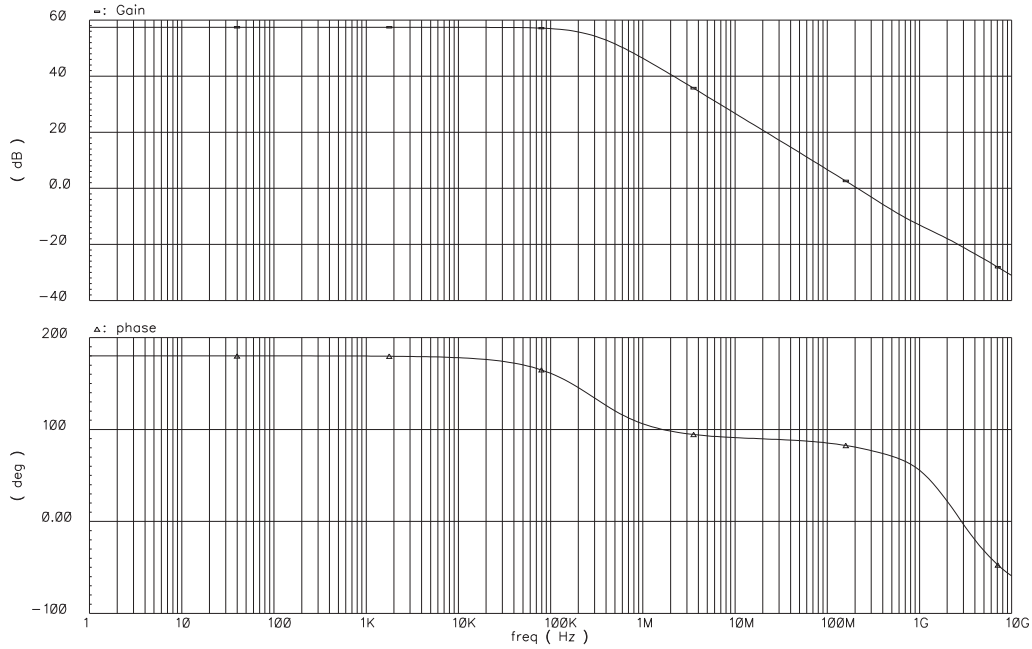


Figure 1.18: Input memory buffer frequency responce

order to pump up the amplifier gain, the input offset voltage can be calculated by:

$$V_{OS} = \sqrt{n_p^2 U_T^2 \left[\left(\frac{\Delta \frac{W}{L}}{\frac{W}{L}} \right)^2 + \left(\frac{\Delta I_B}{I_B} \right)^2 \right] + \Delta V_{th}^2} \tag{1.35}$$

where $\frac{\Delta I_B}{I_B}$ can be deduced from the empiric formula:

$$\sigma_{\Delta I_{ds}}(\%) = C_{nt}(WL)^{-0.5x} \tag{1.36}$$

where C_{nt} and x are constant deduced by devices measurements. In our case, $C_{nt} \simeq 1.6503$ and $x \simeq 0.8592$, which gives $\sigma_{\Delta I_B} = 0.9\%$. Recalling the UMC technology matching paramiters reported in Table 1.1, this leads to $V_{OS} \simeq 0.85$mV, well within the required value.

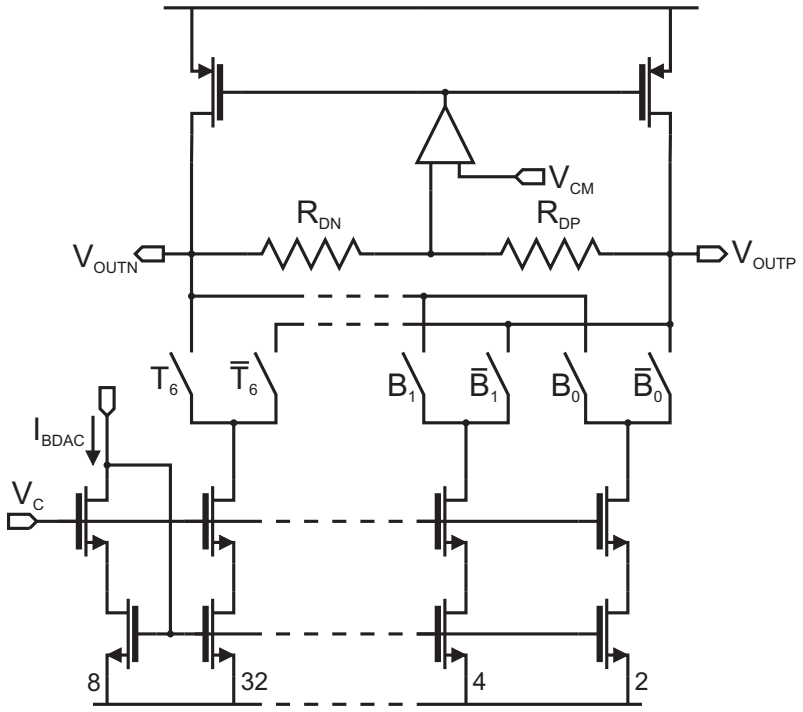


Figure 1.19: Simplified DAC schematic

The buffer frequency responce in typical conditions is shown in Fig.1.18. The buffer exhibits a low frequency gain $A_v = 58$dB with a unit gain frequency of $f_u = 200$MHz.

Corner analysis gives a worst case gain $A_v = 56.8$dB with a unit gain frequency of $f_u = 199$MHz and a phase margin of 78.3deg.

1.2.4 DAC

The DAC has to convert the digital representation of the channel output into the differential voltage that is fed to the input analog memory. We estimated a 7-bit resolution, corresponding to a $V_{LSB} = 2.5$mV, to be high enough to meet the precision requirements of (1.20). To ensure the DAC accuracy, we used for the 3 most significant bits a thermometer-code representation. The DAC is implemented using the differential switched-current architecture reported in Fig.1.19. The two resistors R_{DN} and R_{DP} convert the output current to a differential voltage according to:

$$\begin{aligned} V_{OUTP} - V_{OUTN} &= (R_{DP} + R_{DN}) \cdot \left(\frac{N}{64} - \frac{127}{128} \right) \cdot I_{BDAC} \\ &= 2 \cdot I_{BDAC} \cdot R_D \cdot \left(\frac{N}{64} - \frac{127}{128} \right) \end{aligned} \tag{1.37}$$

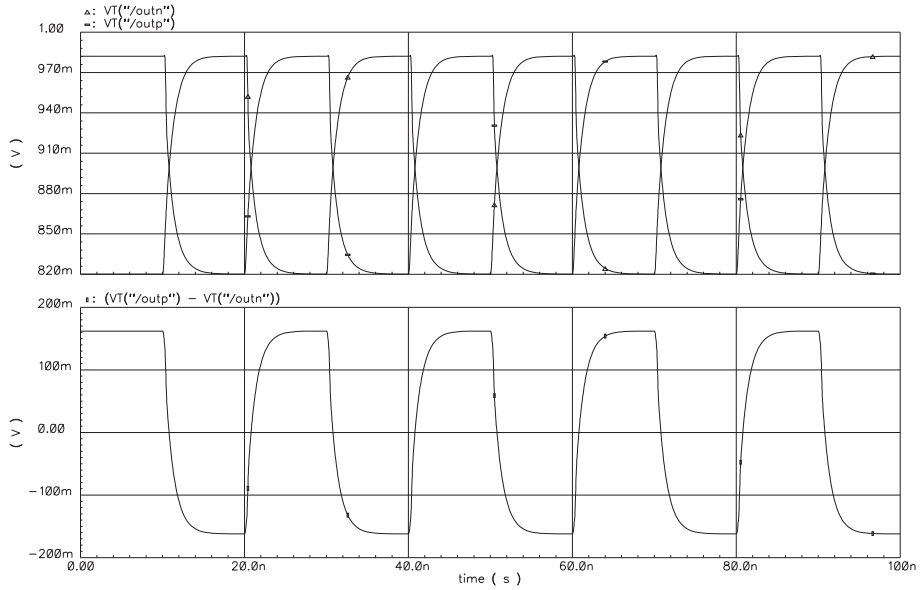


Figure 1.20: DAC transient output waveforms

where I_{BDAC} indicates the DAC reference current and we assume the two resistors to be equal, that is $R_{DN} = R_{DP} = R_D$.

The DAC reference current has been chosen equal to $I_{BDAC} = 102.25\mu$A so as to satisfy the write time specification, considering a load capacitance $C_{LDAC} \simeq 7$pF. According to (1.37), this leads to $R_D = 50\Omega$ and to an overall DAC current consumption of:

$$I_{TOT} = 7 \cdot 4 \cdot I_{BDAC} + 2 \cdot I_{BDAC} + I_{BDAC} + \frac{I_{BDAC}}{2} + \frac{I_{BDAC}}{4} = 3.246\text{mA} \tag{1.38}$$

where the first term is due to the thermometer-code most significant bits while the others are due to the last four significant bits.

The DAC performance have been estimated by means of MonteCarlo simulations considering both process and mismatch variations. The DAC has been fed with a sequence of input words switching between the maximum and minimum channel signal amplitude, which corresponds to $V_{in} = +162$mV and $V_{in} = -162$mV respectively. An example of the corresponding transient DAC output waveform is shown in Fig.1.20.

The results of 500 MonteCarlo iterations are shown in Fig.1.21, Fig.1.22 and Fig.1.23, where the differential voltage has been sampled at the farest memory cell input after a settling time of 9ns, thus considering also the error due to the input buffers. The maximum error mean is equal to $m_{\varepsilon_{DAC}} + m_{\varepsilon_{BUF}} \simeq 0.3$mV with a standard deviation of $\sqrt{\sigma^2_{\varepsilon_{DAC}} + \sigma^2_{\varepsilon_{BUF}}} \simeq 2.2$mV, thus well within the required precision.

The DAC layout has been drawn following the scheme proposed in [23], so as to

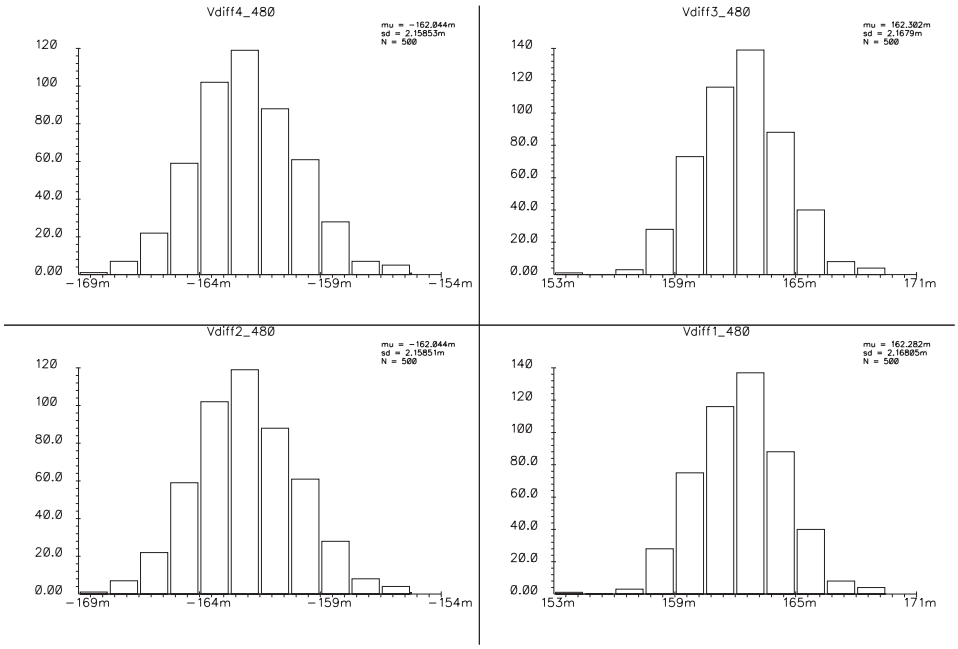


Figure 1.21: Pre-layout simulations error on 600 random codewords

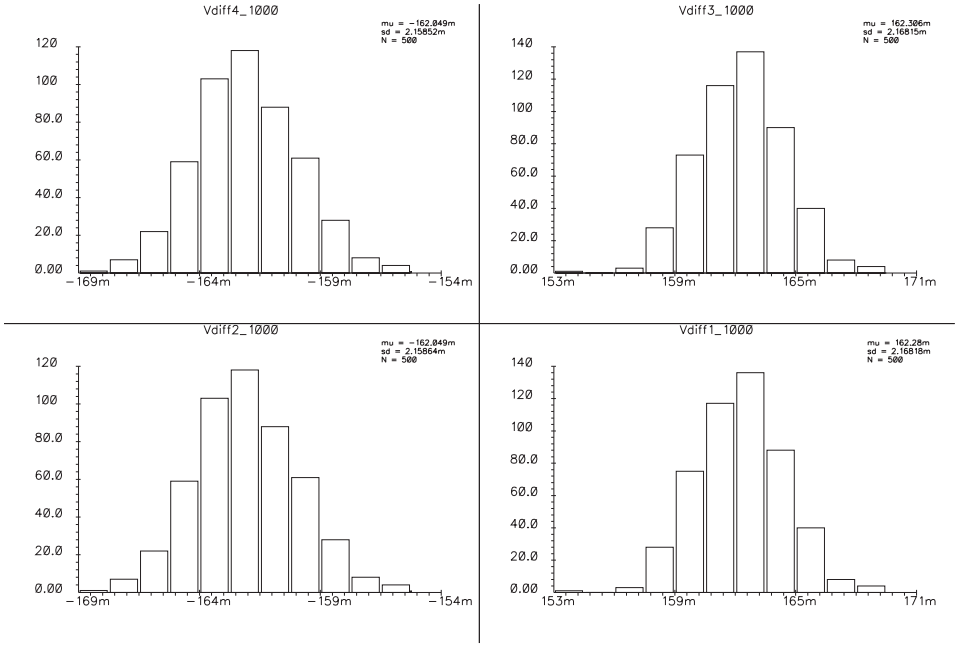


Figure 1.22: Post-layout simulations error on 600 random codewords

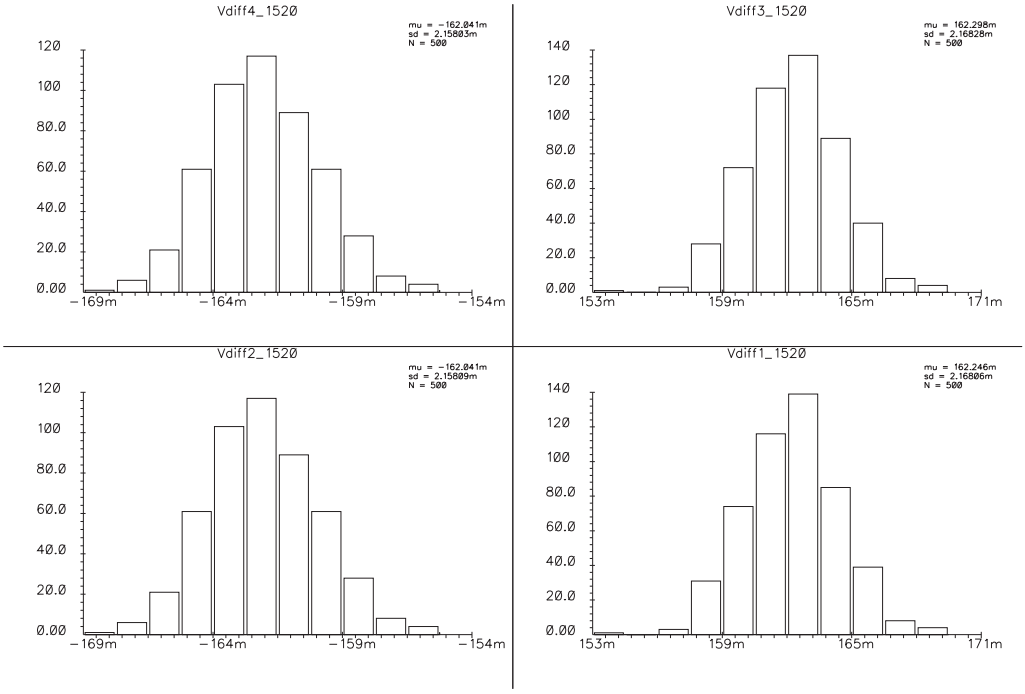


Figure 1.23: Post-layout simulations error on 600 random codewords

ensure an high accuracy.

1.3 Simulations Results

The performace of the decoder interface have been evaluated by simulating the memory read and write phase separately, the overall precision being calculated as:

$$\begin{aligned} m_{\varepsilon_t} &= m_{\varepsilon_r} + m_{\varepsilon_w} \\ \sigma_{\varepsilon_t} &= \sqrt{\sigma_{\varepsilon_r}^2 + \sigma_{\varepsilon_w}^2} \end{aligned} \tag{1.39}$$

where ε_r and ε_w are the errors introduced by the read and write process respectively.

1.3.1 Read Phase Simulations

In order to evaluate the input memory performance during the read phase, extensive MonteCarlo simulations with 500 iterations and considering both process and mismatch variations have been performed. As the error depends on the input voltage V_{in} being read (1.23) but also on the data read by means of the OTA during the previous read cycle $V_{in_{pre}}$, we have considered two different cases:

(a) $V_{in} = 162\text{mV}$ and $V_{in_{pre}} = -162\text{mV}$;

(b) $V_{in} = -162\text{mV}$ and $V_{in_{pre}} = 162\text{mV}$.

In both cases, the memory cell has been read up to 10 times and the corresponding output voltage V_{out}, together with the two voltage to probability converter drain currents I_0 and I_1, has been sampled after 10ns and 30ns from the beginning of each read phase.

As 3 bits of dynamic are required for the log-likelyhood ratio representation, the range of all possible values for I_0/I_1 can be deduced rewriting (1.6) as:

$$\lambda_k = \log \frac{I_0}{I_1} = [-2^{(3-1)}, 2^{(3-1)} - 1] \simeq [-4, 4] \tag{1.40}$$

from which we obtain:

$$\frac{I_0}{I_1} = [e^{-4}, e^4] = \left[\frac{1}{54}, 54\right] \tag{1.41}$$

Thus, when the differential input voltage $V_{in} = 162\text{mV}$, $I_0/I_1 \geq 54$ while when $V_{in} = -162\text{mV}$, $I_1/I_0 \geq 54$. The 1 bit of precision requires:

$$\sigma_{\frac{I_0}{I_1}} \leq e^{0.5} = 1.65 \tag{1.42}$$

The MonteCarlo simulations results are summarized in Table 1.6 for case (a) and in Table 1.7 for case (b). In both cases, two voltage references $V_{bx} = 300\text{mV}$ and $V_{by} = 700\text{mV}$ are considered for the nMOS of the SISO input stage. The common mode output voltage error is also reported, where the nominal common mode value is set to 900mV. As an example, the transient output waveform corresponding to case (a) are shown in Fig.1.24, while the MonteCarlo results for the tenth reading with $V_{in} = -162\text{mV}$, $V_{in_{pre}} = -162\text{mV}$ and $V_{bx} = 300\text{mV}$ are shown in Fig.1.25.

As expected, the error on the differential output voltage worsen with the number of reading to the point of not meeting the precision requirements. However, even if the simulated conditions represent the worst case for the error, when the input voltage $V_{in} = \pm 162\text{mV}$ we are far from the most critical situation for the SISO decoding process, that is represented by the equalprobability condition of two symbols.

	$R_{1,10ns}\,V_{bx}$	$R_{1,10ns}\,V_{by}$	$R_{1,30ns}\,V_{bx}$	$R_{1,30ns}\,V_{by}$	$R_{10,10ns}\,V_{bx}$	$R_{10,10ns}\,V_{by}$	$R_{10,30ns}\,V_{bx}$	$R_{10,30ns}\,V_{by}$
m_{ε_r} [mV]	-0.202	-2.066	0.209	-0.709	5.709	-5.582	6.321	-4.292
σ_{ε_r} [mV]	3.266	3.406	3.269	3.393	4.588	5.092	4.621	5.083
$m_{\frac{I_1}{I_0}}$	2.107	17.417	63.394	60	2.616	22.278	71.667	55.098
$\sigma_{\frac{I_1}{I_0}}$	0.329	3.569	7.568	7.357	0.480	4.469	9.849	8.95
$m_{\varepsilon_{rcm}}$ [mV]			45.92	44.206			25.686	5.927
$\sigma_{\varepsilon_{rcm}}$ [mV]			6.229	6.216			12.43	12.213

Table 1.6: $V_{in} = 162\text{mV}$ and $V_{in_{pre}} = -162\text{mV}$

	$R_{1,10ns}\,V_{bx}$	$R_{1,10ns}\,V_{by}$	$R_{1,30ns}\,V_{bx}$	$R_{1,30ns}\,V_{by}$	$R_{10,10ns}\,V_{bx}$	$R_{10,10ns}\,V_{by}$	$R_{10,30ns}\,V_{bx}$	$R_{10,30ns}\,V_{by}$
m_{ε_r} [mV]	0.169	-1.71	0.565	-0.353	4.22	-5.049	4.828	-3.755
σ_{ε_r} [mV]	3.273	3.437	3.275	3.4133	4.477	5.159	4.516	5.132
$m_{\frac{I_1}{I_0}}$	2.085	17.05	64.07	60.627	2.615	22.285	72.716	55.981
$\sigma_{\frac{I_1}{I_0}}$	0.331	3.571	7.579	7.408	0.482	4.679	9.729	9.204
$m_{\varepsilon_{rcm}}$ [mV]			26.52	24.187			11.424	6.661
$\sigma_{\varepsilon_{rcm}}$ [mV]			6.229	6.216			16.251	9.010

Table 1.7: $V_{in} = -162\text{mV}$ and $V_{in_{pre}} = 162\text{mV}$

	$R_{1,10ns}V_{bx}$	$R_{1,10ns}V_{by}$	$R_{1,30ns}V_{bx}$	$R_{1,30ns}V_{by}$	$R_{10,10ns}V_{bx}$	$R_{10,10ns}V_{by}$	$R_{10,30ns}V_{bx}$	$R_{10,30ns}V_{by}$
m_{ε_r} [mV]	0.959	0.063	0.973	0.064	1.833	0.192	1.904	0.195
σ_{ε_r} [mV]	3.257	3.340	3.267	3.393	4.261	4.854	4.285	4.948
$m_{\frac{I_1}{I_0}}$	0.997	1.002	0.981	1.003	0.962	1.002	0.812	1.004
$\sigma_{\frac{I_1}{I_0}}$	0.014	0.085	0.997	0.105	0.022	0.120	0.089	0.138
$m_{\varepsilon_{rcm}}$ [mV]			26.52	24.187			25.599	5.865
$\sigma_{\varepsilon_{rcm}}$ [mV]			6.229	6.216			12.238	12.081

Table 1.8: $V_{in} = 0\text{mV}$ and $V_{in_{pre}} = 162\text{mV}$

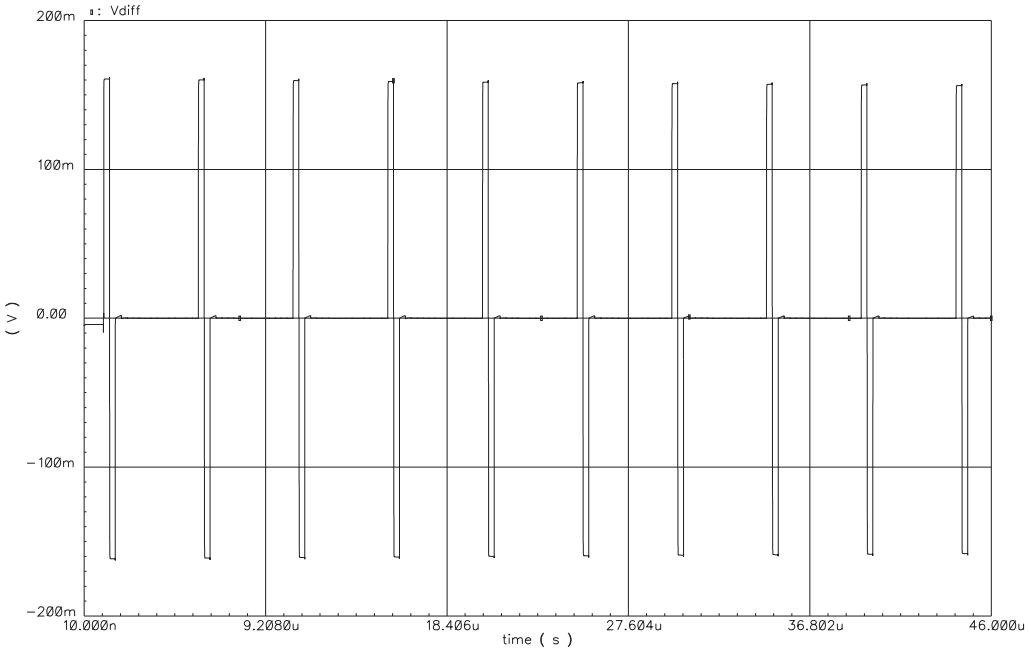


Figure 1.24: Memory transient output waveforms

It is thus useful to analyze the input memory error when $V_{in} = 0$V. From the results reported in Table 1.8 with $V_{in_{pre}} = 162$mV, we see how in this case the error mean is reduced with respect to both case (a) and (c) as expected from (1.23), while its standard deviation is comparable with the one obtained with $V_{in} = \pm 162$mV.

However, we will see how the error introduced by the memory cell during the read phase is dominant with respect to the one introduced by the memory write process, leading to the fact that the obtained resultsare good enough to meet the overall input interface specifications, as we will see in Sec.1.4.

It is important to notice how the voltage to probability output currents I_0 and I_1 require a longer time to settle with respect to the memory output voltage V_{out} when the full dynamic is required. In fact, after a read time of 30ns, they exhibit the desired dynamic but with a large distribution, due to process variations. However, this has a negligible impact on the overall decoder performance as a read time of 40ns brings the standard deviation value back within the required limits given by (1.42).

The error on the common mode voltage is well within the required bounds, as a maximum error of ± 50mV can be tolerated [24].

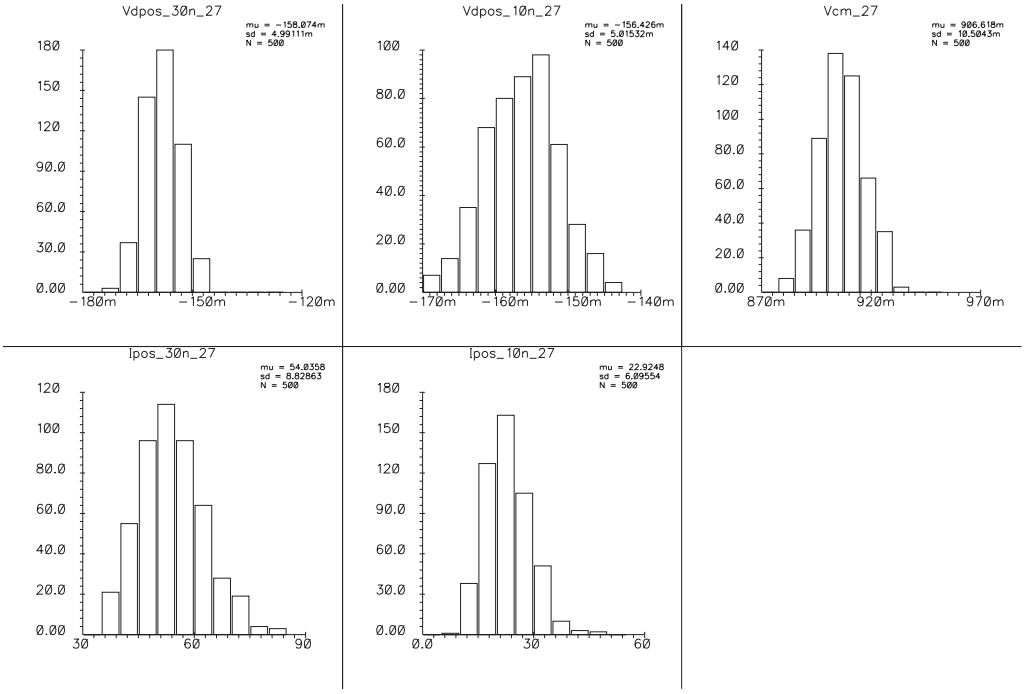


Figure 1.25: MonteCarlo simulation results with $V_{in} = -162\text{mV}$ and $V_{in_{pre}} = -162\text{mV}$

1.3.2 Write Phase Simulations

The performance of the input interface during the write phase have been evaluated by means of MonteCarlo simulations, considering both mismatch and process variations. The DAC has been fed with 600 7-bit random codewords, which has been given a time of 9ns to be stored into the farest memory cells from the DAC output, that are the first and last cell of the eighth memory row. For each codeword, 100 instances have been simulated.

The MonteCarlo simulations results reported in Fig.1.26 show a maximum error mean $m_{\varepsilon_w} = 523\mu\text{V}$ with a standard deviation $\sigma_{\varepsilon_w} = 2.221\text{mV}$. Thus, considering the worst case error during read phase and the one introduced by the voltage to probability converter, from equation (1.39) we obtain:

$$\begin{aligned} m_{\varepsilon_t} &= 6.321 + 0.523 = 6.884\text{mV} \\ \sigma_{\varepsilon_t} &= \sqrt{5^2 + 2^2 + 1.85^2} \simeq 5.7\text{mV} \end{aligned} \tag{1.43}$$

where no systematic offset has been considered for the voltage to probability module. The results obtained are in line with the precision specifications.

The corresponding transient waveforms are shown in Fig.1.27, together with the DAC input value.

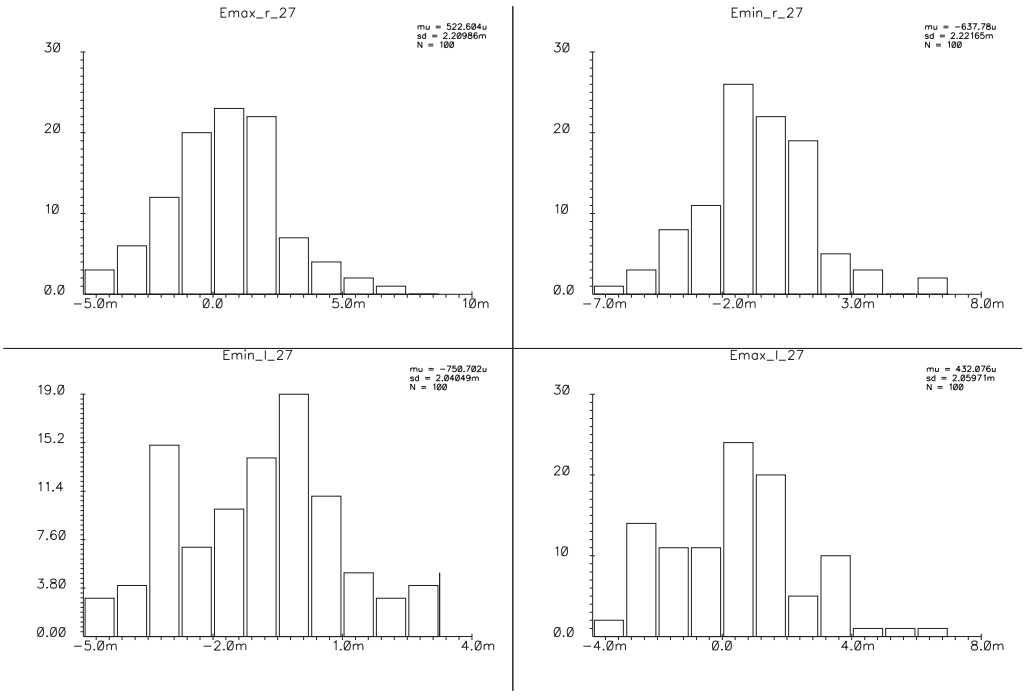


Figure 1.26: MonteCarlo write phase transient simulation results

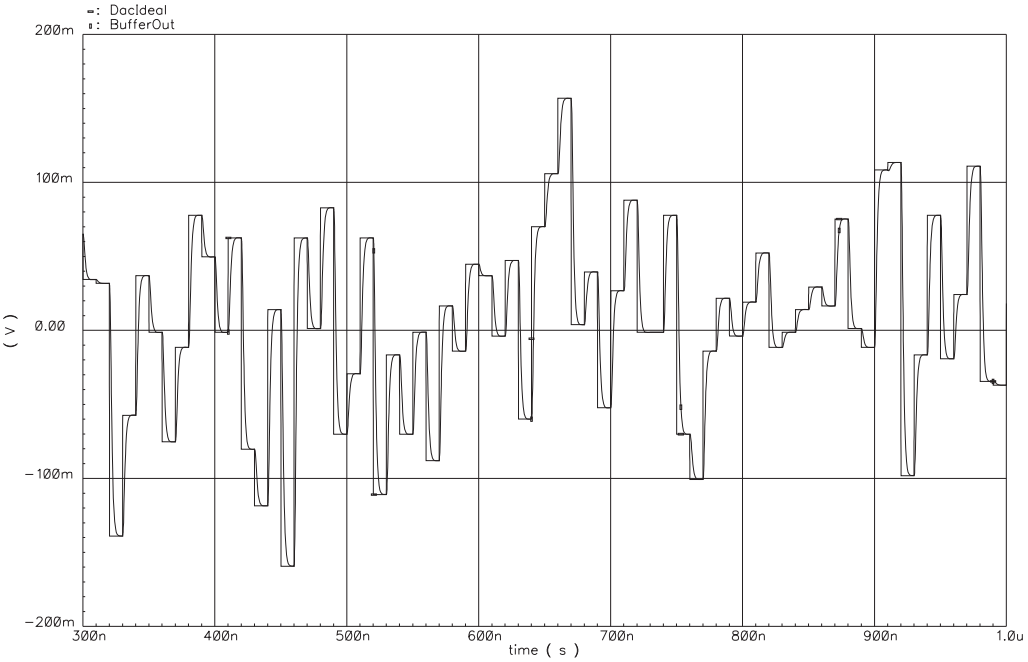


Figure 1.27: MonteCarlo DAC output waveforms

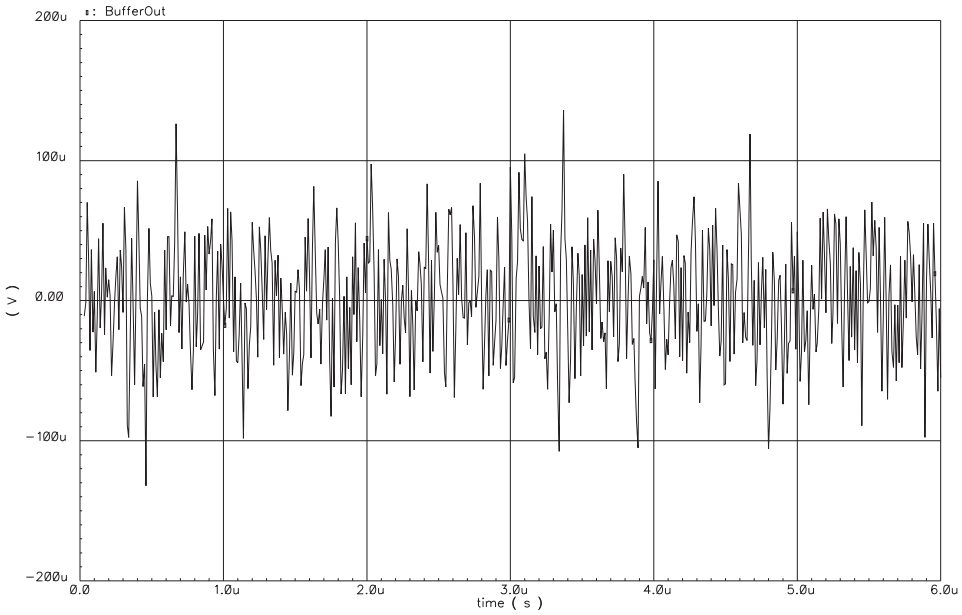


Figure 1.28: Pre-layout simulations error on 600 random codewords

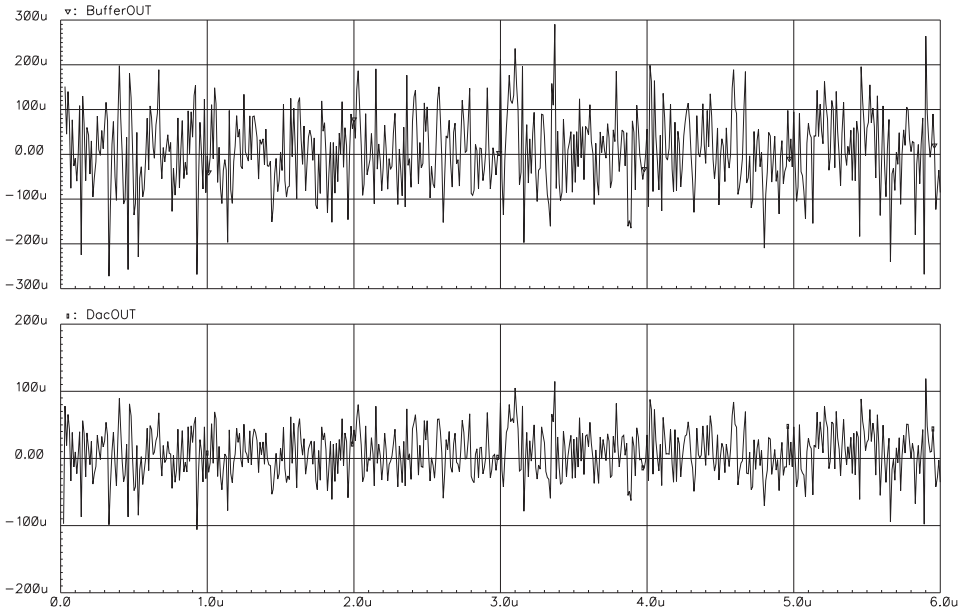


Figure 1.29: Post-layout simulations error on 600 random codewords

Post-layout simulations show how the influence of parasitic on the precision performance is negligible. In Fig.1.28 and Fig.1.29 the error at the DAC and buffer output on 600 random codewords is reported for pre- and post-layout simulations, respectively.

1.3.3 Power Consumption

As the input memory consists of $604 \cdot 2$ OTAs, two for each memory column, and $8 \cdot 4$ buffers, four for each memory row, the total power consumption can be easily computed as:

$$\begin{aligned} P_{memory} &= 604 \cdot 2 \cdot P_{OTA} + 8 \cdot 4 \cdot P_{buffer} \\ &= (604 \cdot 2 \cdot I_{B\_OTA} + 8 \cdot 4 \cdot I_{B\_BUFFER}) V_{DD} \\ &= (604 \cdot 2 \cdot 10.5\mu\text{A} + 8 \cdot 4 \cdot 260\mu\text{A}) 1.8\text{V} \\ &\simeq 38\text{mW} \end{aligned}$$

This result slightly exceeds the power consumption specification given in Sec.1.2.1, leading to an overall decoder power consumption of $\simeq 70$mW. According to (1.7), this translates into an energy per decoded bit of 2.1nJ, which is very close to the original target of 2nJ/bit.

1.4 Conclusions

The input interface for an hybrid analog decoder has been designed in the UMC 0.18-μm CMOS process. Simulations results show how the interface performance meet the required specifications, both in terms of precision and speed. The overall circuitry area occupation is $\simeq 36\text{mm}^2$ with a power consumption of $\simeq 40$mW.

The interface is part of a fully analog iterative decoder for a serially concatenated convolutional code, reconfigurable in both block length and code rate. The decoder exhibits an efficiency of 2.1nJ/bit which outperforms digital decoders with the same block length, that is around 5000, of a factor up to 50 [25]. The chip, whose layout is shown in Fig.1.30, has been fabricated in UMC 0.18-μm CMOS process and is now under test.

The solution of an hybrid decoder implemented with an analog core and memory seems a promising strategy to tackle the limits of traditional analog implementations. In fact, this structure shows the advantages of the analog approach without its drawbacks, which are mainly due to the linear dependence between code block length and decoder dimensions.

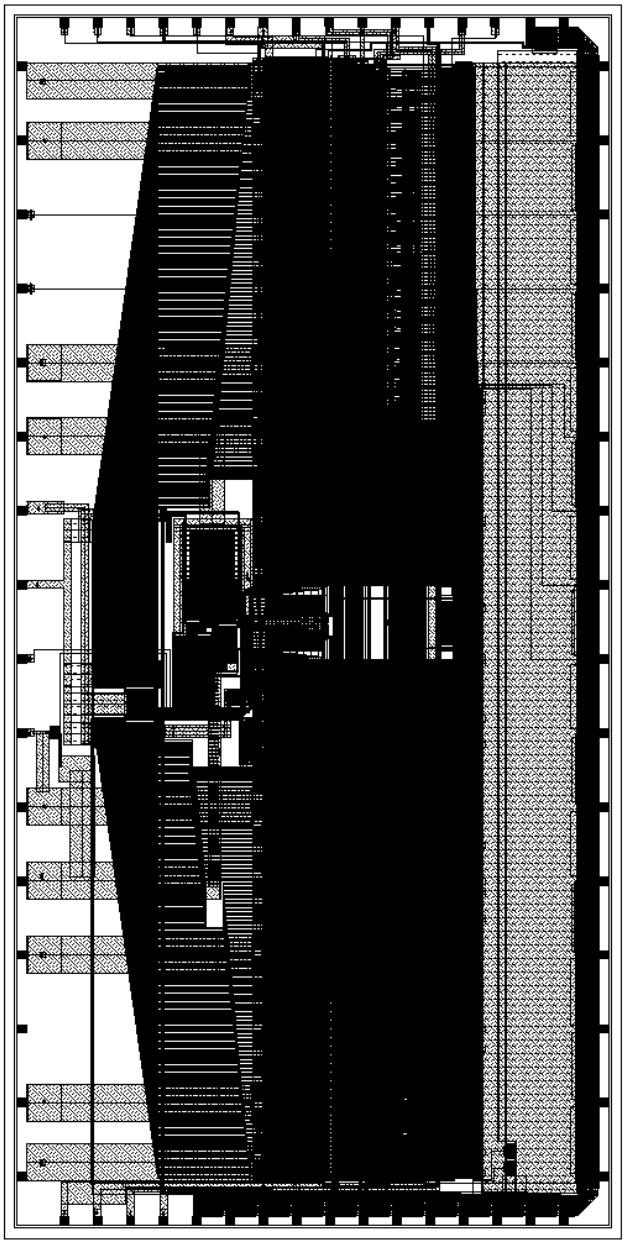


Figure 1.30: Hybrid analog decoder layout

2

Analog Decoding for Data Storage Applications

The demand of high density, high throughput solid-state nonvolatile memories has been constantly increasing in the past decade, due to tendency to convert and store images, videos and music in digital format. The core of all nonvolatile semiconductors memory devices is a matrix of single memory cells, which maintain their state even without supply voltage. In order to reach higher memory density, continual efforts to reduce the single cell area have been made, scaling the process lithography. In addition to shrinkage of the feature size, the memory density can be increased by storing more information bits within a single cell. However this rises new reliability issues, to couple with multilevel cells memories resort to use on-chip error correction code (ECC).

In this chapter, after a brief overview of flash memories technology, the advantages and drawbacks of commonly used ECC, such as linear block codes, are analyzed. Thus a new ECC scheme for multilevel flash memories, based on trellis coded modulation strategy, is proposed.

2.1 Flash Memories

The core of a flash memory consists of an array of memory cells placed on a word-line/bit-line grid. Although in the past different types of flash architectures have been proposed, today two of them can be considered as a standard: the common ground NOR flash that, due to its fast random read access time, is attractive for applications such as program-code storage and the NAND flash, optimized for high density data storage.

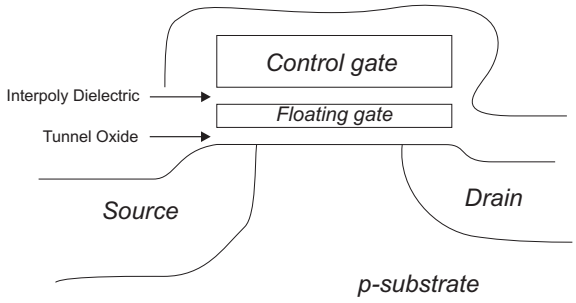


Figure 2.1: Flash cell cross section

2.1.1 Floating Gate Transistor

The flash cell of both memory types is basically the floating-gate MOS transistor shown in Fig.2.1, that is a transistor with a gate completely surrounded by dielectrics, the floating gate, and electrically governed by a capacitive coupled control gate [26]. Applying a high voltage between the source and the gate-drain terminals of the floating-gate MOS, causes electrons to be injected in the floating gate which, being electrically isolated, acts as a storing electrode for the device.

The charge injected onto the floating gate effectively shifts the I-V curves of the transistor, as shown in Fig.2.2, thus allowing modulation of the apparent threshold voltage V_T seen from the control gate. Usually the neutral (or positively charged) state is associated with the logic state “1” while the negatively charged state, corresponding to electrons stored in the floating gate, is associated with the logical “0”.

2.1.2 NOR and NAND Flash

NOR and NAND flash memories [27] use the same basic cell memory described in Sec.2.1.1 but differ in the way the cells are arranged in an array, leading to different characteristics both in terms of memory density and flexibility.

In the NOR architecture, the cells are arranged in a matrix through rows and columns in a NOR-like structure, as shown in Fig.2.3. Flash cells sharing the same gate constitute a *wordline*, while those sharing the same drain contact constitute the *bitline*. In this array organization, every cell contains also a source contact. All the cells sources are connected to a common source electrode, which is usually connected to the ground.

The data stored in a NOR cell can be determined by measuring the threshold voltage of the floating gate MOS transistor. The best and fastest way to do it is by reading the

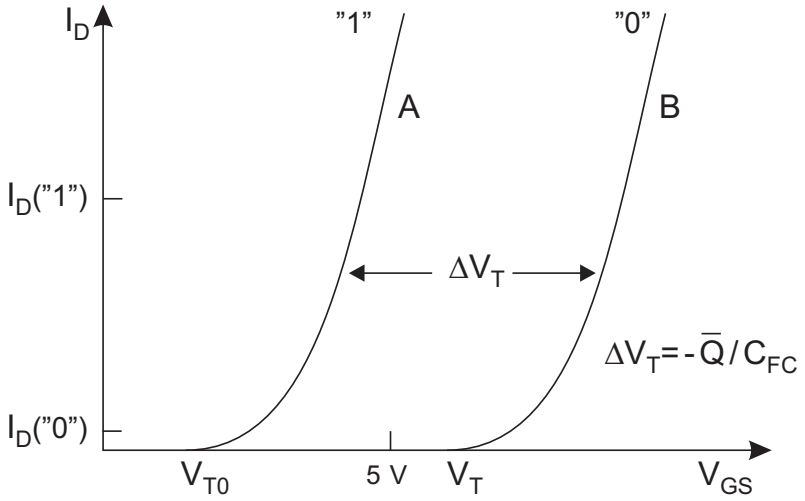


Figure 2.2: I-V curves of a floating-gate MOS without (curve A) and with (curve B) electrons stored in the floating gate

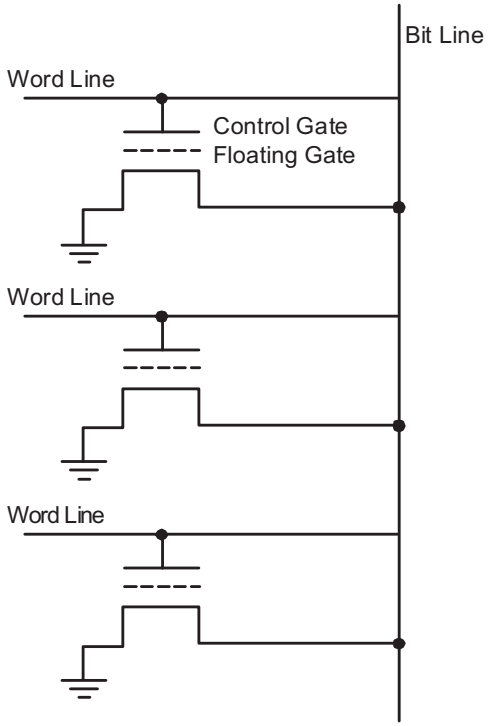


Figure 2.3: NOR flash array

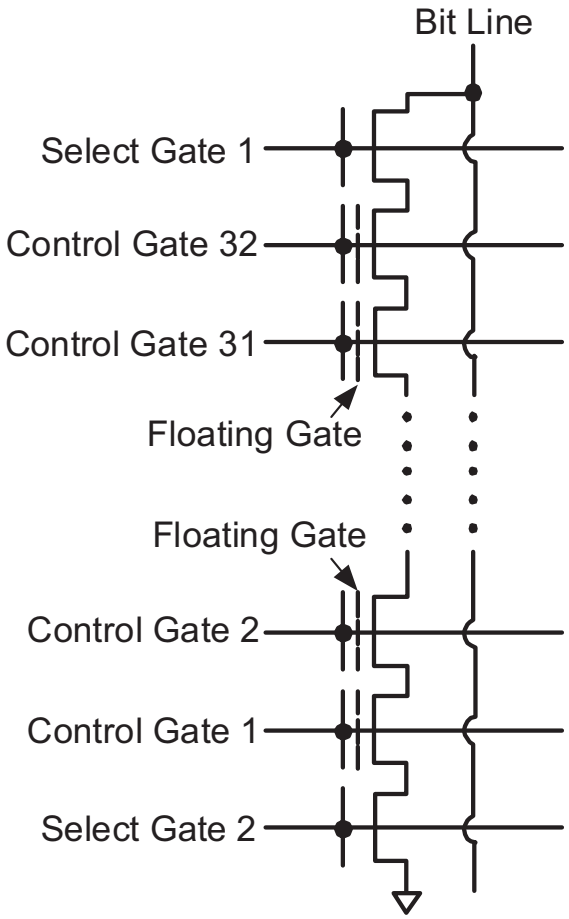


Figure 2.4: NAND flash array

current driven by the selected cell at a fixed gate voltage and then comparing it with the current of a reference cell.

In order to write or to erase a flash cell, that is to force electrons above or across the dielectrics energy barrier so as to inject them onto the floating gate or to remove them from it, two physical mechanisms are used, which exploit two different physical effects:

- the *channel hot electron* mechanism, where electrons gain enough energy to pass the oxide-silicon barrier thanks to the electric field in the transistor channel between source and drain;

- the *Fowler-Nordhein electron tunneling* mechanism, where a quantum-mechanical tunnel is induced across a thin tunneling oxide between the silicon surface and the floating gate by applying a strong electric field.

A NOR flash memory cell is programmed by channel hot electron injection in the floating gate at the drain side and it is erased by means of the Fowler-Nordhein electron tunneling through the tunnel oxide from the floating gate to the silicon surface.

Using the NOR architecture, cells can be accessed directly, thus leading to fast random read access time. At the same time, the programming times are slow due to the need for precise control of the thresholds.

These properties make this style of flash memory attractive for applications such as program-code storage. Other applications, such as video or audio file storage, do not need fast random access, but are better served by large storage density, fast erasure and programming, and fast serial access.

These requirements are more readily provided by the NAND architecture, where the basic module consists of 16 or 32 floating-gate transistors connected in series, as shown in Fig.2.4. This chain is connected to the bit line and to the source line by means of two select transistors. By eliminating all contacts between word lines, the resulting cell size is approximately 40% smaller than the NOR cell.

To read a NAND cell, all the other memory cells connected in series with the selected one have to be activated by applying a gate voltage higher than the maximum programmable threshold voltage. The word line of the selected cell is biased at a fixed voltage, so as to conduct only if in the neutral or logic “1” equivalent state.

The programming and erasing of NAND flash are both performed using the Fowler-Nordhein electron tunneling mechanism, which reduces the current requirements compared to the channel hot electron one, thus allowing for the programming of many modules in parallel while keeping power consumption under control.

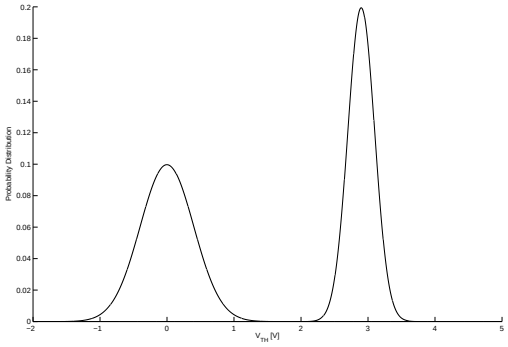
2.1.3 Multilevel Flash Cell

The most efficient way to scale the actual cell size for any given technology is offered by the multilevel concept [28–32].

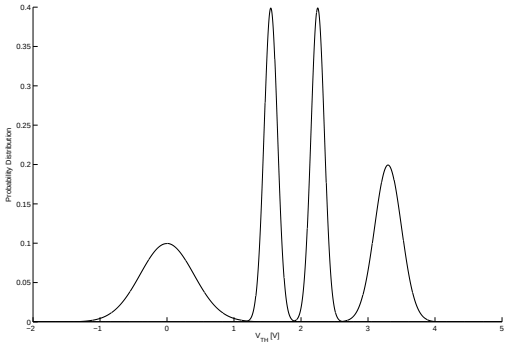
The multilevel idea is based on the ability to precisely control the amount of charge injected in the floating gate, so that the threshold voltage of each cell can be programmed to any of $q = 2^m$ levels, with $m > 1$, each corresponding to a different logical state.

A single cell programmed using the multilevel approach is thus capable of storing m bits, reducing considerably the cost-per-bit.

The threshold distributions of a 1-bit and a 2-bit memory cell are shown in Fig.2.5 [33]. With the multilevel approach, all the programmed threshold voltage levels must be



(a) Bilevel memory cell



(b) 4-level memory cell

Figure 2.5: Conceptual representation of bilevel and multilevel threshold voltage distributions

allocated within a predetermined voltage window. This window is enlarged with respect to the case of conventional bilevel memories to provide more room for the stored levels. However, in practice, this increase can not be very high, so as to limit charge transfer through the gate oxide during program/erase operation and prevent excessive voltages across the oxide during storage time.

This leads to a reduced spacing between adjacent programmed voltage levels, which makes the reliability of multilevel memories more and more critical as the number of bits/cell increases.

2.1.4 Reliability Issues

Reliability issues are particularly critical for flash memories as data retention must be guaranteed after ten years of storage at room temperature and at least 10^5 read, program

and erase cycles.

Several failure mechanisms can affect flash memory reliability, even if they can be traced back to three main effects: read disturb, program distrub and data retention degradation [34].

Read Disturb Read disturb affects the cell under reading and a number of unselected cells, that is the cells placed in the same wordline of the cell being read in a NOR flash memory and the cells belonging to the same wordline and the ones connected in series with the selected one in a NAND-type memory.

Due to the high voltages applied to the gate of these unselected cells, a so called soft-programming mechanism can take place, giving rise to an undesired threshold voltage shift, especially for cells programmed to low level.

Read disturb becomes more critical after program/erase cycling, due to the oxide degradation caused by the high electric fields applied during write and erase operations.

A secondary failure mechanism that takes place in NAND-type flash during read operation is related to the dependency of the reading current on the so called *Background Pattern*. The current driven by the cell being read can vary considerably with the threshold voltage levels programmed in the cells connected in series with the selected one.

Program Disturb Program disturb leads to an undesired threshold voltage shift due to soft programming in the unselected cells that experience the high voltages applied to write the memory cells. In order not to seriously affect the memory reliability, the programming voltages must be carefully chosen.

Program disturb in NAND-type arrays are also due to the capacitive coupling of adjacent cells floating gates. Several different programming sequences and algorithms have been studied to limit this effect, which results particularly detrimental in multilevel memories.

Data Retention Data retention degradation, due to leakage of electrons from the floating gate through the surrounding oxide, depends on the amount of charge stored and hence on the threshold voltage shift of programmed levels. It worsens in the presence of oxides degraded due to program/erase cycling.

All the reliability issues described above become more critical in the case of multilevel flash memories as compared to the bilevel ones, due to the reduced spacing between

adjacent programmed levels and to the higher threshold shift of the highest programmed states.

In particular, the enhanced cell threshold voltage range worsens the extrinsic charge loss, because this phenomena strongly depends on tunnel oxide retention electric field. Moreover, program disturbs are made more severe by the longer programming time necessary for multilevel programming.

2.2 Error Correcting Codes for Multilevel Flash Memories

As already pointed out in Sec.2.1.4, in multilevel flash memories issues such as disturbs and data retention become more and more critical due to the reduced space between adjacent programmed threshold voltage levels. As a consequence, multilevel memories are increasing relaying on error correction code techniques to ensure adequate reliability, in particular in all those applications where a large number of program/erase cycles are required. This is especially true for memories capable of storing more than two bits per cells, such as the 16-level NAND flash presented in [35].

As far as the error correction capability requirement has been moderate, that is for 1 or 2 bit/cell memory, linear block codes, such as Hamming or BCH codes, have been the ideal choice as they combine good performance and relative design simplicity [36]. Nonetheless, their complexity is deemed to increase significantly with the number of bits/cell, since a larger correction capability will be required in order to keep memory reliability the same.

The BCH decoder for 2 bit/cell NAND Flash presented in [36] exhibits a correction capability up to 5 errors with a letency time increasing from the 60μs required to detect a single error to the 250μs needed to correct 5 errors. Once the error condition has been detected by syndromes calculation, the Berlekamp-Massey algorithm [37, 38] is used to compute the errors locator polynomial. Then, the error positions are found by a Chien machine, exploiting polynomial roots search in $GF(2^{15})$. Since this operation is the most time-consuming, parallelism should be exploited. However, the complexity and area overhead of Chien finite-state machine grow dramatically as the parallelism increases. Since single error is more likely to happen than multiple errors in 2 bit/cells, two different Chien cores are implemented: a simplified Chien core finds single-error position while another one manages 2 to 5 errors cases.

However, it is worth to notice that the correction of a single-bit error, which is generally sufficient for bilevel flash, is not satisfactory for multilevel memories, where an error can in principle affect all the bits stored in a faulty cell. Codes that process all the bits belonging to the same cell as a symbol and are therefore capable of detecting and correcting symbol errors rather than bit errors could be more appropriate, even though at the cost of larger area of the decoding circuitry and increased access time penalty.

The key requirements of an ECC for a flash memory are a reduced area overhead, including that coming from the storage of parity information, minimum impact on access time and data transfer rate and a limited power consumption increase due to the coding and decoding circuitry. Moreover, any error corresponding to the failure of a single cell, which involves up to m bits in multilevel memories, must be corrected.

The last requirement can be fulfilled in a multilevel flash by using nonbinary codes, which are based on arbitrary finite alphabets with more than two symbols [39]. The same way as the content of a bilevel cell is associated to a binary digit, the content of a q-level cell, with $q > 2$, can be associated to a q-ary symbol. In such a way, a single-cell error corresponds to a single-symbol error that can be handled easily using a q-ary code. Many of the error correcting schemes used for bilevel memories can be fitted to multilevel ones by replacing binary codes with nonbinary codes.

2.3 q-ary Hamming Codes

As a simple case study, we consider a q-ary Hamming Code, with $q > 2$. The concepts described in Appendix Sec.A.2.2 for binary Hamming code, that is with $q = 2$, can be easily extended to the case of q-ary Hamming codes. In particular, for any integer $r \geq 2$, q-ary Hamming codes have block length $n = (q^r - 1)/(q - 1)$ and data length $k = (q^r - 1)/(q - 1) - r$. The Hamming bound is given by (A.5), where t indicates the error correction capability in terms of symbols.

The parity check matrix $\mathbf{H}$ over the Galois field $GF(2^m)$ can be constructed by choosing, as columns, all the nonzero r-uples of elements from $GF(2^m)$ in such a way that all the columns of $\mathbf{H}$ are linearly independent from one another. In this way we can construct a single-symbol ECC.

To implement this kind of code for nonbinary symbols there are two possible approaches. In the first approach, sum and multiplication operations over $GF(2^m)$ are implemented, so that encoder e decoder circuits are directly obtained from the nonbinary

parity check matrix **H**. In the second, the nonbinary parity check matrix **H** is transformed into a binary form and standard binary operations are then implemented.

2.3.1 Analog Decoding

Linear block codes can be efficiently decoded in the analog domain by means of the Gallager algorithm [40], proposed by Galleger in 1962 to decode binary Low Density Parity Check (LDPC) codes.

The implementation of a CMOS analog decoder for binary Hamming code has already been demonstrate in [41], with encouraging results with respect to the digital counterpart, both in terms of area occupation and power consumption.

The basic building blocks of such a decoder are the *soft-gates* described in Appendix Sec.A.4.2. By choosing a different alphabet rather than the binary one $\{0,1\}$ for X, Y and Z, we can easily realize in the analog domain sum and multiplication operations over $GF(2^m)$, from which the implementation of analog decoders for nonbinary codes is straightforward.

As an example, let's consider the shortened Hamming code $(36,32)$ over $GF(4)$ presented in [42], whose parity-check matrix **H** is given by:

$$\mathbf{H} = \left[\mathbf{P}^T | \mathbf{I}\right] \tag{2.1}$$

where the nonsystematic part $\mathbf{P}^T$ is the 4 by 32 matrix:

$$\mathbf{P}^T_{(36,32)} = \begin{pmatrix} 00011100000001111111110000111111 \\ 01100100111110000011231111000011 \\ 10101011001230012301001123112300 \\ 11010023231002310010002311231123 \end{pmatrix} \tag{2.2}$$

The correspondence between binary and $GF(4)$ notation is given in Table.2.1.

The analog decoder complexity for such a code can be easily estimated from matrix $\mathbf{P}^T$ as each row of $\mathbf{P}^T$ corresponds to a check node, implemented in the analog domain by a *soft-XOR*, while the number of the decoder *equal-gates* is given by the information data symbols, each one represented by a matrix $\mathbf{P}^T$ column. The *soft-XOR* and *equal-gate* inputs are given by all non zeros elements of the corresponding matrix row or column. As the Hamming code $(36,32)$ is defined over $GF(4)$, the *soft-XOR* and *equal-gate* operations, which implement the sum and multiplication operations module 4 described in Table 2.2, have also to be carried out over $GF(4)$.

$GF(4) \leftrightarrow$ *binary notation*
$0 \leftrightarrow 00$
$1 \leftrightarrow 01$
$2 \leftrightarrow 10$
$3 \leftrightarrow 11$

Table 2.1: $GF(4) \leftrightarrow$ binary notation

$+$	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

$\times$	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	3	1
3	0	3	1	2

Table 2.2: Sum and multiplication over $GF(4)$

The equation (A.56) for *soft-XOR* gates over $GF(4)$ can be written as:

$$\begin{bmatrix} p_z(0) \\ p_z(1) \\ p_z(2) \\ p_z(3) \end{bmatrix} = \begin{bmatrix} p_x(0)p_y(0) + p_x(1)p_y(1) + p_x(2)p_y(2) + p_x(3)p_y(3) \\ p_x(0)p_y(1) + p_x(1)p_y(0) + p_x(2)p_y(3) + p_x(3)p_y(2) \\ p_x(0)p_y(2) + p_x(1)p_y(3) + p_x(2)p_y(0) + p_x(3)p_y(1) \\ p_x(0)p_y(3) + p_x(1)p_y(2) + p_x(2)p_y(1) + p_x(3)p_y(0) \end{bmatrix} \tag{2.3}$$

while for $GF(4)$ *equal-gates* equations (A.57) becomes:

$$\begin{bmatrix} p_z(0) \\ p_z(1) \\ p_z(2) \\ p_z(3) \end{bmatrix} = \begin{bmatrix} p_x(0)p_y(0) \\ p_x(1)p_y(1) \\ p_x(2)p_y(2) \\ p_x(3)p_y(3) \end{bmatrix} \tag{2.4}$$

The relative circuit implementations are shown in Fig.2.6 and Fig.2.7 respectively.

The analog decoder core consists of two 18-inputs *soft-XOR* for the first two rows of the matrix $\mathbf{P}^T$, one 20-inputs *soft-XOR* for the third row and one 22-inputs *soft-XOR* for the fourth row. As an N-inputs soft-gate can be translated into $N-1$ 2-inputs soft-gates and each check node must be replied as many times as its input variables number, the decoder will consist of

$$2 \cdot 18 \cdot (18-1) + 20 \cdot (20-1) + 22 \cdot (22-1) = 1454$$

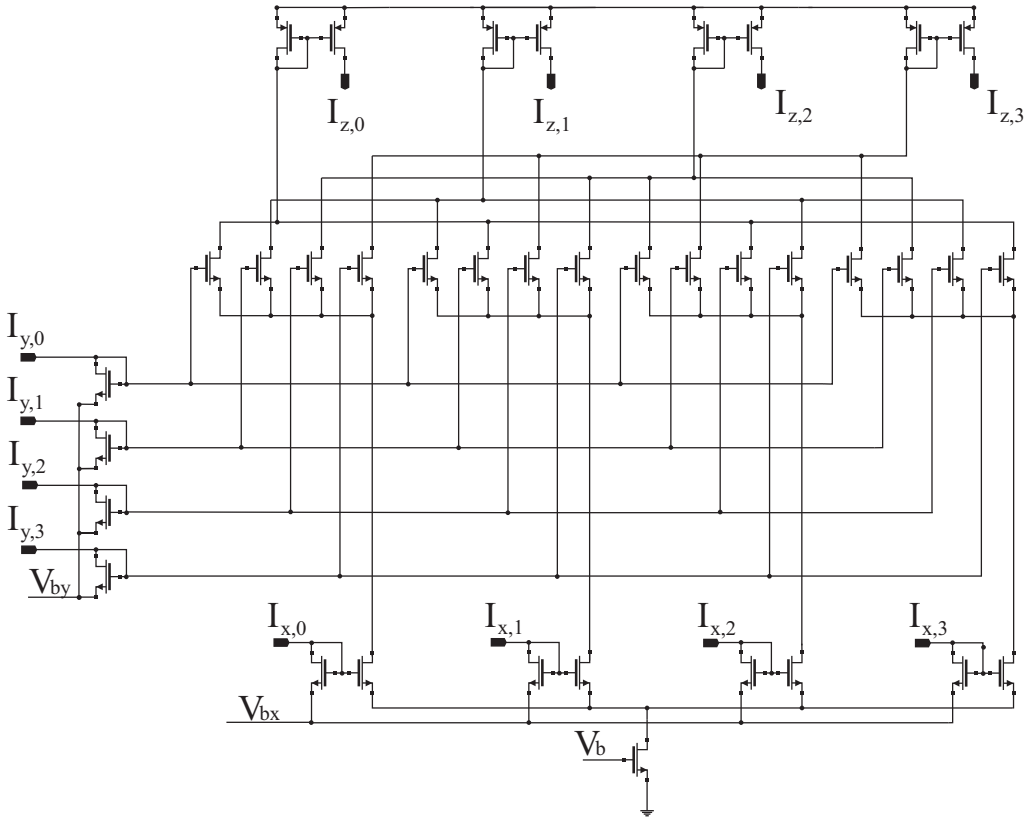


Figure 2.6: 4-ary *soft-XOR* circuit implementation

2-inputs *soft-XOR*.

In the same way, the *equal-gates* number can be estimated by considering that each column of matrix $\mathbf{P}^T$ corresponds to an *equal-gate* whose inputs number is given by the column non zero elements. This leads to 18 3-inputs and 14 4-inputs *equal-gates*, thus giving a total amount of

$$18 \cdot (3-1) + 14 \cdot (4-1) = 78$$

2-inputs *equal-gates*. The decoding circuitry requires a total count of 1532 2-inputs *soft-gates*, which is slighter greater than the equivalent gates count given for a digital decoder in [42].

The soft decoder performance have been estimated by means of a C++ behavioral model of the decoder. In the model, the sum-product operations of the MAP decoder are ideal, with probabilities represented by double precision numbers and no source of distortion, offset or noise is taken into account. In Fig.2.8 the analog decoder performance after a finite number of iterations $N_t = 6$, that is the diamond curve, are compared with

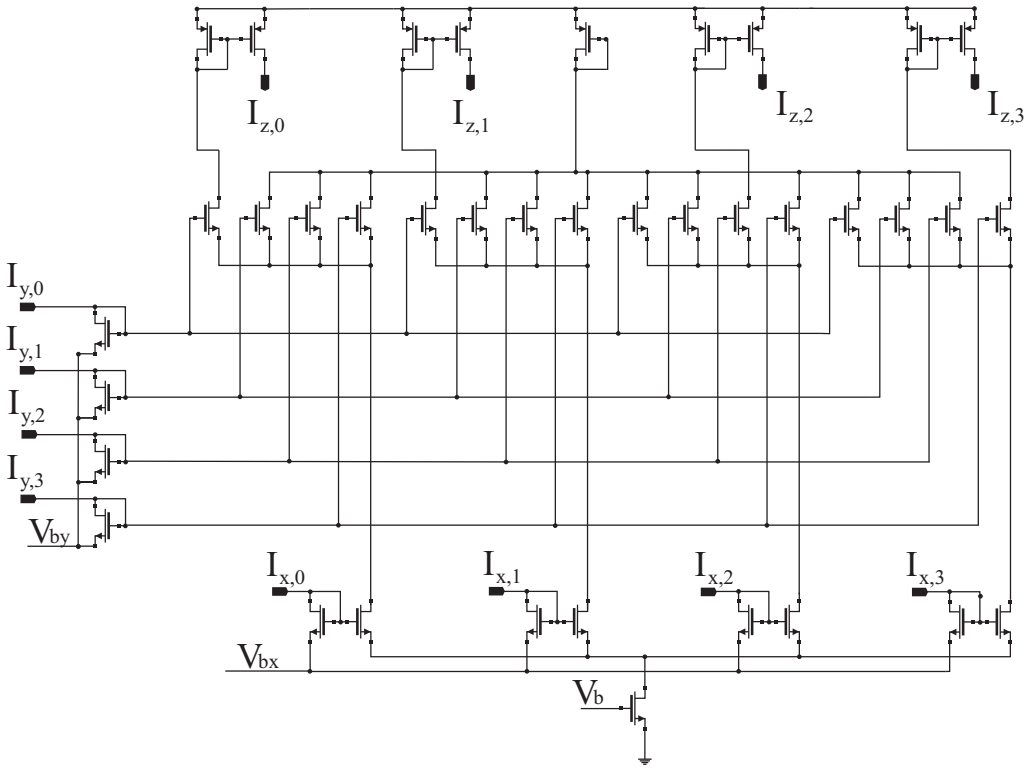


Figure 2.7: 4-ary *equal gate* circuit implementation

those of a hard decoding algorithm, represented by the square curve. The soft decoding algorithm shows a code gain of 4dB with respect to the uncoded channel (circle curve) at BER$= 10^{-3}$, which reduces to 2dB for the hard one. Thus the soft decoding algorithm, which is suitable to be implemented in the analog domain, offers a good advantage in terms of performance with respect to the digital or hard decision one, which can justify the slighter gates count increase required for the analog decoding circuitry implementation.

2.4 Convolutional Punctured Codes

As all the reliability issues described in Sec.2.1.4 become more serious for multilevel memories due to the reduced spacing between adjacent threshold voltage levels, codes with a higher correcting capability as the q-ary Hamming presented in Sec.2.3 may be required to ensure memory reliability.

Convolutional codes described in Appendix Sec.A.2.3 show large free distance, which translates into good error correcting capability, and, due to their trellis structure, can be

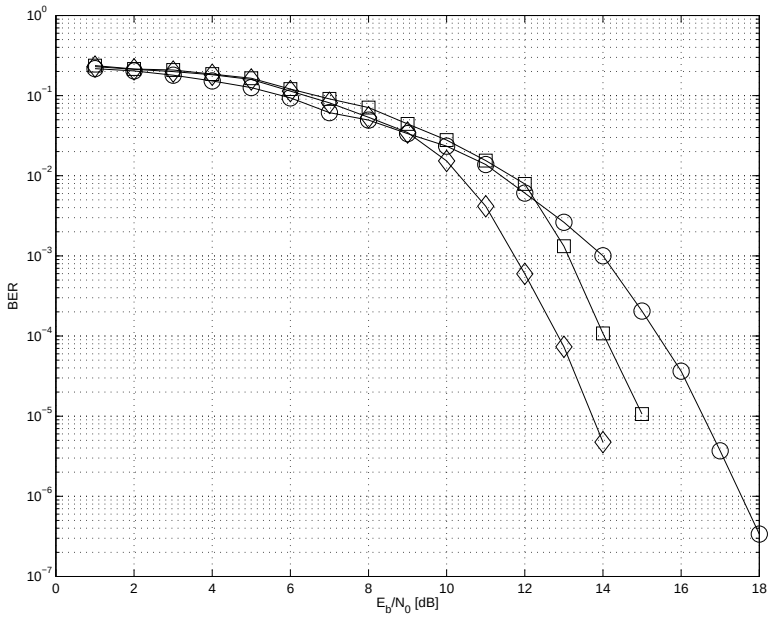


Figure 2.8: BER curves for Hamming $(36,32)$ code over $GF(4)$

naturally decoded in the analog domain, with advantages with respect to digital decoder implementation both in terms of area occupation and power consumption [8, 9, 43].

As one key requirement for multilevel memories ECC is a limited area overhead due both to parity bits and encoder/decoder circuitry, high-rate codes have in any case to be preferred. A way to obtain a high-rate convolutional code starting from a low-rate one is by "puncturing", as already described in Sec.1.1.1.

In general, a punctured rate b/n code can be constructed from a low-rate $1/n$ code, which is described by n generator polynomials G_i, $i+1,2,\cdots,n$. The complexity of decoding such a code is reduced to that of decoding the $1/n$ code. If we further impose a condition on the n generator polynomials so that only two of them at most differ, then the punctured code may also be regarded as having been generated from a rate 1/2 code and this can further reduce the complexity of decoding to that of decoding the corresponding rate 1/2 code [44–46].

2.4.1 Analog Decoder Implementation

The performance of high-rate punctured convolutional codes depend on the original low-rate code and on the perforation patter. In order to evaluate if they are suitable to be used

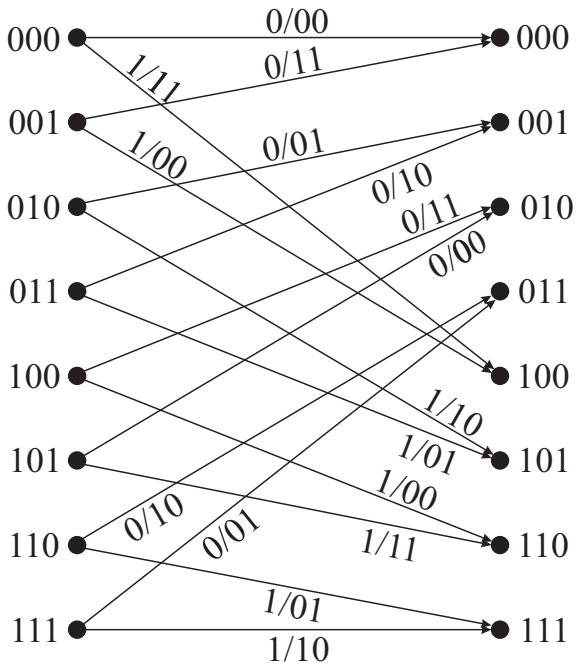


Figure 2.9: Convolutional code trellis section

as ECC for multilevel memories, we consider the rate 15/16 8-state tail-baiting trellis code whose generator polynomials are given by:

$$G(D) = \left[D^3 + D + 1, D^3 + D^2 + D + 1\right] \tag{2.5}$$

with punturation matrix [47]:

$$P = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \end{pmatrix} \tag{2.6}$$

The corresponding trellis without puncturing is shown in Fig.2.9.

Thus k user bits generate $2k$ coded bits that are then punctured so as to obtain a n-bit codeword, where $n = 16/15\,k$. If we use the convolutional code as ECC for multilevel flash memory cells with $q = 4$ levels, that is with 2-bit/cell, the n coded bits are grouped 2 by 2 and saved into $n/2$ memory cells.

If we analyze the structure of the perforation matrix P, we can see as the bits corresponding to the two first columns are written in the same memory cell. This is equivalent to say that the threshold voltage level programmed in the cell corresponds to the branch metric of the trellis section obtained by merging two consecutive trellis sections, as shown in Fig.2.10. Otherwise, if a column contains no zero, the information saved in the corresponding memory cell is relative to the branch metric of a single trellis section.

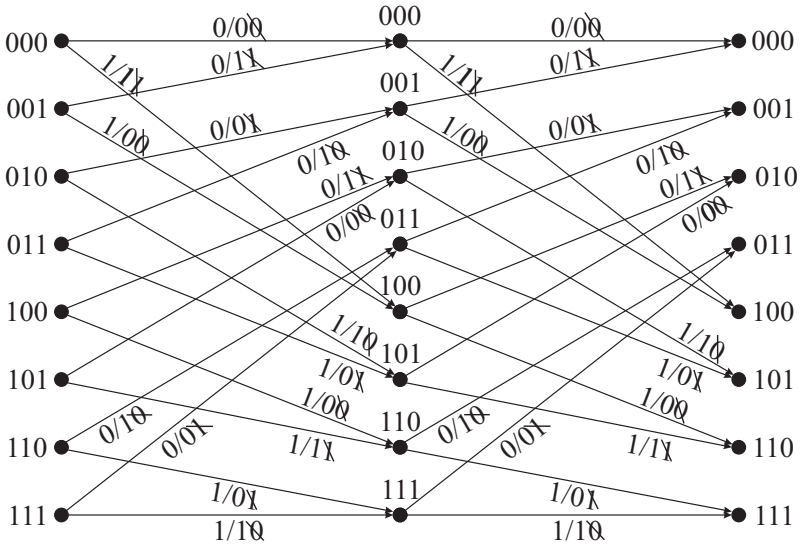


Figure 2.10: Merge of two consecutive trellis sections according to the B_a scheme

The perforation matrix can thus be written as:

$$P' = \left(\begin{array}{cc|cc|cc|cc|c|cc|cc|cc|} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \end{array} \right) \tag{2.7}$$

where column 9 and 10 have been swept. The branch metric of each section is given by the threshold voltage level of the corresponding memory cell.

Convolutional codes can be efficiently decoded by means of the BCJR algorithm described in Sec.A.3.5. The analog implementation of this decoding algorithm is based on Sum-Product modules, which implement the (A.55) equation. A block diagram of the analog decoder implementation is shown in Fig.2.11. In particular, the B and C chains that implement the forward and backward recursion of the BCJR algorithm will consist of 5 different B and C cells types, corresponding to the 4 possible merging schemes of two consecutive trellis sections due to puncturization plus the case without puncturing. The same applies to blocks D, which compute the probability of each user alphabet symbols. In particular, cell B_a corresponds to the perforation matrix section

$$\left| \begin{array}{cc} 0 & 0 \\ 1 & 1 \end{array} \right|$$

which will be indicated as $|\times 1 \times 1|$. In the same way, cell B_b relates to section $|\times 11 \times|$, B_c to $|1 \times 1 \times|$, B_d to $|1 \times \times 1|$ and B to $|11|$. The structure of C cells are specular to those of the corresponding B cells while D cells calculate the user bits probabilities of each

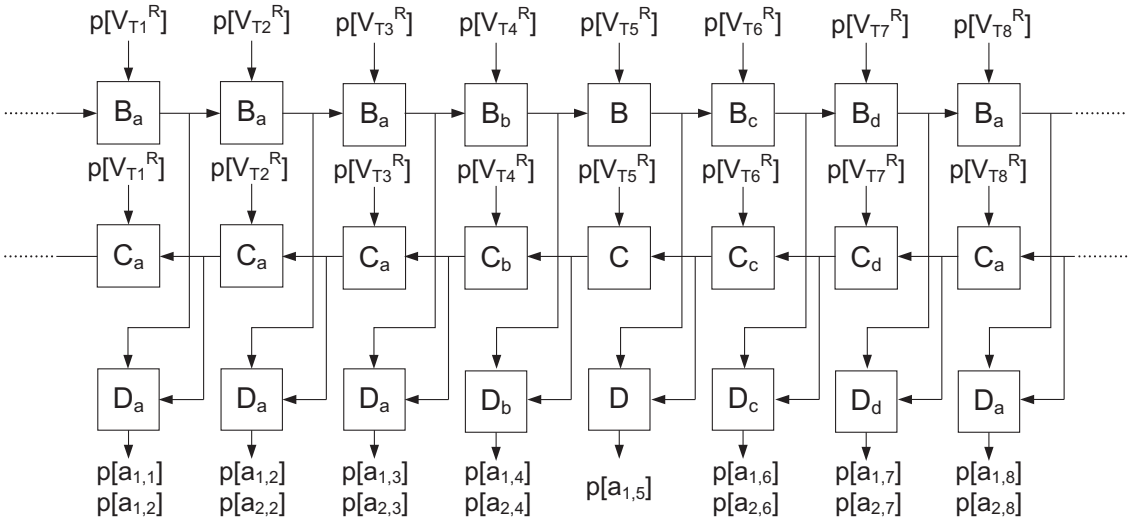


Figure 2.11: Convolutional code SISO block diagram

trellis section combining the information given by the forward and backward recursions. As an example, the trellis of cell B_a and D_a are reported in Fig..2.12.

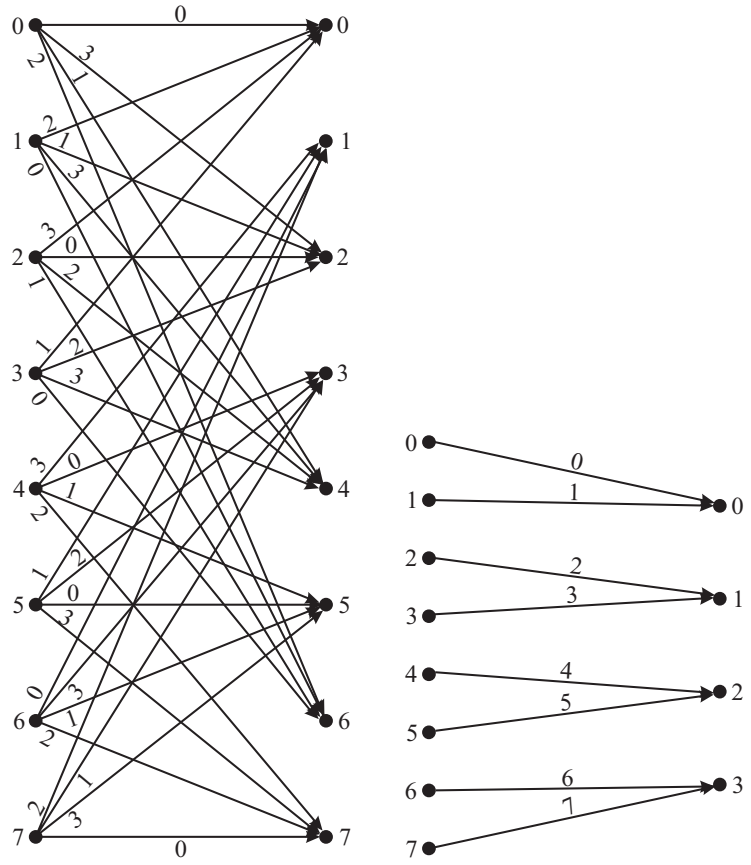
2.4.2 Performance Analysis

The performance of the decoding algorithm have been estimated through a C++ behavioral model of the decoder. To consider a realistic scenario, the analog decoder has been design to work on information data fields $M = 256$ bits wide, as in the 2 bit/cell NOR memory described in [48], that uses as ECC a BCH $(274, 256, 2)$. The code is terminated, that is the 256 user bits plus 3 termination bits are coded by means of the rate 1/2 convolutional code described in Sec.2.4 and then punctured with the pattern given by the perforation matrix

$$P_{Tot} = \begin{pmatrix} P' & \begin{matrix}1\\1\end{matrix} & P'\,P'\,P'\,P'\,P' & \begin{matrix}1\\1\end{matrix} & P'\,P'\,P'\,P'\,P' & \begin{matrix}1\\1\end{matrix} & P'\,P'\,P'\,P'\,P' & \begin{matrix}1\\1\end{matrix} & P' \end{pmatrix} \quad (2.8)$$

where P' is defined by equation (2.7). Thus the resulting code rate is $R = 256/280$. The punctured convolutional code performance are reported in Fig.2.13, together with that of the Hamming code $(36, 32)$ over $GF(4)$ presented in Sec.2.3. The trellis code performance with and without puncturing are compared in Fig.2.14.

The punctured code shows a loss of 5dB with respect to the non-punctured version at



(a) B_a module (b) D_a module

Figure 2.12: Trellis sections according to the merging scheme $|\times 1\times 1|$

BER=10^{-3}, which makes high-rate punctured convolutional codes poor ECC for multi-level memories.

2.5 Trellis Coded Modulation for Multilevel Flash Memories

Neither linear block codes nor convolutional punctured codes seem the right choice when dialing with memory cells with a storage capability equal or greater than 3 bits/cell. The former, because their complexity is deemed to increase when a higher error correction capability is required, as shown by the state-of-the-art BCH decoder demonstrated in [36], where both the BCH decoder area and latency time increase with the error correction capability; the latter due to their trade off between code performance and rate.

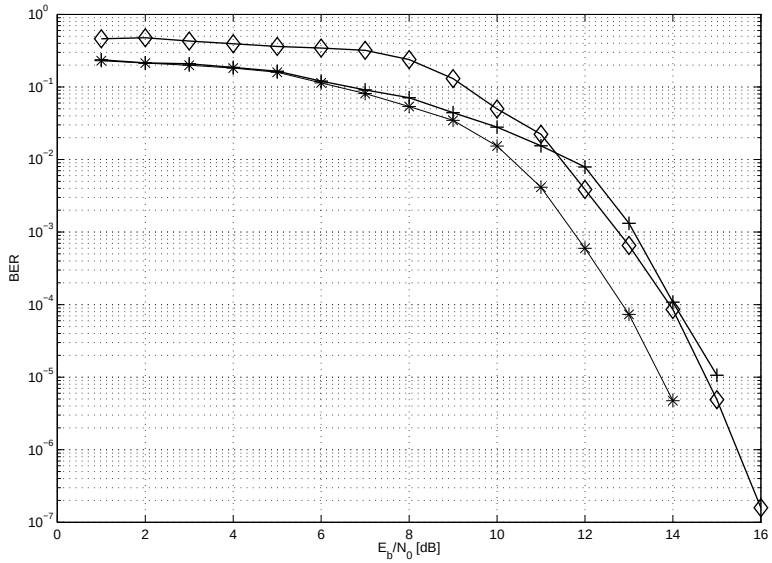


Figure 2.13: BER of rate 15/16 punctured (diamonds) convolutional code, soft-decoded (stars) and hard-decoded (crosses) Hamming$(36,32)$ over $GF(4)$

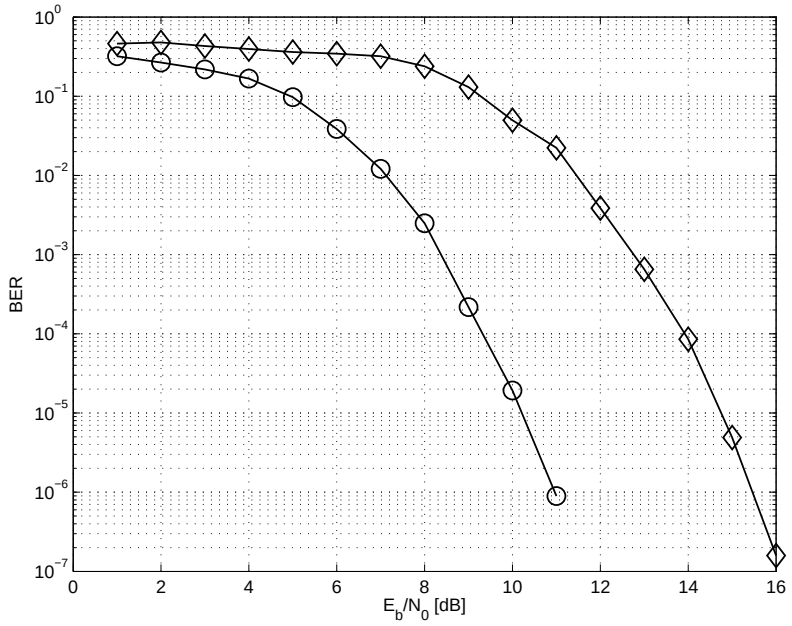
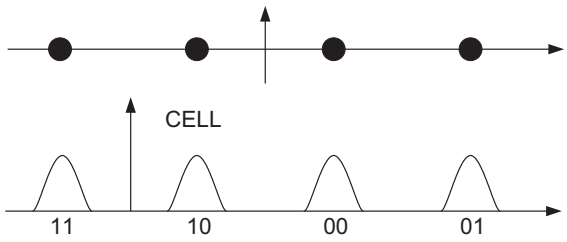
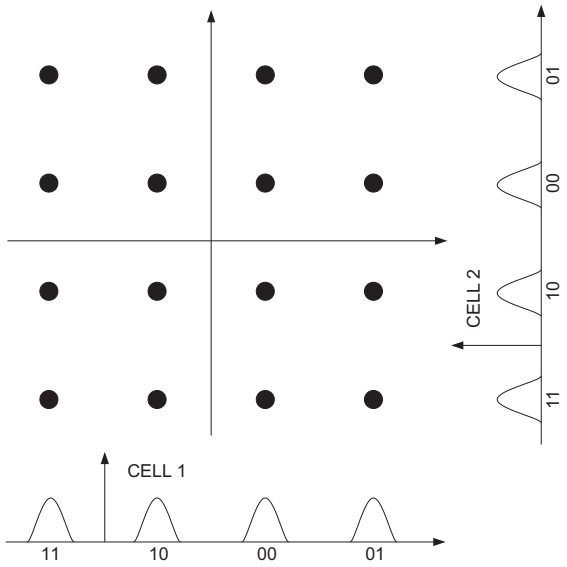


Figure 2.14: BER of rate 15/16 punctured (diamonds) and non-punctured (circles) convolutional code



(a) One 4-levels memory cell as 4 AM modulation



(b) Two 4-levels memory cells as 16 QAM modulation

Figure 2.15: Analogy between memory cell threshold voltage distributions and signal constellations

2.5.1 Multilevel Flash Memories as Signal Constellations

Trellis Coded Modulation (TCM) is a well-known technique for improving band-limited communication systems combining trellis codes (i.e. convolutional codes) and modulation, as described in Appendix Sec.A.2.5.

The concept of signal modulation can be easily extended to a flash memory by considering the analogy between signal constellations and threshold voltage levels (or charge distributions) positions within one or more memory cells as shown in Fig.2.15. In fact, in the case of memory cells, the symbol space is a one- or multi-dimensional discrete grid of (approximately) equally spaced voltage levels.

The noise deteriorating the transmitted signal is due to Gaussian white noise and intersymbol interference in the case of transmission channels, while it can be attributed to process variations and all the disturb effects described in Sec.2.1.4 in the case of flash memory cells. Nevertheless, in both cases the noise effect can be described by means of a triangular or Gaussian distribution.

As a result, a multilevel flash memory cell can be modeled as an Amplitude Shift Keying (ASK) modulation channel plus a white additive Gaussian noise (AWGN), thus allowing the use of TCM to either increase memory reliability or to enable higher effective storage capacity [49].

2.5.2 TCM for Multilevel Flash Memories

The effectiveness of TCM-based solutions for multilevel flash memories in terms of error correcting performance, coding redundancy, silicon cost and operation latency, has been successfully demonstrated in [50].

As TCM requires soft decoding algorithms, analog decoders could offer advantages with respect to digital implementations, both in terms of area occupation and power consumption, as already proved for Trellis and Turbo codes analog decoders [8, 9, 13, 14].

In order to study the feasibility and complexity of the analog approach for a TCM decoder, the case where the effective capacity of a given ML flash memory is to be increased from m to $m+1$ information bits/cell is considered. To maintain the same level of reliability despite the decrease of spacing between adjacent voltage levels, the use a relatively low-rate TCM code is here proposed, based on the following strategy: the number of bits stored in a cell is increased from m to $m+2$, with the additional bit (w.r.t. $m+1$) used as a parity bit. Thus a rate $(m+1)/(m+2)$ TCM code is implemented as, for realistic values of m, is powerful and relatively simple to decode with respect to higher rate block codes.

More specifically, as flash memory cells with up to 16 levels are by now the present technology [35], we consider a case where $m = 2$, thus aiming at an effective memory capacity of 3 information bits/cell, achieved through the use of 4-bit, that is 16-level, multilevel cells protected by a rate 3/4 TCM code. To model the behavior of the multilevel memory cell, the following simplifying (but not unrealistic) assumptions are made:

(i) the q threshold voltage values that can be programmed in a memory cell are equally spaced and bounded between fixed minimum and maximum voltage V_{TMIN} and V_{TMAX};

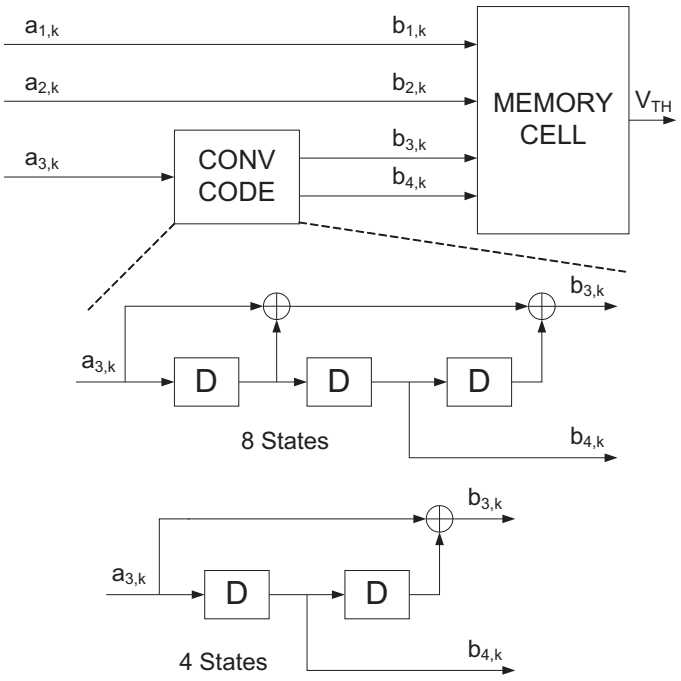


Figure 2.16: TCM encoder block diagram

(ii) the readout threshold voltage has a Gaussian distribution with standard deviation σ_n centered around the nominal programmed value.

To improve memory reliability by means of a TCM code, there are different possibilities, depending on the constituent trellis code [50,51]. The use of a rate 1/2 convolutional code allows to keep the decoder complexity low. Thus, depending on the modulation scheme chosen, three main different scenarios can be considered:

(a) one-dimensional 16-ASK modulation;

(b) bi-dimensional 12-ASK modulation;

(c) bi-dimensional 16-ASK modulation.

Case (a) and (c) are designed for a multilevel memory cell with $q = 16$ levels and distance Δ between two adjacent threshold voltages, while case (b) refers to a multilevel memory cell with $q = 12$ levels and thus with distance between two adjacent threshold voltages $\Delta' = 4/3\Delta$. It can be proved [51–53] that the minimum distance between two codewords that differ only in the uncoded bits decreases from $4 \cdot \Delta$ in case (a) to $8/3 \cdot \sqrt{2} \cdot \Delta$ in case (b) down to $2 \cdot \sqrt{2} \cdot \Delta$ in case (c).

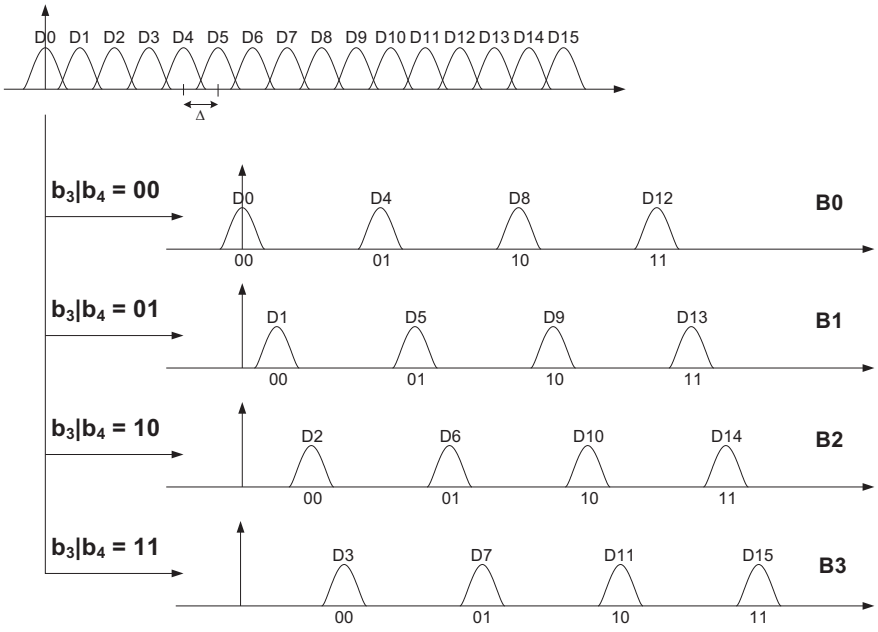


Figure 2.17: Cell mapping

Solution (b) can be discarded since it shows a lower codeword distance with respect to (a) with the same storage density of 3 information bits/cell. As the increased storage capacity of (c) comes at the price of a reduced error correction capability and also of a more complex decoder architecture with respect to (a), the more promising solution seems the latter one, which refers to a rate 3/4 TCM with one-dimensional 16-ASK modulation designed for a 4 bits/cell multilevel memory.

The encoding process can be better understood with the help of Fig.2.16, where a block diagram of the full TCM encoder is shown.

An M-bit information data field is divided into n-bit wide sub-fields ($n = 3$ in Fig.2.16); k bits of the sub-field ($k = 1$ in Fig.2.16) are fed to a convolutional encoder, that generates a $k+1$-bit codeword. The $n-k$ uncoded bits and the $k+1$-bit codeword are fed to a modulator that maps them on a threshold voltage level that is then programmed in a multilevel cell.

In Fig.2.16, bits $a_{1,k}, a_{2,k}, a_{3,k}$ indicate the $n = 3$-bit uncoded information bits, while bits $b_{1,k}, b_{2,k}, b_{3,k}, b_{4,k}$ represent the $(n+1) = 4$-bit codeword. The codewords mapping into memory cell threshold voltage levels, together with the multilevel cells threshold voltage distributions, is shown in Fig.2.17.

The 16 threshold voltages are partitioned into 4 subsets S_0, S_1, S_2, S_3, each consisting

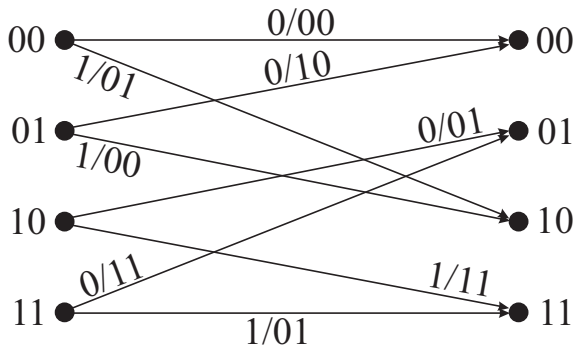


Figure 2.18: 4-state TCM trellis state diagram

of 4 threshold voltages with minimum intra-set distance $\delta = 4 \cdot \Delta$. The two convolutional encoders output bits $b_{3,k}$ and $b_{4,k}$ select one subset out of 4, while the two uncoded bits $b_{1,k}$ and $b_{2,k}$ select one of the 4 voltages within the chosen subset.

In order to analyze in depth the design trade-offs peculiar to the analog approach, the performance and limitations of two different fully analog TCM decoders, a 4-state and an 8-state one, are compared. In both cases, the constituent convolutional code has been chosen in accordance with [51] so as to maximize the asymptotic coding gain.

The generator polynomials for the 4-state trellis code are given by:

$$G(D) = \left[D^2 + 1, D\right] \tag{2.9}$$

Its corresponding trellis state diagram is shown in Fig.2.18. It is worth to notice how each branch of the trellis diagram is actually constituted by 4 parallel branches which differ for the uncoded bits $b_{1,k}$ and $b_{2,k}$.

The generator polynomials for the 8-state code, whose trellis state diagram is shown in Fig.2.19, are given by:

$$G(D) = \left[D^3 + D + 1, D^2\right] \tag{2.10}$$

This choice leads to an asymptotic coding gain of 3.5dB for the 4-state code, which becomes 3.9dB for the 8-state one.

2.5.3 Analog Decoding Algorithm

If implemented in the digital domain, the core blocks of a TCM decoder are a demodulator, which computes the branch metrics for the Viterbi decoder based on the log-likelyhood of the threshold voltage levels read from the memory array, and a soft-Viterbi decoder matched to the convolutional code.

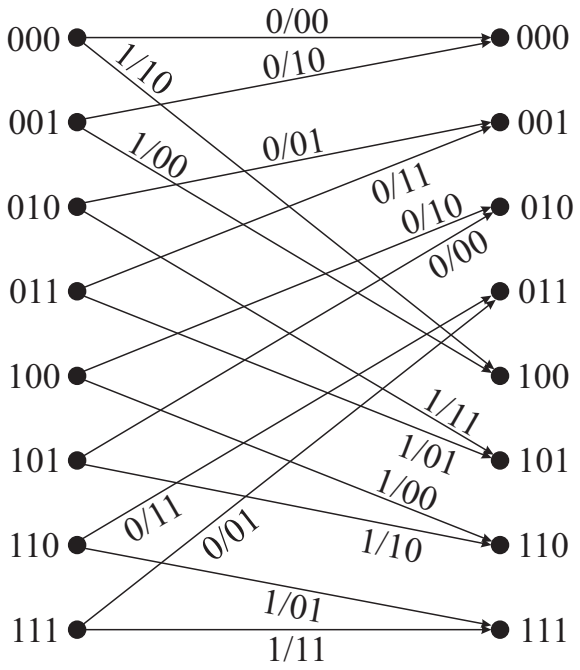


Figure 2.19: 8-state TCM trellis state diagram

In the analog approach, the Viterbi decoder is replaced by a soft-input soft-output (SISO) module implementing the BCJR decoding algorithm with a current mode circuit [6]. In addition, also the demodulator is realized by means of a fully analog circuit.

A simplified block diagram of the decoder core is reported in Fig.2.20. The BCJR decoding algorithm implemented by the decoder core computes, for each cell (i.e., for each trellis section), the most probable subset $S_{i,k}$ of threshold voltage levels, while the demodulator finds the most probable level within each subset $S_{i,k}$.

The SISO is designed following the Sum-Product approach described in [6] and already successfully used in several analog decoder prototypes, such as [14]. In particular, B and C chains implement the forward and backward recursion of the BCJR algorithm; blocks D compute the probability of the trellis coded input bit $a_{3,k}$; blocks E compute the probability of each subset $S_{i,k}$; blocks F1 and F2 realize the demodulation computing the probability of bits $a_{1,k}$ and $a_{2,k}$.

The decoder inputs are the conditional probabilities $p(V_{Tk}^R|S_{i,k})$, that is the probabilities of the threshold voltage V_{Tk}^R read from memory cell k, given that the programmed level belongs to the subset $S_{i,k}$, with $i = 0,1,2,3$, while the demodulator inputs are the conditional probabilities $p(a_{1,k}|S_{i,k})$ and $p(a_{2,k}|S_{i,k})$.

$p(V_{Tk}^R|S_{i,k})$ correspond to the branch metrics required by the MAP decoder and are

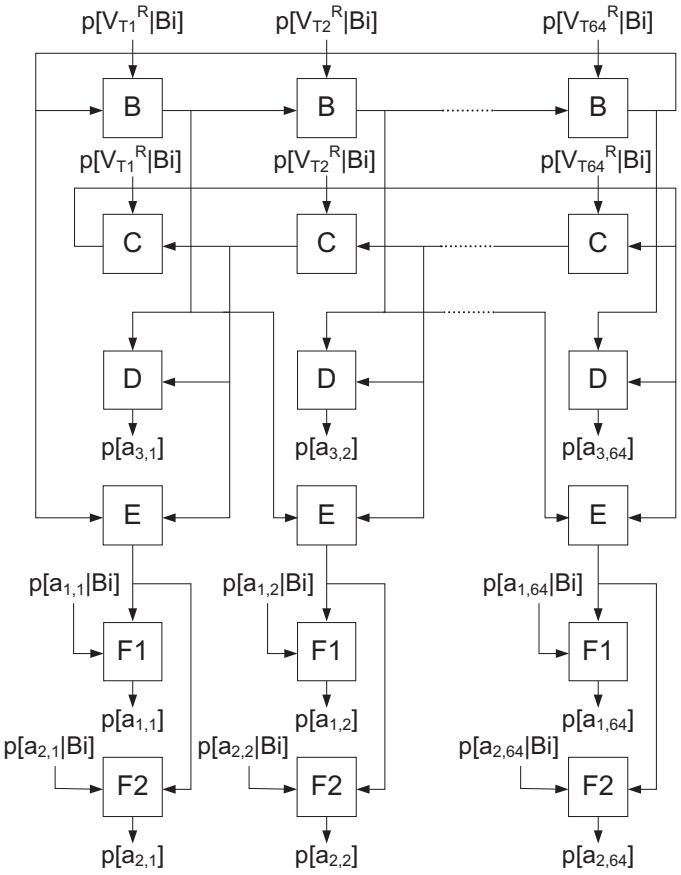


Figure 2.20: Block diagram of the analog decoder core

calculated from the threshold voltage conditional probabilities $p(V_{Tk}^R|V_{Tk}(j))$. These latter ones are the probabilities of V_{Tk}^R given the $V_{Tk}(j)$ programmed voltage level, with $j = 0,1,2,\cdots,15$, and are computed by the voltage to probability module described in Sec.3.2.

Recalling the mapping scheme already shown in Fig.2.17, the conditional probabilities $p(V_{Tk}^R|S_{i,k})$ are given by:

$$
\begin{aligned}
p(V_{Tk}^R|S_{0,k}) &= p(V_{Tk}^R|V_{Tk}(0)) + p(V_{Tk}^R|V_{Tk}(4)) + p(V_{Tk}^R|V_{Tk}(8)) + p(V_{Tk}^R|V_{Tk}(12)) \\
p(V_{Tk}^R|S_{1,k}) &= p(V_{Tk}^R|V_{Tk}(1)) + p(V_{Tk}^R|V_{Tk}(5)) + p(V_{Tk}^R|V_{Tk}(9)) + p(V_{Tk}^R|V_{Tk}(13)) \\
p(V_{Tk}^R|S_{2,k}) &= p(V_{Tk}^R|V_{Tk}(2)) + p(V_{Tk}^R|V_{Tk}(6)) + p(V_{Tk}^R|V_{Tk}(10)) + p(V_{Tk}^R|V_{Tk}(14)) \\
p(V_{Tk}^R|S_{3,k}) &= p(V_{Tk}^R|V_{Tk}(3)) + p(V_{Tk}^R|V_{Tk}(7)) + p(V_{Tk}^R|V_{Tk}(11)) + p(V_{Tk}^R|V_{Tk}(15))
\end{aligned}
$$

where the programmed voltage level $V_{Tk}(j)$ corresponds to the distribution Dj of the k-th memory cell.

In the same way, conditional probabilities $p(a_{1,k}|S_{i,k})$ and $p(a_{2,k}|S_{i,k})$ can be calculated by:

$$
\begin{aligned}
p(a_{1,k}=0|S_{0,k}) &= p(V_{Tk}^R|V_{Tk}(0)) + p(V_{Tk}^R|V_{Tk}(4)) \\
p(a_{1,k}=0|S_{1,k}) &= p(V_{Tk}^R|V_{Tk}(1)) + p(V_{Tk}^R|V_{Tk}(5)) \\
p(a_{1,k}=0|S_{2,k}) &= p(V_{Tk}^R|V_{Tk}(2)) + p(V_{Tk}^R|V_{Tk}(6)) \\
p(a_{1,k}=0|S_{3,k}) &= p(V_{Tk}^R|V_{Tk}(3)) + p(V_{Tk}^R|V_{Tk}(7)) \\
p(a_{1,k}=1|S_{0,k}) &= p(V_{Tk}^R|V_{Tk}(8)) + p(V_{Tk}^R|V_{Tk}(12)) \\
p(a_{1,k}=1|S_{1,k}) &= p(V_{Tk}^R|V_{Tk}(9)) + p(V_{Tk}^R|V_{Tk}(13)) \\
p(a_{1,k}=1|S_{2,k}) &= p(V_{Tk}^R|V_{Tk}(10)) + p(V_{Tk}^R|V_{Tk}(14)) \\
p(a_{1,k}=1|S_{3,k}) &= p(V_{Tk}^R|V_{Tk}(11)) + p(V_{Tk}^R|V_{Tk}(15))
\end{aligned}
$$

and:

$$
\begin{aligned}
p(a_{2,k}=0|S_{0,k}) &= p(V_{Tk}^R|V_{Tk}(0)) + p(V_{Tk}^R|V_{Tk}(8)) \\
p(a_{2,k}=0|S_{1,k}) &= p(V_{Tk}^R|V_{Tk}(1)) + p(V_{Tk}^R|V_{Tk}(9)) \\
p(a_{2,k}=0|S_{2,k}) &= p(V_{Tk}^R|V_{Tk}(2)) + p(V_{Tk}^R|V_{Tk}(10)) \\
p(a_{2,k}=0|S_{3,k}) &= p(V_{Tk}^R|V_{Tk}(3)) + p(V_{Tk}^R|V_{Tk}(11)) \\
p(a_{2,k}=1|S_{0,k}) &= p(V_{Tk}^R|V_{Tk}(4)) + p(V_{Tk}^R|V_{Tk}(12)) \\
p(a_{2,k}=1|S_{1,k}) &= p(V_{Tk}^R|V_{Tk}(5)) + p(V_{Tk}^R|V_{Tk}(13)) \\
p(a_{2,k}=1|S_{2,k}) &= p(V_{Tk}^R|V_{Tk}(6)) + p(V_{Tk}^R|V_{Tk}(14)) \\
p(a_{2,k}=1|S_{3,k}) &= p(V_{Tk}^R|V_{Tk}(7)) + p(V_{Tk}^R|V_{Tk}(15))
\end{aligned}
$$

B and C modules The B and C modules compute the forward and backward recursion of the BCJR algorithm, taking as inputs the trellis branch metrics.

The forward recursion can be described by:

$$
\begin{aligned}
B_k(0) &= B_{k-1}(0) \cdot p(V_{Tk}^R|S_{0,k}) + B_{k-1}(1) \cdot p(V_{Tk}^R|S_{2,k}) \\
B_k(1) &= B_{k-1}(2) \cdot p(V_{Tk}^R|S_{1,k}) + B_{k-1}(3) \cdot p(V_{Tk}^R|S_{3,k}) \\
B_k(2) &= B_{k-1}(0) \cdot p(V_{Tk}^R|S_{2,k}) + B_{k-1}(1) \cdot p(V_{Tk}^R|S_{0,k}) \\
B_k(3) &= B_{k-1}(2) \cdot p(V_{Tk}^R|S_{3,k}) + B_{k-1}(3) \cdot p(V_{Tk}^R|S_{1,k})
\end{aligned}
$$

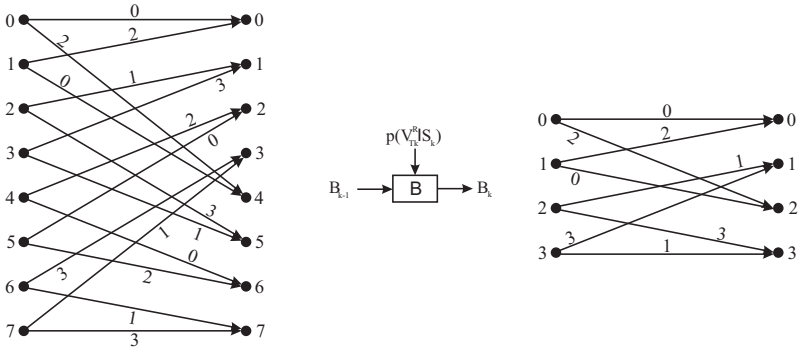


Figure 2.21: B module

for the 4-state code, and by:

$$
\begin{aligned}
B_k(0) &= B_{k-1}(0)\cdot p(V_{Tk}^R|S_{0,k}) + B_{k-1}(1)\cdot p(V_{Tk}^R|S_{2,k})\\
B_k(1) &= B_{k-1}(2)\cdot p(V_{Tk}^R|S_{1,k}) + B_{k-1}(3)\cdot p(V_{Tk}^R|S_{3,k})\\
B_k(2) &= B_{k-1}(4)\cdot p(V_{Tk}^R|S_{2,k}) + B_{k-1}(5)\cdot p(V_{Tk}^R|S_{0,k})\\
B_k(3) &= B_{k-1}(6)\cdot p(V_{Tk}^R|S_{3,k}) + B_{k-1}(7)\cdot p(V_{Tk}^R|S_{1,k})\\
B_k(4) &= B_{k-1}(0)\cdot p(V_{Tk}^R|S_{2,k}) + B_{k-1}(1)\cdot p(V_{Tk}^R|S_{0,k})\\
B_k(5) &= B_{k-1}(2)\cdot p(V_{Tk}^R|S_{3,k}) + B_{k-1}(3)\cdot p(V_{Tk}^R|S_{1,k})\\
B_k(6) &= B_{k-1}(4)\cdot p(V_{Tk}^R|S_{0,k}) + B_{k-1}(5)\cdot p(V_{Tk}^R|S_{2,k})\\
B_k(7) &= B_{k-1}(6)\cdot p(V_{Tk}^R|S_{1,k}) + B_{k-1}(7)\cdot p(V_{Tk}^R|S_{3,k})
\end{aligned}
$$

for the 8-state one. The relative trellises are shown in Fig.2.21

The backward recursion is computed according to the trellises of Fig.2.22, that are specular to the B cells ones. Thus, for the 4-state code we have:

$$
\begin{aligned}
C_{k-1}(0) &= C_k(0)\cdot p(V_{Tk}^R|S_{0,k}) + C_k(2)\cdot p(V_{Tk}^R|S_{2,k})\\
C_{k-1}(1) &= C_k(0)\cdot p(V_{Tk}^R|S_{2,k}) + C_k(2)\cdot p(V_{Tk}^R|S_{0,k})\\
C_{k-1}(2) &= C_k(1)\cdot p(V_{Tk}^R|S_{1,k}) + C_k(3)\cdot p(V_{Tk}^R|S_{3,k})\\
C_{k-1}(3) &= C_k(1)\cdot p(V_{Tk}^R|S_{3,k}) + C_k(3)\cdot p(V_{Tk}^R|S_{1,k})
\end{aligned}
$$

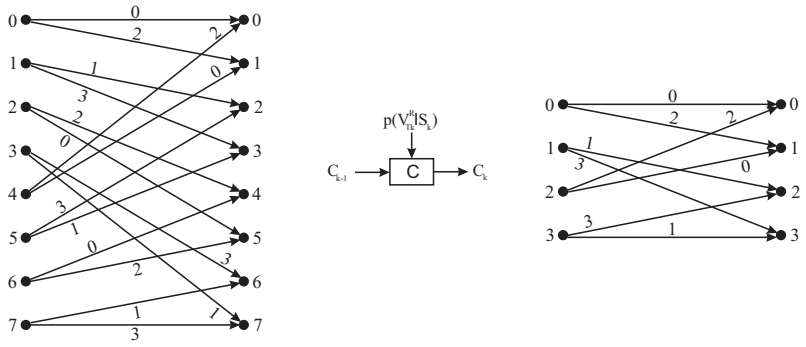


Figure 2.22: C module

which become:

$$
\begin{aligned}
C_{k-1}(0) &= C_k(0)\cdot p(V_{Tk}^R|S_{0,k}) + C_k(4)\cdot p(V_{Tk}^R|S_{2,k})\\
C_{k-1}(1) &= C_k(0)\cdot p(V_{Tk}^R|S_{2,k}) + C_k(4)\cdot p(V_{Tk}^R|S_{0,k})\\
C_{k-1}(2) &= C_k(1)\cdot p(V_{Tk}^R|S_{1,k}) + C_k(5)\cdot p(V_{Tk}^R|S_{3,k})\\
C_{k-1}(3) &= C_k(1)\cdot p(V_{Tk}^R|S_{3,k}) + C_k(5)\cdot p(V_{Tk}^R|S_{1,k})\\
C_{k-1}(4) &= C_k(2)\cdot p(V_{Tk}^R|S_{2,k}) + C_k(6)\cdot p(V_{Tk}^R|S_{0,k})\\
C_{k-1}(5) &= C_k(2)\cdot p(V_{Tk}^R|S_{0,k}) + C_k(6)\cdot p(V_{Tk}^R|S_{2,k})\\
C_{k-1}(6) &= C_k(3)\cdot p(V_{Tk}^R|S_{3,k}) + C_k(7)\cdot p(V_{Tk}^R|S_{1,k})\\
C_{k-1}(7) &= C_k(3)\cdot p(V_{Tk}^R|S_{1,k}) + C_k(7)\cdot p(V_{Tk}^R|S_{3,k})
\end{aligned}
$$

for the 8-state one.

D module The D modules, whose trellises are shown in Fig.2.23, compute the a posteriori probabilities for the convolutional encoder input bit $a_{3,k}$, implementing the equations:

$$
\begin{aligned}
p(a_{3,k}=0) &= B_k(0)\cdot C_k(0) + B_k(1)\cdot C_k(1)\\
p(a_{3,k}=1) &= B_k(2)\cdot C_k(2) + B_k(3)\cdot C_k(3)
\end{aligned}
$$

for the 4-state code, and:

$$
\begin{aligned}
p(a_{3,k}=0) &= B_k(0)\cdot C_k(0) + B_k(1)\cdot C_k(1) + B_k(2)\cdot C_k(2) + B_k(3)\cdot C_k(3)\\
p(a_{3,k}=1) &= B_k(4)\cdot C_k(4) + B_k(5)\cdot C_k(5) + B_k(6)\cdot C_k(6) + B_k(7)\cdot C_k(7)
\end{aligned}
$$

for the 8-state one.

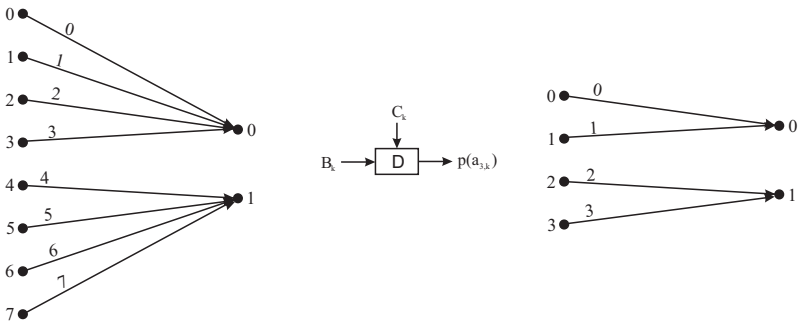


Figure 2.23: D module

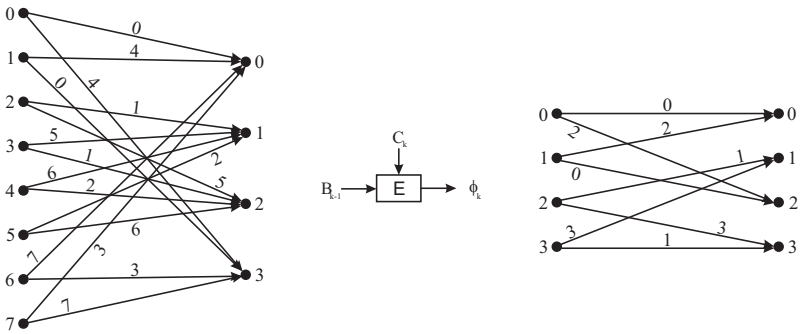


Figure 2.24: E module

E module The E modules recompute the branch metric after the forward-backward recursions, according to the trellises of Fig.2.24 and the following equations:

$$\begin{aligned}
\phi_k(0) &= B_{k-1}(0)\cdot C_k(0) + B_{k-1}(1)\cdot C_k(2)\\
\phi_k(1) &= B_{k-1}(2)\cdot C_k(1) + B_{k-1}(3)\cdot C_k(3)\\
\phi_k(2) &= B_{k-1}(0)\cdot C_k(2) + B_{k-1}(1)\cdot C_k(0)\\
\phi_k(3) &= B_{k-1}(2)\cdot C_k(3) + B_{k-1}(3)\cdot C_k(1)
\end{aligned}$$

for the 4-state case, which become:

$$\begin{aligned}
\phi_k(0) &= B_{k-1}(0)\cdot C_k(0) + B_{k-1}(1)\cdot C_k(4) + B_{k-1}(4)\cdot C_k(6) + B_{k-1}(5)\cdot C_k(2)\\
\phi_k(1) &= B_{k-1}(2)\cdot C_k(1) + B_{k-1}(3)\cdot C_k(5) + B_{k-1}(6)\cdot C_k(7) + B_{k-1}(7)\cdot C_k(3)\\
\phi_k(2) &= B_{k-1}(0)\cdot C_k(4) + B_{k-1}(1)\cdot C_k(0) + B_{k-1}(4)\cdot C_k(2) + B_{k-1}(5)\cdot C_k(6)\\
\phi_k(3) &= B_{k-1}(2)\cdot C_k(5) + B_{k-1}(3)\cdot C_k(1) + B_{k-1}(6)\cdot C_k(3) + B_{k-1}(7)\cdot C_k(7)
\end{aligned}$$

for the 8-state decoder.

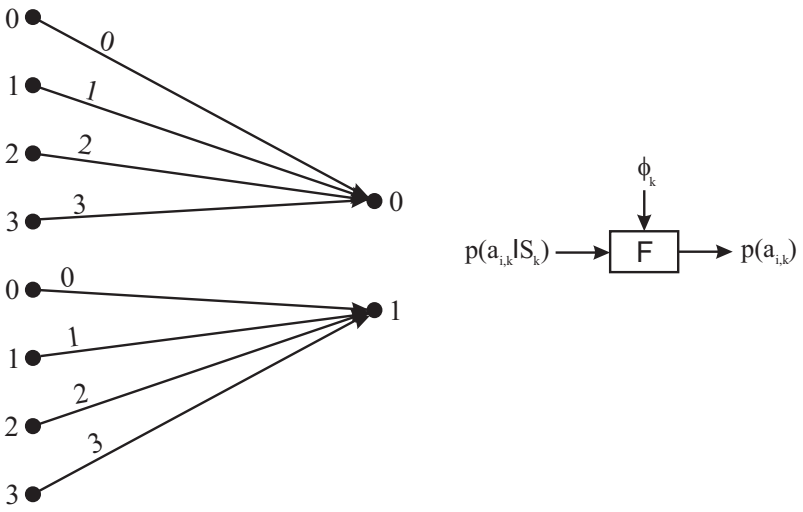


Figure 2.25: F1 and F2 module

F1 and F2 modules The F1 and F2 modules, which exhibit the same trellis depicted in Fig.2.25, compute the a-posteriori probabilities for the uncoded bits $a_{1,k}$ and $a_{2,k}$.

The equations for both modules are the same for both the 4-state and the 8-state decoder. In particular, for F1 modules we have:

$$\begin{aligned} p(a_{1,k}=0) =& p(a_{1,k}=0|S_{0,k})\cdot\phi_k(0)+p(a_{1,k}=0|S_{1,k})\cdot\phi_k(1)+ \\ & p(a_{1,k}=0|S_{2,k})\cdot\phi_k(2)+p(a_{1,k}=0|S_{3,k})\cdot\phi_k(3) \\ p(a_{1,k}=1) =& p(a_{1,k}=1|S_{0,k})\cdot\phi_k(0)+p(a_{1,k}=1|S_{1,k})\cdot\phi_k(1)+ \\ & p(a_{1,k}=1|S_{2,k})\cdot\phi_k(2)+p(a_{1,k}=1|S_{3,k})\cdot\phi_k(3) \end{aligned}$$

while the equations for F2 modules are given by:

$$\begin{aligned} p(a_{2,k}=0) =& p(a_{2,k}=0|S_{0,k})\cdot\phi_k(0)+p(a_{2,k}=0|S_{1,k})\cdot\phi_k(1)+ \\ & p(a_{2,k}=0|S_{2,k})\cdot\phi_k(2)+p(a_{2,k}=0|S_{3,k})\cdot\phi_k(3) \\ p(a_{2,k}=1) =& p(a_{2,k}=1|S_{0,k})\cdot\phi_k(0)+p(a_{2,k}=1|S_{1,k})\cdot\phi_k(1)+ \\ & p(a_{2,k}=1|S_{2,k})\cdot\phi_k(2)+p(a_{2,k}=1|S_{3,k})\cdot\phi_k(3) \end{aligned}$$

2.5.4 Performance Analysis

The error correction capabilities of the analog algorithm described above have been estimated through a C++ behavioral model of the decoder. The analog decoder taken into account is the 8-state one, design to work on information data fields $M = 192$ bits wide.

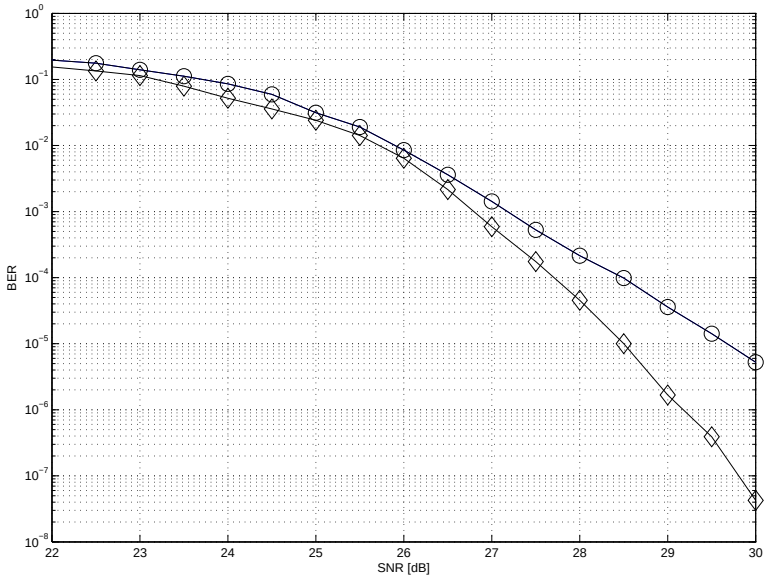


Figure 2.26: BER vs SNR with analog TCM and soft Viterbi decoder

After encoding, the 256 bit codewords are stored in 64 memory cells with $q = 16$ levels. To avoid long connections in the transistor-level implementation, the code is terminated. Thus the 192 bits data field is divided into 189 information bits and 3 termination bits. The C++ analog decoder model assumes the availability at its inputs of the trellis code branch metrics computed by the voltage to probability circuit. The sum-product operations of the MAP decoder are ideal, with probabilities represented by double-precision numbers. No source of distortion, offset or noise due to transistor non-idealities is taken into account.

To compare the analog approach performance with that of a digital implementation based on the Soft-Output Viterbi Algorithm (SOVA) [54], a C++ behavioral model of the digital decoder has also been developed. For a fair comparison with the ideal model of the analog TCM, quantization effects have not been taken into account, so that the input branch metrics and the internal path metrics are represented by double-precision numbers. A survivor depth $D = 64$ has been chosen and the code is terminated, as for the analog one. The resulting BER vs SNR performance obtained by simulation is reported in Fig.2.26 (circles) together with the analog TCM curve (diamonds), where the SNR is calculated as the ratio of the signal power to the threshold voltage distribution variance

σ_n:

$$SNR = \sum_{j=1}^{q} V_T^2(j)/\sigma_n \tag{2.11}$$

The analog TCM outperforms its SOVA counterpart by roughly 1dB at BER=10^{-5}. This result is not surprising as the SOVA is a suboptimum decoding algorithm since it minimizes the whole codeword error probability instead of the single symbol error probability, as done by the MAP algorithm [55].

Based on the analog decoder prototypes reported in the literature [14], the effect of transistor non-idealities, such as mismatch, noise and distortion, will be of the order of a few tenths of a dB with respect to the performance predicted by the ideal model. Thus, the analog implementation of TCM decoders is an interesting candidate to tackle the problem of the enhanced ECC requirements of multilevel flash memories.

3

CMOS Analog TCM Decoders: Design and Performance Analysis

In this chapter, the design and optimization of the fully analog TCM decoder, whose structure has been described in the previous chapter, is presented. In particular, the design trade-offs peculiar to the analog approach are analyzed in depth by comparing the performance and limitations of the 4-state and the 8-state analog decoder, so as to draw some design guidelines for future works.

Both decoders have been designed in the UMC 0.18-μm CMOS process, whose main features have been described in Sec.1.1.5.

3.1 Cells Design

3.1.1 Preliminary Considerations

As already pointed out in Sec.2.5.4, both TCM codes are designed to work on a data field of $M = 192$ bits. After encoding, the 256 bit codewords are stored in 64 memory cells with $q = 16$ levels, that is with a storage capability of 4 bits/cell. As both codes are terminated, the 192 bits wide data field is divided into 190 information bits and 2 termination bits for the 4-state code, which become 189 and 3 respectively for the 8-state one.

The decoder core shown in Fig.2.20 counts 64 identical sections, each constituted by all the 6 different cell types already presented in Sec.2.5.3.

At transistor level, all cells B-F are a variation of the well-known current mode Gilbert multiplier [56]. As an example, the schematic diagram of cell B for the 8-state trellis code is shown in Fig.3.1 with the relative trellis section. In particular, the 8 input currents $I_{y,lmn}$ and 8 output currents $I_{z,rst}$ represent the trellis state probability distributions $p(y)$ and

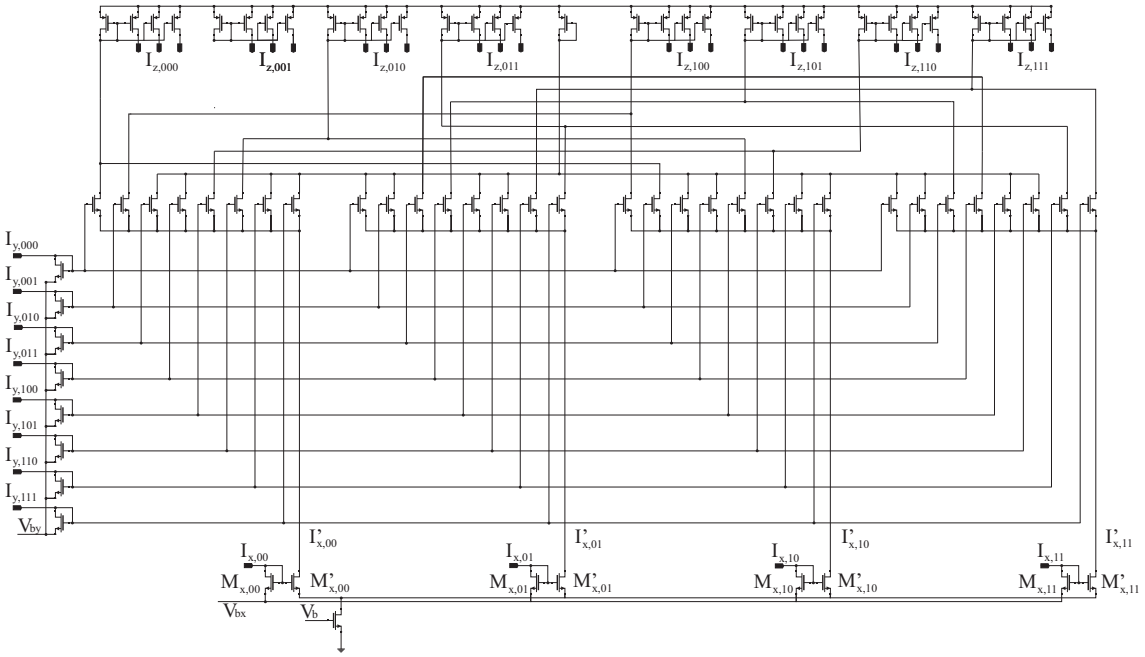


Figure 3.1: B cell schematic for the 8-state analog decoder

$p(z)$, while the 4 input currents $I_{x,ij}$ are the trellis branch metrics probability distributions $p(x)$. The output distribution $p(z)$ is evaluated, in the form of a current vector, according to (A.53). The schematics for cell C, D, E, F1 and F2 can be easily drawn from the corresponding trellis presented in Sec.2.5.3.

The structure common to all the cells consists of a number of diode-connected nMOS transistors operating under weak-inversion, which provide that the input currents $I_{x,ij}$ and $I_{y,lmn}$ are made available to the cell core, an equal number of weak inversion nMOS which compute the products between all possible input values $I_{x,ij}, I_{y,lmn}$ and pMOS current mirrors in strong inversion, used to feed the other cells the output currents $I_{z,rst}$. Every cell also contains a nMOS current source, whose function is to re-normalize the output currents to the bias current I_b according to:

$$I_{x,ij;y,lmn} = \frac{I'_{x,ij} \cdot I_{y,lmn}}{I_y} = \frac{I_{y,lmn}}{I_y} \cdot \frac{I_{x,ij}}{I_x} \cdot I_b \tag{3.1}$$

with $i, j, l, m, n, r, s, t \in \{0, 1\}$. I_x and I_y indicate the sum of all $I_{x,ij}$ and $I_{y,lmn}$ input currents respectively.

The output currents re-normalization is necessary because of the cell trellis, where not all the possible branches between the input and the output states are present, leading to the fact that some currents products items are discarded. Without re-normalization, when

Cell	N_x	N_y	N_z	N_u	N_t	N_c	N_{tot}
B	4	4	4	3	46	64	2944
C	4	4	4	3	46	64	2944
D	4	4	2	1	34	64	2176
E	4	4	4	2	42	64	2688
F1	8	4	2	1	58	64	3712
F2	8	4	2	1	58	64	3712

Table 3.1: 4-state decoder transistors count

Cell	N_x	N_y	N_z	N_u	N_t	N_c	N_{tot}
B	4	8	8	3	82	64	5248
C	4	8	8	3	82	64	5248
D	8	8	2	1	94	64	6016
E	8	8	4	2	102	64	6528
F1	8	4	2	1	58	64	3712
F2	8	4	2	1	58	64	3712

Table 3.2: 8-state decoder transistors count

more cells are connected in sequence as shown in Fig.2.20, the total current reduces along the chain, due to the fact that the output current I_z of every cell is less than the total input one I_x or I_y. This would translate into a decoder performance degradation.

As all the cells show the same structure, the transistor count for each one can be calculated by equation:

$$N_t = 1 + 2N_x + N_y + N_xN_y + N_z + 1 + N_zN_u \tag{3.2}$$

where N_x and N_y are the length of the two input vectors, N_z the length of the output vector, N_u indicates the number of blocks which use the cell output currents, and N_t and N_c are the transistors amount for each cell and the total cells number respectively. The total transistors count for the 4-state decoder core is reported in Table 3.1, while in Table 3.2 the data for 8-state one are shown.

As it was expected, the total transistors count almost double for the 8-state decoder with respect to the 4-state one, as it adds up to 30.464 for the former while it is equal to 18.176 for the latter one.

3.1.2 Weak Inversion Devices

The sum-product cell performance, given the nMOS transistors channel length L, depend mainly on the block bias current density I_b/W, where W is the weak inversion nMOS channel width, as extensively studied in [43, 57]. Increasing I_b/W improves transistors speed but moves the devices progressively out of the pure exponential (weak inversion) region, thus negatively affecting the decoder static accuracy; on the other hand, a faster device settling time translates into a faster transient towards the asymptotic BER of the analog decoder. Transistors sizing has also opposite effects on devices matching, that is on the decoder static accuracy, and on parasitic capacitances, that is on the decoder speed.

As a consequence, the minimum device size in the sum-product cells was set based on accuracy considerations and then the cell bias current was chosen in order to satisfy the decoding time specifications.

Even if systems following the *bio-inspired* design stile [58] seem to be more robust against devices mismatch with respect to the conventional analog ones, nevertheless sub-threshold devices show an exponential relation between the threshold voltage error and the drain current [59–61] due to the exponential dependence of the drain current on the voltage overdrive $V_{GS} - V_{T_h}$. The standard deviation of the threshold voltage error can be modeled by equation (1.21), with the technology dependent parameters A_{V_T} and C_o values still given in Table 1.1.

The choice of the transistors length L has to take into account short-channel effects, which in principle do not seem the major source of performance loss in analog decoder [62], but whose impact is deemed to increase for large decoders.

These considerations lead to set $L = 0.6\mu$m, that is roughly $3 \cdot L_{min}$, and $W = 0.8\mu$m in order to keep the sum-product cells nMOS transistors area roughly 10 times larger than the device minimum area (i.e., $0.24 \times 0.18\,\mu\text{m}^2$). The goodness of this choice will be proved by means of extensive MonteCarlo simulations, which will be discussed in section 3.1.7.

3.1.3 Bias Current

Once the transistors dimensions are fixed, we need to chose the bias current so as to fullfill the decoder settling time specifications and to guarantee that the nMOS will work in their exponential region.

To set a reasonable value for the decoding time, we considered the state-of-the-art

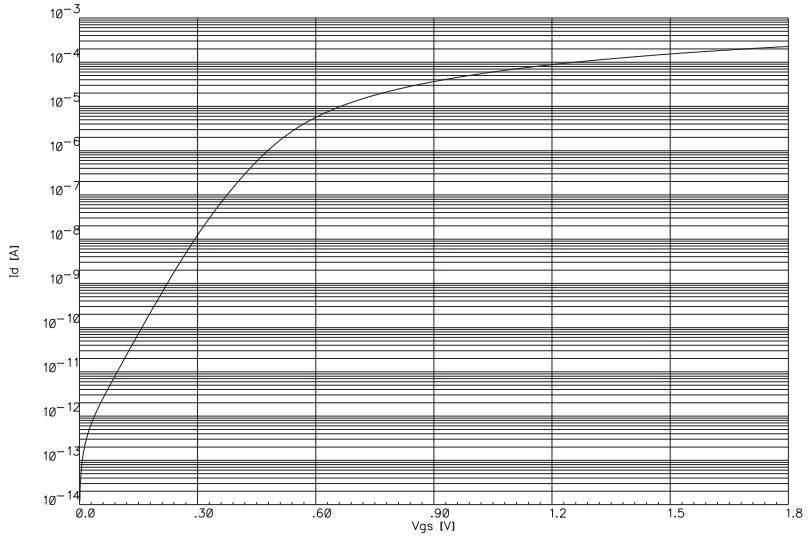


Figure 3.2: Current-voltage characteristic for a $L = 0.6\mu$m $W = 0.8\mu$m nMOS transistor

BCH decoder demonstrated in [36], even if it is designed to work on 2 bit/cell memory. This decoder has a correction capability up to 5 errors with a maximum latency time of 250μs for a data field of 2048 16-bit words. As the analog decoders work on a data field of 190 or 189 information bits, a 2048x16 bit data field is decoded in 173 steps for the 4-state TCM and in 174 steps for the 8-state one, thus leaving a maximum decoding time of 1.4μs for each step in order to guarantee the same latency time of [36].

The current-voltage characteristic for a nMOS transistor with $L = 0.6\mu$m and $W = 0.8\mu$m is shown in Fig.3.2. In order to keep the devices work under true exponential region, the bias current should be in the range of 10^{-10}A, but settling $I_b = 10^{-10}$A would result in a decoding latency too long for our specifications.

If the devices are working under strong inversion, this introduces a performance degradation, as described in [62], especially if some products items are discarded and the total input current is less than the normalization current I_b. It is true that even if the bias current I_b is quite large, some transistors work in the moderate or weak inversion region if the corresponding probabilities are small. However, in these cases, the decisions are clear and the largest probability are nearly unchanged so that we can neglect the error introduced by those devices working under strong inversion. The most critical situation occurs when no probability is dominant. In this case all the transistors work in the strong inversion region if I_b is large enough. As a result, the value of I_b should be chosen so that to avoid all the nMOS to work under moderate or strong inversion when all the inputs currents have nearly the same value.

Following these considerations, a current in the range of μA seems a reasonable choice for the cell biasing.

As already pointed out, the bias current I_b also determines the decoder performance in terms of decoding speed. In order to settle the optimum value for I_b, the decoding time has been evaluated for different bias currents by means of transistor-level simulations.

The inputs of the decoder are generated using a C++ program that implements the following steps:

1. generates random binary 190- or 189-bits wide user words;
2. adds the termination bits;
3. encodes the data;
4. maps them on 64 memory cells threshold voltages;
5. simulates a discrete Gaussian channel (AWGN);
6. calculates the conditional probabilities $p(V_T^R|S_i)$, $p(a_1|S_i)$ and $p(a_2|S_i)$ from the threshold voltage V_T^R read from each memory cell;
7. makes demodulation and detection;
8. saves the decoded words.

The BER as a function of the cell bias current I_b has been simulated at three different SNR, that is 26, 27 and 28dB. As a point of the BER vs SNR curve is considered reliable when a minimum number of 100 erroneous bit is detected, the number of words to be simulated for each SNR has been calculated as $100/(\mathrm{BER}\cdot 192)$, where BER indicates the expected BER and the words are considered 192-bits wide because they include the termination bits. Thus, in order to obtain a reliable BER value, 100 words have been simulated at SNR=26dB, which become 1.000 at SNR=27dB and 11.500 at SNR=28dB.

In the BER simulations, the decoder is feed with a current vector that represents the calculated conditional probabilities $p(V_{Tk}^R|S_{i,k})$, $p(a_{1,k}|S_{i,k})$ and $p(a_{2,k}|S_{i,k})$ values and is given a maximum time of 2μs to settle. The soft outputs are sampled at subsequent time instants, so as to evaluate the BER as a function of the decoding time. The resulting BER curves for the 8-state analog decoder are shown in Fig.3.3 for three different biasing current $I_b = 2, 4 \text{ and } 8\mu$A.

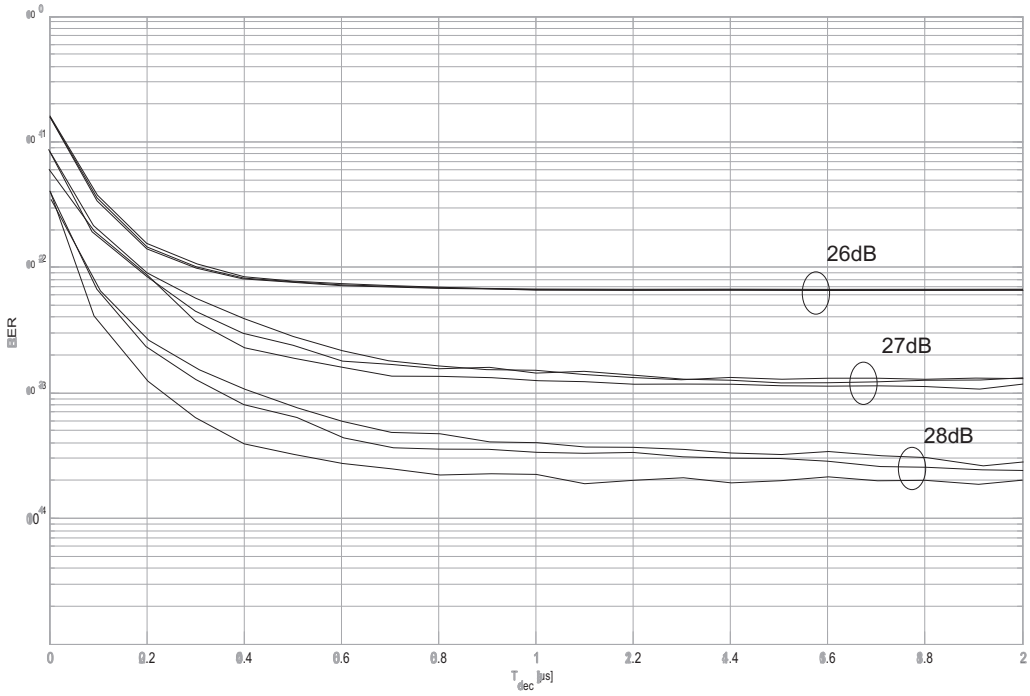


Figure 3.3: BER as a function of the decoding time

At low SNR, where most of the cells have inputs with almost equal probability, the decoding speed is not affected by the cell bias current value, while it improves considerably with I_b at high SNR, when some of the probabilities can be very small. If an error has to be corrected at high SNR, the corresponding cells require a much longer time to commute, because some of the weak inversion nMOS are almost switched off. Increasing the bias current, the nMOS come out of the off region more quickly, thus speeding up the cells settling time.

With $I_b = 8\mu$A we can see that, after a settling time of 1.4μs at SNR=28dB, the decoder BER is equal to $2.1 \cdot 10^{-4}$, which compared with the the ideal C++ model BER of $4.5 \cdot 10^{-5}$, seems a good result.

Following these considerations, we choose $I_b = 8\mu$A for the 8-state decoder.

A similar analysis, performed for the 4-state decoder, has lead to set a bias current of $I_b = 2\mu$A for the 4-state sum-product cells.

3.1.4 Bias Transistor

As the cell input currents are re-normalized at the cell bias current value at each decoding step, the bias current of each cell does not need to be replied with a high precision. As a

consequence, the sizing of the nMOS bias transistor is not particularly critical.

However, it is worth to notice how its drain voltage is subjected to strong variations, as all the $M_{x,ij}$ transistors operate under weak inversion. In fact, it is this very variation of the drain voltage that makes the re-normalization possible. Recalling the low of a MOS working under weak inversion:

$$I_D = I_{D0} e^{\beta(V_G - V_S - V_{T0})} \tag{3.3}$$

where I_{D0} and β are technology dependent constant, if we assume that all the $M_{x,ij}$ and $M'_{x,ij}$ transistors are working in the exponential region, we can write:

$$\begin{aligned} \sum I'_{x,ij} &= \sum I_{D0} \cdot exp(\alpha(V_{G,ij} - V'_S)) \\ &= \sum I_{D0} \cdot exp(\alpha(V_{G,ij} - V_S + V_S - V'_S)) \\ &= \sum I_{x,ij} \cdot exp(\alpha \cdot \Delta V_S) \end{aligned}$$

with $\Delta V_S = V'_S - V_S$. Thus every input current is multiplied by a factor $e^{\beta \Delta V_S}$, so as to make $\sum I'_{x,ij} = I_b$.

As the input current sum decreases with respect to I_b, the multiplying factor increases, lowering the bias transistor drain voltage. As a consequence, the bias nMOS has to be able to work properly with V_{DS} lower that the bias voltage V_{bx}.

3.1.5 pMOS Current Mirrors

The precision of the pMOS current mirrors has a direct impact on the output probabilities accuracy. Thus, they have to be sized following all the design considerations that apply to standard current mirrors. In particular, since the maximum current flowing into the mirror is equal to the cell bias current I_b, the maximum voltage across the current mirror is given by:

$$|V_{GS,max}| = |V_{th,p}| + |V_{eff}(I_b)| = |V_{th,p}| + \sqrt{\frac{2I_b}{k_p \frac{W}{L}}} \tag{3.4}$$

3.1.6 Bias Voltages

As already pointed out, the bias voltage V_{bx} determines the maximum drain to source voltage V_{DS} for the nMOS current source and also the minimum value given the $M'_{x,ij}$ transistors ΔV_S. V_{bx} has thus to be chosen so as to keep the nMOS current source working under strong inversion.

4-states	
M_p	$\frac{0.5}{1}$
M_n	$\frac{0.8}{0.6}$
M_b	$\frac{4}{0.4}$

8-states	
M_p	$\frac{2}{1}$
M_n	$\frac{0.8}{0.6}$
M_b	$\frac{16}{0.4}$

Table 3.3: Transistor size in μm/μm

Cell	m_ε	σ_ε
B	$2.259e^{-3}$	$3.457e^{-3}$
D	$1.511e^{-5}$	$5.676e^{-3}$
E	$2.464e^{-5}$	$4.832e^{-3}$
F1	$1.018e^{-4}$	$6.073e^{-3}$

Table 3.4: Error statistic for the 4-state decoder cells

In the same way, the bias voltage V_{by} determines the V_{DS} range for the $M'_{x,ij}$ transistors. As both the sum currents I_x and I_y are unknown, we can chose V_{by} so as to optimize a "medium" case, when the $M'_{x,ij}$ transistors drain voltage is equal to V_{by}. This leads to:

$$V_{by} \simeq \frac{V_{DD} - |V_{GS,m_{MAX}} - V_{bx}}{2} \tag{3.5}$$

3.1.7 Single Cell Characterization

Following the previous considerations, we set the bias voltages $V_{bx} = 300m\text{V}$ and $V_{by} = 700m\text{V}$. The transistor dimensions for both the 4-state and 8-state decoder, are reported in Table 3.3.

The goodness of this choice has been estimated by characterizing the output error ε of each of the 6 cells types, where ε is defined by:

$$\tilde{f}(x,y) = f(x,y) \cdot (1+\varepsilon) \tag{3.6}$$

As the error ε depends on the cell input values, for each cell 100.000 different input vectors have been simulated. The corresponding error statistic is given in Table 3.4, where m_ε indicates the mean error and σ_ε its standard deviation. As cell C exhibits the same structure as cell B, and cell F1 the same as cell F2, their error statistics have not been simulated.

The effect of mismatch and process variations on the cell error has also been estimated

Cell	m_ε	σ_ε
B	$5.5045e^{-5}$	$9.691e^{-3}$
D	$2.672e^{-5}$	$1.021e^{-2}$
E	$1.062e^{-5}$	$8.823e^{-3}$
F1	$9.26e^{-5}$	$9.22e^{-3}$

Table 3.5: Error statistic for the 8-state decoder cells

Cell	m_{m_ε}	σ_{m_ε}	m_{σ_ε}	$\sigma_{\sigma_\varepsilon}$	max $\sigma_{\sigma_\varepsilon}$	min $\sigma_{\sigma_\varepsilon}$
B	$1.715e^{-3}$	$5.897e^{-3}$	$1.847e^{-2}$	$2.825e^{-2}$	$1.165e^{-1}$	$4.053e^{-7}$
D	$1.035e^{-3}$	$6.612e^{-3}$	$6.721e^{-3}$	$2.13e^{-2}$	$1.654e^{-1}$	$5.145e^{-7}$
E	$1.035e^{-3}$	$5.132e^{-3}$	$1.774e^{-2}$	$3.075e^{-2}$	$1.397e^{-1}$	$7.361e^{-7}$
F1	$8.18e^{-3}$	$5.05e^{-3}$	$9.803e^{-4}$	$8.022e^{-3}$	$1.193e^{-1}$	$1.341e^{-6}$

Table 3.6: Error statistic with MonteCarlo simulations for the 4-state decoder cells

by MonteCarlo simulations: for each cell type 100 instances have been simulated, each one with 1.000 different input vectors. The results are reported in Table 3.6.

As the mean value m_{σ_ε} of all the standard deviations σ_ε calculated on each circuit instance is comparable or even smaller than the standard deviation $\sigma_{\sigma_\varepsilon}$, this indicates that the output error depends on the particular input configurations.

This is due to the fact that some transistors work in the moderate or strong inversion region if the corresponding probabilities are large. However, in these cases, the decisions are clear and the error introduced does not affect the decoder performance. As the output error depends on the input configurations, this indicates that we can not use a second order statistic to describe the random variable ε.

3.2 Voltage to Probability Cell

The interface between the memory array and the decoder core has the task to compute the branch metrics required by the MAP decoder $p(V_{Tk}^R|S_{i,k})$ and the uncoded bits conditional probabilities $p(a_{1,k}|S_{i,k})$ and $p(a_{2,k}|S_{i,k})$, defined in Sec.2.5.3, from the actual threshold voltage V_{Tk}^R programmed in the addressed cell, where k indicates the cell index. Recalling that the branch metrics can be written as:

$$p(V_T^R|S_i) = \sum_{j \in S_i} p(V_T^R|V_T(j)) \tag{3.7}$$

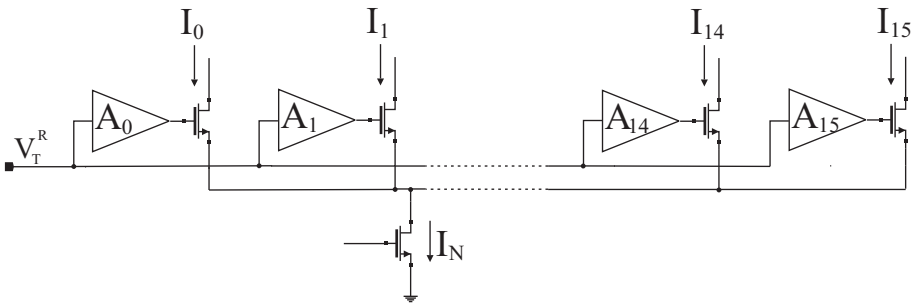


Figure 3.4: Analog circuit for conditional probabilities generation

the voltage to probability module has to calculate the conditional probabilities $p(V_T^R|V_T(j))$.

These probabilities are defined as:

$$p(V_T^R|V_T = V_T(j)) = \frac{1}{\sqrt{2\pi}\sigma_n} exp(-\frac{(V_T^R - V_T(j))^2}{2\sigma_n^2}) \tag{3.8}$$

where σ_n^2 is AWGN channel noise variance, V_T^R is the threshold voltage read from the memory cell, V_T is the threshold voltage programmed in the cell and $V_T(j)$, $j = 0 \div (q-1)$ are all the possible cell voltage levels, that correspond to the $\mathcal{A}$ alphabet symbols.

As already pointed out in Sec.2.5.3, we considered a reference memory cell with $q = 16$ threshold voltage levels equally spaced between fixed minimum and maximum value $V_{TMIN} = 0$V and $V_{TMAX} = 5.6$V, and a channel noise standard deviation σ_n varying between 80 and 300 mV.

In the case of a binary alphabet like $\{-1, +1\}$, the (3.8) can be implemented by a simple differential pair, as described in Sec.1.2.2.

The same concept can be easily extended to the case of non binary alphabet, as proved by Frey in [63], by replacing the differential pair with the circuit shown in Fig.3.4.

In this case the output currents I_j are proportional to the conditional probabilities $p(V_T^R|V_T(j)$ according to:

$$p(V_T^R|V_T = V_T(j)) = I_N \frac{p(V_T^R|V_T = V_T(j))}{\sum_{i\in\mathcal{A}} p(V_T^R|V_T = V_T(i))} \tag{3.9}$$

where the alphabet $\mathcal{A} = \{0 \div 15\}$.

If we assume that all the nMOS transistors are working in the exponential region, we can derive the expression of each output current I_j as a function of the gate voltages V_j, $j \in \mathcal{A}$, as:

$$I_j = \frac{I_N}{\sum_{i\in\mathcal{A}} exp(\frac{V_i - V_j}{nU_T})}, j \in \mathcal{A} \tag{3.10}$$

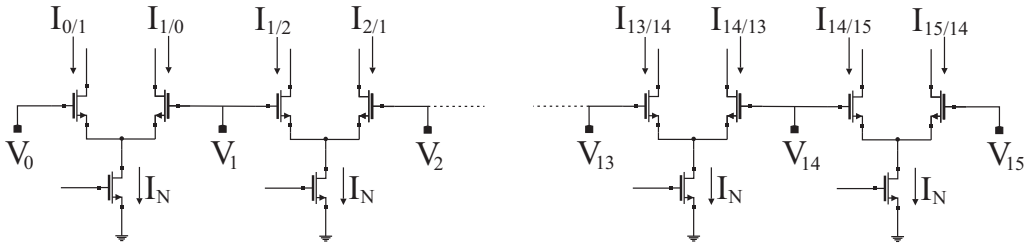


Figure 3.5: Conceptual schematic for approximated conditional probabilities generation

where n is a technology dependent parameter. Its value for the UMC process has been estimated for nMOS and pMOS by means of transistor simulations, which gave $n_n = 1.31$ and $n_p = 1.57$.

It can be proved [63] that, in order to obtain the output currents I_j, the programmed threshold voltage level read from the memory cell has to be amplified by a different factor for each of the alphabet symbols so as to obtain the gate voltages:

$$V_j = V_T^R \cdot \frac{nU_T V_T(j)}{\sigma^2} - \frac{nU_T (V_T(j))^2}{2\sigma^2} \tag{3.11}$$

As $nU_T \simeq 34$mV and $\sigma = 80 \div 300$mV, it is easy to see how this results in a gate voltages range which is not compatible with our technology.

3.2.1 Approximated Normalization

An output current I_j proportional to the conditional probability $p(V_T^R|V_T(j))$ can be obtained [57] with the circuit shown in Fig.3.5, which consists of 16 differential pairs whose nMOS are biased in weak inversion so that their input voltage - output current relation is an hyperbolic tangent. The differential pairs input voltages V_{d_j} are given by:

$$V_{d_j} = \alpha \cdot \left[V_T^R - \frac{V_T(j) + V_T(j-1)}{2} \right] \tag{3.12}$$

where the scaling factor α is equal to:

$$\alpha = \frac{nU_T \Delta}{\sigma_n^2} \tag{3.13}$$

Δ indicates the distance between adjacent threshold voltage levels, which in our case is equal to 400mV.

As an example, let's consider the output current $I'_1 = I_{1/0} - I_{2/1}$ which corresponds to the conditional probability $p(V_T^R|V_T = V_T(1))$. According to (3.9), I_1 can be written as:

$$I_1 = I_N \frac{p(V_T^R|V_T = V_T(1))}{\sum_{i\in\mathcal{A}} p(V_T^R|V_T = V_T(i))} \tag{3.14}$$

Let's define

$$\begin{aligned} p[V_T^R|V_T = V_T(0)] &= p[V_T^R|0] \\ p[V_T^R|V_T = V_T(1)] &= p[V_T^R|1] \\ p[V_T^R|V_T = V_T(2)] &= p[V_T^R|2] \end{aligned}$$

where $V_T(0)$, $V_T(1)$ and $V_T(2)$ correspond to the programmed threshold voltage levels 0V, 400mV and 800mV respectively. The circuit of Fig.3.5, generates the current I'_1 which, according to (1.9), is given by:

$$I'_1 = I_N \frac{p[V_T^R|1]}{p[V_T^R|0] + p[V_T^R|1]} + I_N \frac{p[V_T^R|2]}{p[V_T^R|1] + p[V_T^R|2]} \tag{3.15}$$

The (3.15) can be written as:

$$I'_1 = I_N \frac{p[V_T^R|1]^2 - p[V_T^R|0] \cdot p[V_T^R|2]}{p[V_T^R|1]^2 + p[V_T^R|1] \cdot p[V_T^R|2] + p[V_T^R|0] \cdot p[V_T^R|1] + p[V_T^R|0] \cdot p[V_T^R|2]} \tag{3.16}$$

The factor $p[V_T^R|0] \cdot p[V_T^R|2]$ can be discarded as it is much smaller than all the other ones, as it is related to two non adjacent voltage levels. Thus (3.16) becomes:

$$I'_1 = I_N \frac{p[V_T^R|1]}{p[V_T^R|1] + p[V_T^R|2] + p[V_T^R|0]} \tag{3.17}$$

which shows how the calculated conditional probability is an approximation of the ideal one. However, we will see in Sec.3.2.2 how this approximation is good enough for our purposes.

The expression for the input differential pair voltage V_{d_j} given by equation (3.12) can be deduced by recalling that:

$$I_{D_j} = \frac{I_N}{1 + exp(\frac{-\alpha V_{d_j}}{nU_T})} \tag{3.18}$$

which, in order to satisfy equation (3.15), must be equal to:

$$I_{D_j} = \frac{I_N}{1 + exp(\frac{-(V_T^R - V_T(j-1))^2 + (V_T^R - V_T(j))^2}{2\sigma_n^2})} \tag{3.19}$$

By comparing (3.18) to (3.19), we obtain the expression for V_{d_j}:

$$V_{d_j} = \frac{nU_T(V_T(j) - V_T(j-1))}{\sigma_n^2}[V_T^R - \frac{(V_T(j) + V_T(j-1))}{2}] \tag{3.20}$$

which is equal to (3.12) if we note that $(V_T(j) - V_T(j-1)) = \Delta$.

3.2.2 Voltage to Probability Module Design

The voltage to probability module, which schematic is shown in Fig.3.6, receives as inputs the differential voltages V_{d_j} calculated according to (3.12) and produces as outputs the currents I_j, which are proportional to the conditional probabilities $p(V_T^R|V_T(j))$. In particular, the 16 I_j currents are given by:

$$\begin{aligned}
I_0 &= I_{0/1} \\
I_1 &= I_{1/0} - I_{2/1} \\
I_2 &= I_{2/3} - I_{1/2} \\
I_3 &= I_{3/2} - I_{4/3} \\
I_4 &= I_{4/5} - I_{3/4} \\
I_5 &= I_{5/4} - I_{6/5} \\
I_6 &= I_{6/7} - I_{5/6} \\
I_7 &= I_{7/6} - I_{8/7} \\
I_8 &= I_{8/9} - I_{7/8} \\
I_9 &= I_{9/8} - I_{10/9} \\
I_{10} &= I_{10/11} - I_{9/10} \\
I_{11} &= I_{11/10} - I_{12/11} \\
I_{12} &= I_{12/13} - I_{11/12} \\
I_{13} &= I_{13/12} - I_{14/13} \\
I_{14} &= I_{14/15} - I_{13/14} \\
I_{15} &= I_{15/14}
\end{aligned} \tag{3.21}$$

where $I_{i/j}$ indicates the output current of the differential pair, which compares the threshold voltage levels i and j, that is larger when the threshold voltage level read from the memory cell is equal to $V_T(i)$.

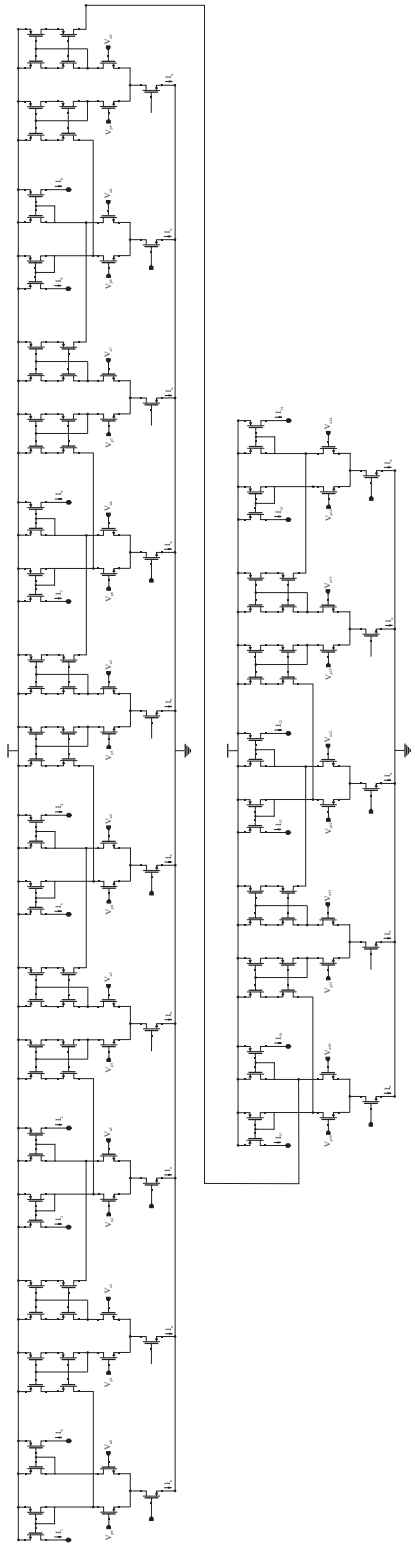


Figure 3.6: Voltage to probability schematic

4-states	
M_p	$\frac{0.5}{2}$
M_{c_1}	$\frac{2}{1}$
M_{c_2}	$\frac{2}{0.5}$
M_n	$\frac{40}{0.4}$
M_b	$\frac{2}{4}$

8-states	
M_p	$\frac{6}{2}$
M_{c_1}	$\frac{10}{1}$
M_{c_2}	$\frac{10}{0.5}$
M_n	$\frac{200}{0.4}$
M_b	$\frac{10}{4}$

Table 3.7: Transistor size in μm/μm

It is worth to notice how, with the scheme described above, each differential pair output current is used just once. This is important because it avoids the introduction of more current mirrors in the V2P cell, thus increasing the cell current consumption.

The design of the voltage to probability cell follows the same considerations as for the decoder core cells B-F. In particular, we first impose the same current consumption for both voltage to probability and decoder core cells, which means a bias current of $I_N = I_b/15$ for each differential pair of the voltage to probability module. However, we saw how this choice had a negative impact on the overall decoder performance due to two different reasons:

- the voltage to probability settling time became the dominant factor in determining the decoding time, thus slowing down the overall decoder speed;
- the decoder precision was heavily affected by the the fact that the sum I_x of the input currents I_{x_j} was 15 times lower than the decoder cell bias current I_b.

As the settling time of cells with the same current density I_b/W is faster for those cell which have a higher bias current I_b, we decided to set $I_N = I_b$.

As the approximate normalization relies on the hyperbolic tangent voltage to current characteristics of the differential pair nMOS, they have been sized so as to work always under weak inversion. This guarantees a more precise input probabilities to the decoder core.

In order to improve the cell accuracy, a cascode current mirror instead of a simple one has been used in all the differential pairs where no current subtraction is performed, as opposite to the nodes where the currents are subtracted, which exhibit a relatively low impedance.

The transistor size for both 4-state and 8-state decoder voltage to probability module are reported in Table 3.7.

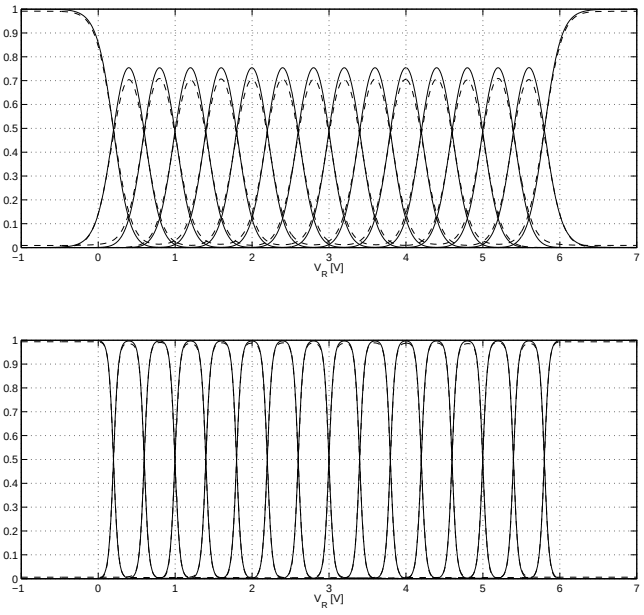


Figure 3.7: Conditional probabilities calculated with the approximated normalization method and ideal at SNR=24dB and SNR=30dB

As already stressed out in Sec.3.2.1, this method calculates an approximated normalization of the conditional probabilities, whose precision increases with SNR.

In Fig.3.7, the ideal conditional probabilities are compared with those calculated with the approximated normalization at two different input SNR of 24dB and 30dB respectively, while in Fig.3.8 the corresponding probability error is reported.

At SNR=24dB the error introduced by the approximated normalization is less than 0.06. If we consider that at this SNR the noise standard deviation $\sigma_n = 0.555 \cdot \Delta = 222$mV, with an error equal to $2\sigma_n$ on the read threshold voltage, we move to the adjacent voltage level, which from Fig.3.7 corresponds to a probability error grater than 0.6.

The maximum probability error as a function of the input SNR is shown in Fig.3.9, where we can see how it decreases at high SNR. For SNR grater than 30dB, the error remains around 0.01. This effect is due to the fact that the differential pair input voltages V_{d_j} are saturated to a maximum peak to peak value of 600mV.

The effect of mismatch and process variations on the voltage to probability error has been estimated by means of MonteCarlo simulations. 100 cell instances have been simulated, each one with 1.000 different input vectors. The results are reported in Table 3.8.

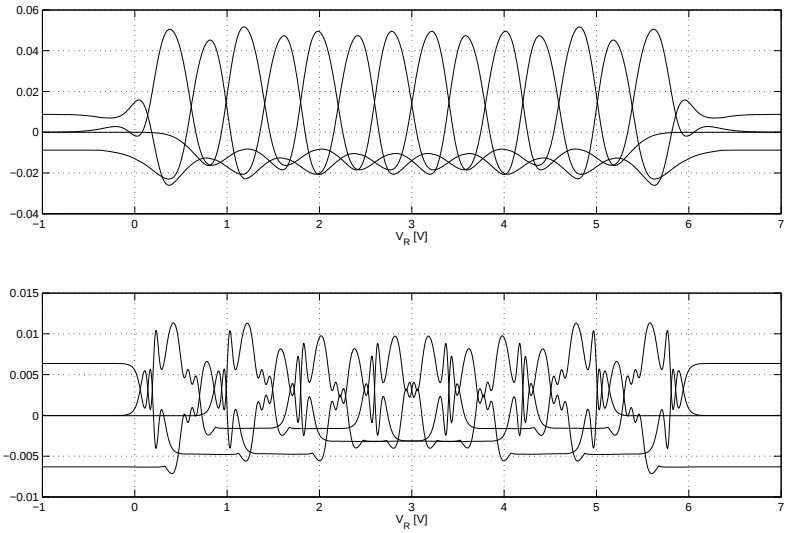


Figure 3.8: Conditional probabilities error at SNR=24dB and SNR=30dB

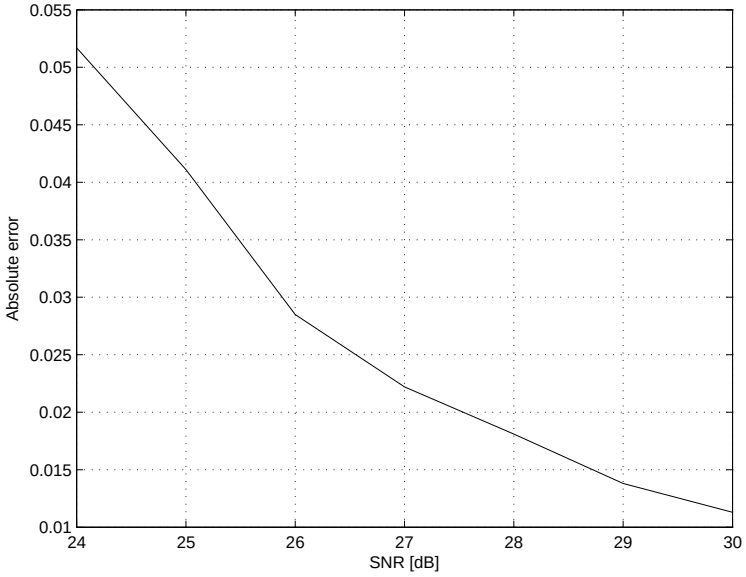


Figure 3.9: Conditional probability error

m_{m_ε}	σ_{m_ε}	m_{σ_ε}	$\sigma_{\sigma_\varepsilon}$	max $\sigma_{\sigma_\varepsilon}$	min $\sigma_{\sigma_\varepsilon}$
$1.218e^{-2}$	$2.248e^{-1}$	$5.802e^{-2}$	$2.397e^{-2}$	$8.739e^{-2}$	$7.287e^{-7}$

Table 3.8: Output currents error statistic for V2P

3.3 Decoder Optimization: the Reset

In order to make the decoding of a new frame independent from the elaboration results of the previous one, we decided to reset the cells to an uniform state probability before starting the decoding of a new word.

The reset has also the advantage to speed up the decoding time, as demonstrated by Fig.3.10, where the output currents transient with and without reset is shown.

We can see that if the decoding of a new frame starts from a reset state, we can avoid a lot of spurious commutations in the decoder output currents.

The uniform state probability is obtained by forcing the output currents of cells B and C to be all equal to $I_b/8$. This is done by adding pass-transistors to the output current mirrors, so as to short the gate voltage of all the pMOS and thus forcing all the currents flowing through the output transistors to be equal.

At the same time, cell B, C, F1 and F2 inputs are reset to the uniform state probability, so as to generate output currents all equal to $I_b/8$. Thus cell B and C inputs are set to:

$$p(V_{Tk}^R|S_{0,k}) = p(V_{Tk}^R|S_{3,k}) = 0.5$$
$$p(V_{Tk}^R|S_{1,k}) = p(V_{Tk}^R|S_{2,k}) = 0$$

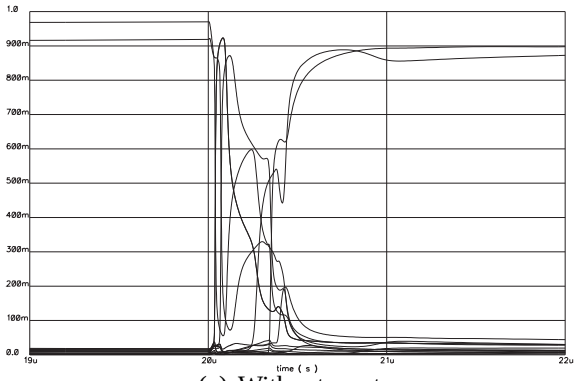
while cell F1 and F2 inputs reset configurations are given by:

$$p(a_{1,k} = 1|S_{0,k}) = p(a_{1,k} = 0|S_{3,k}) = 0.5$$
$$p(a_{1,k} = 1|S_{1,k}) = p(a_{1,k} = 1|S_{2,k}) = p(a_{1,k} = 1|S_{3,k}) = 0$$
$$p(a_{1,k} = 0|S_{0,k}) = p(a_{1,k} = 0|S_{1,k}) = p(a_{1,k} = 0|S_{2,k}) = 0$$

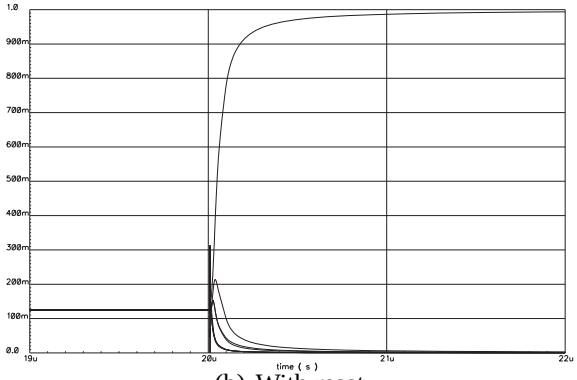
and:

$$p(a_{2,k} = 1|S_{3,k}) = p(a_{2,k} = 0|S_{0,k}) = 0.5$$
$$p(a_{2,k} = 1|S_{0,k}) = p(a_{2,k} = 1|S_{1,k}) = p(a_{1,k} = 1|S_{2,k}) = 0$$
$$p(a_{2,k} = 0|S_{1,k}) = p(a_{2,k} = 0|S_{2,k}) = p(a_{1,k} = 0|S_{3,k}) = 0$$

respectively.



(a) Without reset



(b) With reset

Figure 3.10: Output probabilities transient with and without reset

The decoding time for the 8-state decoder with and without reset has been simulated at SNR=27dB and SNR=28dB, showing how the reset speeds up the decoding latency especially at high SNR. This is due to the fact that at high SNR the decisions are more clear, which means that after a frame has been decoded the transistors are either switched off or completely on. If a nMOS has to commute for the next frame from the switched off state, this requires a long time. During this time, the information coming from the other cells in the chain can be more updated and cause the cell outputs to have spurious commutations.

If the decoding of each codeword starts from an uniform state probability, these spurious commutations can be avoided, thus improving the decoding speed.

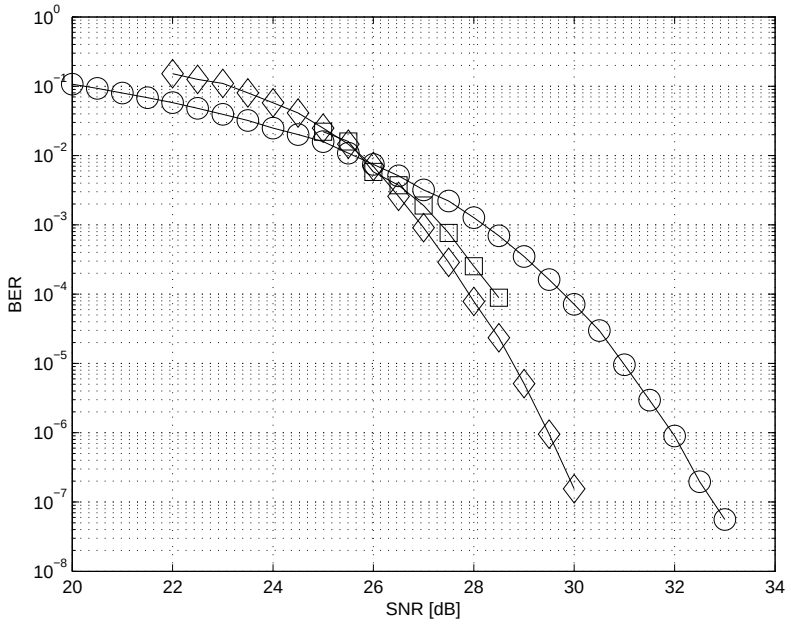


Figure 3.11: BER vs SNR for a memory cell with $q = 8$ without ECC (circle), a memory cell with $q = 16$ and 4-state TCM C++ model (diamonds) and with TCM analog decoder (squares)

3.4 Overall Decoder Performance

The performance of both 4-state and 8-state decoder have been estimated by means of transistor-level simulations. In each simulation, the threshold voltage levels stored in 64 memory cells are read and applied to an AWGN channel. Then the conditional probabilities are calculated and the corresponding 4 input currents I_x are fed to the decoders, which are given a time $T_D = 1.3\mu$s to settle. The soft output of the decoders is then sliced and used to estimate the BER and the decoders are reset to uniform state probability in a time $T_R = 100$ns, thus giving an overall decoding time of 1.4μs.

The BER vs SNR curves for the 4-states and 8-states analog decoder are reported in Fig.3.11 and Fig.3.12 respectively. These curves are compared with the benchmark obtained with a C++ behavioral model of the decoders. In the model, the sum-product operations of the MAP decoder are ideal, with probabilities represented by double precision numbers and no source of distortion, offset or noise is taken into account. In both cases the simulations results show a performance loss of 0.5dB with respect to the benchmark at a BER=10^{-4}, in line with the results already found for all-analog Turbo decoders [14].

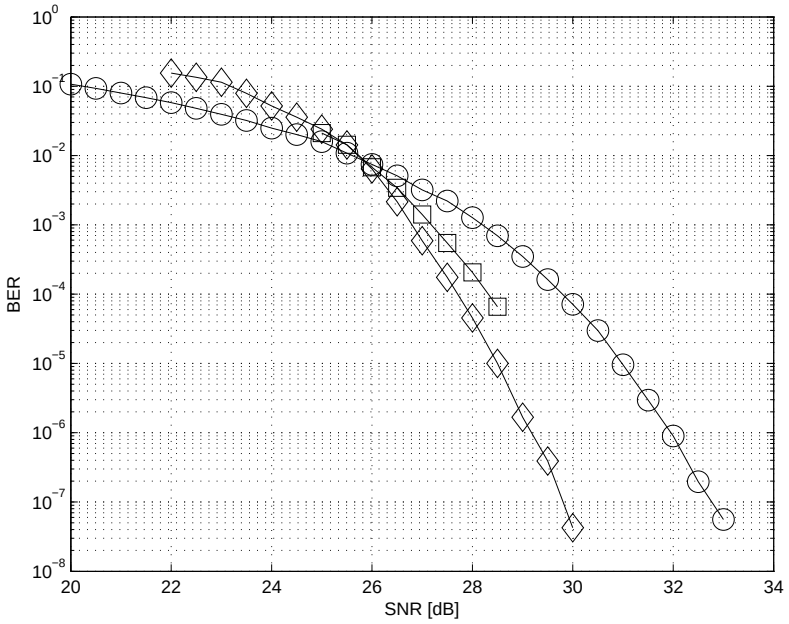


Figure 3.12: BER vs SNR for a memory cell with $q = 8$ without ECC (circle), a memory cell with $q = 16$ and 8-state TCM C++ model (diamonds) and with TCM analog decoder (squares)

Each point of the BER vs SNR curves is considered reliable when a minimum number of 100 erroneous bit is detected. As the number of frames to be simulated increases at lower BER, it was not possible to simulate the decoders behavior at SNR greater than 28.5dB, due to the excessive computational load of transistor-level simulations. For both decoders, the BER at SNR=28.5dB was estimated with a reduced number of 50 errors.

The effect of transistor mismatch has been estimated for the 4-state decoder by means of transistor-level MonteCarlo simulations at BER=10^{-3}, which corresponds to a SNR = 27dB. In this case, the BER has been estimated with a reduced number of 50 errors, due to the long computational time required. The simulations results with 100 MonteCarlo iterations reported in Fig.3.13 show for the BER a mean value $m = 2.67 \cdot 10^{-3}$ with standard deviation $\sigma = 1.54 \cdot 10^{-4}$, which corresponds to a loss between 0.4 and 0.6 dB with respect to the BER benchmark of $9.07 \cdot 10^{-4}$. The deviation with respect to the typical case is of 0.1dB, in accordance with the simulation and experimental data reported in [14].

The decoder core estimated area occupation is 0.32mm^2 for the 4-state one with a power consumption of 4mW at 1.8V supply, which become 0.55mm^2 and 16.5mW re-

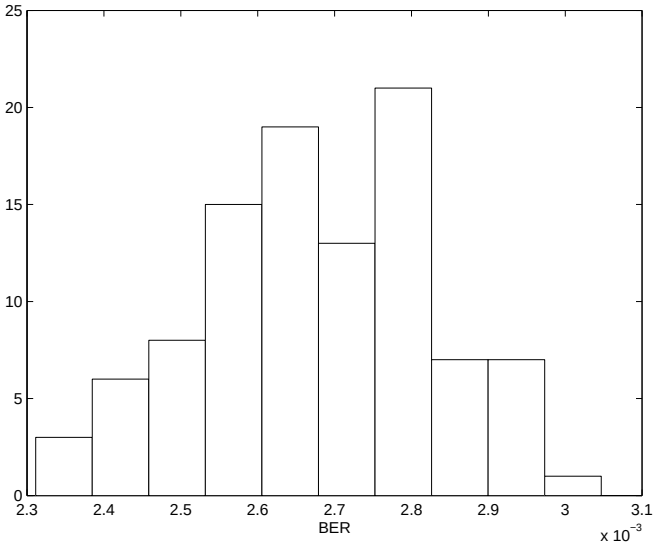


Figure 3.13: 4-state decoder MonteCarlo simulations at SNR=27dB

spectively for the 8-state decoder.

The area occupation for the interface circuitry, that is the voltage to probability modules, has been estimated in 4.8mm^2 for the 8-state decoder, with a power consumption of 13.8mW, which reduce to 1.25mm^2 and 3.4mW for the 4-state one respectively.

3.5 Conclusions

The simulation results reported in this work suggest that a full analog implementation of a TCM decoder for multilevel flash memories can achieve a decoding speed comparable with the state-of-the-art linear block codes occupying a small area, with a BER close to that of the ideal decoding algorithm.

It is worth to notice how the most area and power consuming circuitry is not the one implementing the decoder core, but that used to realize the interface between the memory array and the decoder core itself. This can be traced back to the complexity of dealing with memory cells with 16 levels.

Transistor-level MonteCarlo simulations show how the analog decoder is robust with respect to transistors mismatch. However, as the performance loss in terms of BER of the 4-state decoder with respect to the 8-state one is only 0.5dB while the power consumption increases by over a factor of 2, our work demonstrates how the analog approach is all the

more a competitive solution for ECC in multilevel flash memories as far as the decoder states number is kept low.

4

UWB-IR Transceiver Chipset for Sensor Network Applications

The change in FCC regulations that allows unlicensed communication using pulsed ultra-wideband (UWB) signalling has given new momentum to the research in this field, which has roots that can be traced back to the original Marconi spark gap radio. UWB signaling has many attributes that make it attractive for a wide range of applications, from ultra-low-power RFID tags and wireless sensors to streaming wireless multimedia and wireless USB at data rate greater than 1Gb/s.

This chapter introduces UWB signaling and regulations, with more details on wireless sensor networks applications, to present a transceiver chipset for UWB Impulse Radio. The specifications, architecture and implementation of a UWB-IR non-coherent receiver are then outlined and a synchronization algorithm is proposed and successfully tested, while the transmitter design will be discussed in details in Chap.5.

4.1 UWB Definition

In February 2002, the FCC approved the use of the 3.1-10.6GHz band for UWB communication [64, 65], giving birth to a new technology for wireless communication.

The noise emissions limit for digital electronics above 960MHz is set by the FCC at a constant -41.3dBm/MHz [66]. For example, personal computers are allowed to radiate noise below this level at any frequency above 960MHz. The original intent of UWB communication was to transmit data within the emissions limits already placed on personal computers. However, due to interference concerns from UWB radiators to other existing wireless services the FCC placed “conservative” requirements on UWB emissions. These limits are shown in Fig.4.1 and reported in Table 4.1. Instead of a constant

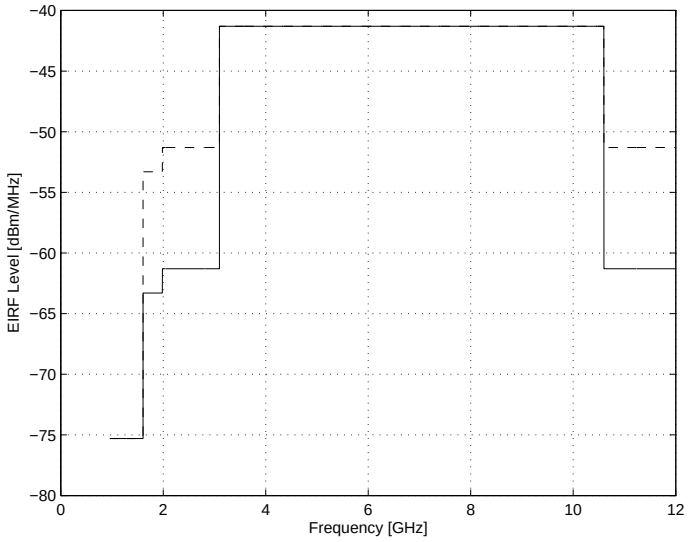


Figure 4.1: FCC emissions limit for indoor (dashed) and outdoor (solid) UWB communication

-41.3dBm/MHz above 960MHz, a deep notch is placed around GPS and PCS services because these receivers have higher sensitivities. GPS operates at 1.2 and 1.6GHz and PCS at 1.9GHz. There are also stricter requirements on outdoor or handheld UWB devices than indoor UWB devices.

UWB signaling has been used in the military since 1960's for both communication and radar. The UWB pulses used in radar applications were low-frequency, high power and generated with non-linear devices and transmission lines that can not be integrated in a high volume process. Some of this technologies for low-frequency pulse generation are still actively researched today [67], even if UWB low-power applications are gaining more and more popularity within the international research community.

UWB has several advantages over traditional narrowband architectures. From a channel prospective, the wide bandwidth can offer excellent robustness to multi-path fading [68]. Additionally, the narrow pulses in time offer the ability to perform precise locating combined with communication. UWB has the potential for spatial capacity that is orders of magnitude above other popular wireless standards such as 802.11a, 802.11b and Bluetooth [69, 70].

The main limitations of UWB communication is the presence of strong, in-band interference that can easily saturate the UWB receiver front-end. The overlap between UWB

Frequency Range $[MHz]$	Indoor Limit $[dBm/MHz]$	Outdoor Limit $[dBm/MHz]$
Below 960	FCC 15.209	
$960-1610$	-75.3	-75.3
$1610-1990$	-53.3	-63.3
$1990-3100$	-51.3	-61.3
$3100-10600$	-41.3	-41.3
Above 10600	-51.3	-61.3

Table 4.1: FCC Mask Limits

and existing services is a major concern in both the transmitter and receiver design, since UWB transmitters will also raise the noise floor seen by narrowband victim receivers.

4.2 UWB Sensor Networks

Wireless sensor networks consist of tens to thousands of distributed low complexity nodes that have limitations both on process power and memory, and severe restriction on power consumption. By the very nature of the application, traffic in sensor networks is often bursty with long periods of no activity. For event detection operations, a device may remain idle for long periods, sending only ”heart-beat” information, then suddenly be required to send significant amounts of data when an event occurs. For devices involved in continuous monitoring, the flow of traffic will be more stable. However, efficient multiple access, reliability and battery life are still major concerns.

Impulse-Radio-based UWB technology proprieties make it well suited to sensor networks applications. In particular, as already outlined, UWB-IR systems have potentially low complexity and low cost [71] with respect to classical narrow-band radio [72–74] or multi-band OFDM UWB [75, 76]. Moreover, they exhibit noise-like signals, are resistant to severe multipath and jamming and have a very good time domain resolution, allowing for location and tracking applications.

The low complexity and low cost of impulse radio UWB systems arise from the essentially baseband nature of the signal transmission. Unlike conventional radio systems, UWB transmitters produce very short time domain pulses that are able to propagate without the need of an additional radio frequency (RF) mixing stage [77]. At the receiver side, the non-coherent energy detection approach may be adopted [78]. Non-coherent com-

munication does not require precise phase control, which allows both the transmitter and the receiver architecture to be simplified, particularly the high frequency circuits that can consume the majority of power in a wireless transceiver. A non-coherent energy detection scheme can further reduce hardware complexity while providing resilience to multi-path fading without the cost of high frequency Rake-base techniques [79]. Non-coherent solutions suffer from a reduced robustness with respect to narrow-band interferers, which can easily cause the receiver front-end to saturate. The IEEE 802.15.4a standard [80] has recognized the advantages offered by non-coherent communication and includes support for it.

The aim of our project is to realize a UWB-IR transceiver chipset for low-data rate wireless sensor netwoks. In particular, our target is a transmission data rate of 100kb/s over a link distance d of at least 10 metres. As non-coherent communication advantages in terms of power consumption are significant especially for short distance links, we decided to use it for our system.

4.3 UWB Signal Choice

According to FCC standard, UWB signals must have a minimum continuous signal bandwidth of 500MHz, a spectral mask of -41.3dBm/MHz within the 3.1-10.6GHz bandwidth and a peak power limit that can not be exceeded. Thus, the three major design choices for UWB signals are bandwidth, modulation and pulse type.

4.3.1 Bandwidth

Although the IEEE 802.15.4a standard physical layer for UWB-IR [81] can operate in several bands of 500MHz or 1.5GHz from 3.1GHz to 10.6GHz, we prefer to limit the pulse bandwidth from 7.25GHz to 8.5GHz. The advantage is twofold: first the system exploits only that subset of the UWB band that is allowed for transmission in USA, Europe and Japan [82]; secondly, the signal is concentrated in the upper part of the UWB spectrum, maximizing the frequency separation from WLAN interferes around 2.4GHz and 5GHz.

Thus, our system will operate in a single band B of 1.25GHz instead of bands of 500MHz, in order to maximize the transmitted power.

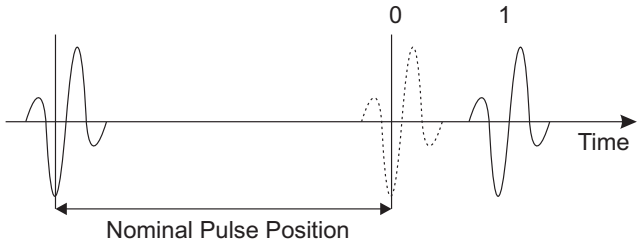


Figure 4.2: Pulse position modulation

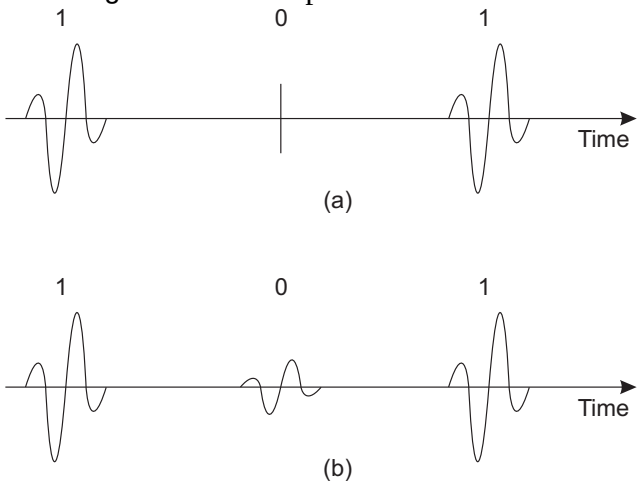


Figure 4.3: Pulse amplitude modulation

4.3.2 Modulation

Although information can be encoded in a UWB signal in a variety of methods, only some modulation schemes are suitable for non-coherent energy detection. Among these, the most popular modulation schemes developed up to date for UWB are pulse-position modulation (PPM) [83], pulse-amplitude modulation (PAM) [84], on-off keying modulation (OOK) [85] and binary phase-shift keying modulation (BFSK).

PPM PPM is based on the principle of encoding information with two or more positions in time, referred to the nominal pulse position, as shown in Fig.4.2. A pulse transmitted at the nominal position represents a 0 and a pulse transmitted after the nominal position represents a 1. The drawing shows a two-position modulation, where one bit is encoded in one pulse. Additional positions can be used to provide more bits per symbol.

The time delay between positions is typically a fraction of a nanosecond, while the time between nominal positions is typically much longer to avoid interference between pulses.

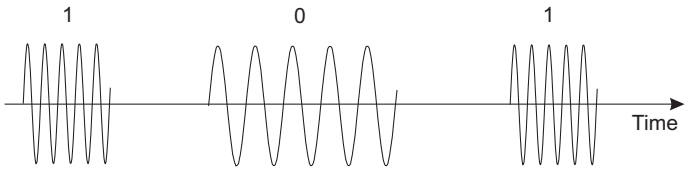


Figure 4.4: Binary frequency-shift keying modulation

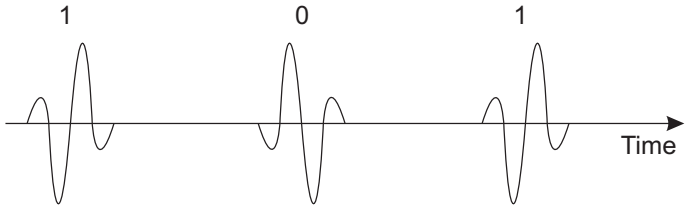


Figure 4.5: Binary phase-shift keying modulation

PAM and OOK PAM is based on the principle of encoding information with the amplitude of the pulses, as shown in Fig.4.3. The drawing shows a two-level modulation, respectively, for zero and lower amplitude, where one bit is encoded in one pulse. When the binary 0 is associated with the zero amplitude pulse, as shown in Fig.4.3.a, we speak of OOK.

As with pulse position, with PAM more amplitude levels can be used to encode more than one bit per symbol.

BFSK In frequency-shift keying (FSK) the digital information is encoded through discrete frequency changes of a carrier wave. The simplest FSK is binary FSK (BFSK). BFSK literally implies using a couple of discrete frequencies to transmit binary information. With this scheme, the 1 is called the mark frequency and the 0 is called the space frequency. The time domain of an FSK modulated carrier is illustrated in Fig.4.4.

BPSK In binary phase-shift keying modulation, information is encoded with the polarity of the pulse, as shown in Fig.4.5. The polarity of the pulses is switched to encode a 0 or a 1. In this case, only one bit per impulse can be encoded because there are only two polarities available to choose among. Although BPSK modulation stand alone is not suited to non-coherent energy detection, nevertheless BPSK scrambling in addition to PPM is used to eliminate PPM spectrum tone lines.

The choice of the PPM scheme leads to a reduced receiver complexity with respect

to the other modulation schemes but, on the other side, PPM spectrum exhibits tone lines $10\log_{10}(PRF/1\text{MHz})$ above the BPSK spectrum while keeping all other factors, in particular the total pulse energy and the pulse repetition frequency PRF, equal [86]. This results in a PPM transmitter having to lower its power by this factor relative to a BPSK transmitter in order to meet the FCC mask. Therefore, high-order PPM or BPSK scrambling in addition to PPM is used to eliminate these tones and thus the need to reduce power [87]. Because BPSK decouples the scrambling problem from the modulation, it is typically preferred over high-order PPM, which adds complexity to the receiver hardware.

The advantage of PPM over BFSK consists in the removal of the two additional filters, centered at the mark and space frequency respectively, required for BFSK non-coherent detection. Indeed, while PAM or OOK needs the comparison with a reference threshold in order to extract the encoded information, with 2-PPM the bit value can be deduced by simply comparing the energy received in two time slots centered at the nominal and shifted time position respectively.

4.3.3 Pulse Shape Analysis

There are several pulse shapes found in literature for UWB communication, ranging from spectral inefficient [68, 88–90] to precisely controlled frequency tolerance [91, 92]. The performance in terms of BER has been analyzed for a range of pulse shapes and modulation techniques [93–95]. However, we consider three metrics to quantify a pulse shape, that are: spectral efficiency, out-of-band emissions and time-bandwidth product.

Spectral Efficiency The spectral efficiency of a pulse quantifies how well the pulse spectrum utilizes the available bandwidth. The system performances in terms of BER depends only on the received pulse energy [96] and not on its actual shape. Therefore, given an average power limit and a -10dB channel bandwidth in the receiver, the transmitter must fill the channel spectrum as tightly as possible. The spectral efficiency of a pulse is the loss incurred from incomplete filling of the -10dB channel bandwidth calculated by

$$\eta_{ch} = \frac{E_{ch}}{P_{FCC} \cdot B_{-10dB}} \tag{4.1}$$

where E_{ch} is the pulse energy within the -10dB channel bandwidth, P_{FCC} is the maximum average power spectral density in W/MHz and B_{-10dB} is the -10dB bandwidth in MHz.

Out-of-Band Emissions The out-of-band emissions metric of a pulse is the ratio of the energy outside the -10dB channel to the energy within the -10dB channel. This metric is used to analyze the adjacent channel interference and it is calculated by

$$\eta_{out} = (E_{tot} - E_{ch})/E_{ch} \tag{4.2}$$

where E_{tot} is the total pulse energy given by

$$E_{tot} = \int_{-\infty}^{+\infty} p^2(t)dt \tag{4.3}$$

Time-Bandwidth Product The time-bandwidth product is a figure of merit which indicates the localization of a pulse both in time and frequency. The lower this number, the more localized a pulse is in both time and frequency, which generally produces the best combination of performance in both time and frequency domains. The time-bandwidth product is calculated by

$$TB_w = D_p \cdot d_p \tag{4.4}$$

where

$$D_p^2 = \frac{1}{2\pi E} \int_{-\infty}^{+\infty} \omega^2 |F(\omega)|^2 d\omega \tag{4.5}$$

and

$$d_p^2 = \frac{1}{E} \int_{-\infty}^{+\infty} t^2 |f(t)|^2 dt \tag{4.6}$$

$F(\omega)$ is the Fourier transform of the time domain pulse $f(t)$ and E is the pulse energy calculated by

$$E = \int_{-\infty}^{+\infty} |f(t)|^2 dt = \frac{1}{2\pi} \int_{-\infty}^{+\infty} |F(\omega)|^2 d\omega \tag{4.7}$$

We consider five different time domain pulses: *sinc*, *square*, 2^{nd} *order filtered*, *root-raised cosine* [97, 98] and *Gaussian* pulse. Their performance metrics are reported in Table 4.2. The 2^{nd} *order filtered* pulse is a square pulse filtered by a 2^{nd} order low-pass filter.

The *sinc* and *root-raised cosine* pulses have the highest spectral efficiencies but require the most complex transmitters to be generated. The *square* pulse is the simplest to generate but it results in the highest out-of-band emissions. The *Gaussian* pulse has the lowest time-bandwidth product, that is why it is typically preferred and the most common pulse shape found in the literature. The 2^{nd} *order filtered* pulse performs similarly to the *Gaussian* pulse, but it requires area and power consuming filters to be generated.

	Spectral Efficiency	Out-of-Band Emissions	Time-BW Product
Sinc	100% (0dB)	0% ($-\infty$dB)	∞
Square	60.0% (-2.2dB)	12.8% (-8.9dB)	∞
2^{nd} order filtered	59.2% (-2.3dB)	2.8% (-15.6dB)	0.55
Root-raised cosine	84.6% (-0.7dB)	0.4% (-23.8dB)	0.85
Gaussian	56.5% (-2.5dB)	3.3% (-14.9dB)	0.50

Table 4.2: Comparison of different pulse shapes

Following these considerations, we chose the Gaussian pulse shape for our application. However, as this pulse is a relatively complex pulse shape to generate with circuits, we will see in Chap.5 how a good approximation with almost the same performance can be obtained with a very simple circuit.

4.3.4 Channel Model

A fundamental aspect to be taken into account in the design of our transceiver is the channel model. The equation for the path loss of a UWB signal is given by [99, 100]:

$$L(d) = L_0 + 10 \cdot \gamma \cdot \log_{10}(d/d_0) + S \tag{4.8}$$

where

$$L_0 = 10 \cdot \gamma \cdot \log_{10}(\frac{4\pi d_0 f_c}{c}) \tag{4.9}$$

Usually the reference distance d_0 is chosen as 1m. f_c is the signal central frequency, c the light speed in m/sec and S represents the shadowing factor of the channel, which is a zero-mean Gaussian random variable which indicates the deviation of $L(d)$ from its nominal value. Finally, γ represents the severity of the path loss. For line-of-sight, γ is measured to be 2 but can grow as large as 3.34 for non-line-of-sight measurements.

The system level paramiters for our project are summarized in Table 4.3.

4.4 CMOS Technology

Both receiver and transmitter are designed in the UMC 0.13-μm Mixed-Mode and RFC-MOS process. The key features of this technology are:

Modulation	2-PPM
Pulse bandwidth	$7.25 - 8.5$GHz
Pulse shape	Gaussian
Data rate	100kb/s
Link distance	$d \geq 10$m

Table 4.3: System level parameters

- minimum channel length: 0.13-μm;
- dual supply voltage: 1.2V and 3.3V;
- P-substrate;
- single poly, eight metal layers (1P8M);
- Twin-Well and Triple Well;
- Metal Metal capacitors;
- high performance mixed-mode signal capabilities;
- radio frequency MOS transistors.

4.5 Receiver

As already outlined in Sec.4.2, the non-coherent energy-detection approach [101] in the receiver may be preferred because it allows to avoid the integration of both a template pulse generator [102, 103] and a quadrature frequency synthesizer [104]. In such a non-coherent case, the received energy has to be estimated, performing a windowed integration on the received signal squared. The block diagram of a non-coherent receiver is reported in Fig.4.6. It consists of a low-noise amplifier (LNA), a variable-gain amplifier (VGA) to accommodate variations of the received signal strength and an energy detector, composed of a squarer and a windowed integrator. For each received bit, the integral of the received signal squared is computed separately over the time windows T_{int1} and T_{int2} so that a comparator allows to decide whether more energy is allocated in the first or in the second window.

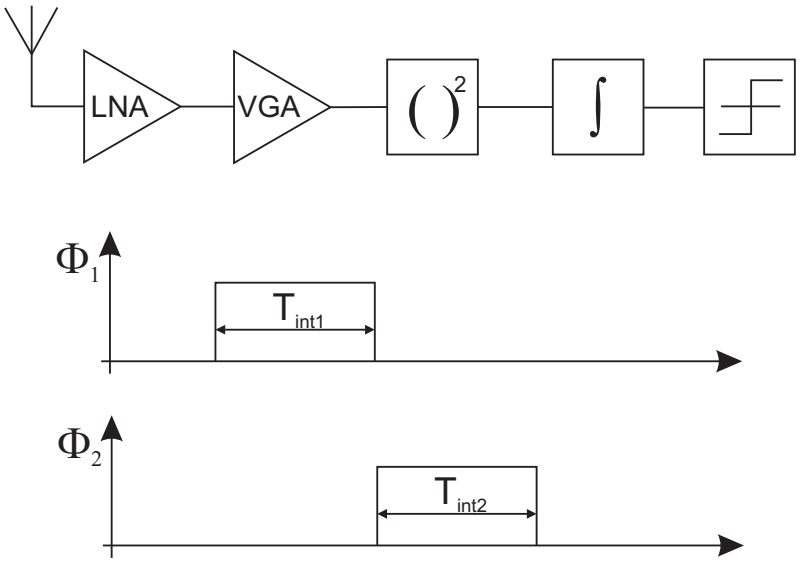


Figure 4.6: Receiver block diagram

The receiver has been designed in UMC 0.13-μm technology by Andrea Gerosa and Marco D'Aguanno and it is now under test.

The main features of each block are summarized hereafter.

LNA and VGA The LNA is an inductively degenerated common-source amplifier with a resonant load. It exploits fully differential conversion with embedded impedance matching by means of a monolithic integrated transformer.

The VGA has been designed with two stacked stages that exploit the same bias current, in order to minimize its power consumption. It also has resonant loads with capacitor tuning to adjust the tank center frequency.

The LNA and VGA parameters are reported in Table 4.4.

Energy detector The energy detector squarer exploits the quadratic non-linearity of MOS devices working under strong inversion region, as described in [105], to obtain an output current proportional to the squared input voltage. The integrator, realized by means of a transimpedance amplifier, integrates the squared signal over a capacitor. Two capacitors are used to estimate the received energy in two consecutive time slots.

The overall energy detector main parameters are reported in Table 4.5.

Parameter	*Value*
LNA gain	14dB
LNA noise figure	5dB
LNA *iIP*3	-10dBm
LNA bias current	500μA
LNA + VGA gain	$15 \div 30$dB
LNA + VGA noise figure	5.2dB
LNA + VGA *iIP*3	-10dBm
LNA + VGA bias current	1mA
LNA + VGA area	$800\mu\text{m} \times 500\mu\text{m}$

Table 4.4: LNA + VGA parameters

Parameter	*Value*
Energy detector conversion gain	$8.5\text{mA}/\text{V}^2$
Current consumption	1.4mA

Table 4.5: Energy detector parameters

4.5.1 Behavioral Model

In order to evaluate the performance of the receiver at system level so as to derive the transmitter specifications, a behavioral model of the whole receiver has been realized using Matlab. The block diagram of the model is shown in Fig.4.7. Signal V_{RX} is generated modulating an ideal Gaussian pulse at a given energy with a random bitstream and adding the thermal noise at the antenna. The blocks in the first row of Fig.4.7 model the two gain stages. The circuit noise due to the LNA and the VGA is added to signal V_{RX} as a white Gaussian noise, whose power depends on the amplifying stages noise figure that has been estimated by means of transistor-level simulations. A third-order non linearity is also accounted for, modeling the LNA transfer function with a third-order power series, extrapolating coefficient b_3 from the input-referred intercept point *iIP*3. Finally, three bi-quadratic filters, whose pass-band corresponds to the pulse bandwidth, model the inherent frequency selectivity of the LNA input matching network and of the resonant loads in the LNA and VGA.

The squarer is modeled using a power series to account for its non linear relationship

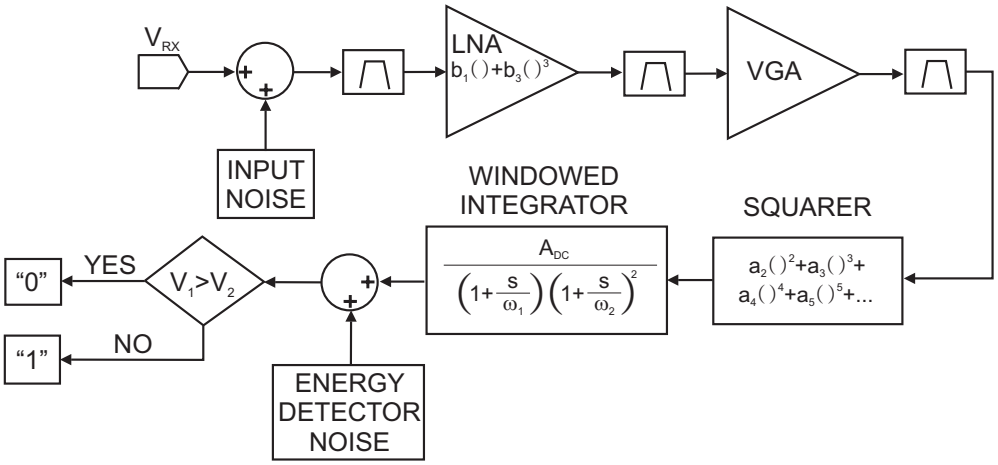


Figure 4.7: Matlab receiver equivalent model

between the input voltage and the output current. The power series is given by

$$I_{out} = \sum_{i=0}^{N} a_i \cdot V_{in}^i \tag{4.10}$$

where the a_i coefficients have been extracted from transistor-level simulations. In particular, a classical two-tone test with input signal components spaced by 100MHz, namely at 8GHz and 8.1GHz, has been performed. The magnitude of the different harmonic components allows to estimate the coefficients a_i values which are reported in Table 4.6. Similarly, the integrator model mimics its transistor-level frequency responce.

The circuit noise due to the squarer and to the integrator has been estimated with transistor-level simulations as well, and it is added as an equivalent noise source at the integrator output.

The integration results at the end of the two integrating phases Φ_1 and Φ_2 of Fig.4.6 are sampled and held before being compared to decide which of the two windows contains more energy. It is worth to notice that the integrator output voltage is clipped at 500mV, in order to account for limited output swing of the real circuit.

4.5.2 BER and Sensitivity

In order to quantify the receiver performance, the uncoded BER is estimated as a function of the input power at the antenna, as reported in Fig.4.8. As shown, the minimum input power for which a BER lower than 10^{-3} is estimated, is $\simeq -95$dBm. The receiver performance are summarized in Table 4.7, where a process gain $G_p = 5$dB has been accounted

Parameter	*Value*
Data rate	100kb/s
Pulses per bit	10
LNA + VGA gain	6dB - 28dB
LNA + VGA noise figure	5.2dB
LNA + VGA *iIP*3	-10dBm
a_1, a_2, a_4	$1.6 \cdot 10^{-4}$, $8.83 \cdot 10^{-3}$, -0.15
A_{DC}	233dBΩ
ω_1	3Hz
ω_2	1.9GHz
Energy detector noise power	$5.11 \cdot 10^{-6}\mathrm{V}^2$

Table 4.6: Parameters of the receiver behavioral model

for, deriving from the transmission of 10 pulses per bit and the use of a *by majority* decision rule, as described in details in Sec.5.1.

Although the expected link margin of 6.8dB would allow a link distance greater than 10m, as low-data rate transmissions are peak power limited, we will see in Sec.5.1 how we need to reduce the average transmitted power to -17.7dBm in order to be compliant with the FCC limits. This leads to a link margin of $\simeq 2$ dB.

4.6 Synchronization Algorithm

In order to maintain the low complexity nature of the receiver, we decided to use a synchronization algorithm based on the energy collection strategy to synchronize the receiver and the transmitter clocks before demodulation, as the one described in [106]. The algorithm is based on a preamble which contains 4 repetition of a $N_c = 31$ bits Gold code sent at the maximum repetition frequency $PRF_{syn} = 1/(2 \cdot T_{int})$, followed by an inverted Gold code sequence used to indicate the end of the synchronization phase. The structure of each data packet is reported in Fig.4.9. During the synchronization phase, the analog front end just integrates and then compares the energy received in consecutive time slots of duration equal to T_{int}. The data generated is then parallelized into two data streams sent to two identical correlators banks, as shown in Fig.4.10. The 31 correlators of each bank correlate the received data with shifted version of the 31 bit Gold code to account

Parameter	*Value*
Throughput (R_b)	100 kb/s
Pulses per bit (N_b)	10
Bandwidth (B)	1.25GHz
FCC Limit (P_{FCC})	-41.3dBm/MHz
Maximum TX Power $(P_{max} = P_{FCC} + 10\log_{10}(B/1\text{MHz})$	-10.33dBm
Gaussian pulse spectral efficiency (E)	-2.5dB
Average TX Power $(P_T = P_{max} + E)$	-12.83dBm
Path Loss @1m (L_1)	50.37dB
Path Loss @10m (L_2)	20dB
RX Power $(P_R = P_T - L_1 - L_2)$	-83.2dBm
Process Gain $(G_p = 10\log_{10}(\sqrt{N_b}))$	5dB
Average noise power $(N = -174 + 10\log_{10}(R_b N_b))$	-114dBm
RX noise figure (N_f)	12dB
Total noise power $(P_N = N + N_f)$	-102dBm
Minimum E_b/N_0 for 10^{-3} BER (S)	17dB
Link margin $M = P_R + G - P_N - S$	6.8dB

Table 4.7: Parameters of the receiver behavioral model

for all the possible time differences between transmitter and receiver. Synchronization is declared if any, but only one, of the accumulator exceeds a programmed threshold.

The drawback of a such a simplicity is that the synchronization time resolution is equal to the integration window duration T_{int} itself. In fact, different synchronization strategies have been developed to achieve higher synchronization time resolution, but at the price of an increased receiver complexity [107] or of a long locking time [108].

In most cases, synchronization is declared such as the pulse to be detected lies completely within a single integration window, as sketched in Fig.4.11.a. However, it may happen that the receiver is synchronized in a way such that a fraction of the pulse falls outside the correct integration window, as shown in Fig.4.11.b. The latter event impairs the detection capability of the receiver, because part of the signal energy is not accounted for in the integration result. Due to the mentioned finite time resolution of the algorithm, the only way to preserve the receiver performance is to make the probability of this event negligible, using a window duration T_{int} sufficiently larger than the pulse duration. This

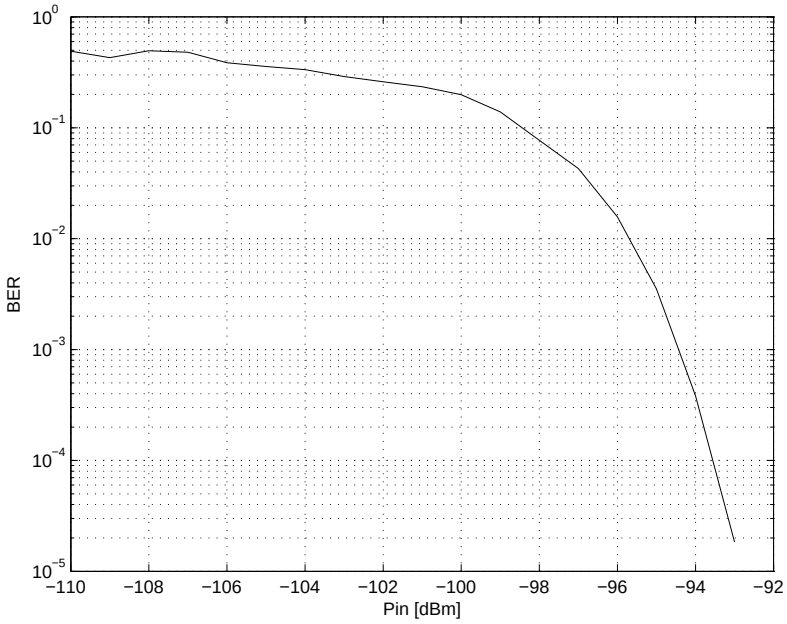


Figure 4.8: Simulated BER as a function of the input signal power at the antenna, using the model parameters given in Table 4.6

31 bit Gold Code	31 bit Gold Code	31 bit Gold Code	31 bit Gold Code	Inverted Gold Code	Header	1024 bit Payload

Figure 4.9: Data packet structure

has also the advantage of reducing the clock frequency of the correlators, as synchronization data stream is sent at a frequency equal to $2 \cdot PRF_{syn} = 1/T_{int}$. On the other hand, a large integration window would worsen the receiver performance in terms of sensitivity, as shown in Fig.4.12 where the receiver BER is simulated for different T_{int} values. In fact, larger integration windows reduce the SNR for the same signal power at the antenna [109], due to both a larger noise energy and to the finite output resistance of the integrator. Following these considerations we singled out $T_{int} = 15$ns as a good design compromise.

The performance of the synchronization algorithm in terms of probability of detection P_d, that is the probability that synchronization is declared when the input and the Gold code are aligned, is reported in Fig.4.13. Such a probability is always larger than 0.9 that is generally considered a reasonable benchmark [110]. The probability of false acquisition

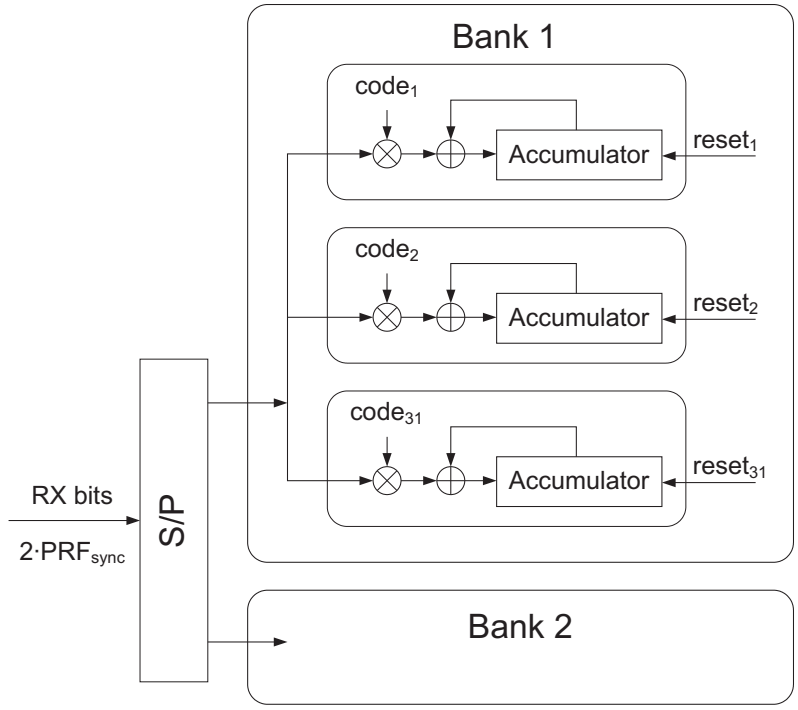


Figure 4.10: Block diagram of synchronization correlator banks

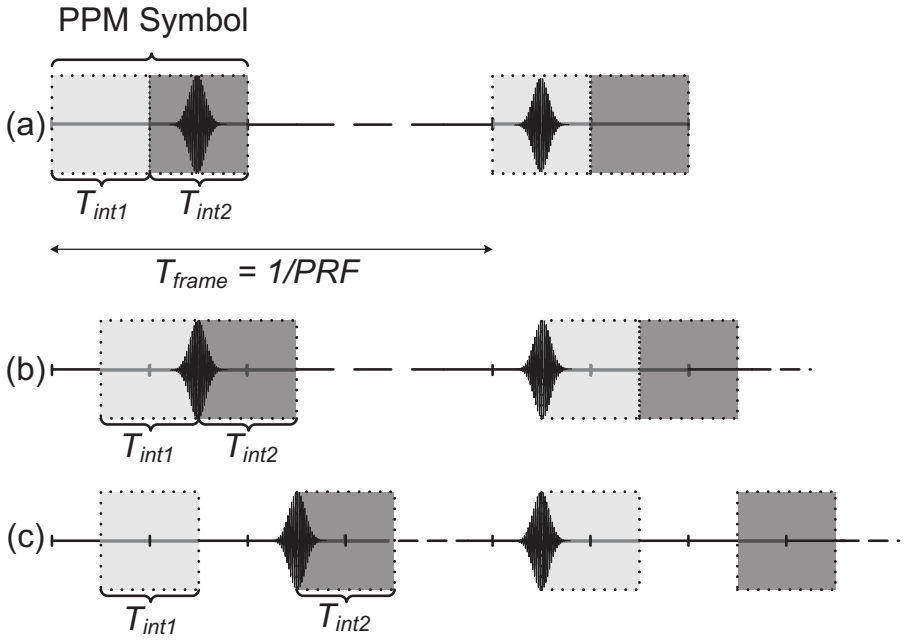


Figure 4.11: PPM signaling scheme with: (a) perfect synchronization; (b) half T_{int} misalignment; (c) modified PPM scheme with half T_{int} misalignment

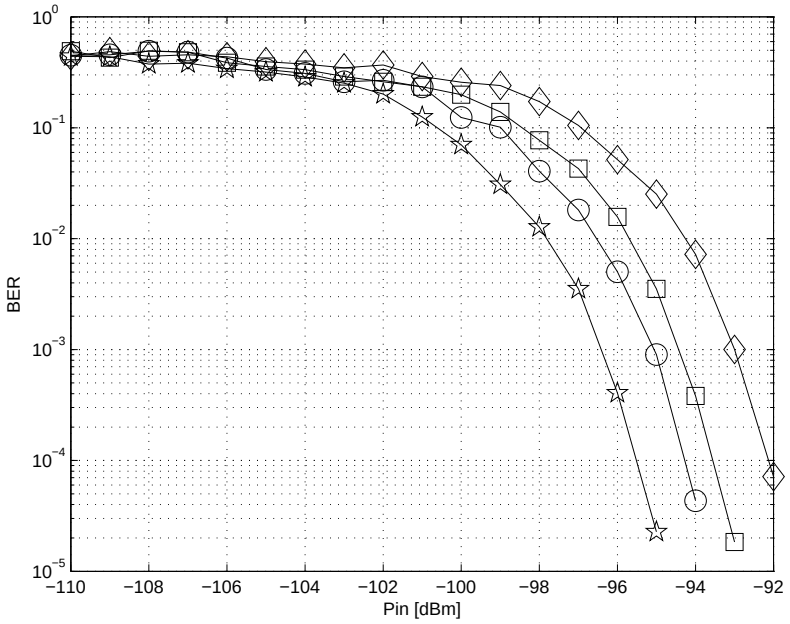


Figure 4.12: Simulated BER as a function of the input signal power at the antenna with $T_{int} = 5$ns (star), $T_{int} = 10$ns (circle), $T_{int} = 15$ns (square) and $T_{int} = 30$ns (diamond)

P_{fa}, that is the probability that synchronization is declared when the input and the Gold code are misaligned, is always less than 10^{-7}.

Fig.4.14 compares the BER in case of perfect synchronization (as in Fig.4.8) with the one obtained including the synchronization phase, assuming a payload size equal to 1024 bits. Almost no power loss can be observed at the sensitivity; however the BER curve shows a floor at 10^{-4}.

The floor is caused by the residual probability of the event illustrated in Fig.4.11.b. In particular, whenever the integration window misalignment is such that the received pulse is split across the two PPM windows, the noise influence on the demodulation result dominates regardless the signal power. This generates a BER floor above that input signal power for which this misalignment becomes the most relevant error source. A way to avoid this eventuality, is to insert a time slot $T_{int} = 15$ns between the two PPM windows, as shown in Fig.4.11.c. As a consequence, the BER curve does not exhibit any floor, as reported in Fig.4.14.

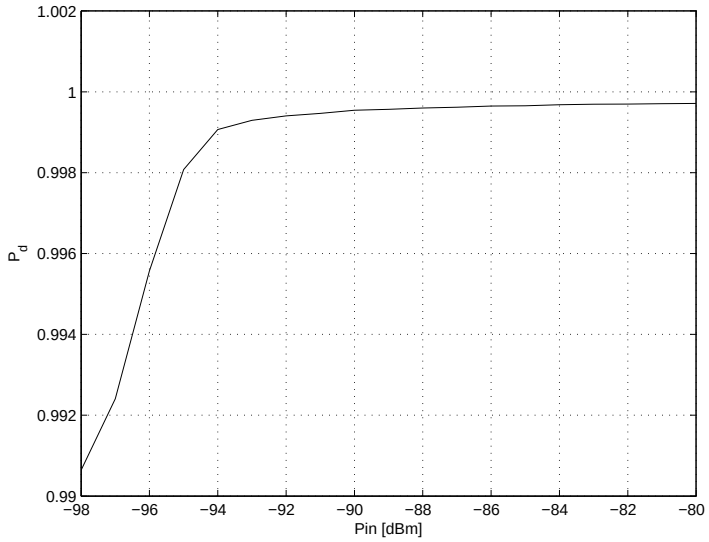


Figure 4.13: Probability of detection

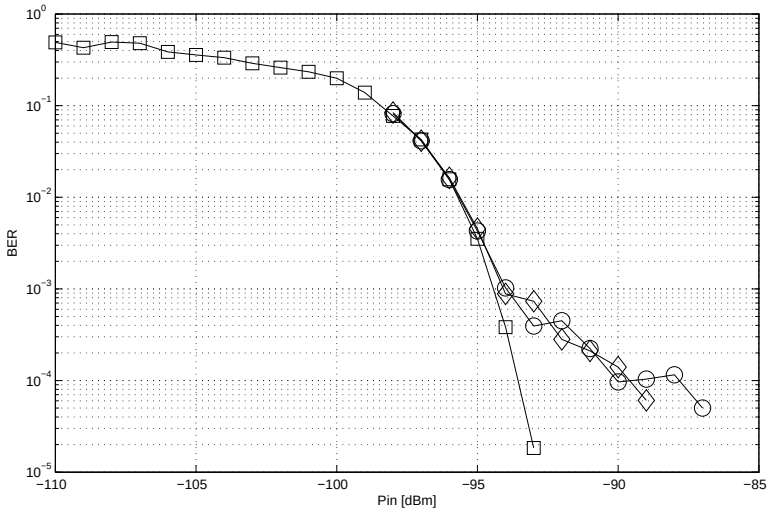


Figure 4.14: Simulated BER as a function of the input signal power at the antenna with perfect synchronization (square), original PPM synchronization (circle) and modified PPM synchronization (diamond)

5

Transmitter

This chapter presents the design of the transmitter for the UWB-IR low-data rate wireless sensor network described in Chap.4. In particular, the transmitter specifications are derived first; then, a possible architecture for the overall system is presented to focus on the design of a novel energy efficient Gaussian pulse generator.

5.1 Specifications

As already outlined in Sec.4.1, the FCC limits the output power in the 3.1-to-10.6GHz band in two ways [64, 65]:

1. The *average* power spectral density must be less or equal to -41.3dBm. This corresponds to a theoretical maximum total power of -10.3dBm for a 1.25GHz bandwidth signal. In practice, this number is reduced by $2-4$dB due to pulse generation constraints.

2. The *peak* power may not exceed 0dBm at the UWB signal center frequency f_c in a 50MHz resolution bandwidth (RBW). Since most spectrum analyzers are not equipped with a 50MHz IF filter, the peak power measurement is typically performed at a lower RBW and the limit is conservatively set to be

$$P_{pk} \leq 0\text{dBm} + 20\log_{10}(RBW/50\text{MHz})$$

Communication distance in a non-coherent energy-detecting UWB system is maximized when the SNR seen at the receiver during the integration window is maximized. This occurs when the transmitter generates maximum total output power under the regulatory limits. Since sensor networks typically communicate at low data rates, large amplitude pulses transmitted at the data rate are required to maximize power, and thus

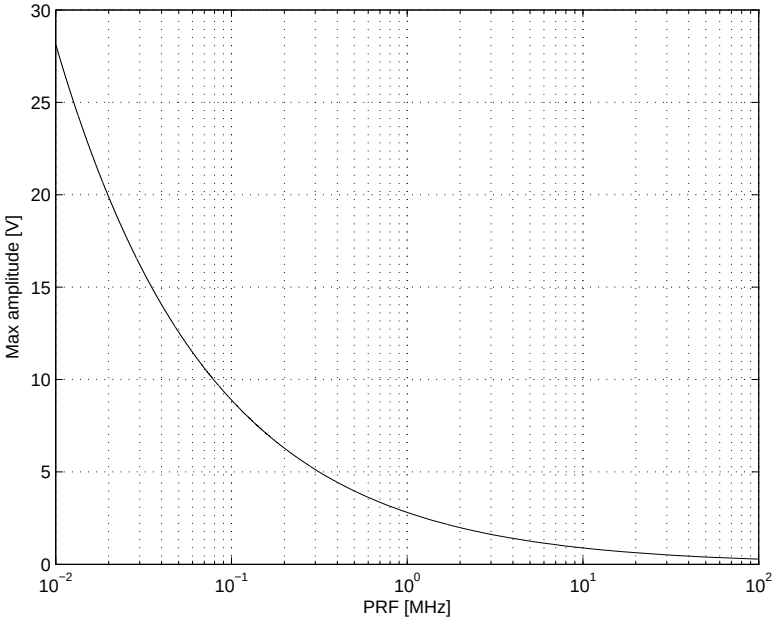


Figure 5.1: Maximum single pulse amplitude allowed by the FCC average power spectral mask as a function of the PRF

communication distance, under FCC spectral masks. The voltage amplitude V_{max} for single pulses transmitted at a data rate equal to PRF are reported in Fig.5.1. We can see as a peak-to-peak voltage swing of $17.8V_{pp}$ would be required to maximally satisfy FCC average power spectral mask at a PRF of 100kHz.

On the other hand, while high data rate pulsed-UWB transmitters are typically average power limited, low data rate transmitters are typically peak power limited [111], as shown in Fig.5.2 where the peak power corresponding to the transmission of a single pulse as a function of the pulse amplitude V_{max} together with the FCC limit is reported. For the peak power estimation a spectrum analyzer resolution bandwidth of 3MHz has been considered.

The peak power as a function of the pulse voltage amplitude for a data rate transmission of 100kb/s is reported in Fig.5.3. We can see how a maximum peak-to-peak voltage swing of $9V_{pp}$, corresponding to a $V_{max} = 4.5$V is allowed to be compliant with the FCC mask limit. However, this is still impractical for deep-submicron CMOS technologies, where supply voltages are of the order of 1V.

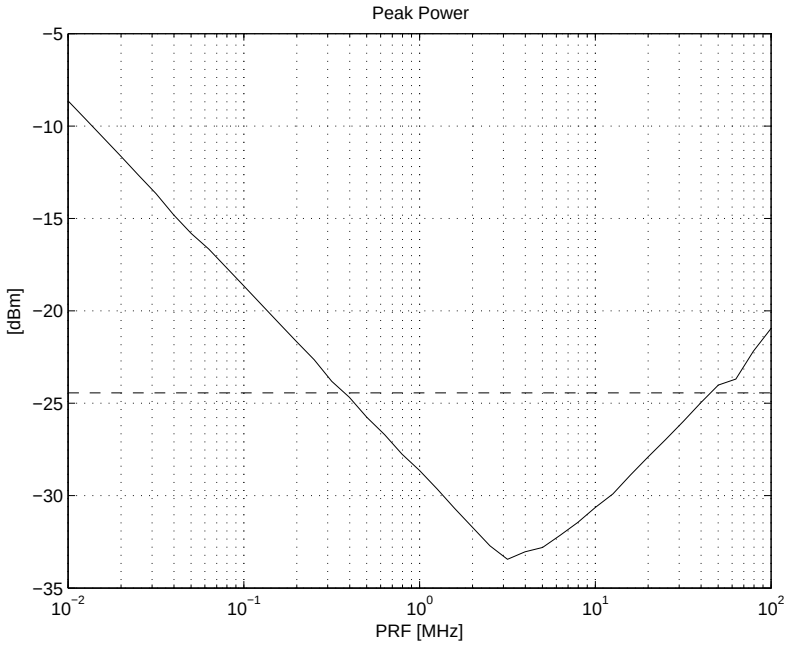


Figure 5.2: Peak power of a maximum amplitude single pulse as a function of the PRF

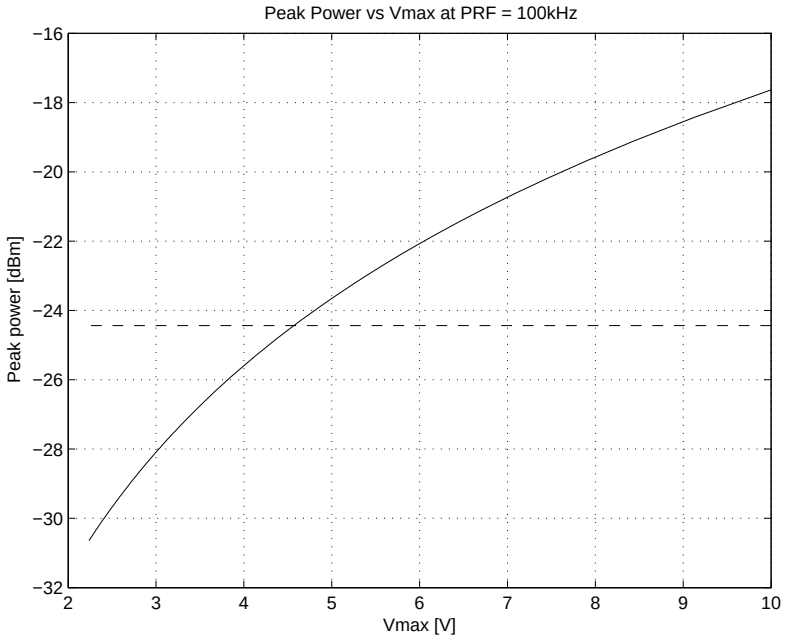


Figure 5.3: Peak power as a function of the single pulse voltage amplitude for a 100kb/s data rate transmission

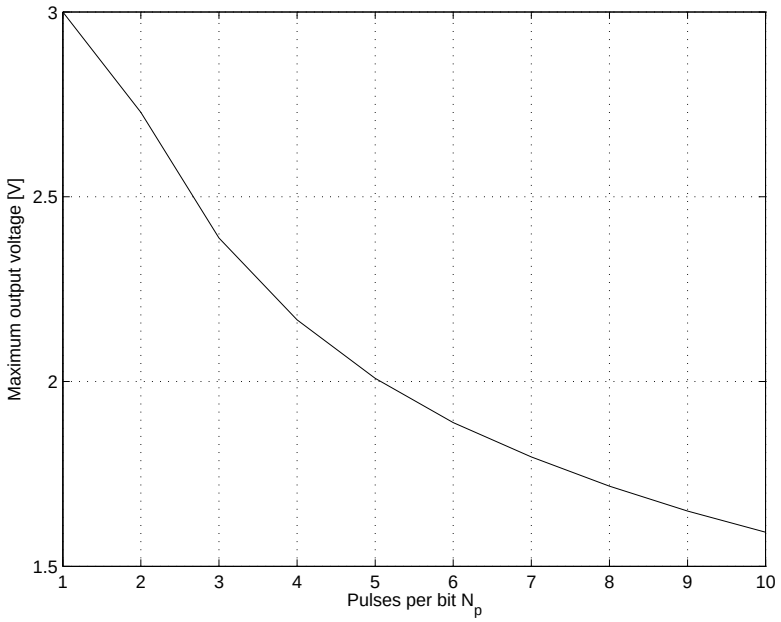


Figure 5.4: Maximum output voltage as a function of the pulses number per bit

An alternative approach to generate large swing pulses while maximizing total power under FCC masks is to reduce output voltage swings and increase the PRF, that is equivalent to transmit multiple pulses per bit. As the transmitter is to be implemented in the UMC 0.13-μm CMOS process described in Sec.4.4 with a maximum supply voltage of 1.2V, and monolithic transformer with transformer ratio greater than 3 show poor performance, we fixed as a reasonable limit for the maximum output swing $\simeq$ 3V. Then we calculated the link distance as a function of the number of pulses per transmitted bit with the receiver parameters reported in Sec.4.5.2.

The maximum output voltage depends not only on technology limits but also on the FCC average power mask, as during the synchronization phase pulses are transmitted at a maximum $PRF_{syn} = 1/(2 \cdot T_{int})$ and the average power is measured on a data packet basis, according to [64]. Increasing the output swing leads to a reduction of the pulse repetition frequency during synchronization phase PRF_{syn}, thus protracting the time required to synchronize transmitter and receiver. This has a negative impact on the energy per bit of the overall system, as during the synchronization phase the receiver is always switched on.

As pulses with a smaller amplitude can be more efficiently generated than large voltage swing ones, we decided to transmit more pulses per bit instead of a single pulse with

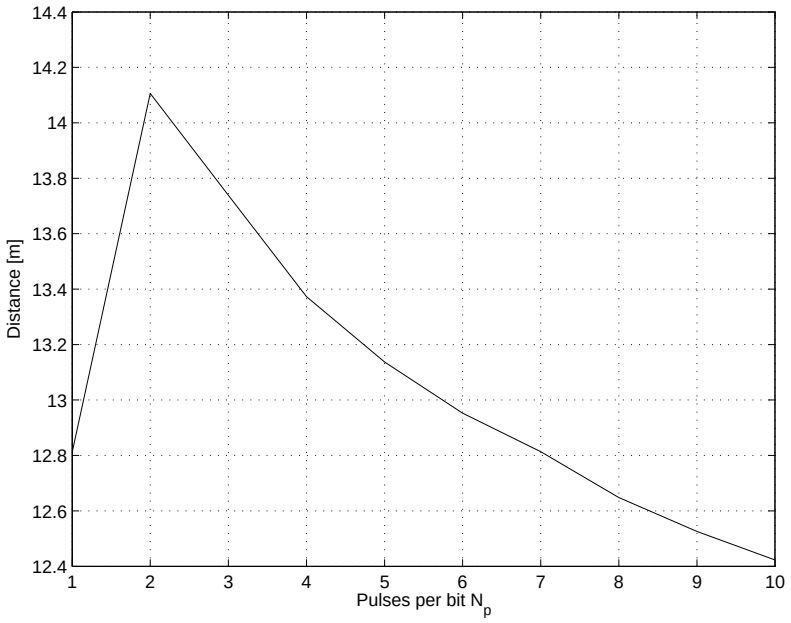


Figure 5.5: Link distance as a function of the pulses number per bit with the pulses amplitude given by Fig.5.4

a larger amplitude. We can see from Fig.5.4 and Fig.5.5 that by transmitting $N_b = 10$ pulses per bit with a pulse output swing of $V_{max} = 1.6$V, that is $3.2V_{pp}$, we can reach a link distance of 12.4m, which corresponds to a link margin of $\simeq 2$dB.

However, due to FCC peak power limits, the pulse repetition frequency during synchronization PRF_{syn} must be reduced to $1/(2 \cdot 3 \cdot T_{int}) = 11.1$MHz instead of 33.3MHz. This has no impact on the BER performance but it slightly increases the receiver hardware complexity as two banks of 6 correlators each, instead of 2, have to be implemented.

The estimated energy per pulse, that is the energy spent to transmit a single pulse, is shown in Fig.5.6, together with the energy per bit of the overall system, including the amount of power spent during the synchronization phase. In particular, for a transmission of 10 pulses per bit, the estimated energy per bit is 3.2nJ. This shows better performance than the result reported in [106], where an energy/bit of 2.5nJ over a link distance of 3m is declared, without including the energy spent to synchronize the receiver and the transmitter.

Following these considerations, we can derive the transmitter specifications, which are summarized in Table 5.1.

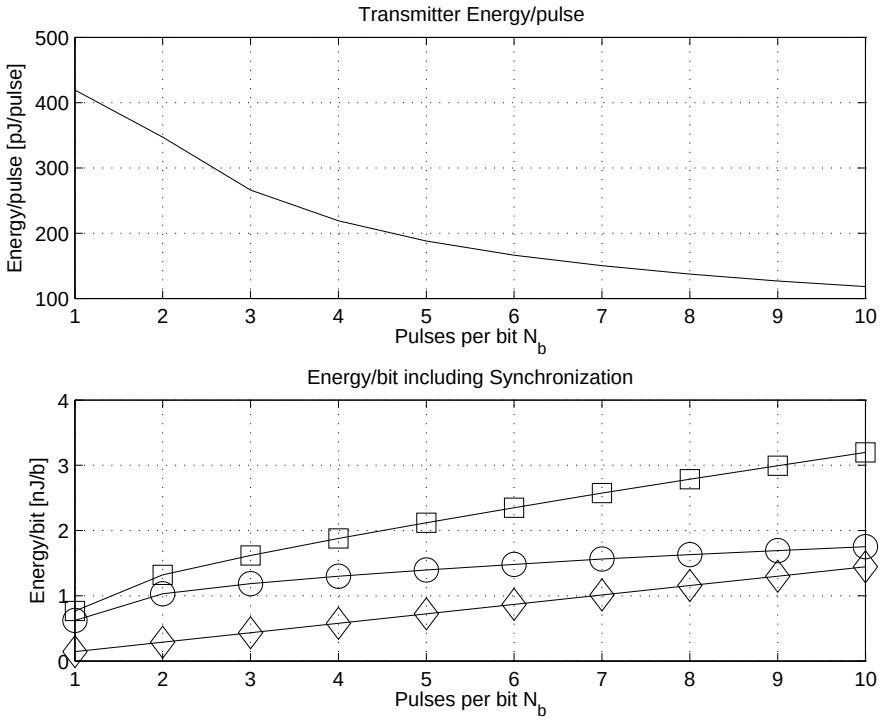


Figure 5.6: Energy/pulse and receiver (diamond), transmitter (circle) and overall (square) energy/bit as a function of the number of pulses per bit

Parameter	*Value*
Throughput (R_b)	100 kb/s
Pulses per bit (N_b)	10
Bandwidth (B)	1.25GHz
Carrier frequency (f_c)	7.875GHz
Maximum output voltage (V_{max})	1.6V
Energy per pulse (E_p)	$\leq$ 120pJ/pulse

Table 5.1: Main transmitter parameters

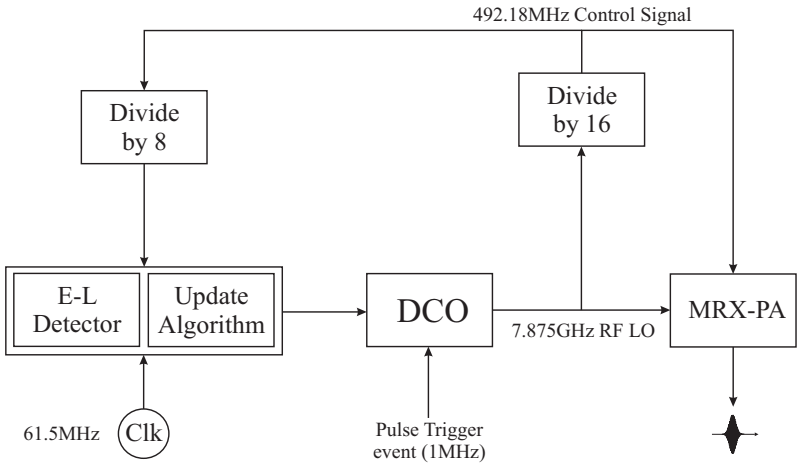


Figure 5.7: Transmitter block diagram

5.2 Architecture

The transmitter block diagram is shown in Fig.5.7. It consists of a digital controlled oscillator (DCO) which generates the carrier frequency $f_c = 7.875$GHz and a frequency divider by 16, which generates the control signal V_{RF} for the combined mixer and power amplifier (MXR-PA). The MXR-PA receives at its input the differential carrier frequency signal V_{LO} and the pulse control signal V_{RF} and produces at its output a Gaussian pulse with central frequency $f_c = 7.875$GHz and bandwidth $B = 1.25$GHz. The carrier frequency f_c accuracy is controlled by means of a phase-aligned frequency-locked loop (PA-FLL). The control signal V_{RF} is further divided by 8 and compared with an external reference clock at 61.5MHz by means of an early-late detector. A binary search algorithm is then implemented to adjust the carrier frequency f_c. However, it is worth to notice how a non-coherent signaling scheme does not require precise frequency tuning. Thus, the control system proposed reaches the required accuracy with a relative implementation simplicity. To exploit the low duty cycle nature of UWB-IR systems, the transmitter is activated at each pulse transmission by means of an external control signal. As we transmit a number of 10 pulses per bit, the pulses are sent at a repetition frequency of 1MHz, that is 10 times the nominal data rate of 100kb/s.

As the frequency accuracy requirement can be relaxed in non-coherent energy detecting systems, the VCO is implemented by means of a three-stage ring oscillator in order to ensure a fast start-up. However, its design is still at the preliminary stage, as the one of PA-FLL loop, and will not be discussed further. In the next section, we will thus describe

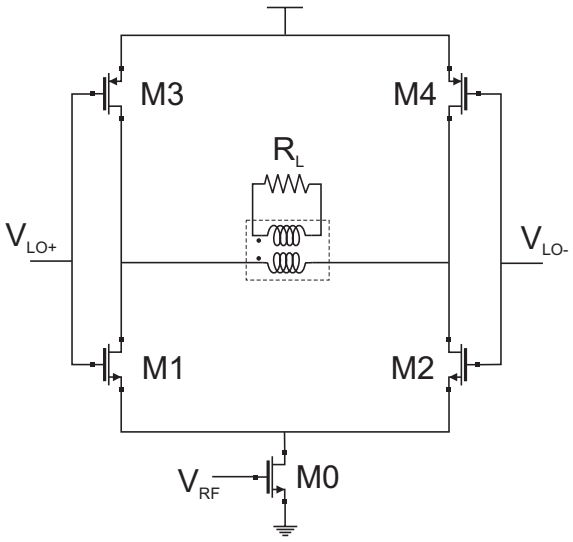


Figure 5.8: Gaussian pulse generator

in details the design of the Gaussian pulse generator.

5.3 Gaussian Pulse Generator

The variety of Gaussian pulse generators present in literature is very wide, ranging from all-digital CMOS pulse generators [112–114] to Gilbert cell BiCMOS based solutions [111]. However, the former reach a maximum output swing which is of the order of hundreds of millivolts, while the latter usually require discrete low-pass filters to meet the FCC mask requirements. We will see how a Gaussian pulse can be efficiently generated by the circuit shown in Fig.5.8 that simultaneously permforms up-conversion to the central frequency $f_c = 7.875$GHz and mixing.

The basic idea is to generate a Gaussian pulse by multiplying a triangular shape pulse and the differential local oscillator output signal. In order to generate the required peak-to-peak output swing of $3.2V_{pp}$ and perform the differential to single ended conversion, a monolithic transformer with transformer ratio $r = 2$ is used.

The nMOS transistor M0 of Fig.5.8 is driven by a triangular pulse V_{TR} while transistors M1, M2, M3 and M4 act as switches driven by the local oscillator output signals V_{LO+} and V_{LO-}. To have an idea of the design parameters values, we can consider the small signal analysis model of our circuit, even if the triangular pulse driving the bias transistor is a large signal. If we assume V_{LO+} and V_{LO-} to be square waves, the conversion gain is

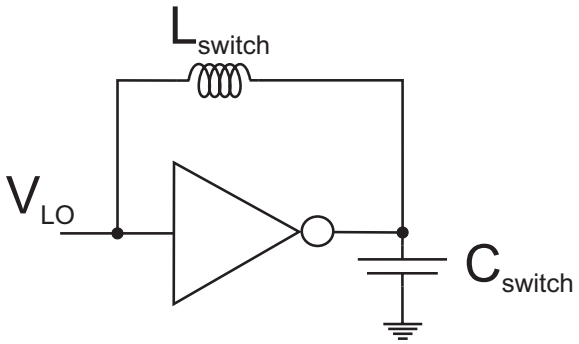


Figure 5.9: Simplified buffer schematic

given by

$$\frac{2}{\pi} \cdot g_{m0} \cdot \frac{R_L}{r^2} \tag{5.1}$$

where $R_L = 50\Omega$ is the antenna resistance and r the transformer ratio. If the local oscillator output waves are not square but sinusoidal, the pMOS and nMOS switches will be on simultaneously for a certain amount of time, giving rise to a conversion gain loss.

The triangular pulse V_{TR} has a slope of 1.2V/ns as it has to rise from 0V to $V_{dd} = 1.2$V in 1ns. However, the accuracy requirements for the V_{TR} generation are very relaxed, as the output signal spectrum does not show any significant variation due to imprecise triangular pulse generation. As a consequence, V_{TR} can be easily generated from the square control signal V_{RF} by means of a cascade of two inverters.

It is worth to notice how the proposed circuit does not need any bias current, thus greatly improving the overall system efficiency.

However, in order to reduce the gain loss, we need to keep the switches resistance quite low. In particular, if we assume a minimum drain-source voltage of $\simeq 200$mV for M0, so as to keep it always working in saturation, as the maximum transformer primary coil current is equal to

$$I_{max} \simeq \frac{r \cdot V_{max}}{R_L} = 64\text{mA}$$

the switches resistance as to be in the order of few Ohms. This leads to very large nMOS and pMOS transistors with a gate capacitance of the order of pFs. In order to drive such a capacitance, we use an inductor as feedback network for the switches buffers, so as to resonate the gate capacitance, as shown in Fig.5.9. The buffers are realized by means of an inverter. In order to avoid DC power consumption, both buffers are activated by means of a switch in series with the inverters nMOS transistors.

MXR-PA		
$M0$	$\frac{32}{0.12}$	14
$M1$	$\frac{32}{0.12}$	14
$M3$	$\frac{32}{0.12}$	28

Table 5.2: MXR-PA transistor size in μm/μm

Buffer		
M_N	$\frac{4.8}{0.12}$	24
M_P	$\frac{1.8}{0.12}$	24
Passive Elem.		
C_{switch}	1.2pF	
L_{switch}	340pH	

Table 5.3: Passive elements values and buffer transistor size in μm/μm

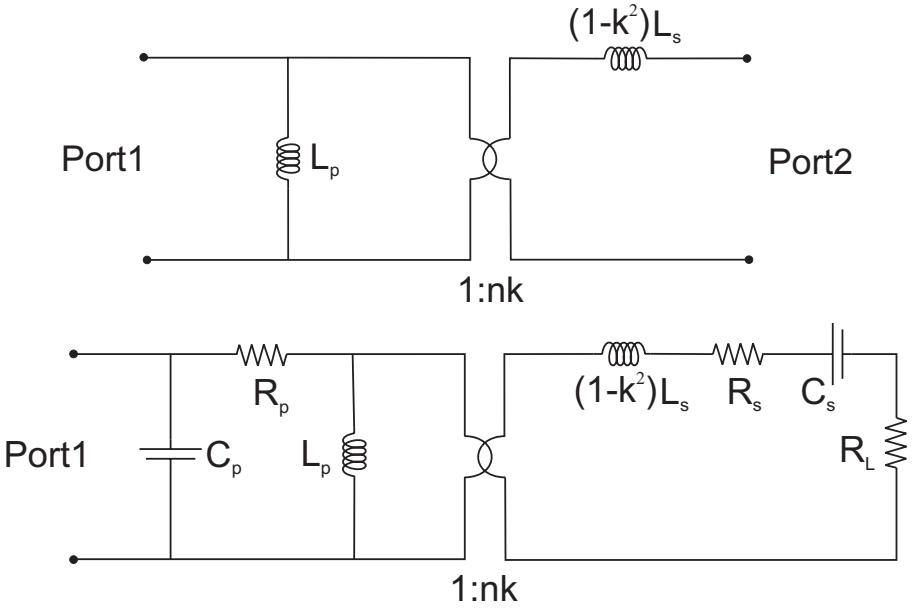


Figure 5.10: Transformer equivalent models

The transistor size for the Gaussian pulse generator are reported in Table 5.3, while those for the buffers together with the switches gate capacitance and the feedback inductor values are reported in Table 5.3. The third column of both tables indicates how many transistors are connected in parallel.

5.3.1 Transformer Design

As already outlined, a transformer with transformer ratio $r = 2$ is needed in order to generate the required transmitter output power. We decided to implement it by means of an on-chip monolithic transformer, created by magnetically coupling two inductors.

As the transformer is used to achieve output matching in our power amplifier, it will be necessary to resonate some of the transformer inductance to minimize the loss [115]. A capacitor is also necessary at the primary side of the transformer to adjust its input reactance to the desired value for the driving transistors. This can be done using a parallel capacitor on the primary and another capacitor in series with the secondary, as shown in Fig.5.10, where a simplified transformer model is reported, together with the series and parallel tuning capacitance C_s and C_p. In the model, k indicates the coupling factor, n the turn ratio between primary and secondary coils, R_L the load resistance and R_p and R_s the primary and secondary inductor series resistances. R_p and R_s can be calculated from the inductors quality factors Q_p and Q_s as:

$$R_p = \frac{\omega L_p}{Q_p} \qquad R_s = \frac{\omega L_s}{Q_s} \tag{5.2}$$

The two transformer inductors, combined with two capacitors, can be used not only to achieve output matching but also to obtain a fourth-order bandpass ladder filter, so as to reduce the power amplifier out-of-band emissions. Thus, the four reactive elements values will be chosen so as to ensure a good efficiency and at the same time a filtering effect.

Given a load resistance of 50Ω, we want to determine the capacitors and inductors values that allow to maximize the transformer efficiency η. This is defined as the ratio of the power delivered to the load P_{load} to the total power delivered into PORT1 of the network, that is $P_{total} = P_{diss} + P_{load}$. It can be shown that the transmitter efficiency η is equal to

$$\begin{aligned} \eta &= \frac{P_{load}}{P_{diss} + P_{load}} \\ &= \frac{R_L/n^2}{\frac{R_L}{n^2} + \frac{\omega L_p}{Q_s} + \frac{\omega L_p}{Q_p} \cdot \left(\frac{R_L/n^2 + \omega L_p/Q_s}{k\omega L_p} \right)^2} \end{aligned} \tag{5.3}$$

where we assume $L_s \simeq n^2 L_p$ and we use for C_s the value given by

$$C_s = \frac{1}{\omega^2 L_s} \tag{5.4}$$

as it allows to cancel one of the terms at the denominator of (5.3). By differentiating equation (5.3), we can obtain the optimum value of L_p resulting in the highest possible η, which is:

$$\omega L_p = \frac{R_L}{n^2 \sqrt{\frac{1}{Q_s^2} + \frac{Q_p}{Q_s} \cdot k^2}} \tag{5.5}$$

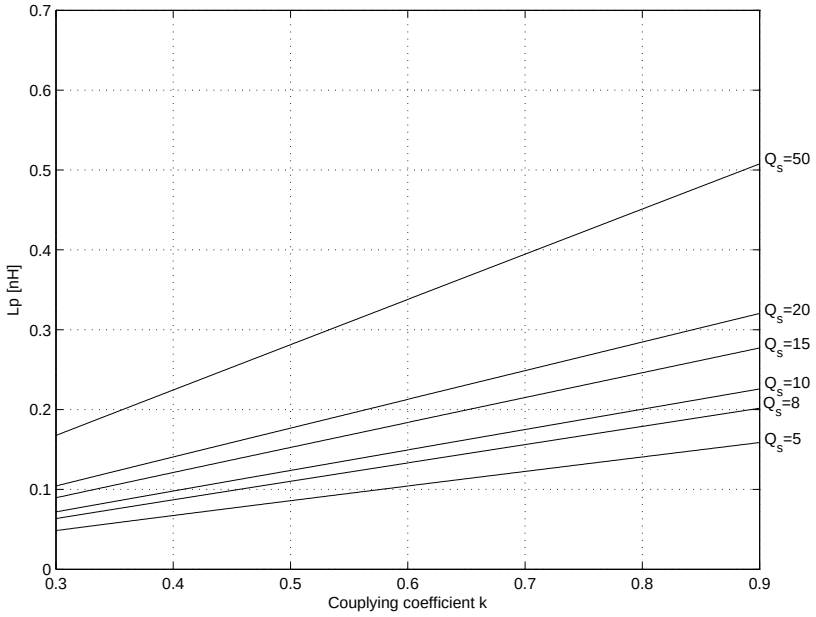


Figure 5.11: L_p optimum values as a function of the coupling coefficient k

In Fig.5.11 the optimum value of L_p as a function of the coupling coefficient k with transformer ratio $r = n \cdot k = 2$ and $Q_p = 10$ for different values of Q_s is reported, while the optimum values for L_s and C_s are shown in Fig.5.12 and Fig.5.13 respectively. The corresponding transformer efficiency is shown in Fig.5.14. If we assume $Q_p = Q_s = 10$, as on-chip inductors with quality factor of the order of 10 can be realized without too much effort,, with components values

$$\begin{aligned} L_p &= 98\text{pH} \\ L_s &= 2.45\text{nH} \\ C_s &= 166.7\text{fF} \end{aligned} \tag{5.6}$$

we can reach a theoretical efficiency of $\simeq 65\%$. With these reactive elements values, we will see how we can also create a fourth-order bandpass ladder filter, as the one shown in Fig.5.15, where the input source is represented with a current generator with infinite output resistance.

The values for the two capacitors C_1 and C_2, together with those for the two inductors L_1 and L_2, can be derived for different types of filters from the values tabulated for low-pass ladder filters [116] applying a low-pass to band-pass transformation. By means of

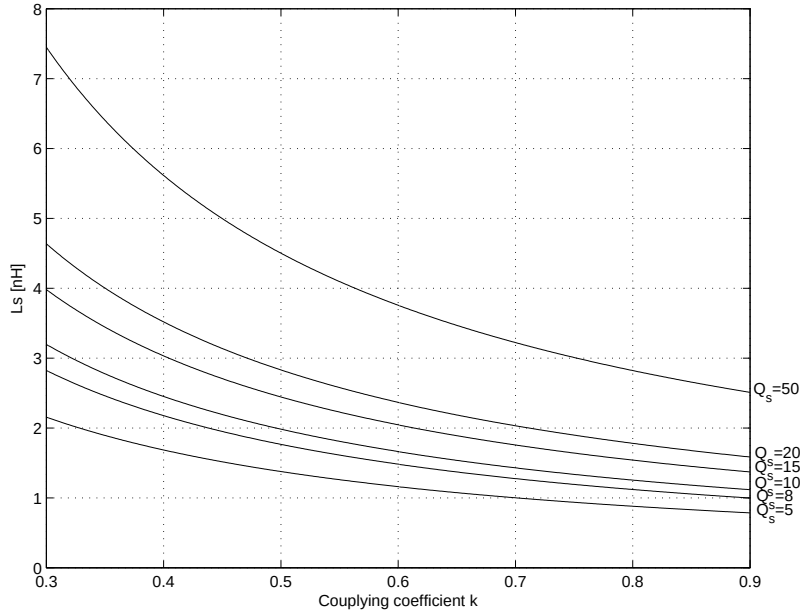


Figure 5.12: L_s optimum values as a function of the coupling coefficient k

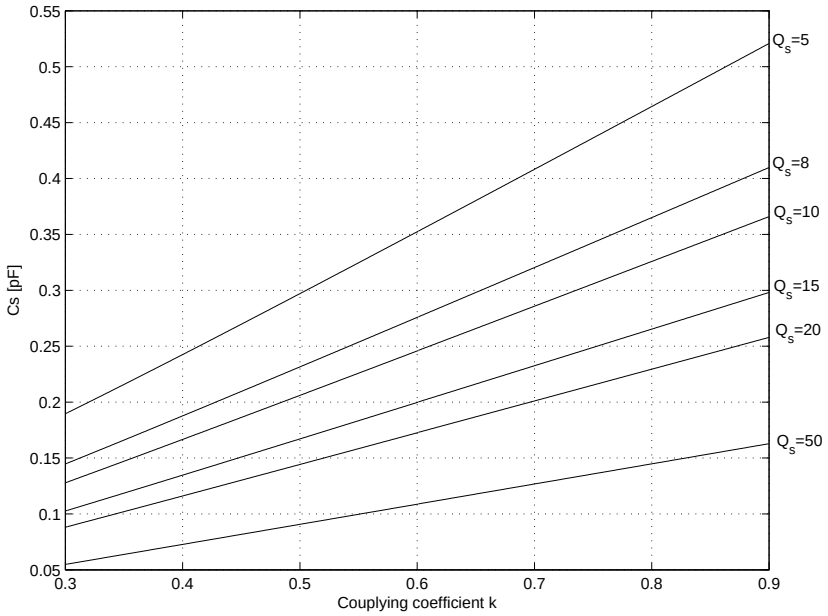


Figure 5.13: C_s optimum values as a function of the coupling coefficient k

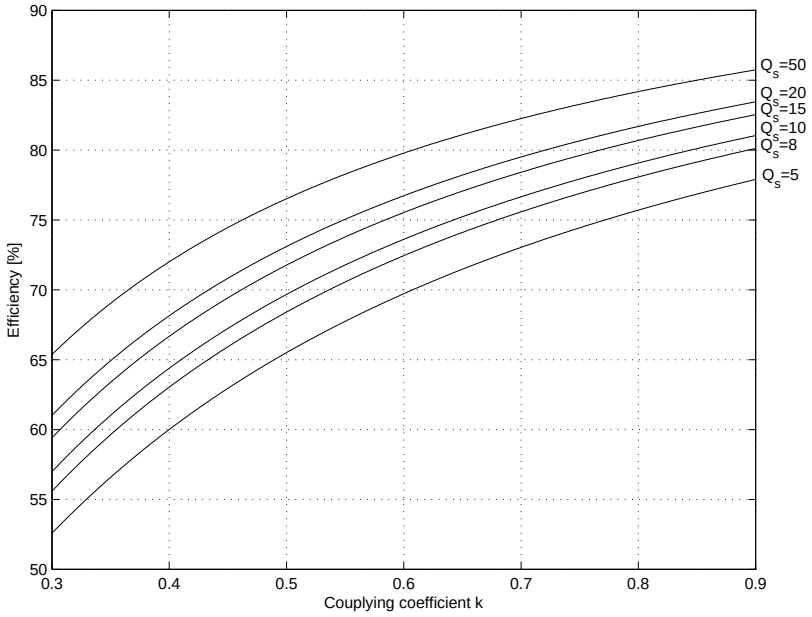


Figure 5.14: Transformer efficiency as a function of the coupling coefficient k

the transformer equivalent model of Fig.5.10, we can derive the primary and secondary coils values as:

$$\begin{aligned} L_p &= L_1 \\ L_s &= \frac{n^2 \cdot k^2 \cdot L_2}{1-k^2} \end{aligned} \tag{5.7}$$

while the capacitors value are given by

$$\begin{aligned} C_p &= C_1 \\ C_s &= \frac{C_2}{n^2 \cdot k^2} \end{aligned} \tag{5.8}$$

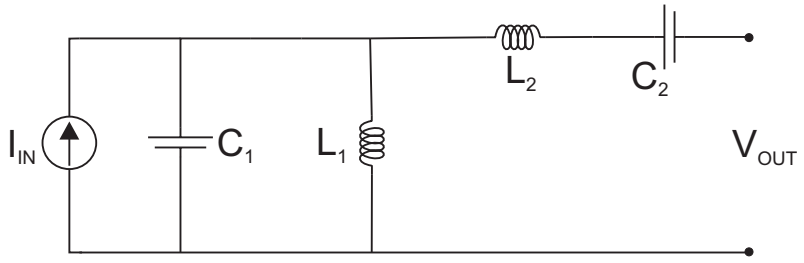


Figure 5.15: Forth-order bandpass ladder filter

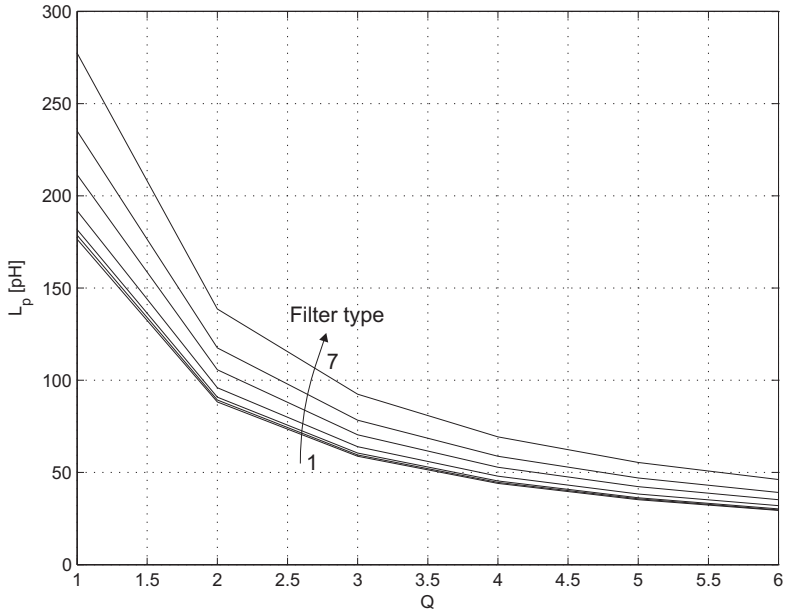


Figure 5.16: L_p as a function of the filter Q factor for different filter types

We consider seven different filters types, that is a Butterworth filter and six different Chebyshev filters with in-band ripple ranging from 0.1dB to 3dB. They correspond to filter type 1 up to 7 respectively. Figures 5.16, 5.17 and 5.18 show L_p, C_p and C_s values for filter Q factors ranging from 1 to 6, where Q is defined as

$$Q = \frac{1}{\Delta} = \frac{\omega_c}{\omega_2 - \omega_1} \tag{5.9}$$

where Δ is the filter fractional bandwidth and ω_c the central frequency, which in our case is equal to

$$\omega_c = 2\pi f_c = \sqrt{\omega_2 \omega_1} \tag{5.10}$$

We can see how a Q factor grater than 2 requires a primary inductance smaller than 100pH, which is hardly implemented on-chip. Thus, we chose for our design $Q = 2$. The secondary inductor value has to simultaneously satisfy two equations: $L_s = n^2 L_p$ in order to ensure the required transformer ratio, and equation (5.7) so as to obtain the desired ladder filter. The corresponding values for the secondary inductor are plotted in Fig.5.19.a for the first condition and in Fig.5.19.b for the second one. By recalling the optimum components values to maximize the transformer efficiency given by (5.6), we

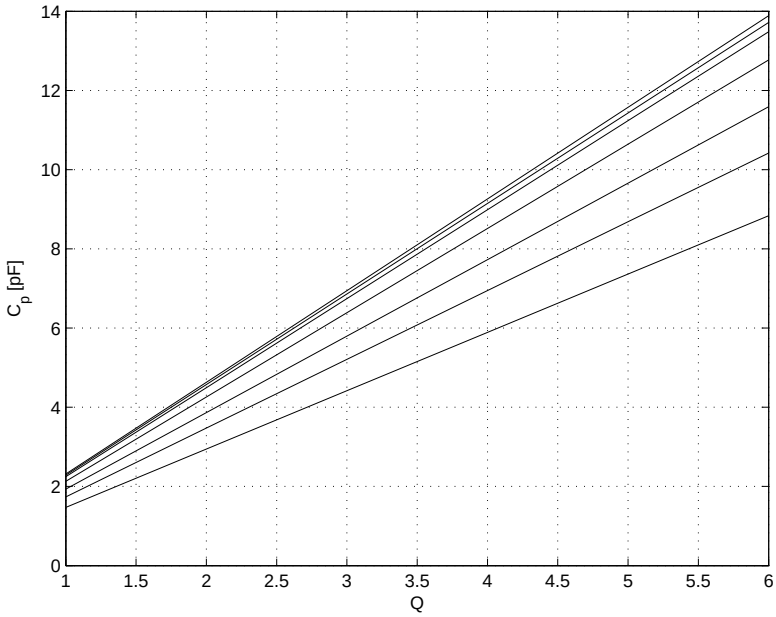


Figure 5.17: C_p as a function of the filter Q factor for different filter types

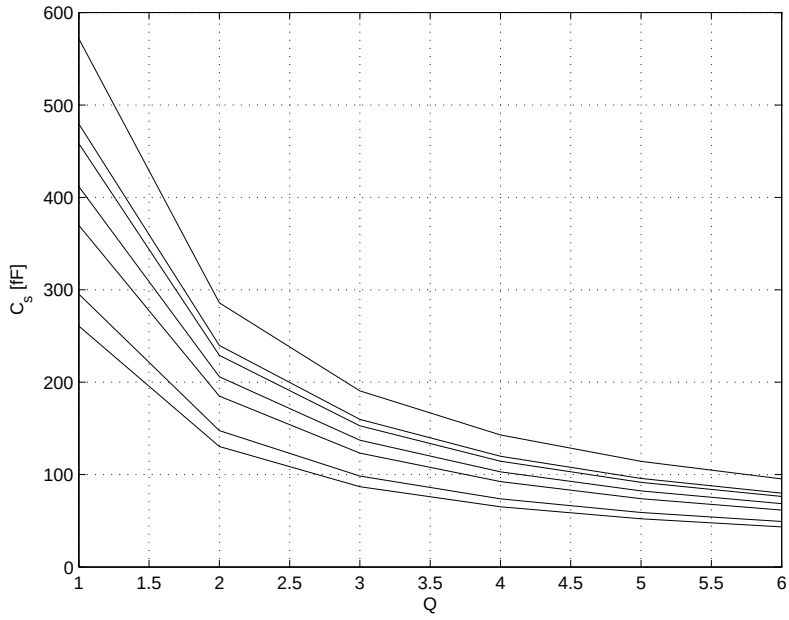
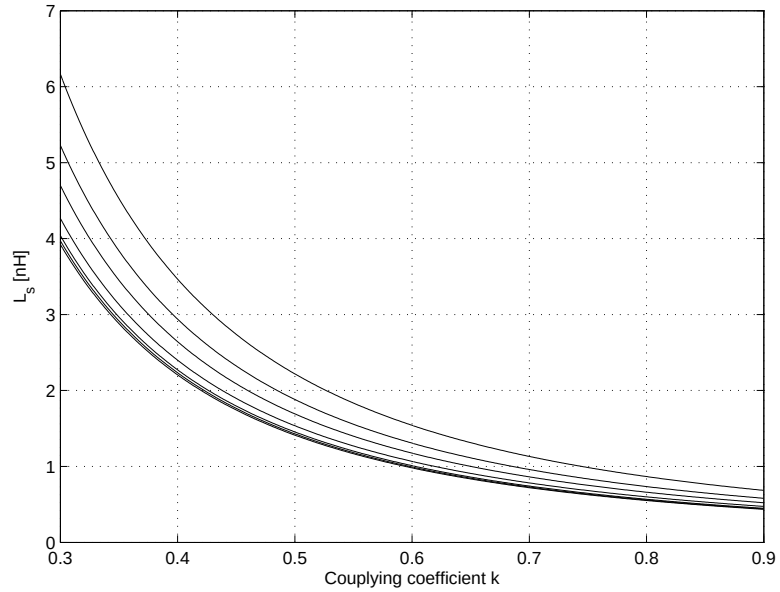
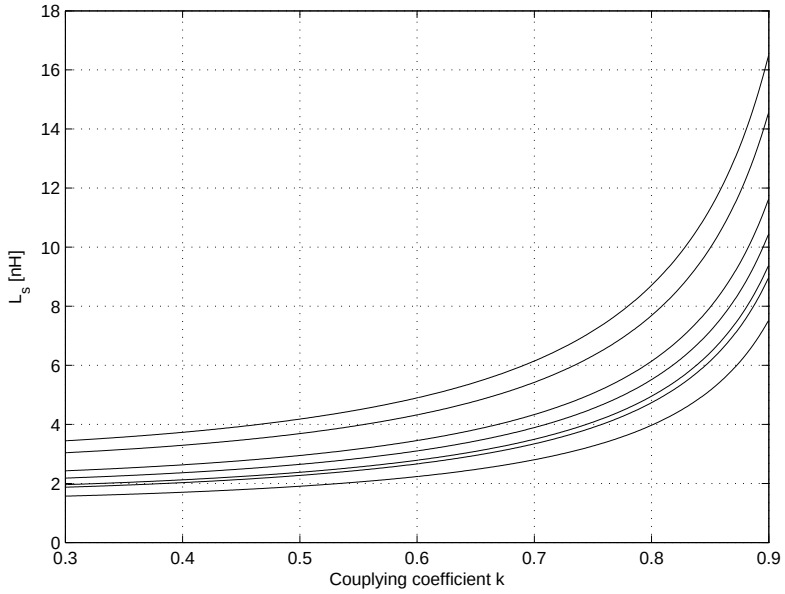


Figure 5.18: C_s as a function of the filter Q factor for different filter types



(a) $L_s = n^2 L_p$



(b) $L_s = n^2 k^2 L_2/(1-k^2)$

Figure 5.19: L_s values with $Q = 2$ as a function of the coupling coefficient k for different filter types

$L_p[pH]$	$L_s[nH]$	$C_p[pF]$	$C_s[fF]$	k
98	2.45	4.25	167	0.4

Table 5.4: Transformer parameters

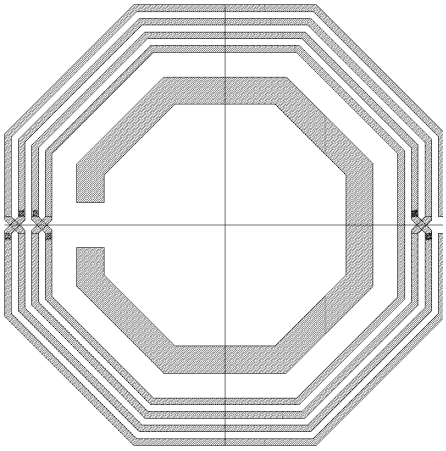


Figure 5.20: Transformer layout

can see that a fourth order bandpass filter of type 4, which corresponds to a Chebyshev filter with in-band ripple of 0.5dB, can be implemented with the reactive element values reported in Table 5.4.

Implementation The transformer has been implemented using concentrically wound planar spirals [117], as shown in Fig.5.20, where the actual transformer layout is reported. Using this configuration, the common periphery between the two windings is limited to just a single turn. Therefore, mutual coupling between adjacent conductors contributes mainly to the self-inductance of each winding and not to the mutual inductance between the windings. As a result, the concentric spiral transformer has less mutual inductance and more self-inductance than the interwound configuration, giving it a lower k-factor. However, this does not represent a limitation in our case, as we need a coupling coefficient $k = 0.4$.

The electrical lumped model of the transformer has been derived from the physical

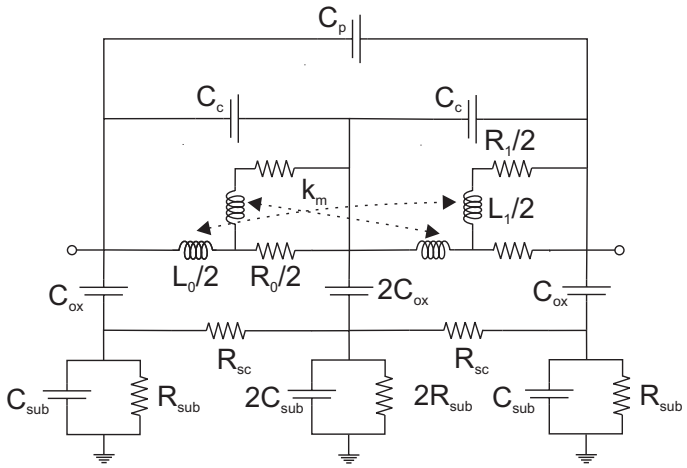


Figure 5.21: Transformer primary coil lumped model

layout by means of electromagnetic simulations. These have been performed with the software Momentum^{TM} from Agilent Technologies. In particular, first the primary and secondary inductor models have been derived separately. Then, a transformer compact model, as the one described in [118], has been derived.

To model the primary inductor, the two-PI model [119] reported in Fig.5.21 has been used. In this model, L_0 is the inductance, R_0 the resistance of the metal strip, L_1 and R_1 model the surface layer inductance and resistance. They are evenly split into two parts and the coupling coefficient k_m cross couples the two parts in order to properly capture the inductive coupling among metal lines. C_{ox} is the oxide capacitance, C_{sub} and R_{sub} are the substrate capacitance and resistance while C_s is the edge-to-edge feedthrough capacitance. In addition, we use C_s and R_{sc} to model the line-to-line coupling capacitance and direct turn-to-turn electric coupling through the dielectric materials and the conductive substrate respectively.

The equivalent inductance, resistance and quality factor for the EM-simulation and the lumped model, whose parameters are reported in Table 5.5, are shown in Fig.5.22.

$L_0[pH]$	$R_0[m\Omega]$	$L_1[nH]$	$R_1[\Omega]$	k_m
97.79	479	2.81	856.4	0.177

$C_{ox}[fF]$	$R_{sub}[\Omega]$	$C_{sub}[fF]$	$R_{sc}[\Omega]$	$C_s[fF]$	$C_c[fF]$
4.53	540.03	40.21	9.40	0.4	0

Table 5.5: Primary inductor lumped model parameters

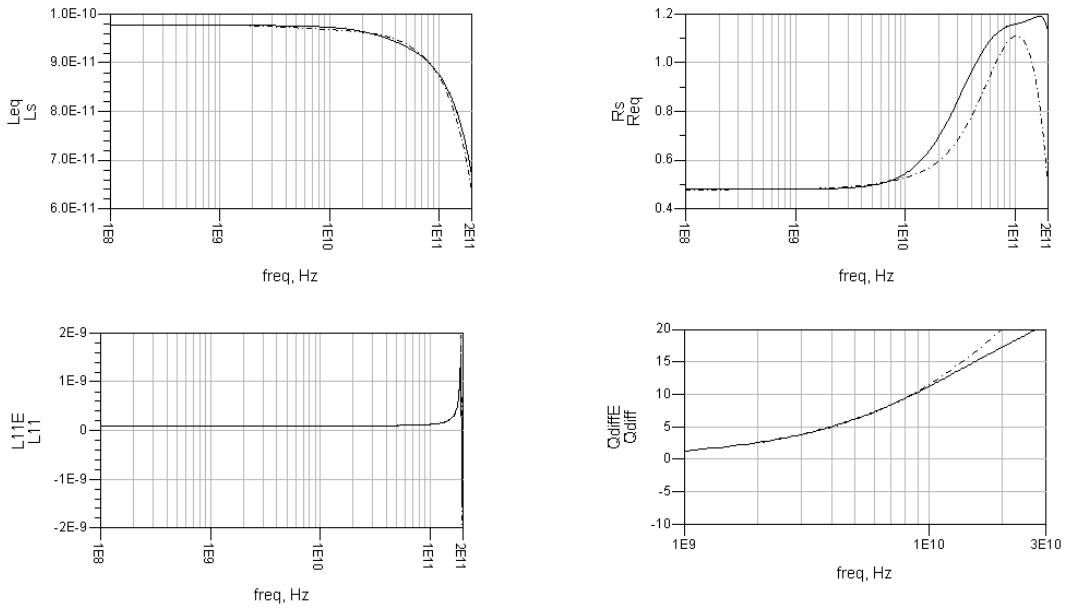


Figure 5.22: Primary equivalent inductance, resistance and quality factor from the EM-simulation (solid) an the lumped model (dashed)

The secondary inductor can be better modeled by means of the PI-model described in [120] and reported in Fig.5.23. In this case, the skin effect is modeled by means of resistor R_{skin} and inductor L_{skin} whose coupling coefficient with L_0 is given by K_{skin}. An additional branch constituted by L_{ed} and R_{ed} takes into account the magnetic coupling between the coil and the substrate. The equivalent inductance, resistance and quality factor for the EM-simulation and the lumped model with the parameters reported in Table 5.6, are shown in Fig.5.24.

$L_0[nH]$	$R_0[m\Omega]$	$L_{skin}[fH]$	$R_{skin}[\Omega]$	k_{skin}
2.46	11.95	446.2	3.66	0.9

$C_{ox}[fF]$	$R_{sub}[\Omega]$	$C_{sub}[fF]$	$C_s[fF]$	$L_{ed}[pH]$	$R_{ed}[\Omega]$	k_{ed}
17.08	220.76	90	12.29	83.26	14.04	0

Table 5.6: Secondary inductor lumped model parameters

Finally, the overall transformer lumped model is derived from the two inductors models by adding a capacitor C_a connected between primary and secondary to model the

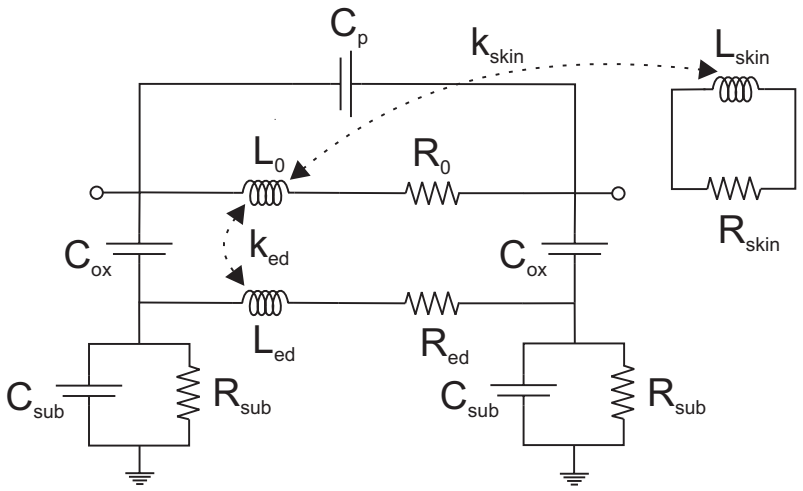


Figure 5.23: Transformer secondary coil lumped model

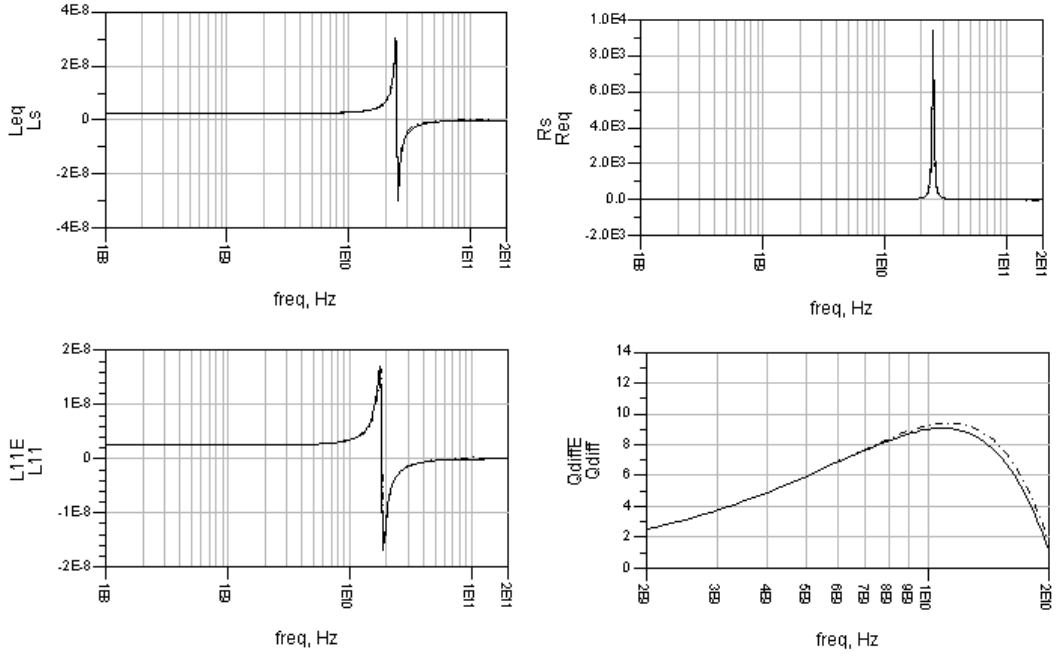


Figure 5.24: Secondary equivalent inductance, resistance and quality factor from the EM-simulation (solid) an the lumped model (dashed)

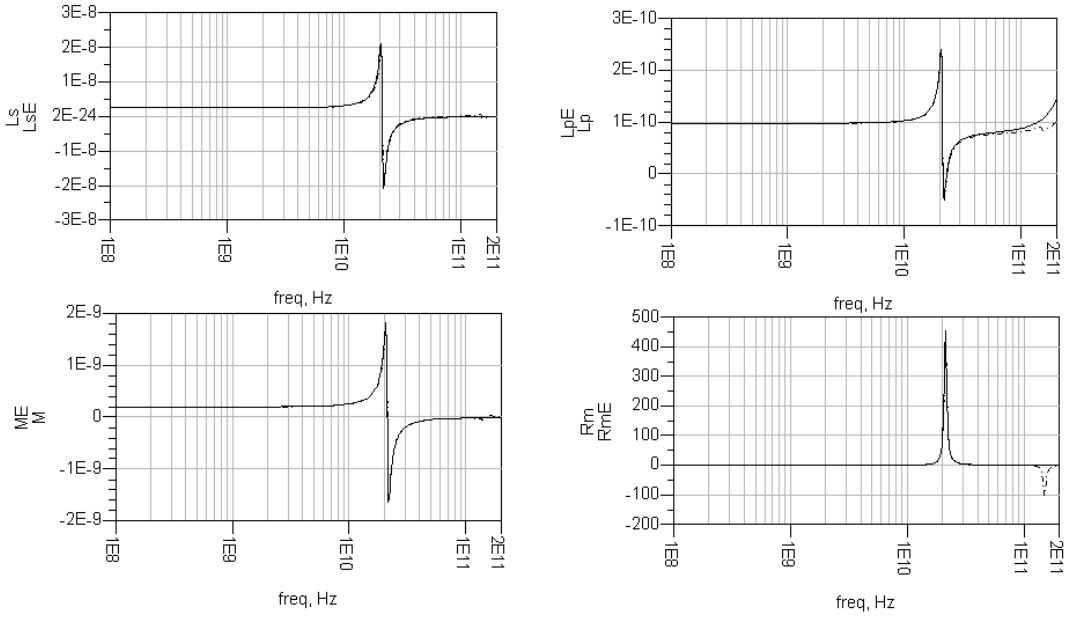


Figure 5.25: Transformer equivalent inductance, resistance and quality factor from the EM-simulation (solid) an the lumped model (dashed)

$C_a[fF]$	k
3.4	0.403

Table 5.7: Transformer lumped model parameters

interwinding capacitance, as described in [118].

The simulations result for the transformer model are shown in Fig.5.25, while its additional parameters values are reported in Table 5.7.

5.3.2 Simulation Results

The Gaussian pulse generator has been simulated in typical conditions. Its output transient waveform is reported in Fig.5.26, while the corresponding spectrum is shown in Fig.5.27. We can see how it reaches a maximum output power of -46dBm/MHz, according to specification, and it meets the FCC out-of-band masks limits without the need of external filtering. The estimated energy/pulse is $\simeq$ 120pJ/pulse, thus in line with the transmitter specifications reported in Table 5.1. This result is in line with the most efficient transmitters for UWB-IR, as the one reported in [114], where 113pJ/pulse are declared at a data

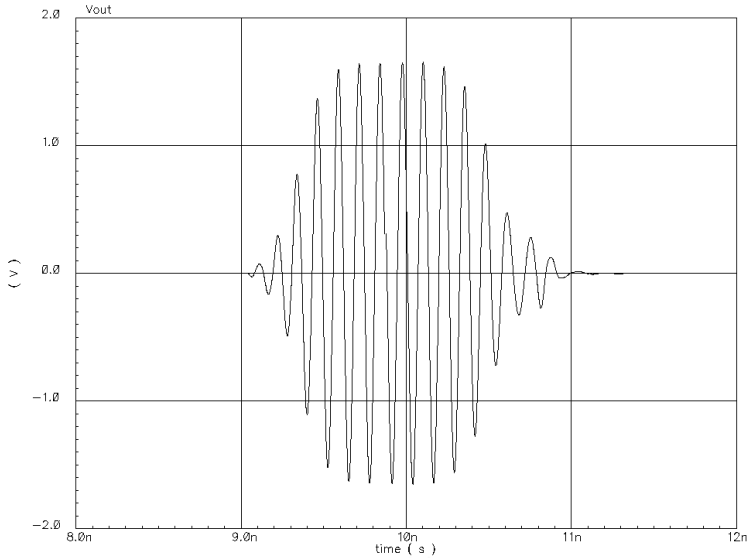


Figure 5.26: Transient output waveform

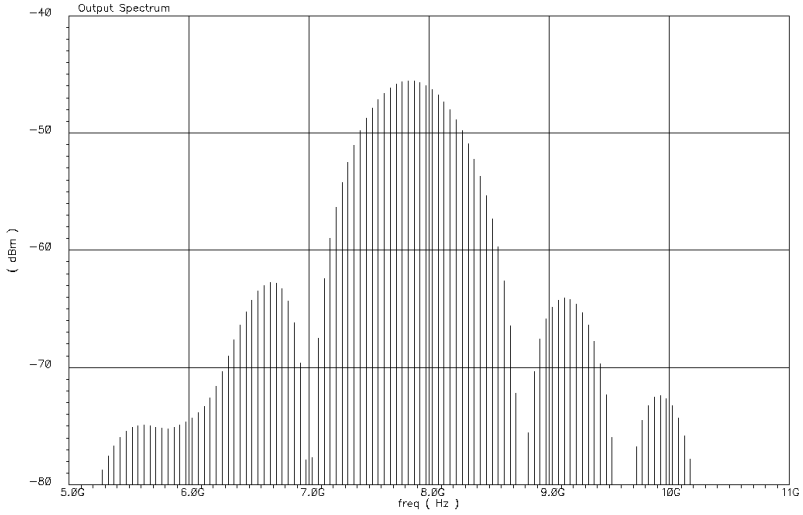


Figure 5.27: Output spectrum

Specification	[114]	[112]	[113]	This work
Data rate	100kb/s	1Mb/s	10Mb/s	100kb/s
Bandwidth	500MHz	500MHz	500MHz	1.25GHz
Center frequency	3.5GHz	10GHz	4.05GHz	7.875GHz
Energy/pulse	113pJ	87.5pJ	43pJ	124pJ
Efficiency	0.02	0.014	0.03	0.14

Table 5.8: Performance summary of UWB-IR transmitters

rate of 100kb/s. However, a more fair comparison can be carried out by considering the transmitter efficiency η_T defined as

$$\eta_T = \frac{\text{load energy/pulse}}{\text{energy/pulse}} \tag{5.11}$$

where the load energy/pulse indicates the energy per pulse transferred to the load. In [114], with a maximum output swing of 710mV_{pp}, a central frequency $f_c = 3.5\text{GHz}$ and a bandwidth $B = 500\text{MHz}$, we have a transmitter efficiency $\eta_T = 2.16\text{pJ}/113\text{pJ} \simeq 0.02$. Our transmitter reaches an efficiency of $\eta_T = 17.5\text{pJ}/120\text{pJ} \simeq 0.15$, thus outperforming the result reported in [114] by a factor of almost 10.

The performance comparison with state of the art transmitters are reported in Table 5.8, where the ring oscillator power consumption has also taken into account, leading to an estimated energy/pulse of 124pJ/pulse.

Although the energy consumed by the frequency dividers plus the early-late detector circuit has not been taken into account, results reported in Table 5.8 show how the proposed solution outperforms the state of the art transmitter efficiency by a factor of almost 10.

5.4 Conclusions and Future Work

A novel energy efficient transmitter for UWB-IR has been proposed, which uses an original combined mixer and power amplifier to generate a Gaussian pulse with 1.25GHz bandwith and center frequency of 7.875GHz. The combined MRX-PA includes a monolithic transformer to reach the maximum output voltage swing required to ensure a link distance of 10 meters with the non-coherent energy detector receiver described in Chap.4.

The transformer has been designed so as to maximize the power efficiency and at the

same time to realize a fourth-order ladder filter, in order to reduce the transmitter out-of band emissions.

Conclusions

The first part of this work, devoted to analog decoding, reports the design of the input interface for an iterative fully analog decoder for a SCCC and of two analog TCM decoders for multi-level Flash memories, all realized in submicron CMOS technologies.

Both projects exhibit the advantages already demonstrated by the analog decoders with respect to their digital counterparts, that is a reduced area occupation, a lower power consumption and an higher throughput. In fact, the SCCC hybrid decoder reaches an efficiency of 2.1nJ/bit which outperforms digital decoders with the same block length, that is around 5000, of a factor up to 50 [25], while the full analog implementations of TCM decoders presented in Chapter 3 can achieve a decoding speed comparable with the state-of-the-art linear block codes occupying a smaller area, with a BER close to that of the ideal decoding algorithm.

At the same time, some traditional limitations of the analog implementations, due mainly to the fact that their circuitry complexity increases linearly with the codeword length, are overcame as the hybrid SCCC decoder is reconfigurable in both block length and code rate.

However, it is worth to notice how the most area and power consuming circuitry for both analog decoder projects is not the one implementing the decoder core intself, but the so-called I/O interface circuitry. This consists of an analog memory and a voltage to probability converter for the SCCC decoder, while it reduces only to the voltage to probability converter for the TCM decoders. Even if the design of an analog memory can be avoided in this latter case, as the data to be processed are already stored in the Flash memory array, the complexity of the interface between the memory array and the decoder core itself increases with respect to the binary case as we deal with memory cells with 16 levels. As a consequence, it is mandatory to optimize the interface design in terms of area, power and speed in order not to spoil the overall system performance.

Moreover, as the performance loss in terms of BER of the 4-state TCM decoder with respect to the 8-state one is only 0.5dB while the power consumption increases by over a factor of 2, our work demonstrates how the analog approach is all the more a competitive solution for ECC as far as the decoder states number is kept low.

The second part of this thesis is focused on the design of a transceiver for low data

rate UWB-IR. The transceiver is first analyzed at system level in order to draw the specifications for both receiver and transmitter so as to guarantee a link distance of at least 10 meters.

At the receiver side, the non-coherent energy detection approach has been adopted [78], as it does not require precise phase control, which allows both the transmitter and the receiver architecture to be simplified.

The transmitter uses a novel combined mixer and power amplifier to generate a Gaussian pulse with 1.25GHz bandwith and center frequency of 7.875GHz. The combined MRX-PA includes a monolithic transformer to reach a maximum output voltage swing of $3.2V_{pp}$, necessary to ensure the required link distance. The transformer has been designed in order to maximize the power efficiency and at the same time to realize a fourth-order ladder filter, so as to reduce the transmitter out-of band emissions.

The efficiency of our design has been compared with state-of-the-art UWB-IR transmitters, showing how the proposed solution leads to an improvement in the transmitter efficiency of a factor of almost 10.

A synchronization algorithm has also been proposed and successfully tested, showing how the transceiver chipset is a promising candidate for a low-power UWB-IR.

A

Fundamentals of Error Correcting Coding

This appendix, after a brief introduction about general communication/storage systems, presents some basic error correcting codes, with a particular emphases on Trellis Coded Modulation and Turbo codes. The decoding issue is then tackled, to concentrate on all those aspects peculiar to the analog decoding.

A.1 The Shannon Limit

The most intuitive way to represent a data communication or storage system is shown in Fig.A.1. This is the famous *Figure 1* of most books on error correcting coding theory. An information source emits a sequence of binary digits (bits), called the uncoded sequence **u**. This sequence is transformed into the coded sequence **x** by an encoder and transmitted over a communication channel or a storage medium. During the transmission, the coded sequence **x** is corrupted by a noise vector **n**, where we assume that the noise is of additive nature and that no inter-symbol interference is present. Thus a noisy sequence **y** is received at the input of the decoder, whose task is to estimate the most probably sent data sequence $\hat{\mathbf{u}}$ using **y**.

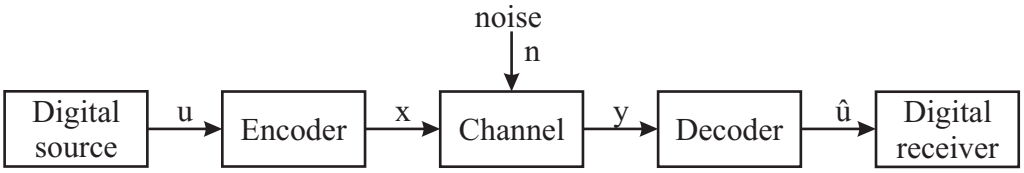


Figure A.1: SISO simplified diagram with switch between *inner* and *outer* configuration

In his 1984 pioneering work [121], Shannon showed that every communication channel has a maximum rate for reliable data transmission, which he called the channel capacity C, measured in bits per second. He demonstrated that it is always possible to send information at a rate R lower than C through a channel with an error probability as small as desired by properly encoding the information source. This statement on controlled error probability is not true for rates above C.

The capacity of an ideal band-limited channel corrupted by an additive white Gaussian noise (AWGN) is given by the famous formula:

$$C = B\log_2\left(1+\frac{S}{N}\right) \tag{A.1}$$

where C is the capacity in bits per second, B is the channel bandwidth in Hertz and S/N is the signal-to-noise power ratio at the receiver.

However, until the advent of complex Turbo codes [2, 122], practical error control schemes have been far away from this theoretical limit. In fact, Shannon's theorem sets a limit on the maximum transmission rate over a channel, but it is silent about the way to reach it. After Turbo codes, an even higher rate has been reached using very large low-density parity-check codes [123], which were originally invented by Gallager [124].

A.2 Types of Codes

Most of the codes that are common use today can be distinguish between two main types, *block codes* and *convolutional codes*. The output of a block encoder is strictly block oriented and it is generated by combinatorial operations, whereas the convolutional encoders create data streams of possibly infinite length. Additionally, the output of a convolutional encoder is created by a finite-state machine, that is the encoder incorporates memory that tracks the history of the incoming data bits.

A.2.1 Block Codes

A block code is defined as an algebraic mapping from the vector space $GF(q)^k$ over the Galois field $GF(q)$ into the vector space $GF(q)^n$, with $n > k$ [39]. If this mapping from one vector space to another is linear, we speak of *linear codes*.

In block coding, the incoming data stream is segmented into blocks of length k and then mapped into n-symbol long codewords. If we assume the information is in the form

of a sequence of symbols belonging to a given alphabet of q symbols (if $q = 2$, the symbols are named bits), than we speak of a (n,k) q-ary block code. Thus, a (n,k) q-ary block code is a set of q^k codewords corresponding to the q^k possible data blocks.

A *linear block code* is entirely described by its generator matrix **G**. A codeword is built using the relation

$$\mathbf{x} = \mathbf{u} \cdot \mathbf{G} \tag{A.2}$$

where both the codeword **x** and the user word **u** are assumed to be row vectors. Thus linear codes transform the all-zero input vector into the all-zero codeword. Equivalently, every codeword **x** has always to satisfy the equation

$$\mathbf{H} \cdot \mathbf{x}^T = \mathbf{0}^T \tag{A.3}$$

where **H** is the parity-check matrix, that can be derived from the generator matrix **G** such as $\mathbf{G}\mathbf{H}^T = 0$.

The *code rate* of a block code is defined as the ratio between the number of bits carrying information and the total codeword length. Thus, if the generator matrix is a full rank $k \times n$ matrix, the code rate is given by

$$R = \frac{k}{n} \tag{A.4}$$

The *Hamming distance* between two codewords is the number of positions in which they differ. The minimum distance d_{min} of a code is the Hamming distance of the pair of codewords with the smallest Hamming distance [39]. A code with a minimum distance d_{min} can correct up to $t = \lfloor (d_{min} - 1)/2 \rfloor$ errors. If d_{min} is even, the code can simultaneously correct $t = (d_{min} - 2)/2$ errors and detect $d_{min}/2$ errors.

To ensure a minimum distance equal or greater than $2t + 1$ in any (n,k) q-ary block code, the following condition must be satisfied:

$$n - k \geq \log_q \left\{ \sum_{i=0}^{t} \left[\binom{n}{i} (q-1)^i \right] \right\} \tag{A.5}$$

which is know as Hamming bound.

A.2.2 Hamming Codes

Hamming codes are a whole class of linear block codes that can correct single errors. If we consider a binary alphabet, that is $q = 2$, Hamming codes of length $n = 2^r - 1$ with

$r \geq 2$ are defined to have a parity-check matrix $\mathbf{H}$ whose columns consist of all non-zero binary vectors of length r, each used once.

A Hamming code is thus a $n = 2^r - 1, k = 2^r - 1 - r, d = 3$ block code. The Hamming code as it was first defined is the $(7,4,3)$ Hamming code, whose parity-check matrix is given by [125]:

$$\mathbf{H} = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \tag{A.6}$$

Any code consisting of rows that are created using linear combinations and column permutations of the original matrix $\mathbf{H}$ is said to be *equivalent*.

A.2.3 Convolutional Codes

Unlike block codes, convolutional codes work on data streams of possibly infinite length. An encoder for a convolutional code can be seen as a finite-state machine, that is a sequential logic circuit, with a memory of order m. The generator matrix $\mathbf{G}$ of a convolutional code has a general form given by:

$$G(D) = \begin{pmatrix} G_{11}(D) & G_{12}(D) & \cdots & G_{1n}(D) \\ G_{21}(D) & G_{22}(D) & \cdots & G_{2n}(D) \\ \vdots & \vdots & \ddots & \vdots \\ G_{k1}(D) & G_{k2}(D) & \cdots & G_{kn}(D) \end{pmatrix} \tag{A.7}$$

Each element $G_{ij}(D)$ represents a transfer function of a linear discrete-time system of order m:

$$G_{ij}(D) = \frac{a_{ij,m}D^m + a_{ij,m-1}D^{m-1} + \cdots + a_{ij,0}}{b_{ij,m}D^m + b_{ij,m-1}D^{m-1} + \cdots + b_{ij,0}} \tag{A.8}$$

where D indicates the unit delay element.

The block diagram of a simple convolutional code with generator polynomials

$$G(D) = \left[D^2 + 1, D^2 + D + 1\right] \tag{A.9}$$

is shown in Fig.A.2.

The rate of a convolutional code is still given by the ratio of the generator matrix dimensions. Thus, the convolutional code of Fig.A.2 has a code rate $R = 1/2$.

If the uncoded data $\mathbf{u}$ is part of the codeword, we speak of *systematic* code. Given the generator matrix of a convolutional code, it is always possible to build its systematic

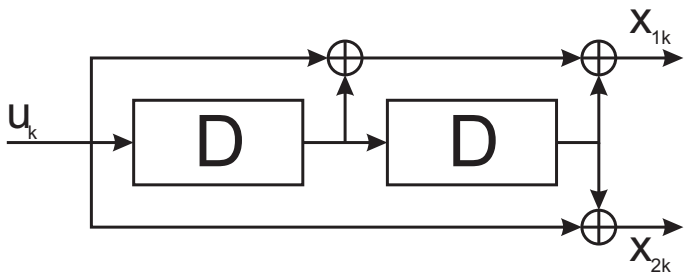


Figure A.2: 4-state rate 1/2 binary convolutional encoder

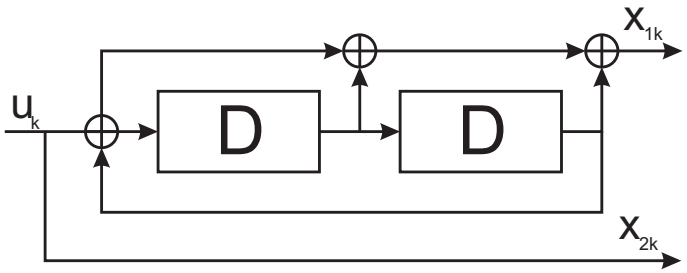


Figure A.3: Systematic version of code of Fig.A.2

version as

$$G(D) = \begin{pmatrix} 1 & \frac{G_{12}(D)}{G_{11}(D)} & \cdots & \frac{G_{1n}(D)}{G_{11}(D)} \\ 1 & \frac{G_{22}(D)}{G_{21}(D)} & \cdots & \frac{G_{2n}(D)}{G_{21}(D)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \frac{G_{k2}(D)}{G_{k1}(D)} & \cdots & \frac{G_{kn}(D)}{G_{k1}(D)} \end{pmatrix} \tag{A.10}$$

The systematic version of the code of Fig.A.2 is reported in Fig.A.3. Its generator matrix is given by:

$$G(D) = \left[1, \frac{D^2 + D + 1}{D^2 + 1}\right] \tag{A.11}$$

The systematic version of a convolutional code is also called *recursive* and exhibits the same performance of the original code.

The definition of the Hamming distance given in Sec.A.2.1 can not be applied to convolutional codes, as they work on codewords of possibly infinite length. Instead, for convolutional code, we speak of *minimum Euclidean distance* d_{free} of a code, referring to the same concept [126].

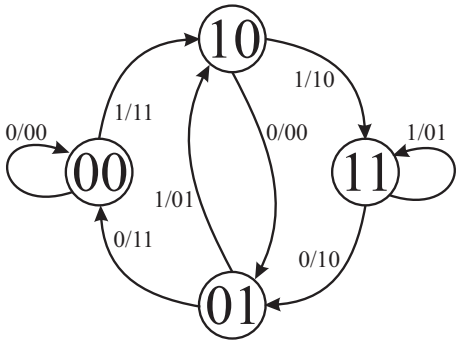


Figure A.4: 4-state transition diagram of code of Fig.A.2

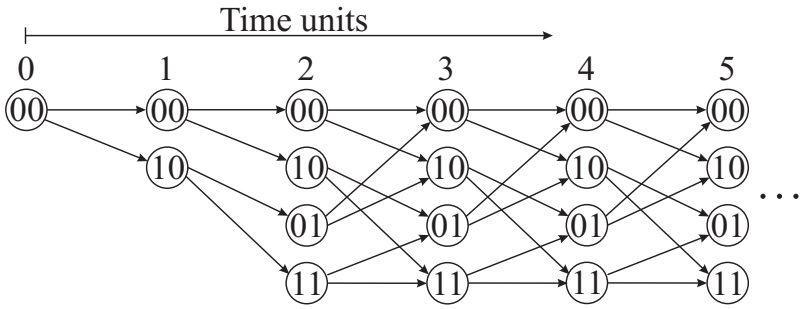


Figure A.5: 4-state trellis diagram of code of Fig.A.2

A.2.4 Convolutional Codes Trellis Diagram

Since a convolutional encoder is a finite-state machine, it may be completely defined by a finite *state-transition diagram*, such as the 4-state diagram shown in Fig.A.4 for the encoder of Fig.A.2. The nodes in the transition diagram are the states of the finite-state machine and the branches represent the possible transitions between states. Each branch is labeled by the user bits **u** which cause the transition as well as by the corresponding output codeword **x**.

If we index the state-transition diagram by both the states and the time index r, Fig.A.4 expands into the *trellis diagram* of Fig.A.5. This is a two dimensional representation of the operation of the encoder, capturing all possible state transitions starting from an originating state that is usually state 0. If the finite-state machine is driven back into the original state at a certain time $r = L$, as shown in Fig.A.6 with $L = 8$, we speak of *terminated* codes. To force the encoder back to the original state, the last m branches, where m is the number of memory elements, are predetermined and no information is

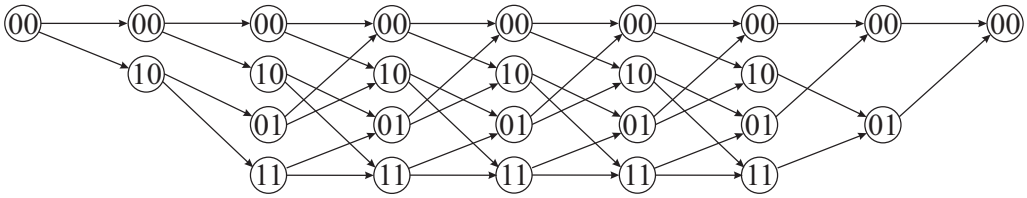


Figure A.6: 4-state terminated trellis diagram of code of Fig.A.2

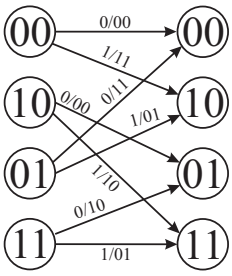


Figure A.7: Trellis section of code of Fig.A.2

transmitted in those time units. This results in a rate loss of

$$\frac{mk}{n(n+m)}$$

. In order to avoid this rate loss, *tail baiting* codes have been proposed in 1986 [127]. The trellis of a tail baiting code is formed by connecting the outgoing states of the last trellis section to the incoming states of the first trellis sections. Such a tail baiting code forms a closed ring structure with no need for termination bits. A valid codeword is then defined by a path starting in any state at a certain point, i.e. non necessary the zero state, and terminating in the same state after one turn.

Since the size of the trellis transition diagram explodes for long user data sequences, a complete description of a convolutional code can also be given by a single section of the trellis diagram. The trellis section for the code of Fig.A.2 is shown in Fig.A.7.

A.2.5 Trellis Coded Modulation

‘

The use of an error correcting scheme with rate $R = k/n$ to increase the reliability of binary transmission or storage systems, reduces the spectral efficiency from its maximum value $\mu = 1$ bit/sec/Hz to $\mu = R < 1$ bps/Hz. This leads to a faster signalling rate or to a larger bandwidth if the aim is to guarantee the same rate of the uncoded system.

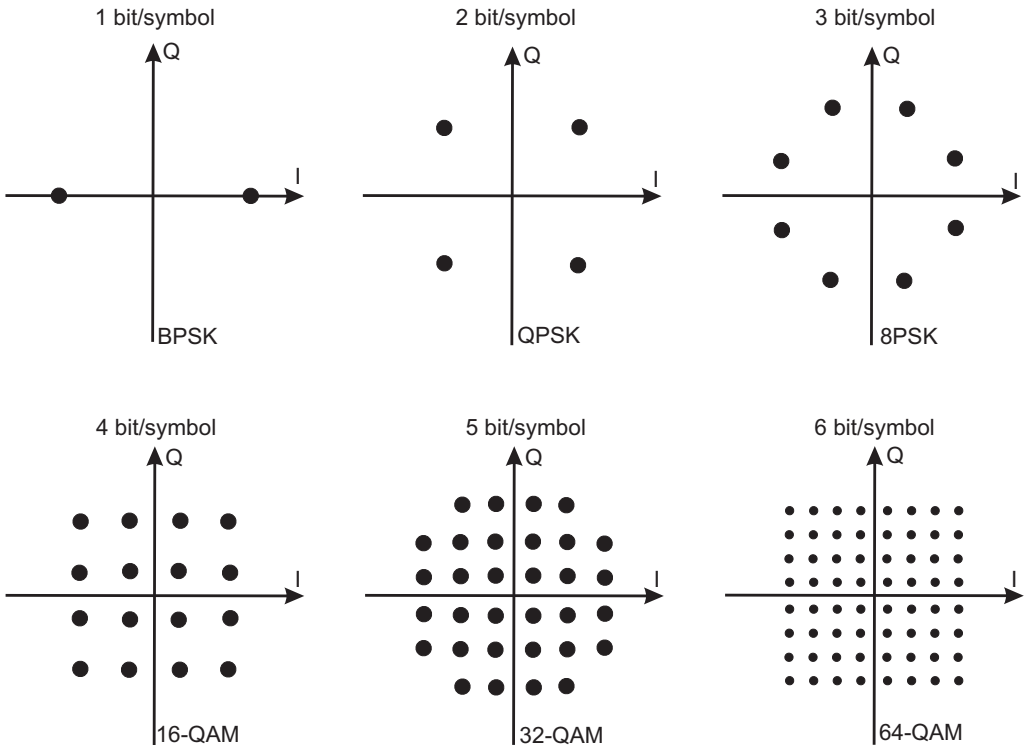


Figure A.8: Signal constellations

Equivalently, the bit rate has to be reduced by a factor of $1/B$ so as to keep the transmission symbol rate or bandwidth constant, where B is the channel bandwidth.

To increase the data rate without increasing the bandwidth, Ungerboeck [51] and Imai and Hirakawa [128] used an expanded signal set, such as 2^m-ary PSK or QAM digital modulation, and then applied an error correcting code to increase the Euclidean distance between codewords.

Several signal constellations used in digital communication systems are shown in Fig.A.8. From the viewpoint of digital signal processing, modulation is mapping, that is the process of assigning a m-dimensional binary vector b to a signal point $(x(b), y(b))$ in the constellation. Using 2^m-ary modulation instead of the binary one has the advantage that the number of bits per symbol is increased by a factor of m, thus increasing the spectral efficiency of the system. On the other hand, the required average energy of the signal increases, as in the QAM case, or the distance between modulation symbols decreases, as with PSK modulation. In practice, transmitted power or storage level is limited to a maximum value. This implies that the signal points become closer to each other. As a result, an error correcting code is needed to reduce the increased error probability and to

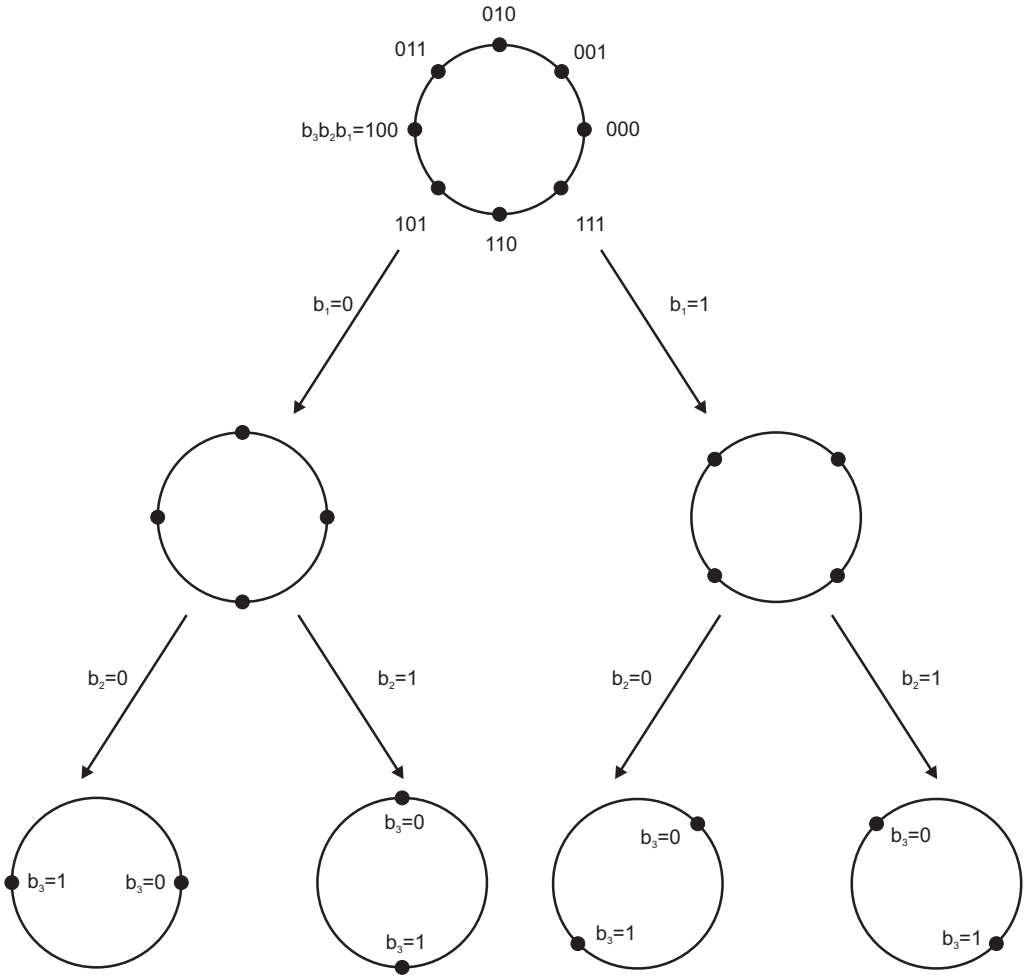


Figure A.9: Trellis section of code of Fig.A.2

improve the system reliability.

The basic idea of TCM is thus to expand the signal constellation in order to obtain the redundancy needed for error correction coding and then to design a trellis code to increase the minimum Euclidean distance between codewords. In fact, the *asymptotic code gain* of a TCM scheme is given by:

$$G = 10\log_{10}\left(\frac{d^2_{free}}{d^2_{unc}}\right) \tag{A.12}$$

where d^2_{unc} indicates the minimum squared Euclidean distance between uncoded signal sequences.

The design of a TCM, which is the joint design of a trellis code and a modulation scheme, is performed using a *mapping by set partitioning*, as proposed by Ungerboeck

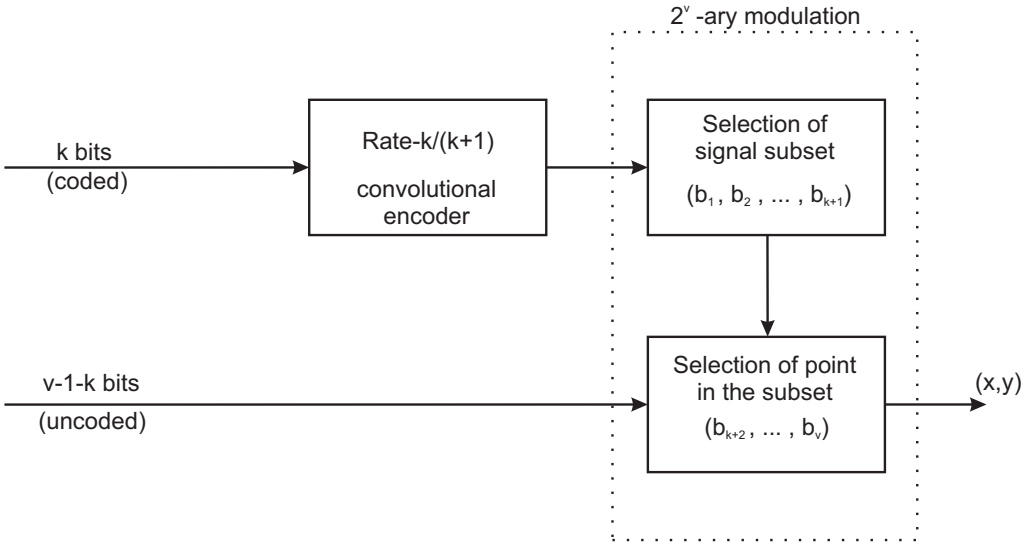


Figure A.10: TCM encoder

in [51]. A basic trellis structure, associated with the state transitions of a finite-state machine, is selected and signal subsets mapped to trellis branches. Uncoded signals are assigned to parallel branches so as to increase system spectral efficiency.

Thus, a 2^m-ary modulation signal set S is *partitioned* in m levels. For $1 \leq i \leq m$, at the i-th partition level, the signal set is divided into two subsets $S_i(0)$ and $S_i(1)$, such that the *intra-set distance*, δ_i, is maximized. A label bit $b_i \in \{0,1\}$ is associated with the subset choice, $S_i(b_i)$, at the i-th partition level. This partition process results in a *labelling* of the signal points. Each signal point in the set has a unique m-bit label $b_1 b_2 \cdots b_m$ and is denoted by $s(b_1, b_2, \cdots, b_m)$. With this Ungerboeck partitioning of a 2^m-ary modulation signal constellation, the intra-set distances are in nondecreasing order $\delta_1^2 \leq \delta_2^2 \leq \cdots \leq \delta_m^2$. This strategy corresponds to a natural labelling for M-PSK modulations, i.e., binary representations of integers, whose value increases clockwise (or counter-wise). Fig.A.9 shows a natural mapping of bits to signals for the case of a 8-PSK modulation, with $\delta_1^2 = 0.586$, $\delta_2^2 = 2$ and $\delta_3^2 = 4$. Ungerboeck regarded the encoder *"simply as a finite-state machine with a given number of states and specified state transitions"*. He gave a set of pragmatic rules to map signal subsets and points to branches in a trellis. These rules can be summarized as follows:

1. all subsets should occur in the trellis with equal frequency and with a fair amount of regularity and symmetry;

2. state transitions that begin or end in the same state should be assigned to subsets

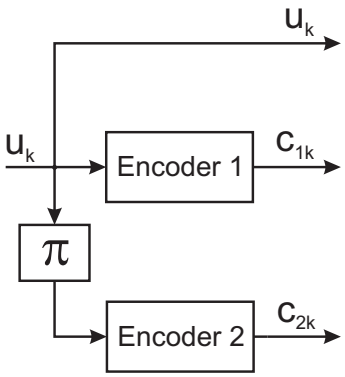


Figure A.11: Parallel concatenated convolutional code block diagram

separated by the largest Euclidean distance;

3. parallel transitions are assigned to signal points separated by the largest Euclidean distance, that is the highest partition levels.

The general structure of a TCM encoder is shown in Fig.A.10. In the general case of a rate $(m-1)/m$ TCM system, the trellis structure is inherited from a $k/(k+1)$ convolutional encoder. The uncoded bits introduce parallel branches in the trellis.

A.2.6 Turbo Codes

In 1993, Berroux [2] presented his first article on Turbo codes, which has turned the view of coding theory upside down. This new coding scheme, whose performance are very close to the Shannon limit [129], consists of two or more convolutional codes connected in series or in parallel by a bit-interleaving structure π.

In particular, *Parallel Concatenated Convolutional Codes PCCC* consist of two convolutional encoders working in parallel, as shown in Fig.A.11. The first encoder receives a copy of the user data **u** while the second encoder is fed with a scrambled version of the same user data obtained by means of an *interleaver* π.

In the *Serial Concatenated Convolutional Codes SCCC* scheme [130], the two encoder are connected in series, as depicted in Fig.A.12. The first encoder (*Outer*) transforms the user data **u** into a temporary codeword, whose bits are permuted by the interleaver and then fed to the second encoder (*Inner*). A *puncturer* can be inserted between the two encoders to delete some parity bits so as to increase the code rate.

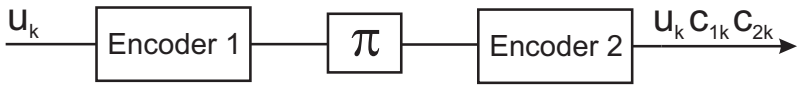


Figure A.12: Serial concatenated convolutional code block diagram

A.3 Decoding

Basically, the decoding is a decision-making process. Based on the observed data vector $\hat{\mathbf{u}}$, the decoder tries to figure out which information bit or information vector has been generated by the information source.

The so called *hard decision* algorithms have been widely used in the past, due to their straightforward implementation in the digital domain. In fact, the Viterbi algorithm has been the standard decoding algorithm for most convolutional codes for over a decade.

With the advent of Turbo codes, *soft decision* algorithms have become popular, because they allow the implementation of an iterative decoding process, necessary to the Turbo code decoding.

A.3.1 Viterbi Algorithm

In 1967, Viterbi [131] introduced a new algorithm for decoding convolutional codes. It works on the trellis diagram of the code and it *decide* what codeword have the *maximum-likelihood* (or the *minimum-distance*) from the received string.

If we consider a trellis diagram of a convolutional code, as the one shown in Fig.A.5, we can associate to each possible path $\mathbf{x}$ the *log-likelihood* function

$$\log L(\mathbf{y}|\mathbf{x}) = \log P(\mathbf{y}|\mathbf{x}) = \sum_{i=0}^{n-1} \log P(y_i|x_i) \tag{A.13}$$

where $\mathbf{y}$ is the received string.

We observe that if the *max-log-likelihood* path includes the state s_k at time unit k, then the first k branches of that path constitute the *max-log-likelihood* path of the partial trellis from time 0 to time k.

Therefore it suffices at time k to determine and retain for each possible state s_k only the biggest path from the unique state at time 0 to that state. This path is called the “survivor”.

The time-$k+1$ survivors may be determined from the time-k survivors by the following recursive “add-compare-select” rules:

1. for each branch from a state at time k to a state at time $k+1$, add the metric of that branch to the metric of the time-k survivor to get a candidate path metric at time $k+1$;

2. for each state at time $k+1$, compare all the candidate path metrics arriving at that state and select the path corresponding to the largest as the survivor. Store the new survivor path.

At the end of the trellis, there is a unique state, whose survivor is the *max-log-likelihood* path for the received string **y**.

This regular recursive structure is attractive for software or hardware implementation because it requires only some memory to store the metrics and the temporary paths.

The Viterbi algorithm can be applied to decode the most likely TCM sequence as well, provided that the branch metric generator is modified to include parallel branches. The selection of the winning branch and surviving uncoded bits should be changed as well. The survivor path (or trace-back) memory should include the $(m-1-k)$ uncoded bits, as opposed to just one bit for rate-$1/n$ binary convolutional codes.

However, for practical considerations, it was suggested in [132] that 2^m-ary modulation signal constellations be partitioned in such a way that the cosets at the top two partition levels are associated with the output of a rate 1/2 convolutional encoder. This mapping leads to a *pragmatic TCM system*. With respect to the general encoder structure shown in Fig.A.10, the value of $k=1$ is fixed. As a result, the trellis structure of a pragmatic TCM remains the same, as opposed to the first TCM proposed by Ungerboeck, for all values of $m>2$. The difference is that the number of parallel branches $m-2$ increases with the number of bits per symbol. This suggests a two-stage decoding method in which, at the first stage, the parallel branches in the trellis "collapse" into a single branch and a conventional off-the-shelf Viterbi decoder can be used to estimate the coded bits associated with the two top partition levels. In a second decoding stage, based on the estimated coded bits and the positions of the received symbols, the uncoded bits are estimated.

A.3.2 MAP Decision Rule

Recalling the general communication system of Fig.A.1, if the data transmission is assumed to be over a time-invariant, memory-less and feedback-less channel, we can define

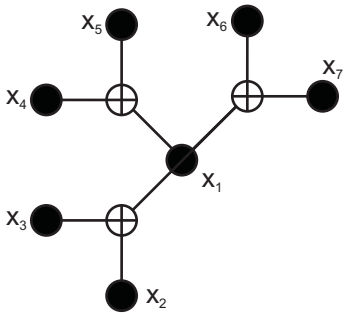


Figure A.13: $(7,4,2)$ code tanner graph

the conditional probability

$$P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}) = \prod_i P_{Y|X}(y_i|x_i) \tag{A.14}$$

This conditional probability, called the *a posteriori probability* APP, represents the probability of receiving at the decoder input the noisy codeword $\mathbf{y}$ given the sent user data $\mathbf{x}$.

Given the received symbols sequence $\mathbf{y}$, a *Maximum a Posteriori Probability* algorithm finds the user data sequence $\mathbf{x}_{MAP}$ that maximizes (A.14), which means:

$$P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}_{MAP}) = \max_{\mathbf{x}\in\mathbf{X}} P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}) \tag{A.15}$$

An algorithm implementing the MAP decision rule is the *Sum-Product* algorithm [133].

A.3.3 Sum-Product Algorithm

The Sum-Product algorithm can be better described with the help of an example. Let's consider a simple binary block code $(7,4,2)$ with parity-check matrix:

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \tag{A.16}$$

Each row of the parity-check matrix corresponds to a parity equation. Thus, from the parity-check matrix (A.16), we can derive the parity check equations:

$$\begin{aligned} x_1 \oplus x_2 \oplus x_3 &= 0 \\ x_1 \oplus x_4 \oplus x_5 &= 0 \\ x_1 \oplus x_6 \oplus x_7 &= 0 \end{aligned} \tag{A.17}$$

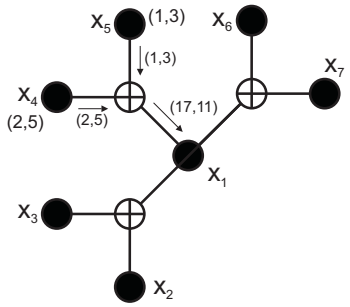


Figure A.14: Example of weights propagation through a function node

which have to be simultaneously satisfied from any valid codeword.

This code can also be described by means of its Tanner graph [134] reported in Fig.A.13, where the black circles called *variable nodes* represent the codeword symbols x_i, with $i = 1,2,\cdots,7$, and the so-called *function nodes* $\oplus$ indicate the parity relations (A.17).

After receiving a codeword $\mathbf{y}$, we assign a weight to each possible alphabet symbol a_k, where $a_k \in \{0,1\}$ for a binary code:

$$w_k(a_k) = P(y_k|x_k = a_k) \tag{A.18}$$

Thus, the weight of a codeword can be calculated as:

$$w(\mathbf{x}) = \prod_k w_k(x_k) = P(\mathbf{y}|\mathbf{x}) \tag{A.19}$$

For each alphabet symbol a_k, the sum of the weights of all those codewords that present the value a_k in correspondence of the variable x_k, given by:

$$\sum_{\mathbf{x}\in\mathbf{X}:x_k=a_k} w_k(x_k)w(\mathbf{x}) \tag{A.20}$$

is proportional to the APP value (A.14) according to:

$$P(x_k = a_k|\mathbf{y}) \propto \sum_{\mathbf{x}\in\mathbf{X}:x_k=a_k} P(\mathbf{y}|\mathbf{x}) \tag{A.21}$$

The most probable sent codeword is chosen as the one that maximizes the APP.

The weights associated with each variable node can be seen as messages that propagate thought the graph. The messages passing can be scheduled so as to make the calculation of the maximum APP simple and automatic. The first step consists in assigning to the graph leaf nodes, that are the variable nodes connected to just one function node,

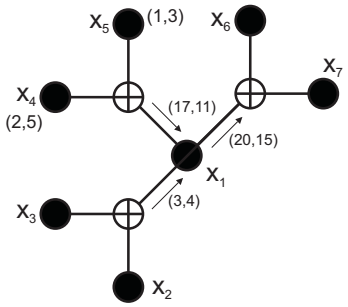


Figure A.15: Example of weights propagation through a variable node

the weights corresponding to all the possible variable nodes values. These weights are then passed on to the closest function node. Each function nodes receives the leaf nodes weights as inputs and calculates the output weights according to the the XOR gate function. In the example if Fig.A.14, the two function node inputs are

$$\begin{cases} (w_5(0), w_5(1)) = (1,3) \\ (w_4(0), w_4(1)) = (2,5) \end{cases}$$

while its output is given by:

$$\begin{cases} w(0) = w_4(0) \cdot w_5(0) + w_4(1) \cdot w_5(1) = 2 + 15 = 17 \\ w(1) = w_4(0) \cdot w_5(1) + w_4(1) \cdot w_5(0) = 5 + 6 = 11 \end{cases} \Rightarrow (17,11)$$

Thus the variable node x_1 receives the three weights coming from the three neighbor function nodes. The local weight of the node x_1 is then propagated towards each of the function nodes after being multiplied by the sum of all the weights of the incoming branches but the one towards which it is propagated, as described in Fig.A.15. This sum-product process is repeated until reaching the leaf nodes. At the end we obtain a pair of weights (one incoming and one outgoing) for each graph edge. The total weight associated with each edge, that is the weight of the corresponding node, is given by the sum of these two weights. Thus, if we chose for each node the grater weight among all the ones obtained for all the possible alphabet symbols, we have a MAP decision.

Thus the Sum-Product algorithm computations, as described by Gallager for decoding of LDCP codes [124], can be summarized in the following steps:

initialization: each variable node x_k is initialized with the conditional probabilities $P(y_k|x_k) = \lambda y_k|x_k = \lambda_{x_k}^{(0)}$, $\forall k = 1, \cdots, q$ where q indicates the alphabet symbols number

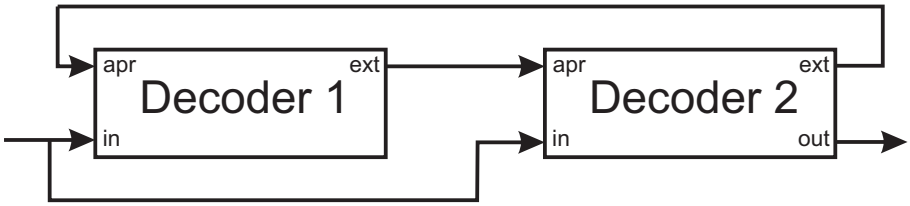


Figure A.16: Iterative decoding scheme

nth iteration, function-to-variable nodes: for each variable node x_k, we compute the $\oplus$ of all the messages coming from the neighbor variable nodes but x_k, that is $\forall r$: $h_{r,k} = 1$:

$$\lambda^{(n)}_{\oplus_r \to x_k} = \bigoplus_{\forall c: h_{r,c}=1, c \neq k} \lambda^{(n-1)}_{x_c \to \oplus_r} \tag{A.22}$$

nth iteration, variable-to-function nodes: for each function node $\oplus_k$, we need to compute the values of all the variable nodes connected to the function node, that is $\forall c : h_{k,c} = 1$:

$$\lambda^{(n)}_{x_c \to \oplus_k} = \lambda y_k | x_k \odot \left(\bigodot_{\forall r: h_{r,c}=1, r \neq k} \lambda^{(n)}_{\oplus_r \to x_c} \right) \tag{A.23}$$

where $\odot$ corresponds to the logical function $\overline{EXOR}$.

final decision: after a fixed iteration number N, the sent codeword symbols are computed according to:

$$\lambda_{\hat{x}_k} = \lambda y_k | x_k \odot \left(\bigodot_{\forall r: h_{r,c}=1} \lambda^{(N)}_{\oplus_r \to x_c} \right) \tag{A.24}$$

A.3.4 Iterative Decoding

As the Sum-Product algorithm complexity increases linearly with the code states number, it can not be used for Turbo code decoding. Indeed, due to the interleaver presence, the Turbo codes states number is very high and besides difficult to compute.

Concatenated codes can be efficiently decoded by means of a suboptimum algorithm, whose complexity is almost independent on the interleaver length. This algorithm, called *iterative*, shows performance close to the Shannon limit [135, 136], even if its effectiveness has not been analytically proved yet.

The iterative decoding scheme is reported in Fig.A.16. The Turbo decoder uses two MAP decoders, one for each constituent code. Each MAP decoder has two inputs, the

channel output and an *a priori* probability generated by the other decoder, and generates at its output an *extrinsic* information, that is the information on the received symbols known the constituent code. This extrinsic information constitutes the a priori probability for the other decoder, as it is not correlated with the knowledge of its own constituent code.

The decoding algorithm As already pointed out, each decoder has to take a MAP decision on the bases of two inputs, the channel information and the extrinsic probabilities generated by the other decoder. Thus, the decoder has to compute for all the possible codewords the corresponding APP and then chose among them the user word $\hat{\mathbf{u}}$ with the maximum APP, that is:

$$\hat{u}_k = \max_i [APP(k,i)] \tag{A.25}$$

where k is the symbol under analysis index, i represents every possible alphabet symbol value and $APP(k,i)$ is defined by:

$$APP(k,i) \doteq p(\mathbf{r}_1,\mathbf{r}_2)|u_k = 1) = \sum_{\mathbf{u}:u_k=i} p(\mathbf{r}_1|c_1(\mathbf{u}))p(\mathbf{r}_2)|c_2(\mathbf{u}))p_a(\mathbf{u}) \tag{A.26}$$

$$p(\mathbf{r}_1|c_1(\mathbf{u})) = \prod_{j=1}^{N_1} p(r_{1j}|c_{1j}(\mathbf{u})) \tag{A.27}$$

$$p(\mathbf{r}_2|c_2(\mathbf{u})) = \prod_{m=1}^{N_2} p(r_{2m}|c_{2m}(\mathbf{u})) \tag{A.28}$$

$$p_a(\mathbf{u}) = \prod_{l=1}^{K} p_a(u_l) \tag{A.29}$$

where $\mathbf{u}$ is the transmitted codeword, $\mathbf{r}_1$ and $\mathbf{r}_2$ indicate the received codewords relative to the two codes, $c_1(\cdot)$ and $c_2(\cdot)$ are the ideal decoding functions of the two constituent codes, $p_a(\cdot)$ represents the a priori probability and K, N_1 and N_2 are the symbols number of the user word and of the two codes codeword respectively.

According to Berroux [2], equations (A.28) and (A.29) can be expressed as function of the single codeword symbols instead of the whole codeword $\mathbf{u}$. As a consequence, we can write equation (A.27) as a product of functions defined on the single symbol as well, that is:

$$APP(k,i) = \tilde{P}_{1k}(i) \cdot \tilde{P}_{2k}(i) \cdot p_a(i) \tag{A.30}$$

where $\tilde{P}_{1k}(i)$ and $\tilde{P}_{2k}(i)$ solve an opportunely derived non-linear system [2]. For the sake of brevity, only the solutions are reported hereafter:

$$\tilde{P}_{1k}(i) = \sum_{\mathbf{u}:u_k=i} p(\mathbf{r}_1|c_1(\mathbf{u})) \prod_{l\neq k} \tilde{P}_{2l}(u_l) p_a(u_l) \tag{A.31}$$

$$\tilde{P}_{2k}(i) = \sum_{\mathbf{u}:u_k=i} p(\mathbf{r}_2|c_2(\mathbf{u})) \prod_{l\neq k} \tilde{P}_{1l}(u_l) p_a(u_l) \tag{A.32}$$

that can be computed as:

$$\begin{aligned}
\tilde{P}_{1k}^{(0)}(i) &= 1, k = 1, \cdots, K \\
&\vdots \\
\tilde{P}_{1k}^{(m)}(i) &= \sum_{\mathbf{u}:u_k=i} p(\mathbf{r}_1|c_1(\mathbf{u})) \prod_{l\neq k} \tilde{P}_{2l}(u_l) p_a(u_l), k = 1, \cdots, K \\
\tilde{P}_{2k}^{(m)}(i) &= \sum_{\mathbf{u}:u_k=i} p(\mathbf{r}_2|c_2(\mathbf{u})) \prod_{l\neq k} \tilde{P}_{1l}(u_l) p_a(u_l), k = 1, \cdots, K
\end{aligned} \tag{A.33}$$

Log-Likelihood Ratio Algorithm If the symbol alphabet is binary, it can be convenient to use an additive version of the decoding algorithm previously described, especially in digital implementations where multiplications are expensive to implement. The additive version of the iterative decoding algorithm works on the so called *Log-Likelihood Ratio, LLR*, which are defined as:

$$\begin{aligned}
L_k(APP) &\doteq \log \frac{\sum_{\mathbf{u}:u_k=0} p(\mathbf{r}_1|c_1(\mathbf{u})) p(\mathbf{r}_2|c_2(\mathbf{u})) p_a(\mathbf{u})}{\sum_{\mathbf{u}:u_k=1} p(\mathbf{r}_1|c_1(\mathbf{u})) p(\mathbf{r}_2|c_2(\mathbf{u})) p_a(\mathbf{u})} \\
L_k &\doteq \log \frac{p(r_k|0)}{p(r_k|1)} \\
L_{1j} &\doteq \log \frac{p(r_{1j}|0)}{p(r_{1j}|1)}, j = 1, \cdots, N_1 \\
L_{2m} &\doteq \log \frac{p(r_{2m}|0)}{p(r_{2m}|1)}, m = 1, \cdots, N_2 \\
L_a &\doteq \log \frac{p_a(0)}{p_a(1)} \\
\pi_{1l} &\doteq \log \frac{\tilde{P}_{1l}(0)}{\tilde{P}_{1l}(1)} \\
\pi_{2m} &\doteq \log \frac{\tilde{P}_{2m}(0)}{\tilde{P}_{2m}(1)}
\end{aligned} \tag{A.34}$$

After some computations, for which we refer to the literature [137], we obtain the desired formula:

$$L_k(APP) = \pi_{1k} + \pi_{2k} + L_a \tag{A.35}$$

Figure A.17: Soft-Input Soft-Output module

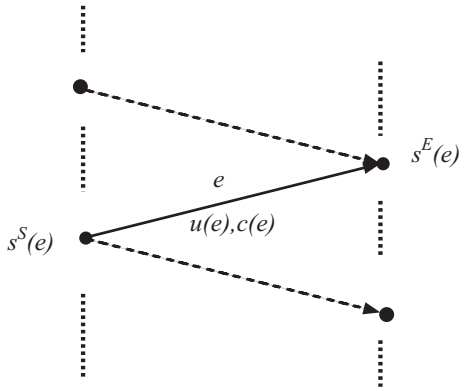


Figure A.18: An edge of the trellis section

where π_{1k} and π_{2k} can be calculated with an iterative process, as described by (A.3.4). It is worth to notice how, using the additive version of the iterative algorithm, only one value has to be propagated as opposite to the two required by its multiplicative version.

A.3.5 Soft-Input Soft-Output Algorithm

The iterative decoding algorithm, as is it stated in Sec.A.3.4, can not be efficiently used for Turbo codes decoding because it requires two MAP decoders, whose complexity and memory grows linearly with the decoding latency.

In order to overtake this limit, a novel version of the iterative algorithm, called *Soft-Input Soft-Output, SISO* has been introduced [138]. The SISO algorithm uses more modules, one for each constituent code, with real or *soft* input and output values.

An example of a SISO module is reported in Fig.A.17. Both its two inputs, $\mathbf{P}(c;I)$ and $\mathbf{P}(u;I)$ and two outputs, $\mathbf{P}(c;O)$ and $\mathbf{P}(u;O)$, represent probability distributions.

To illustrate the decoding algorithm, first presented by Bahl, Cocke, Jelinek and Raviv in 1974 [139], whence the name BCJR, we consider a code trellis section, as the one shown in Fig.A.18. Every trellis section is characterized by:

- a set of N states $\mathcal{S} = \{s_1, \cdots, s_N\}$. The state of the trellis at time k is $S_k = s$, with

$s \in \mathcal{S}$;

- a set of $N \cdot N_I$ edges obtained by the Cartesian product

$$\mathcal{E} = \mathcal{S} \times \mathcal{U} = \{e_1, \cdots, e_{N \cdot N_I}\}$$

which represent all possible transitions between trellis states.

Indeed, the following functions are associated to each edge $e \in \mathcal{E}$:

- the original state $s^S(e)$ (the projection of e onto $\mathcal{S}$);
- the final state $s^E(e)$;
- the input symbol $u(e)$ (the projection of e onto $\mathcal{U}$);
- the output symbol $c(e)$.

In the case of systematic encoders the pair $(s^S(e), c(e))$ also identifies the edge since $u(e)$ is uniquely determined by $c(e)$. In the following, we consider only the case in which the pair $(s^S(e), u(e))$ uniquely identifies the final state $s^E(e)$; this assumption is always verified, as it is equivalent to say that, given the initial trellis state, there is a one-to-one correspondence between input sequences and state sequences, a property required for the code to be uniquely decodable.

We also indicate with $\mathbf{P}(c;I)$ and $\mathbf{P}(u;I)$ the code and the input a priori distribution respectively, while $\mathbf{P}(u;O)$ is the user data a posteriori distribution.

If k indicates the discrete index $(k \in \{1, \cdots, n\})$, the BCJR algorithm can be divided into two steps:

- at time k, the SISO output probability distributions are computed according to

$$\tilde{P}_k(c;O) = \tilde{H}_c \sum_{e:c(e)=c} A_{k-1}[s^S(e)] P_k[u(e);I] P_k[c(e);I] B_k[s^E(e)] \tag{A.36}$$

$$\tilde{P}_k(u;O) = \tilde{H}_u \sum_{e:u(e)=u} A_{k-1}[s^S(e)] P_k[u(e);I] P_k[c(e);I] B_k[s^E(e)] \tag{A.37}$$

- the quantities $A_k(\cdot)$ and $A_k(\cdot)$ are obtained through the so-called *forward* and *backward* recursions, respectively, as:

$$A_k(s) = \sum_{e:s^S(e)=s} A_{k-1}[s^S(e)] P_k[u(e);I] P_k[c(e);I] \quad , k = 1, \cdots, n-1 \tag{A.38}$$

$$B_k(s) = \sum_{e:s^S(e)=s} B_{k+1}[s^S(e)] P_{k+1}[u(e);I] P_{k+1}[c(e);I] \quad , k = n-1, \cdots, 1 \tag{A.39}$$

with initial conditions

$$A_0(s) = \begin{cases} 1 & \text{if } s = S_0 \\ 0 & \text{otherwise} \end{cases} \tag{A.40}$$

$$B_n(s) = \begin{cases} 1 & \text{if } s = S_0 \\ 0 & \text{otherwise} \end{cases} \tag{A.41}$$

The two quantities $\tilde{H}_c$ and $\tilde{H}_u$ are normalization constants defined as:

$$\tilde{H}_c \ : \ \sum_c \tilde{P}_k(c;O) = 1 \tag{A.42}$$

$$\tilde{H}_u \ : \ \sum_u \tilde{P}_k(u;O) = 1 \tag{A.43}$$

$P_k[c(e);I]$ in (A.36) and $P_k[u(e);I]$ in (A.37) are constant with respect to the correspondent sum terms. Thus, defining

$$P_k(c;O) \doteq H_c \frac{\tilde{P}_k(c;O)}{P_k(c;I)} \ , \ H_c : \sum_u P_k(c;O) = 1 \tag{A.44}$$

$$P_k(u;O) \doteq H_u \frac{\tilde{P}_k(u;O)}{P_k(u;I)} \ , \ H_u : \sum_u P_k(u;O) = 1 \tag{A.45}$$

it can be easily verified that

$$\begin{aligned} P_k(c;O) &= H_c \tilde{H}_c \sum_{e:c(e)=c} A_{k-1}[s^S(e)] P_k[c(e);I] B_k[s^E(e)] \\ P_k(u;O) &= H_u \tilde{H}_u \sum_{e:u(e)=u} A_{k-1}[s^S(e)] P_k[c(e);I] B_k[s^E(e)] \end{aligned} \tag{A.46}$$

In literature, $P_k(c;O)$ and $P_k(u;O)$ are called *extrinsic information*, as they represent the added value by the SISO module to the *a priori* distributions $P_k(c;I)$ and $P_k(u;I)$.

It is worth to notice how the implementation of the two equations (A.46) requires less hardware resources than that of (A.3.4), that is the reason way we used the SISO decoding algorithm for our decoder implementation.

As for the MAP algorithm, an additive version of the SISO algorithm can be easily

derived by considering the logarithm of all the quantities previously defined, that is:

$$\begin{aligned}
\pi(c:I) &\doteq \log[P_k(c:I)] \\
\pi(u:I) &\doteq \log[P_k(u:I)] \\
\pi(c:O) &\doteq \log[P_k(c:O)] \\
\pi(u:O) &\doteq \log[P_k(u:O)] \\
\alpha_k(s) &\doteq \log[A_k(s)] \\
\beta_k(s) &\doteq \log[B_k(s)]
\end{aligned} \tag{A.47}$$

The forward and backward recursions can be written as:

$$\alpha_k(s) = \log\left[\sum_{e:s^E(e)=s} exp\{\alpha_{k-1}[s^S(e)]\pi_k[u(e);I]\pi_k[c(e);I]\}\right] \tag{A.48}$$

$$\beta_k(s) = \log\left[\sum_{e:s^E(e)=s} exp\{\beta_{k+1}[s^S(e)]\pi_{k+1}[u(e);I]\pi_{k+1}[c(e);I]\}\right] \tag{A.49}$$

where $k \in \{1,\cdots,n-1\}$ for both equations. The initial conditions are given by:

$$\alpha_0(s) = \begin{cases} 0 & \text{if } s = S_0 \\ -\infty & \text{otherwise} \end{cases} \tag{A.50}$$

$$\beta_n(s) = \begin{cases} 0 & \text{if } s = S_0 \\ -\infty & \text{otherwise} \end{cases} \tag{A.51}$$

The algorithm computation can be further simplified considering the approximation

$$\log\left[\sum_{i}^{L} exp(a_i)\right] \simeq \max_{i=1,\cdots,L} a_i \tag{A.52}$$

which leads to good results for medium to high signal-to-noise power ratio. Using this simplification we can avoid to compute exponential and logarithm functions, which are quite complex to implement in the digital domain.

However, in the analog domain the implementation of the non additive SISO algorithm is straightforward.

A.4 Analog Decoding

In 1998, Hagenauer [140] showed how analog networks could be efficiently used to decode binary Turbo codes. The main attractiveness of the proposed solution was the possibility to get rid of the iteration cycles presented in all the Turbo codes decoding schemes.

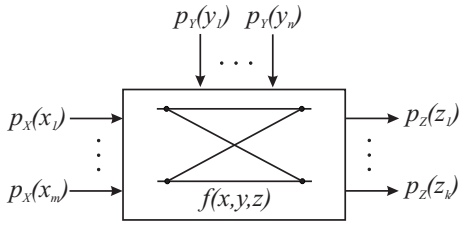


Figure A.19: Analog decoders building block

Moreover, he highlighted how there is a simple and straightforward correspondence between the Sum-Product modules and simple analog circuits.

In his work [6], Loeliger proposed an implementation of such a decoder by means of analog VLSI based on Sum-Product module realized with simple analog transistor circuits. In particular, he showed how the whole family of Sum-Product modules can be obtained with small variations of the same basic analog circuit. Thus, the decoder becomes an asynchronous analog network that, after a certain time evolution, settles towards a state that corresponds to the decoded information data. The iterative cycles of the decoding algorithm are then substituted by the continuous time feedbacks of the analog circuit.

Furthermore, analog decoders proved to be robust against all analog circuits non-idealities and able to overcome the performance of their digital counterparts by a factor up to 100 in terms of decoding speed and power consumption.

A.4.1 Sum-Product Module

The building blocks of an analog decoder are the Sum-Product modules. A generic Sum-Product module, as the one shown in Fig.A.19, computes the output probability distribution P_Z, with $\mathcal{Z} \doteq \{z_1, \cdots, z_k\}$, from the two input probability distributions P_X and P_Y, with $\mathcal{X} \doteq \{x_1, \cdots, x_m\}$ and $\mathcal{Y} \doteq \{y_1, \cdots, y_n\}$, according to

$$p_Z(z) = \gamma \sum_{x \in X} \sum_{y \in Y} p_X(x) p_Y(y) f(x,y,z) \; \forall z \in Z \tag{A.53}$$

where f is a function from $\mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$ into $\{0,1\}$ and where γ is an appropriate scale factor that does not depend on z.

When implemented with analog circuits, a probability distribution p_X is represented by means of a current vector $(I_{x_1}, \cdots, I_{x_k})$ such as

$$\sum_i I_{x_i} = I_x \geq 0 \tag{A.54}$$

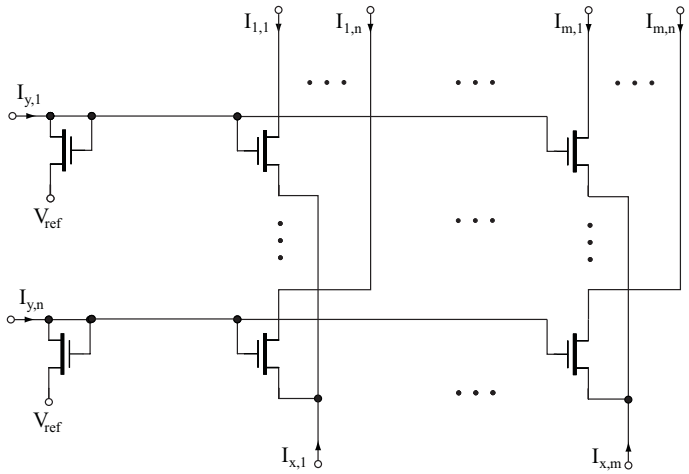


Figure A.20: Basic circuit for Sum-Product module implementation

On the other hand, every current vector $(I_1, \cdots, I_n)$ with non-negative and all non-zero elements, can be seen as a probability distribution p_Y with alphabet $\mathcal{Y} \doteq \{y_1, \cdots, y_n\}$, whose values are equal to $p_Y(y_i) \doteq I_i/(I_1 + \cdots I_n)$. The analog building block proposed by Loeliger and shown in Fig.A.20 is derived from the well-known *Gilbert multiplier*, from the name of the person who proposed it for the first time in 1968 [56]. Even if implemented with bipolar transistors in its original version, the Gilbert multiplier can also be realized in CMOS technology by means of MOS transistors working in their exponential or weak inversion region.

The Gilbert cell behavior is described by the following input-output equation:

$$I_{z,i,j} = I_z \cdot \frac{I_{x,i}}{I_x} \cdot \frac{I_{y,j}}{I_y} \tag{A.55}$$

where $I_x \doteq \sum_i I_{x,i}$, $I_y \doteq \sum_j I_{y,j}$ and $I_x \doteq \sum_i \sum_j I_{z,i,j} = I_x$. Thus the Sum-Product module computes the products of the two probability mass functions $p_X(i) \doteq I_{x,i}/I_x$ and $p_Y(j) \doteq I_{y,j}/I_y$.

A.4.2 Soft-gates

The different computational modules needed for analog decoding can be easily obtained from the basic Sum-Product cell by an adequate choice of the $\{0,1\}$ valued function f.

In particular, if the function f is defined as $f(x,y,z) = 1$ if and only if $z = x \oplus y$ with $\oplus$ denoting the standard module-2 addition and $f(x,y,z) = 0$ otherwise, the Sum-Product module becomes a *soft-XOR* gate. If p_X and p_Y are the distributions of two independent

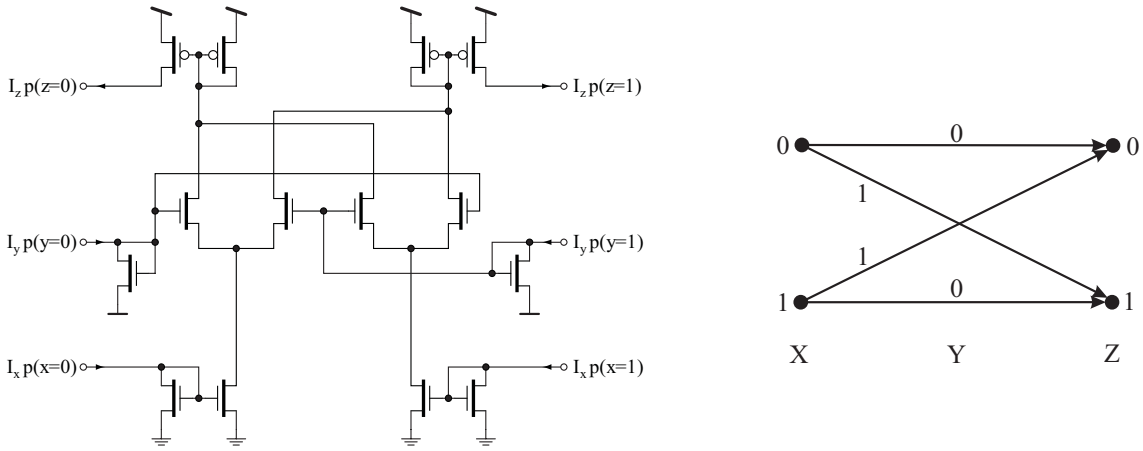


Figure A.21: *Soft-XOR* trellis and circuit implementation

binary random variables X and Y, respectively, then the distribution of $X \oplus Y$, p_Z, is given by:

$$\begin{bmatrix} p_z(0) \\ p_z(1) \end{bmatrix} = \begin{bmatrix} p_x(0)p_y(0) + p_x(1)p_y(1) \\ p_x(0)p_y(1) + p_x(1)p_y(0) \end{bmatrix} \tag{A.56}$$

The corresponding circuit implementation, together with the trellis diagram, is reported in Fig.A.21.

It is worth to highlight the biunique correspondence between the $\{0,1\}$ valued function f and its trellis diagram, which uniquely defines the Sum-Product module.

If the function f is equal to 1 if and only if $x = y = z$ and $f(x,y,z) = 0$ otherwise, the Sum-Product module realizes an *equal gate*. The output probability distribution can be computed as:

$$\begin{bmatrix} p_z(0) \\ p_z(1) \end{bmatrix} = \gamma \begin{bmatrix} p_x(0)p_y(0) \\ p_x(1)p_y(1) \end{bmatrix} \tag{A.57}$$

where γ is a scale factor to satisfy $p_Z(0) + p_Z(1) = 1$.

Both soft gates can be obtained from a generic Sum-Product module, as the one shown in Fig.A.22, by a proper configuration of the *interconnections*. The paths without a corresponding branch in the trellis are connected together to a *dummy* transistor (not shown in the figure). The two bias voltages and the cell bias current are chosen so as to allow the transistors to work under their weak inversion region.

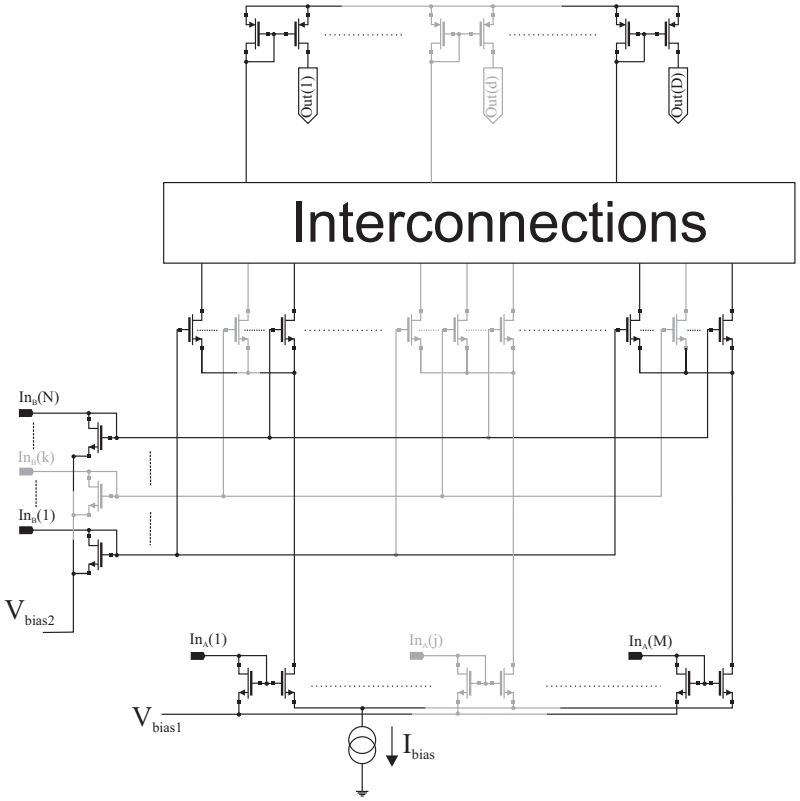


Figure A.22: Transistor network implementation of a general Sum-Product module

Bibliography

[1] R. King, “Averting the it energy crunch,” Online, 2007.

[2] C. Berroux, A. Glavieux, and P. Thitimjshima, “Near Shannon limit error-correcting coding and decoding: Turbo codes,” in *IEEE Proc. of International Conference on Communications*, Geneva, Switzerland, May 1993, pp. 1064–1070.

[3] *Third Generation Patnership Project (3GPP)*, 3G TS 25.212, v3.5.0, Multiplexing and Channel Coding (FDD), 2000.

[4] *Digital Video Bradcasting (DVB). Interaction channel for satellite distribution systems*, ETSI EN 301 790, v1.2.2, 2000.

[5] G. Aiello and G. Rogerson, “Ultra-wideband wireless systems,” *IEEE Microwaves Magazine*, vol. 4, no. 2, pp. 36–47, June 2003.

[6] H.-A. Loeliger, F. Lustenberg, M. Helfenstein, and F. Taröy, “Probability propagation and decoding in analog VLSI,” *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 837–843, February 2001.

[7] J. Hagenauer and M. Winklhofer, “The analog decoder,” in *Information Theory, 1998. Proceedings. 1998 IEEE International Symposium on*, August 1998, p. 145.

[8] F. Lustenberg, M. Helfenstein, G. Moschytz, H.-A. Loeliger, and F. Taröy, “All-analog decoder for a binary (18,9,5) tail-biting trellis code,” in *Proc. Eur. Solid State Circuits Conf.*, Duisburg, Germany, September 1999, pp. 362–365.

[9] M. Moerz, 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.*, San Francisco, CA, February 2000.

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

[11] M. Moerz, “Analog sliding window decoder core for mixed signal Turbo decoder,” in *Proc. ITG Conf. Source and Channel Coding*, February 2004, pp. 63–70.

[12] M. Arzel, F. Seguin, C. Lahuec, and M. Jézéquel, "Semi-iterative analog Turbo decoding," in *Proc. Int. Symp. Circ. and Syst., ISCAS'06*.

[13] V. Gaudet and P. 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, no. 11, pp. 122–131, January 2004.

[14] D. Vogrig, A. Gerosa, A. Neviani, A. G. i Amat, G. Montorsi, and S. Benedetto, "A 0.35-μm CMOS analog Turbo decoder for 40-bit rate 1/3 UMTS channel code," *IEEE J. Solid-State Circuits*, vol. 40, no. 3, pp. 735–762, March 2005.

[15] A. G. i Amat, G. Montorsi, and F. Vatta, "Analysis and design of rate compatible serial concatenated convolutional code," in *Proc. IEEE Int. Symp. Inform. Theory, ISIT'05*, September 2005, pp. 607–611.

[16] C. Winstead, J. Dai, W. Kim, S. Little, Y.-B. Kim, C. Mayers, and C. Schlegel, "Analog MAP decoder for (8,4) Hamming code in subthhreshold CMOS," in *Proc. of Advanced Research in VLSI Conf.*, March 2001, pp. 132–147.

[17] G. Montorsi and S. Benedetto, "Design of fixed-point iterative decoders for concatenated codes with interleavers," *IEEE Journal on Select Areas in Communications*, vol. 19, no. 5, pp. 871–882, May 2001.

[18] C. Winstead, C. Myers, C. Schlegel, and R. Harrison, "Analog decoding of product codes," in *Proc. of Information Theory Workshop*, Cairns, Australia, September 2001, pp. 131–133.

[19] G. Haller and B. Wooley, "A 700-MHz switched-capacitor analog waveform sampling circuit," *IEEE J. Solid-State Circuits*, vol. 29, no. 4, pp. 500–508, 1994.

[20] J. Krupar, R. Srowik, J. Schreiter, A. Graupner, R. Schüffny, and U. Jörges, "Minimizing charge injection errors in high-precision, high-speed SC-circuits," in *Proc. Int. Symp. Circ. and Syst., ISCAS'01*, vol. 1, May 2001, pp. 727–730.

[21] D. Johns and K. Martin, *Analog Integrated Circuit Design*, J. Wiley and Sons, Eds., 1997.

[22] J. Rabay, A. Chandrakasan, and B. Nikolić, *Digital Integrated Circuits*, second edition ed., C. Sodini, Ed. Prentice Hall, 2003.

[23] J. Deveugele, G. V. der Plas, M. Steyaert, G. Gielen, and W. Sansen, "A gradient-error and edge-effect tolerant switching scheme for a high-accuracy DAC," *IEEE Trans. on Circuits and Systems*, vol. 51, no. 1, pp. 191–195, 2004.

[24] S. Massaro, "Progettazione di una memoria RAM analogica per un decodificatore Turbo ibrido," Ph.D. dissertation, Universita' di Padova, Padova, 2005.

[25] M. Bickerstaff, D. Garret, T. Prokop, C. Thomas, B. Widdup, G. Zhou, L. Davis, G. Woodward, C. Nicol, and R.-H. Yan, "A unified Turbo/Viterbi channel decoder for 3GPP mobile wireless in 0.18μm CMOS," *IEEE J. Solid-State Circuits*, vol. 37, no. 11, pp. 1555–1564, 2002.

[26] P. Cappelletti, C. Golla, P. Olivo, and E. Zanoni, *Flash Memories*, K. A. Publishers, Ed. Kluwer Academic Publishers, 1999.

[27] R. Bez, E. Camarlenghi, A. Modelli, and A. Visconti, "Introduction to flash memory," *Proc. IEEE*, vol. 91, no. 4, pp. 489–502, April 2003.

[28] B. Ricco, G. Torelli, M. Lanzoni, A. Manstretta, H. Maes, D. Montanari, and A. Modelli, "Nonvolatile multilevel memories for digital applications," *Proc. IEEE*, vol. 86, no. 12, pp. 2399–2423, December 1998.

[29] S.-P. Sim, W. Kwon, C. Lee, J. I. Han, W. Lee, C. Jung, H. Lee, Y. Jang, S. Park, J. Park, C.-K. Park, K. Kim, and K. Kim, "A 90nm generation NOR flash multilevel cell (MLC) with 0.44μm^2/bit cell size," in *IEEE VLSI-TSA International Symposium on VLSI Technology*, April 2005, pp. 35–36.

[30] G. Servalli, D. Brazzelli, E. Camerlenghi, G. Capetti, S. Costantini, C. Cupeta, D. D. Simone, A. Ghetti, T. Ghilardi, P. Gulli, M. Mariani, A. Pavan, and R. Somaschini, "A 65nm NOR flash technology with 0.042μm^2 cell size for high performance multilevel application," in *IEEE International Electron Devices Meeting, 2005*, December 2005, pp. 849–852.

[31] S. Lee, Y.-T. Lee, W.-K. Han, D.-H. Kim, M.-S. Kim, S.-H. Moon, H. Cho, J.-W. Lee, D.-S. Byeon, Y.-H. Lim, H.-S. Kim, S.-H. Hur, and K.-D. Suh, "A 3.3 V 4 Gb four-level NAND flash memory with 90 nm CMOS technology," in *Proc. IEEE Int. Solid-State Circuits Conf.*, vol. 1, San Francisco, CA, February 2004, pp. 52–513.

[32] T. Hara, K. Fukuda, K. Kanazawa, N. Shibata, K. Hosono, H. Maejima, M. Nakagawa, T. Abe, M. Kojima, M. Fujiu, Y. Takeuchi, K. Amemiya, M. Morooka, T. Kamei, H.Nasu, C.-M. Wang, K. Sakurai, N. Tokiwa, H. Waki, T. Maruyama, S. Yoshikawa, M. Higashitani, T. Pham, Y. Fong, and T. Watanabe, "A 146-mm^2 8-Gb multi-level NAND flash memory with 70-nm CMOS technology," *IEEE J. Solid-State Circuits*, vol. 41, no. 1, pp. 161–169, January 2006.

[33] G. Atwwood, A. Fazio, D. Mills, and B. Reaves, "Intel strataflash^{tm} memory technology overview," in *Intel Technology Journal*, 4th Quarter 1997, pp. 1–8.

[34] P. Pavan, R. Bez, P. Olivo, and E. Zanoni, "Flash memory cell – An overview," *Proc. IEEE*, vol. 85, no. 8, pp. 1248–1271, August 1997.

[35] N. Shibata, H. Maejima, K. Isobe, K. Iwasa, M. Nakagawa, M. Fujiu, T. Shimizu, M. Honma, S. Hoshi, T. Kawaaiand, K. Kanebako, S. Yoshikawa, H. Tabata, A. Inoue, T. Takahashi, T. Shano, Y. Komatsu, K. Nagaba, M. Kosakai, N. Motohashi, K. Kanazawa, K. Imamiya, H. Nakai, M. Lasser, M. Murin, A. Meir, A. Eyal, and M. Shlick, "A 70 nm 16 Gb 16-level-cell NAND flash memory," *IEEE J. Solid-State Circuits*, vol. 43, no. 4, pp. 929–937, April 2008.

[36] R. Micheloni, R. Ravasio, A. Marelli, E. Alice, V. Altieri, A. Bovino, L. Crippa, E. D. Martino, L. D'Onofrio, A. Gambardella, E. Grillea, G. Guerra, D. Kim, C. Missiroli, I. Motta, A. Prisco, G. Ragone, M. Romano, M. Sangalli, P. Sauro, M. Scotti, and S. Won, "A 4Gb 2b/cell NAND flash memory with embedded 5b BCH ECC for 36MB/s system read throughput," in *ISSCC Dig. Tech. Papers*, February 2008, pp. 497–506.

[37] E. Berlekamp, *Algebraic Coding Theory*, rev. ed. ed., 1984.

[38] J. Massey, "Shift register syntesis and BCH decoding," *IEEE Trans. Inform. Theory*, vol. IT-15, no. 1, pp. 122–127, January 1969.

[39] R. Blahut, *Theory and Practice of Error Control Codes*, A.-W. P. Company, Ed. Reading, MA, 1983.

[40] R. Gallager, "Low density parity check codes," *IRE Trans. Inform. Theory*, vol. IT, pp. 21–28, January 1962.

[41] C. Winstead, J. Dai, S. Yu, C. Mayers, R. Harrison, and C. Schlegel, "CMOS analog MAP decoder for (8,4) Hamming code," *IEEE J. Solid-State Circuits*, vol. 39, no. 1, pp. 122–131, January 2004.

[42] S. Gregori, A. Cabrini, O. Khouri, and G. Torelli, "On-chip error correction techniques for new-generation flash memories," *Proc. IEEE*, vol. 91, no. 4, pp. 602–616, April 2003.

[43] A. Xotta, A. Gerosa, and A. Neviani, "CMOS implementation of all-analogue APP decoders: analysis of performance and limitations," *Electronics letters*, vol. 37, no. 25, pp. 1501–1503, December 2001.

[44] J. Cain, G. Clark, and J. Geist, "Punctured convolutional codes of rate $(n-1)/n$ and simplified maximum likehood decoding," *IEEE Trans. Inform. Theory*, vol. 25, no. 1, pp. 97–100, January 1979.

[45] D. Haccoun and G. Bégin, "High-rate punctured convolutional codes for Viterbi and sequential decoding," *IEEE Trans. Commun.*, vol. 37, no. 11, pp. 1113–1125, November 1989.

[46] G. Bégin, D. Haccoun, and C. Paquin, "Further results on high-rate punctured convolutional codes for Viterbi and sequential decoding," *IEEE Trans. Commun.*, vol. 38, no. 11, pp. 1922–1928, November 1990.

[47] A. G. i Amat, G. Montorsi, and S. Benedetto, "Design and decoding of optimal high-rate convolutional codes," *IEEE Trans. Inform. Theory*, vol. 50, no. 5, pp. 867–881, May 2004.

[48] C. Villa, D. Vimercati, S. Schippers, E. Confalonieri, M. Sforzin, S. Polizzi, M. L. Placa, C. Lisi, A. Magnavacca, E. Bolandrina, A. Martinelli, V. Dima, A. Scavuzzo, B. Calandrino, N. D. Gatto, M. Scardaci, F. Mastroianni, M. Pisasale, A. Geraci, M. Gaitiotti, and M. Sali, "A 125 MHz burst-mode flexible read-while-write 256 Mbit 2b/c 1.8V NOR flash memory," in *Proc. IEEE Int. Solid-State Circuits Conf.*, San Francisco, CA, February 2005.

[49] H.-L. Lou and C.-E. Sundberg, "Increasing storage capacity in multilevel memory cells by means of communications and signal processing techniques," *IEE Proc.-Circuits Devices Syst.*, vol. 147, no. 4, pp. 229–236, August 2000.

[50] F. Sun, S. Devarajan, K. Rose, and T. Zhang, "Design of on-chip error correction systems for multilevel NOR and NAND flash memories," *IET Circuits Devices Syst.*, vol. 1, no. 3, pp. 241–249, June 2007.

[51] G. Ungerboeck, "Channel coding with multilevel/phase signals," *IEEE Trans. Inform. Theory*, vol. IT-28, no. 1, pp. 55–67, January 1982.

[52] ——, "Trellis-coded modulation with redundant signal sets Part I: Introduction," *IEEE Commun. Mag.*, vol. 25, no. 2, pp. 5–11, February 1987.

[53] ——, "Trellis-coded modulation with redundant signal sets Part II: State of the art," *IEEE Commun. Mag.*, vol. 25, no. 2, pp. 12–21, February 1987.

[54] J. Hagenauer and P. Hoher, "A Viterbi algorithm with soft-decision outputs and its applications," in *Proc. 1989 IEEE Global Telecomm. Conf. (GLOBECOM'89)*, Dallas, Texas, November 1989, pp. 47.1.1–47.1.7.

[55] J. Hagenauer, P. Robertson, and L. Papke, "Iterative (TURBO) decoding of systematic convolutional codes with the MAP and SOVA algorithms," in *IEEE Proc. ITG Fachtagung*, Munich, Germany, October 1994, pp. 21–29.

[56] B. Gilbert, "A precise four-quadrant multiplier with subnanosecond response," *IEEE J. Solid-State Circuits*, vol. 3, no. 4, pp. 365–373, December 1968.

[57] A. Xotta, "Implementazione analogica di turbo decoder CMOS per applicazioni hard disk," Ph.D. dissertation, Universita' di Padova, Padova, 2002.

[58] C. Mead, *Analog VLSI and Neural Systems*, ser. Addison Wesley Computation and Neural Systems Series, A. Wesley, Ed. Addison Wesley, 1989.

[59] M. Polgrom, A. Duinmaijer, and A. Welbers, "Matching properties of MOS transistors," *IEEE J. Solid-State Circuits*, vol. 21, no. 5, pp. 1433–1440, October 1989.

[60] R. Gregor, "On the relationship between topography and transistor matching in an analog CMOS technology," *IEEE Trans. on Electron Devices*, vol. 39, no. 2, pp. 275–282, February 1992.

[61] F. Lustenberger and H.-A. Loeloger, "On mismatch errors in analog-VLSI error correcting decoders," in *IEEE Proc. of International Symposium on Circuits and Systems*, vol. 4, Sydney, Australia, May 2001, pp. 198–201.

[62] J. Dai, "Design methodology for analog VLSI implementations of error control decoders," Ph.D. dissertation, University of Utah, Salt Lake City, Utah, 2002.

[63] M. Frey, H.-A. Loeliger, and P. Merkli, "Analog circuits for symbol-likelihood computation," in *Proc. 13th IEEE Int. Conf. on Electronics, Circuits and Systems*, Nice, France, December 2006.

[64] *First Report and Order*, FCC 02-48, Federal Communications Commission, ET Docket 98-153, February 2002.

[65] *Second Report and Order and Second Memorandum Opinion and Order*, FCC 04-285, Federal Communications Commission, ET Docket 98-153, December 2004.

[66] *Code of Federal Regulations*, Federal Communications Commission, Title 47, Chapter 1, Part 15, October 2004.

[67] J. Han and C. Nguyen, "Ultra-wideband electronically tunable pulse generators," *IEEE Microwave and Wireless Components Letters*, vol. 14, no. 3, pp. 112–114, March 2004.

[68] J. Gerritis and J. Farserotu, "Wavelet generation circuit for UWB impulse radio applications," *Electronics Letters*, vol. 38, no. 25, pp. 1737–1738, December 2002.

[69] J. Foerster, E. Green, S. Somayazulu, and D. Leeper, "Ultra-wideband technology for short- or medium-range wireless communication," in *Intel Technology Journal*, 2nd Quarter 2001.

[70] I. Oppermann, M. Hämäläinem, and J. Iinatti, *UWB Theory and Applications*, Wiley, Ed., 2004.

[71] D. Wentzloff, R. Blazquez, F. Lee, B. Ginsburg, J. Powell, and A.P.Chandrakasan, "System design considerations for ultra-wideband communication," *IEEE Communications Magazine*, vol. 43, no. 8, pp. 114–121, August 2005.

[72] W. Kluge, F. Poegel, H. Roller, M. Lange, T. Ferchland, L. Dathe, and D. Eggert, "A fully integrated 2.4-GHz IEEE 802.15.4-compliant transceiver for ZigBee^{tm} applications," *IEEE J. Solid-State Circuits*, vol. 41, no. 12, pp. 2767–2775, December 2006.

[73] M. Camus, B. Butaye, L. Garcia, M. Sie, B. Pellat, and T. Parra, "A 5.4 mW/0.07 mm^2 2.4 GHz front-end receiver in 90 nm CMOS for IEEE 802.15.4 WPAN standard," *IEEE J. Solid-State Circuits*, vol. 43, no. 6, pp. 1372–1383, June 2008.

[74] N. Stanic, A. Balankutty, P. Kinget, and Y. Tsividis, "A 2.4-GHz ISM-band sliding-IF receiver with a 0.5-V supply," *IEEE J. Solid-State Circuits*, vol. 43, no. 5, pp. 1138–1145, May 2008.

[75] M. Ranjan and L. Larson, "A low-cost and low-power CMOS receiver front-end for MB-OFDM ultra-wideband systems," *IEEE J. Solid-State Circuits*, vol. 42, no. 3, pp. 592–601, March 2007.

[76] G. Cusmai, M. Brandolini, P. Rossi, and F. Svelto, "A 0.18μm CMOS selective receiver front-end for UWB applications," *IEEE J. Solid-State Circuits*, vol. 41, no. 8, pp. 1764–1771, August 2006.

[77] D. Wentzloff and A. Chandrakasan, "A 47pJ/pulse 3.1-to-5GHz all-digital UWB transmitter in 90nm CMOS," in *ISSCC Dig. Tech. Papers*, February 2007, pp. 118–591.

[78] S. Dubouloz, A. Rabbachin, S. de Rivaz, B. Denis, and L. Ouvry, "Performance analysis of low complexity solutions for UWB low data rate impulse radio," in *Proc. Int. Symp. Circ. and Syst., ISCAS'06*, May 2006.

[79] L. Stoica, A. Rabbachin, H. Repo, S. Tiuraniemi, and I. Oppermann, "An ultra-widebena system architecture for tag based wireless sensor networks," *IEEE Transactions on Vehicular Technology*, vol. 54, pp. 1632–1645, September 2005.

[80] "IEEE 802.15.4a wireless MAC and PHY specifications for LR-WPANs," Online, 2007.

[81] M. Win and R. Scholtz, "Impulse radio: how it woks," *IEEE Communications Letters*, vol. 2, no. 2, pp. 36–38, February 1998.

[82] W. Hirt, "The european UWB radio regulatory and standards framework: Overview and implications," in *Ultra-Wideband, 2007. ICUWB 2007. IEEE International Conference on*, September 2007, pp. 733–738.

[83] R. Scholtz and M. Win, "Impulse radio," in *Wireless Communications: TDMA vs. CDMA*, S. Glisic and P. Leppänen, Eds. Norwell, MA: Kluwer, 1997, pp. 245–264.

[84] M. Ho, L. Taylor, and G. Aiello, "UWB architecture for wireless video networking," in *Proc. IEEE Int. Conf. Consumer Electronics*, June 2001, pp. 18–19.

[85] R. Fontana, A. Ameti, E. Richley, L. Beard, and D. Guy, "Recent advances in ultra wideband communications systems," in *IEEE Conf. UWB Systems and Technologies Dig.*, May 2002, pp. 129–133.

[86] Y. Nakache and A. Molisch, "Spectral shaping of UWB signals for time hopping impulse radio," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 4, pp. 738–744, April 2006.

[87] E. Barajas, R. Cosculluela, D. Coutinho, M. Molina, D. M. amd J.L. Gonzalez, I. Cairo, S. Banda, and M. Ikeda, "A low-power template generator for coherent impulse-radio ultra wide-band receivers," in *IEEE International Conference on Ultra-Wideband*, September 2006, pp. 97–102.

[88] B. Jung, Y.-H. Tseng, J. Harvey, and R. Harjani, "Pulse generator design for UWB IR communication systems," in *IEEE International Symposium on Circuits and Systems*, vol. 5, May 2005, pp. 4381–4384.

[89] ——, "A novel CMOS/BiCMOS UWB pulse generator and modulator," in *IEEE International Microwave Symposium*, vol. 2, June 2004, pp. 1269–1272.

[90] ——, "Low-power full-band UWB active pulse shaping circuit using 0.18-μm CMOS technology," in *IEEE Radio Frequency Integrated Circuits Symposium*, June 2006, pp. 510–513.

[91] S. Iida, K. Tanaka, H. Suzuki, N. Yoshikawa, N. Shoji, B. Griffiths, D. Mellor, F. Hayden, I. Butler, and J. Chatwin, "A 3.1 to 5 GHz CMOS DSSS UWB transceiver for WPANs," in *Proc. IEEE Int. Solid-State Circuits Conf.*, San Francisco, CA, February 2005, pp. 214–594.

[92] Y. Choi, "Gated UWB pulse signal generation," in *IEEE Conference on Ultra Wideband Systems and Technologies*, May 2004, pp. 122–124.

[93] M. Welborn, "System considerations for ultra-wideband wireless networks," in *IEEE Radio and Wireless Conference*, August 2001, pp. 5–8.

[94] X. Chen and S. Kiaei, "Monocycle shapes for ultra wideband systems," in *IEEE International Symposium on Circuits and Systems*, vol. 1, May 2002, pp. 597–600.

[95] F. Ramirez-Mirales, "On the performance of ultra-wideband signals in gaussian noise and dense multipath," *IEEE Transactions on Vehicular Technology*, vol. 50, no. 1, pp. 244–249, January 2001.

[96] J. Proakis and M. Salehi, *Digital Communications*, 5th ed., M. Hill, Ed. McGraw-Hill Science Engineering, 2007.

[97] W. Jones and J. Dill, "The square root raised cosine wavelet and its relation to the meyer functions," *IEEE Transactions on Signal Processing*, vol. 49, no. 1, pp. 248–251, January 2001.

[98] S. Chennakeshu and G. Saulnier, "Differential detection of pi/4-shifted-dqpsk for digital cellular radio," *IEEE Transactions on Vehicular Technology*, vol. 42, no. 1, pp. 46–57, February 1993.

[99] H. Hashemi, "The indoor radio propagation channel," *Proc. IEEE*, vol. 81, no. 7, pp. 943–968, July 1993.

[100] S. Ghassemzadeh, L. Greenstein, A. Kavicic, T. Sveinsson, and V. Tarokh, "UWB indoor path loss model for residential and commercial buildings," in *IEEE Transactions on Vehicular Technology Conference*, vol. 5, October 2003, pp. 3120–3125.

[101] F. Lee and A. Chandrakasan, "A 2.5nJ/bit 0.65V pulsed UWB receiver in 90nm CMOS," *IEEE J. Solid-State Circuits*, vol. 42, no. 12, pp. 2851–2859, December 2007.

[102] Y. Tong, Y. Zheng, and Y.-P. Xu, "A coherent ultra-wideband receiver IC system for WPAN application," in *Ultra-Wideband, 2005. ICU 2005. IEEE International Conference on*, September 2005, pp. 60–64.

[103] H. Danneels, M. Verhelst, P. Palmers, W. Vereecken, B. Boury, W. Dehaene, M. Steyaert, and G. Gielen, "A low-power mixing DAC IR-UWB receiver," in *Proc. Int. Symp. Circ. and Syst., ISCAS'08*, vol. 5, May 2008, pp. 2607–2700.

[104] D. Barras, F. Ellinger, H. Jackel, and W. Hirt, "A robust front-end architecture for low-power UWB radio transceivers," in *Microwave Theory and Techniques, IEEE Transactions on*, vol. 54, no. 4, June 2006, pp. 1713–1723.

[105] A. Demosthenous and M. Panovic, "Low-voltage MOS linear transconductor/squarer and four-quadrant multiplier for analog VLSI," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 52, no. 9, pp. 1721–1731, September 2005.

[106] D. Wentzloff, F. Lee, D. Daly, M. Bhardwaj, P. Mercier, and A. Chandrakasan, "Energy efficient pulsed-UWB CMOS circuits and systems," in *Ultra-Wideband, 2007. ICUWB 2007. IEEE International Conference on*, September 2007, pp. 282–287.

[107] A. Rabbachin and I. Oppermann, "Synchronization analysis for UWB systems with a low-complexity energy collection receiver," in *Ultra Wideband Systems, 2004. Joint with Conference on Ultrawideband Systems and Technologies. Joint UWBST & IWUWBS. 2004 International Workshop on*, May 2004, pp. 288–292.

[108] M. Crepaldi, M. Casu, M. Graziano, and M. Zamboni, "A 1-bit synchronization algorithm for a reduced complexity energy detection UWB receiver," in *Ultra-Wideband, 2007. ICUWB 2007. IEEE International Conference on*, September 2007, pp. 703–708.

[109] M. Weisenborn and W. Hirt, "Robust non-coherent receiver exploiting UWB channel properties," in *Proc. of UWBST-IWUWBS*, May 2004, pp. 156–160.

[110] R. Blazquez, P. Newaskar, and A. Chandrakasan, "Coarse acquisition for ultra wideband digital receivers," in *Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03). 2003 IEEE International Conference on*, vol. 4, April 2003, pp. 137–140.

[111] D. Wentzloff and A. Chandrakasan, "Gaussian pulse generators for subbanded ultra-wideband transmitters," *IEEE Trans. Microwave Theory Tech.*, vol. 54, pp. 1647–1655, June 2006.

[112] F. Lee and A. Chandrakasan, "A 0.65-to-1.4nJ/burst 3-to-10 GHz UWB all-digital TX in 90nm CMOS for IEEE 802.15.4a," *IEEE J. Solid-State Circuits*, vol. 42, no. 12, pp. 2860–2869, December 2007.

[113] D. Wentzloff and A. Chandrakasan, "A 47pJ/pulse 3.1-to-5GHz all-digital UWB transmitter in 90nm CMOS," in *Proc. IEEE Int. Solid-State Circuits Conf.*, San Francisco, CA, February 2007, pp. 118–591.

[114] P. Mercier, D. Daly, and A. Chandrakasan, "A 19pJ/pulse UWB transmitter with dual capacitively-coupled digital power amplifiers," in *Radio Frequency Integrated Circuits Symposium, 2008. RFIC 2008. IEEE*, June 2008, pp. 47–50.

[115] I. Aoki, S. Kee, D. Rutledge, and A. Hajimiri, "Distributed active transformer - A new power-combining and impedance-transformation technique," *IEEE Trans. Microwave Theory and Techniques*, vol. 50, no. 1, pp. 316–331, January 2002.

[116] G. Matthaei, L. Young, and E. Jones, *Microwave filters, impedance-matching networks, and coupling structures*, McGraw-Hill, Ed., 1964.

[117] S. Mohan, C. Yue, M. del Mar Hershenson, S. Wong, and T. Lee, "Modeling and characterization of on-chip transforms," in *Proc. IEDM*, December 1998, pp. 531–534.

[118] J. Long, "Monolithic transformers for silicon RF IC design," *IEEE J. Solid-State Circuits*, vol. 35, no. 9, pp. 1368–1382, September 2000.

[119] Y. Cao, R. Groves, X. Huang, N. Zamdmer, J.-O. Plouchart, R. Wachnik, T.-J. King, and C. Hu, "Frequency-independet equivalent-circuit model for on-chip spiral inductors," *IEEE J. Solid-State Circuits*, vol. 38, no. 3, pp. 419–426, March 2003.

[120] I. Lai and M. Fujishima, "A new on-chip substrate coupled inductor model implemented with scalable expressions," *IEEE J. Solid-State Circuits*, vol. 41, no. 11, pp. 2491–2499, November 2006.

[121] C. Shannon, "A mathematical theory of communication," *Bell Systems Technical Journal*, vol. 27, pp. 379–423 (part I), 623–656 (part II), July 1984.

[122] S. Benedetto and G. Montorsi, "Unveiling Turbo codes: some results on parallel concatenated coding schemes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 409–428, March 1996.

[123] T. Richardson, A. Shokrollahi, and R. Urbanke, "Design of provably good low-density parity check codes," in *Proc. IEEE Int. Symp. on Inform, theory*, Sorrento, Italy, June 2000, p. 199.

[124] R. Gallager, "Low density parity check codes," *IRE Trans. Inform. Theory*, vol. IT, pp. 21–28, January 1962.

[125] R. Hamming, "Error detecting and error correcting code," *Bell Systems Technical Journal*, vol. 26, pp. 147–150, July 1950.

[126] C. Schlegel, *Trellis Coding*, 1997.

[127] H. Ma and J. Wolf, "On tail baiting convolutional codes," *IEEE Trans. Communications*, vol. COM-34, no. 2, pp. 104–11, February 1986.

[128] H. Imai and S. Hirakawa, "A new multilevel coding method using error-correcting codes," *IEEE Trans. Inform. Theory*, vol. IT-23, no. 3, pp. 371–377, May 1977.

[129] S. Benedetto and G. Montorsi, "Performance evaluation of Turbo codes," *Electronics Letters*, vol. 31, no. 3, pp. 163–165, February 1995.

[130] S. Benedetto, D. Divslar, G. Montorsi, and F. Pollara, "Serial concatenation of interleaved codes: Performance analyisis, design and iterative decoding," *IEEE Trans. Inform. Theory*, vol. 44, no. 3, pp. 909–926, May 1998.

[131] A. Viterbi, "Error bounds for convolutional codes and asymptotically optimum decoding algorithm," *IEEE Trans. Inform. Theory*, vol. 13, no. 2, pp. 260–269, April 1967.

[132] A. Viterbi, J. Wolf, E. Zehavi, and R. Padovani, "A pragmatic approach to trellis-coded modulation," *IEEE Commun. Mag.*, vol. 27, no. 7, pp. 11–19, July 1989.

[133] F. Kschischang, B. Frey, and H. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inform. Theory*, vol. 17, no. 2, pp. 498–519, February 2001.

[134] R. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inform. Theory*, vol. 27, no. 5, pp. 533–547, September 1981.

[135] S. Benedetto and G. Montorsi, "Iterative decoding of serially concatenated convolutional codes," *Electronics Letters*, vol. 32, no. 16, pp. 1186–1188, June 1996.

[136] S. Benedetto, D. Divslar, G. Montorsi, and F. Pollara, "Analysis, design and iterative decoding of double serial concatenated codes with interleavers," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 2, pp. 231–244, February 1998.

[137] S. Lin and D. Costello, *Error Control Coding: Fundamentals and Applications*, Prentice-Hall, Ed., 1983.

[138] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Trans. Inform. Theory*, vol. 12, no. 2, pp. 429–445, March 1996.

[139] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, vol. 20, pp. 284–287, March 1974.

[140] J. Hagenauer, "Decoding of binary codes with analog network," in *Proc. IEEE Int. Symp. on Information Theory*, San Diego, CA, February 1998, pp. 13–14.