You see, wire telegraph is a kind of a very, very long cat. You pull his tail in New York and his head is meowing in Los Angeles. Do you understand this? And radio operates exactly the same way: you send signals here, they receive them there. The only difference is that there is no cat.
Albert Einstein

University of Alberta

Analog FEC Decoders in Sub-100nm CMOS Technologies

by

Mohammad Meysam Zargham 

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Master of Science**.

in

Communications

Department of Electrical and Computer Engineering

Edmonton, Alberta
Fall 2008

Your file Votre référence
ISBN: 978-0-494-47452-5
Our file Notre référence
ISBN: 978-0-494-47452-5

صدای پای آب

I'm from Kashan
My life is not so bad
All I have is a loaf of bread, a bit of
intelligence and
a tiny amount of taste!
A mother better than the green leaf,
Friends, better than the running brook
And a God who is nearby

اهل کاشانم
روزگارم بد نیست،
تکه نانی دارم، خرده هوشی، سرسوزن ذوقی
مادری دارم، بهتر از برگ درخت،
دوستانی، بهتر از آب روان
وخدایی که در این نزدیکی است.

Life is a pleasant custom
Life owns wings as wide as death
and leaps as high as love
Life is the attraction of a hand that reaps
Life is the earth multiplied by our
heartbeats
Life is a simple and monotonous geometry
of breaths

زندگی رسم خوشایندی است.
زندگی بال و پری دارد با وسعت مرگ،
پرشی دارد اندازه عشق.
زندگی جذبه دستی است که می چیند.
زندگی "ضرب" زمین در ضربان دل ما،
زندگی "هندسه" ساده و یکسان نفسهاست.

Where ever I might be, let it be so,
The sky is mine
The window, thought, air, love, The earth
is mine
What significance does it make,
If mushrooms of loneliness
Sometimes grow?

هر کجا هستم، باشم،
آسمان مال من است.
پنجره، فکر، هوا، عشق، زمین مال من است.
چه اهمیت دارد
گاه اگر می رویند قارچهای غربت؟

I am content with an apple
or with the smell of a chamomile plant,
I am content with a mirror or even a pure
feeling.

من به سیبی خوشنودم
و به بوییدن یک بوته بابونه.
من به یک آینه، یک بستگی پاک قناعت دارم.

Let's remove the curtains
Let's allow our feeling to drink fresh air

پرده را برداریم:
بگذاریم که احساس، هوایی بخورد.

Let's undress
The brook is just a step away

رخت‌ها را بکنیم:
آب در یک قدمی است.

Parts of the poem from Sohrab Sepehri

سهراب سپهری

Preface

To my mom and dad for their love and support

Abstract

Recently, Low Density Parity Check codes have shown a near optimum performance with reasonable decoder complexity. Analog implementations of small decoders in the recent years have shown advantage in terms of speed and power over their digital counterparts. In this thesis, we are addressing the difficulties and different strategies for the implementation of large analog LDPC decoders in sub-100nm CMOS technologies. The leakage currents in nano-scale processes tend to reduce the dynamic range of operation. In addition, the errors due to mismatch and short channel effects tend to become more prominent. These effects introduce additional problems and non-idealities to the decoder. We have addressed these problems in detail at both the system level and the transistor level. In addition, several guidelines on transistor sizing are proposed. As a proof of concept, four fundamental LDPC nodes are implemented and sent for fabrication. The post-layout simulation results are available in this thesis.

Acknowledgements

I would like to thank Dr. Vincent Gaudet for his guidance, funding and support. I also wish to thank Dr. Christian Schlegel for his encouragement, late night discussions, funding and trust. Both of my supervisors were very nice to me, gave me scientific freedom and allocated their time whenever I needed them.

My special thanks to Jorge Perez for all his help during the tape out nights, useful discussions and patience. The back to back tape-outs could not have been possible without his help.

Many thanks to Dr. Mani Vaidyanathan for the productive discussions on device physics, everything I learnt from him and cheering up my day with his jokes. I would like to thank Dr. Duncan Elliott for everything I learnt from him during the high voltage chip implementation.

All colleagues at the HCDC laboratory always helped me out when needed. I especially want to thank Charmaine Ramdass for being so nice to me. I would also say "thank you" to Sheehan Khan for being a good friend and for all our long meaningless chats. People in the VLSI lab spent time to solve my problems with Cadence whenever I needed them. I want to specially thank Tyler Brandon, Leendert van den Berg and John Koob.

Special thanks to Dr. MacGregor for joining my defense committee My deepest gratitude goes to my family. My mom, dad and my two sisters.

Contents

List of Tables

List of Figures

List of Acronyms & Symbols

CMOS	Complementary metal oxide semiconductor	1
LDPC	low density parity check	2
MAP	maximum a posteriori	3
XOR	exclusive-or	7
SNR	signal to noise ratio	11
LLR	log-likelihood ratio	11
SP	sum-product	13
AWGN	additive white Gaussian noise	15
DE	density evolution	16
PDF	probability density function	16
BER	bit error rate	18
I/O	input/output	20
MOSFET	metal oxide semiconductor field effect transistor	21
nMOS	n-type metal oxide semiconductor	21
pMOS	p-type metal oxide semiconductor	21
MM9	MOS model 9	30
BSIM4	Berkeley short-channel IGFET model	30
SCE	short channel effect	31
STI	shallow trench isolation	33
DIBL	drain induced barrier lowering	35
BTBT	band to band tunneling	41
GIDL	gate induced drain leakage	41
ECB	electron tunneling from conduction band	42

Chapter 1

Introduction

Effectively communicating information forms an important and unavoidable part of our everyday life. The rapid flow of technological advancements has enabled the possibility of novel, fast and easy methods for communication. Wireless applications have recently received a special attention. Therefore, the demand for higher processing capabilities have increased exponentially over time. However, the lifetime of a wireless mobile device depends on its battery and unfortunately battery technology has not advanced much over the past few decades [2].

The first constraint in any design is usually its power budget. Due to practical constraints, power consumption is a major issue in applications such as monitoring devices, cochlear implants and handheld wireless devices [3] [4] . Depending upon the nature of these constraints, changing the batteries of these technologies can range from difficult to impossible [5]. Minimizing power consumption with low power circuit design techniques compensates for this drawback. By following such low power design strategies, we can offer enough energy to both extend the operating life of communication devices and still guarantee a high quality of service.

A large number of today's circuits are designed based on digital CMOS logic. These digital circuits offer many benefits but their relatively high power dissipation rapidly becomes an issue in power-constrained devices. Also the fact that this power consumption generates a significant amount of heat can be a problem. These notions have boosted the tendency towards analog implementation of circuits. Implementation of

different functions that exploit the basic physical characteristics of a microelectronic device typically saves hardware resources and is very power efficient. This analog implementation of circuits has proved to be ten to one hundred times more power efficient in hearing aids, speech recognition systems, analog decoders and echo cancellation circuits for high-speed cable channel modems [6] [7] [8].

Scaling transistor technologies has also shown to be an effective solution for power-constrained systems. According to Synopsys [9], the trend towards using smaller size transistors is growing rapidly. Circuits fabricated in sub-100nm CMOS processes are good candidates for ultra low power high-frequency systems. The majority of these circuits are digital circuits. The reason digital designers love scaling is the savings in area and power in addition to faster cicuits. While the cost for adapting a digital circuit to a new scaled technology is not very high, analog designers often do not benefit from scaling, and in many cases suffer from it.

Analog design deals with the physical quantities such as voltages and currents. Smaller processes provide less voltage headroom and current which makes the design process very challenging. In a mixed digital/analog design, such as most communication systems, the integration onto a so-called "System on a Chip" solution containing both analog and digital parts is very desirable. Therefore there is huge demand for analog design techniques in smaller processes, for the sake of integration, area and potential power or speed savings.

The key performance metric in a communication system is its data integrity. Unfortunately, noise corrupts the data that travels through a system. Reliability of data transmission has to be ensured through sophisticated error control coding techniques. For this reason, decoder circuitry that detects and corrects errors is an essential part of any receiver. Recently, low density parity check (LDPC) codes have shown near optimum performance with reasonable decoder complexity [10]. As a result, high throughput LDPC decoders have been increasing in demand. Both digital and analog implementation of these decoders have been reported [11] [12] [13]. Digital implementation of practical codes requires large chip area and a huge power budget. State of the art digital decoders consume power on the order of nano joules per decoded bit [13]. This amount of power dissipation quickly becomes unacceptable in

applications requiring gigabytes of data to be transferred each second. On the other hand, analog implementations of small decoders have shown promising advantage in terms of speed and power [14] [11] [12] [15].

Analog decoders exploit the physical properties of semiconductors to perform basic operations in decoding algorithms. For instance summation in an analog circuit is easily realized by tying two current-carrying wires together. Analog designers have employed these characteristics to design functional prototypes of turbo decoders [14] [16] [17], MAP decoders [11] [12] [18] and LDPC decoders [19] [20]. To the best of our knowledge, all the implemented analog decoders use CMOS or BiCMOS processes between 0.5μm and 0.18μm, whereas the digital signal processing and RF blocks in transceivers are constantly scaling down. Therefore analog decoders should scale with CMOS technologies to maintain integrability, and possibly consume less power.

In this thesis we address the problems, difficulties and strategies for implementing analog LDPC decoders in sub-$100nm$ CMOS technologies. In these technologies leakage currents tend to reduce the dynamic range of operation. As a result, signals in decoders get clipped and some of the information gets lost. Also, the effects of mismatch tend to become more prominent.

In the following chapters, we address these issues in detail at both the systems level and the transistor level. In addition, the sizing of transistors has a much greater contribution in the circuit performance of analog decoders. A detailed study of these effects is presented and we propose several design recommendations for transistor sizing. We also demonstrate functionality of the basic LDPC circuit modules in both $90nm$ and $65nm$ CMOS processes. As a proof of concept several fundamental decoder nodes were fabricated in these technologies. Our System-level simulations use the post layout simulation results directly from Cadence BSIM4 models to predict the behavior of a decoder. System-level simulation results confirm that while the performance of analog decoder circuit modules degrades in nano-scale CMOS processes, by following some design strategies from the fourth chapter, it is still possible to build functional decoders.

The thesis is organized as follows: The next Chapter reviews the required error control coding background. Chapter three reviews circuit background and fundamentals of

analog decoders. Chapter four discusses design issues in nano-scale CMOS circuits and provides guidelines for sub-threshold nano-scale analog circuit design. Chapter five presents the fundamental decoder nodes we have fabricated in 90nm and 65nm CMOS processes. Chapter six is dedicated to a system-level analysis of decoder performance, considering nano-scale circuit imperfections. The last chapter concludes the thesis and provides suggestions for future work.

Chapter 2

Forward Error Control

In this chapter we will briefly touch upon the fundamentals of coding theory. These concepts are the basis for the subsequent chapters. Discussions are restricted to the binary case, which means the information is in terms of binary symbols. It is also assumed that the information sent across the channel is random.

2.1 Communication Theory and Coding

The goal of a communication system is to provide the means for reliable transfer of information across a noisy environment. Fig. 2.2 shows the block diagram of a simple communication system. We know that information is corrupted as it is sent through the channel. The error control coding blocks are responsible for efficient reduction or elimination of these errors. The way error control codes do the job is by introducing correlation between the transmitted bits. This correlation is presented by adding redundant bits to the data. The correlation is later used by a decoder to correct errors. In 1948 Claude E. Shannon established an upper limit for the error free achievable transmission rate in an unreliable channel [21]. He also proved that there exist codes that can achieve this rate. Ever since then people have tried to come up with codes that are capable of reaching this limit. The complexity of optimal decoding algorithms was the biggest issue. Most decodable proposed codes had a rich mathematical structure but they were far from the Shannon bound. As

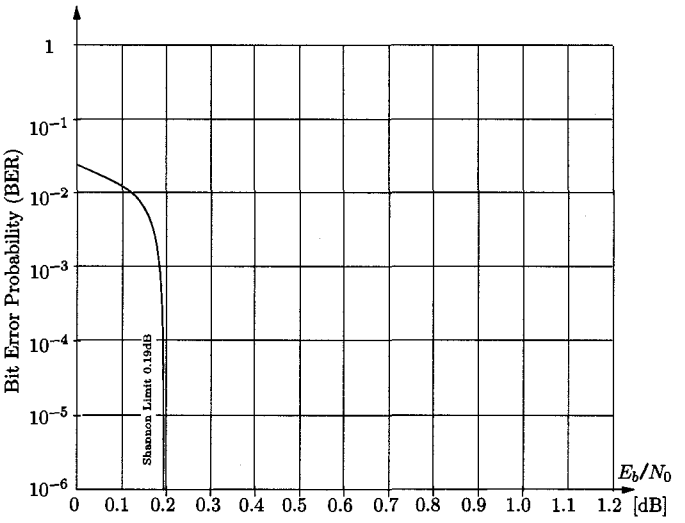

Figure 2.1: Shannon limit for rate 1/2 figure from [1]

optimal decoding seemed to be impractical, people started looking for sub-optimal algorithms, an engineering way of dealing with problems. In 1962, Robert Gallager [22] proposed LDPC codes along with an iterative decoding scheme. Unfortunately the algorithm was too complex computationally at the time to be verified. Not long after their invention, these codes were largely forgotten, but reinvented independently, for different reasons, twice in the next 30 years [23]. These codes were the first codes to approach Shannon's limit. Fig. 2.1 shows the Shannon bound for an AWGN channel. The main advantage of these codes is probably the decoding complexity that is linear with the block length of the code.

2.2 Coding: The Fundamentals

The most popular type of codes are *block codes*. A systematic block code encoder maps every k information bits at its input onto n output coded bits by introducing $(n-k)$ extra redundant parity bits. The mapping process is completely independent of the previous or future data words. The n bit output of the encoder is called a *codeword*. Practical block codes are usually linear. An important property of linear

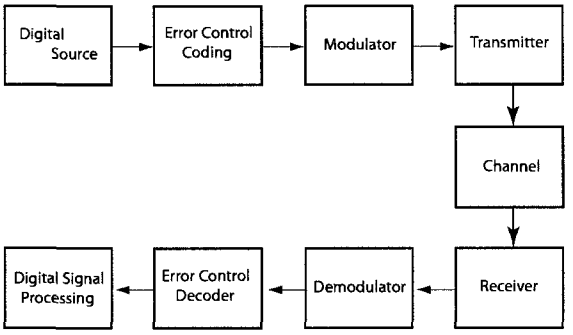

Figure 2.2: Block diagram of a simple communication system

codes is that if two codewords in a *linear* block code are summed, the result is also a valid codeword. These codes are usually referred to as an (n, k) code. If the data vector is visible in the codeword we have a *systematic* code. The fraction of information bits in the codeword is referred to as the *Code Rate* and is defined as:

$$R = \frac{k}{n}. \tag{2.1}$$

The mapping can be described by a matrix called the **generator matrix** G. To better clarify these concepts, a small $(7, 4)$ linear symmetric block code (known as Hamming code) is presented as an example. The $\mathbf{d} = [d_1, d_2, d_3, d_4]$ vector is used as the input data and the following equations are used to generate the extra three parity bits $\mathbf{p} = [p_1, p_2, p_3]$

$$p_1 = d_1 \oplus d_2 \oplus d_4 \tag{2.2}$$

$$p_2 = d_1 \oplus d_3 \oplus d_4 \tag{2.3}$$

$$p_3 = d_2 \oplus d_3 \oplus d_4. \tag{2.4}$$

This code has a rate of $R = 4/7$ and codewords are of the form $\mathbf{c} = [d_1, d_2, d_3, d_4, p_1, p_2, p_3]$. The addition sign $\oplus$ in equations is modulo-two addition also known as *exclusive-or (XOR)* operation. This code is systematic and the input data maps directly to the codeword. A matrix representation of these equations is as follows, where G is the

generator matrix.

$$\mathbf{c} = \mathbf{d}.\mathbf{G}. \tag{2.5}$$

Using these equations one realization of G is as follows:

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

We can also use the parity equations directly to form another matrix called the *Parity check matrix.* This matrix can be derived directly from the generator matrix, G. The generator and parity check matrix, $\mathbf{H}$ of any binary linear systematic block code satisfies the following equations:

$$\mathbf{G} = [\, \mathbf{I}_k \mid \mathbf{P} \,], \tag{2.6}$$

and

$$\mathbf{H} = [\, \mathbf{P}^T \mid \mathbf{I}_{n-k} \,], \tag{2.7}$$

where $\mathbf{I}$ is the identity matrix, $\mathbf{P}$ is the parity matrix and $\mathbf{P}^T$ is the transpose of $\mathbf{P}$. The parity check matrix is used in a decoder to check if the received data is a codeword. Codewords satisfy the following equation:

$$\mathbf{c}.\mathbf{H}^T = 0 \tag{2.8}$$

The $\mathbf{H}$ matrix contains all the information about the block code. In the rest of the thesis the codes are defined either by their parity check matrix or the factor graph representation which is discussed in the next section [24]. The parity check matrix of the above example has the following form:

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

2.3 Factor Graphs

A factor graph is a graphical representation (bipartite graph) of a large (global) function that has been broken down into a product of smaller (local) functions in which each local function's argument is a subset of the global function's argument [24]. A special case known as Tanner graph is widely used in LDPC literature. Linear block codes can be graphically represented by Tanner graphs [25]. A Tanner graph is a bipartite graph with two type of nodes. Variable nodes represent information bits and check nodes correspond to a set of parity check equations. The information bits which are involved in a parity check equation are connected to that check node using an edge in the graph. Going back to our example we can represent the H matrix with a Tanner graph. Fig. 2.3 shows the graphical representation. The circles on the top are the variable nodes and the boxes on the bottom correspond to the check nodes. The role of check nodes is based on the parity check equations in the H matrix. We can rearrange the parity check equations (2.2) and generate the f_i functions:

$$f_1 : p_3 \oplus d_2 \oplus d_3 \oplus d_4 = 0 \tag{2.9}$$

$$f_2 : p_2 \oplus d_1 \oplus d_3 \oplus d_4 = 0 \tag{2.10}$$

$$f_3 : p_1 \oplus d_1 \oplus d_2 \oplus d_4 = 0. \tag{2.11}$$

This graphical representation fully describes the H matrix. Each of the n bits in the code (columns of H) is represented by a variable node and each row of H is described by a check node. The ones in the H determine the edge connections between variable nodes and check nodes.

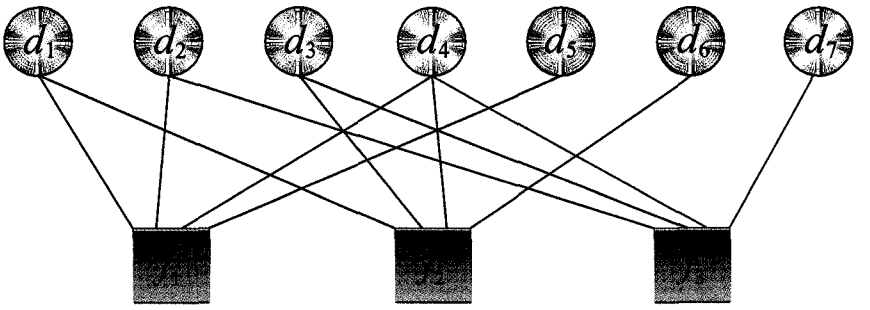

Figure 2.3: Factor Graph of a (7,4) code

2.4 Introduction to LDPC codes

LDPC codes are linear block codes with a sparse[1] parity check matrix **H**. In the original Gallager proposal, the parity check matrix requires a low density of ones and is constructed randomly subject to a single constraint: all rows in H must have the same number of ones, d_v, and each column has to have the same number of ones, d_c. In the literature these types of LDPC codes are known as (d_v, d_c)-regular codes. Here, each information bit is involved in d_v parity check equations and each parity check bit involves, d_c information bits. The rate of such an LDPC code is:

$$R = 1 - \frac{d_v}{d_c}. \tag{2.12}$$

Large LDPC codes tend to have a good performance. A slightly different class of LDPC codes, known as irregular LDPC codes and have shown to be able to get as close as $0.0045dB$ to the Shannon bound in additive white Gaussian noise channels [10].

[1]No one has mathematically defined the meaning of "sparse"

2.5 Decoding of LDPC Codes

LDPC codes are decoded using a general class of decoding algorithms called message passing algorithms [26]. These algorithms are iterative and depending on the structure of the underlying factor graph, might be the optimal or suboptimal in terms of error correcting performance. In the context of LDPC decoding these messages are *a posteriori* probabilities of the bits being ones or zeros. During the decoding process, at each step, every variable node sends a number to the check nodes, indicating its confidence about its value. The check nodes process this data and try to satisfy the parity check equations by sending a feedback message to the variable nodes. The variable nodes update their values based on the feedback from check nodes. If the independence of the messages is guaranteed through the code structure, then the algorithm has been shown to be optimal [25]. Once again we will use our example to show how the algorithm works and its basis. Assume we have sent the vector $\mathbf{d} = [d_1, d_2, d_3, d_4, p_1, p_2, p_3]$. We assume these bit are modulated using antipodal signaling where we send -1 for zero and $+1$ for one and received $\mathbf{y} = [y_1, y_2, y_3, y_4, y_5, y_6, y_7]$ in a memoryless channel with additive white noise. Where

$$y_i = d_i + n_i,$$

and n_i is the additive noise with a Gaussian distribution dependent on the channel signal to noise ratio (SNR) . In order to make a decision about the transmitted bit p_3 using the received vector we need to calculate the following probability:

$$P(p_3 = 0 | y_1, y_2, y_3, y_4, y_5, y_6, y_7). \tag{2.13}$$

It is often easier to work with *log-likelihoods* instead of directly using the probabilities. The complexity of implementation using LLRs is less compared to probabilities. log-likelihood ratios (LLR) are defined as the natural log of the ratio between the probability of the bit being zero over the probability of it being one:

$$\mathrm{LLR}(x) = \ln\left(\frac{P(x=0)}{P(x=1)}\right). \tag{2.14}$$

A positive LLR shows that the probability of zero is higher while a negative LLR has a higher probability of being one and the magnitude indicates our confidence in the values. Looking back at our example we want to find

$$\mathrm{LLR}(p_3) = \ln\left(\frac{P(p_3 = 0|y_1, y_2, y_3, y_4, y_5, y_6, y_7)}{P(p_3 = 1|y_1, y_2, y_3, y_4, y_5, y_6, y_7)}\right).$$

Applying Bayes'rule to both the numerator and denominator we can rewrite the above LLR in terms of mutual probabilities. We can then further simplify the results, considering the fact that the value of y_7 only depends on p_3 and the channel. In other words it is independent of the other received bits.

$$\mathrm{LLR}(p_3) = \ln\left(\frac{P(y_7|p_3 = 0, y_1, y_2, y_3, y_4, y_5, y_6) \times P(p_3 = 0, y_1, y_2, y_3, y_4, y_5, y_6)}{P(y_7|p_3 = 1, y_1, y_2, y_3, y_4, y_5, y_6) \times P(p_3 = 1, y_1, y_2, y_3, y_4, y_5, y_6)}\right) =$$

$$\ln\left(\frac{P(y_7|p_3 = 0) \times P(p_3 = 0, y_1, y_2, y_3, y_4, y_5, y_6)}{P(y_7|p_3 = 1) \times P(p_3 = 1, y_1, y_2, y_3, y_4, y_5, y_6)}\right) =$$

$$\ln\left(\frac{P(p_3 = 0|y_7)}{P(p_3 = 1|y_7)}\right) + \ln\left(\frac{P(p_3 = 0, y_1, y_2, y_3, y_4, y_5, y_6)}{P(p_3 = 1, y_1, y_2, y_3, y_4, y_5, y_6)}\right) =$$

$$\mathrm{LLR}(y_7) + \ln\left(\frac{P(p_3 = 0, y_1, y_2, y_3, y_4, y_5, y_6)}{P(p_3 = 1, y_1, y_2, y_3, y_4, y_5, y_6)}\right). \tag{2.15}$$

Now all we need to calculate is the second part of (2.15). This part is code-dependent and depends on the number of connections from check nodes. Fig. 2.3 verifies that p_3 has no connection to y_1,y_5 or y_6. Therefore p_3 is independent of these received bits. Further simplification of the second term can take place by writing it in terms of LLRs from the dependent nodes:

$$\ln\left(\frac{P(p_3 = 0, y_1, y_2, y_3, y_4, y_5, y_6)}{P(p_3 = 1, y_1, y_2, y_3, y_4, y_5, y_6)}\right) = \ln\left(\frac{P(p_3 = 0, y_2, y_3, y_4)}{P(p_3 = 1, y_2, y_3, y_4)}\right).$$

Using (2.9), we have:

$$P(p_3 = 0, y_2, y_3, y_4) = P(d_2 \oplus d_3 \oplus d_4 = 0, y_2, y_3, y_4), \tag{2.16}$$

$$P(p_3 = 1, y_2, y_3, y_4) = P(d_2 \oplus d_3 \oplus d_4 = 1, y_2, y_3, y_4), \tag{2.17}$$

and:

$$\ln\left(\frac{P(d_2 \oplus d_3 \oplus d_4 = 0, y_2, y_3, y_4)}{P(d_2 \oplus d_3 \oplus d_4 = 1, y_2, y_3, y_4)}\right) =$$

$$\ln\left(\frac{\frac{1+\prod_{i=2}^{4}(2P(d_i=0|y_i)-1)}{2}}{1-\frac{1+\prod_{i=2}^{4}(2P(d_i=0|y_i)-1)}{2}}\right) = \ln\left(\frac{1+\prod_{i=2}^{4}(2P(y_i=0)-1)}{1-\prod_{i=2}^{4}(2P(y_i=0)-1)}\right) = \tag{2.18}$$

Now if we rewrite the probabilities $P(y_i = 0)$ in terms of LLRs, it will have the following form:

$$\ln\left(\frac{1+\prod_{i=2}^{4}(\frac{\exp(\mathrm{LLR}(y_i)-1)}{\exp(\mathrm{LLR}(y_i)+1)})}{1-\prod_{i=2}^{4}(\frac{\exp(\mathrm{LLR}(y_i)-1))}{\exp(\mathrm{LLR}(y_i)+1)}}\right) = 2\,\mathrm{atanh}\left(\prod_{i=2}^{4}\tanh(\frac{\mathrm{LLR}(y_i)}{2})\right) \tag{2.19}$$

In the process of calculating these probabilities we used the received noisy bits from the channel. Therefore it is still possible to have a better estimate of these probabilities by feeding the updated data back in to the process and recalculating these probabilities. In belief propagation algorithms [26] we go through many iterations. At each step we use the results from the previous iteration to obtain more accurate results. This algorithm is also known as the *sum-product* (SP) algorithm. The outgoing messages from each node, at the l^{th} round of the algorithm are computed using the following equations:

$$\mathbf{m}_{vc}^{(l)} = \begin{cases} \mathbf{m}_v & l = 0 \\ \mathbf{m}_v + \sum_{c' \in C_v \setminus \{c\}} \mathbf{m}_{c'v}^{(l-1)} & l \geq 1, \end{cases} \tag{2.20}$$

where $\mathbf{m}_{vc}^{(l)}$ is the LLR message sent from the variable node v to check node c at the

l^{th} round of the algorithm and $\mathbf{m}_{cv}^{(l)}$ is the message send from the check node c' to variable node v at the l^{th} round. The set C_v is the set of check nodes affiliated with the variable node v. The message $\mathbf{m}_v$ is the LLR of the variable node v while the check node messages $\mathbf{m}_{cv}$ are calculated using (2.21). It should be noted that these equations are a straightforward generalization of (2.19), (2.15)

$$\mathbf{m}_{cv}^{(l)} = 2\,\mathrm{atanh}\left(\prod_{v' \in V_c \setminus \{v\}} \tanh(\mathbf{m}_{v'c}^{(l)}/2)\right), \tag{2.21}$$

where V_c is set of variable nodes affiliated with the check node c. We also express (2.21) in terms of probabilities:

$$\mathbf{P}_{cv}^{(l)}(v=0) = \forall v'_{i,j,\ldots,k} \in V_c \setminus \{v\} \mathbf{P}(v'_i \oplus v'_j \oplus \ldots v'_k) = 0 \tag{2.22}$$

$$\mathbf{P}_{cv}^{(l)}(v=1) = \forall v'_{i,j,\ldots,k} \in V_c \setminus \{v\} \mathbf{P}(v'_i \oplus v'_j \oplus \ldots v'_k) = 1 \tag{2.23}$$

The derivation of these formulas depends on the assumption of independence that we made earlier. Unfortunately in practical codes, after a number of iterations, this assumption does not hold anymore. A factor graph can better clarify this. Fig. 2.4 highlights a number of edges in our example code. These edges form a loop referred to as a "*cycle*". In the first iteration, v_2 sends a message to the first check node. This information is sent to v_4 in the second step. The third check node then receives this information and feeds it back to v_2 in the fourth step; therefore, the message from the third check node is no longer independent of v_2. This dependence is caused by the cycles in the code. The girth of a graph is the length of the shortest cycle contained in the graph. The bigger the minimum cycle girth in a code the better the code will be decoded using the Sum-Product algorithm. This is because large cycles allow for a greater number of iterations before we lose the independence assumption. Therefore a larger portion of the code can converge in the early phase which gives more reliable *a posteriori* information for the rest of the decoding process.

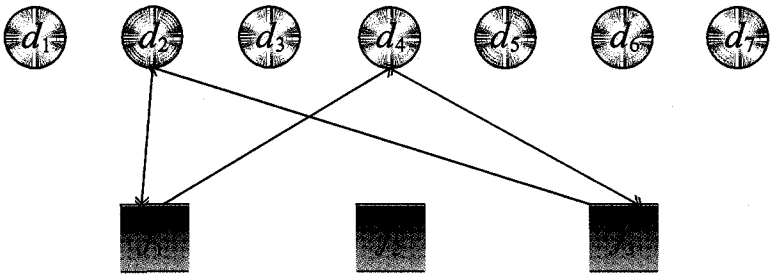

Figure 2.4: A cycle of length four, in the (7,4) Hamming code

2.6 Channel Model

Communications signals get corrupted as they travel through real channels. One of the basic yet practical channel models in communications is the *"additive white Gaussian noise channel"* (AWGN) channel model. This model simulates the behaviour of thermal noise. This noise has a white spectrum and adds directly to the signal:

$$Y = X + N, \tag{2.24}$$

where X is the sent symbol, N is the added noise and Y is the received bit. Noise, N has a Gaussian form with variance $\sigma_N^2 = N_0/2$. Therefore The received bits Y at the receiver have the following distribution:

$$P(Y|X) = \frac{1}{\sigma_N\sqrt{2\pi}} \exp\left(\frac{-(Y-X)^2}{2\sigma_N^2}\right), \tag{2.25}$$

in the case of antipodal signaling modulation. Using (2.25) we can calculate the log-likelihood of the received bits as follows:

$$\mathrm{LLR}(x|y) = \ln\left(\frac{\frac{1}{\sigma_N\sqrt{2\pi}} \exp\left(\frac{-(y+1)^2}{2\sigma_N^2}\right)}{\frac{1}{\sigma_N\sqrt{2\pi}} \exp\left(\frac{-(y-1)^2}{2\sigma_N^2}\right)}\right) = -4y/N_0 \tag{2.26}$$

Using (2.26) the probability density function of log-likelihood messages at the receiver, when we send the all-zero codeword is:

$$f_Y(Y) = \sqrt{\frac{N_0}{16\pi}} \exp\left(-\frac{N_0}{16}\left(y - \frac{4}{N_0}\right)^2\right) \tag{2.27}$$

2.7 Density Evolution

As we pointed out in the previous sections, practical codes have cycles and so the messages passed around are not fully independent. This problem is less severe in large codes. These codes, if they are designed well, usually tend to have low density of small cycles. Therefore in the first l rounds, where l is a large number, the independence assumption is valid for the majority of the bits, hence they converge to the correct values. This will help the remaining nodes to converge under the sub-optimal belief propagation conditions. Therefore the result will be very close to optimal decoding. One way to simulate a very big code without any cycles is a method called "*density evolution*" (DE) [27] [28]. Using this method we can find the ultimate error correcting performance of a code ensemble. A code *ensemble* is a set of all possible regular bipartite graphs with n variable nodes and nd_v/d_c check nodes. Density evolution tracks the probability density function (PDF) of variable nodes and check nodes at each iteration. Above some threshold signal to noise ratio, the error goes asymptotically to zero. In a linear code the all zero codeword can be sent and then we can track the PDFs. Assuming antipodal modulation, the negative tail of the PDF represents the error. The input densities depend on the channel, Noise and the input power. Ultimately density evolution will tell us the minimum SNR = E_b/N_0 required for a code ensemble to converge. In this thesis we are dealing with AWGN channels. Hence we use (2.27) as the density of received messages. The function of variable node was to add LLRs from the channel and check nodes connected to it. Since we are dealing with density functions, the density of sums would be the convolution of these values. The check node on the other hand has to deal with densities in the complicated hyperbolic functions but numerical methods allows us to numerically calculate them. In density evolution, we are dealing with densities. All nodes start off with the same input density, therefore all the messages passed at each round will be the same and all we need for simulation is one variable node and one check node. This

way we are also treating the code as a tree which has the independency assumption inherited. We have programmed a MATLAB code for density evolution. The variable node (d_v) and check node (d_c) distribution inputs indicate the code ensemble. For each signal to noise ratio we form the input densities at the receiver using (2.27). At each round of the algorithm, the variable node convolves the input from the channel with $d_v - 1$ identical check node densities from the previous round. A check node on the other hand computes the density of (2.21) assuming $(d_c - 1)$ identical inputs from the variable node. A look up table is generated for fast numerical calculation of check node densities. The negative tail of the PDF at the output of the variable node is the indicator of the error. In this thesis we run the density evolution procedure for 2000 iterations at each SNR value. We consider that convergence is achieved if the error probability gets smaller than 10^{-8}. For numerical calculation of the densities we divided the LLR range into 12 bits = 4096 bins which provides enough accuracy and reasonable speed. This number guarantees that the LLR quantization will not exceed 0.012 LLR. The program will converge in less than a minute for a (3,6) code. According to our simulations, more precise quantization values do not significantly change the result. The (2.27) model is used as the channel input density for the density evolution.

2.8 Monte Carlo Decoder Simulation

The error correcting capabilities of specific LDPC codes as opposed to code ensembles as in (2.7) cannot be computed analytically with a deterministic algorithm in reasonable time. Therefore to verify the error correcting power of these codes we need to rely on statistical approaches. Monte Carlo simulation estimates the performance of system using repeated random test cases. For the purpose of this thesis we programmed an LDPC decoder in Matlab/Mex C++ and used the all-zero codeword as the input to the channel. This assumption is valid since we have a linear code and our channel is memoryless. This condition would not apply if we were using a real circuit simulator as analog decoders tend to prefer some codewords over others. This is because the analog implementation can be biased depending on the input values.

We have verified the bias independency of our decoder by testing the decoder under both all-zero and all-ones codeword conditions.

At each iteration the channel noise, which is a random function of input SNR is added to our all-zero codeword. Then the decoder program goes through a finite number of iterations (usually 50 t0 90 my simulations) to decode the data. The 90 frame errors provides an 80% confidence interval between $[0.87 \times 10^{-6} \;\; 1.14 \times 10^{-6}]$ for the frame error rate. The decoded codeword is then compared with the sent codeword and we calculate the number of bits in error. We repeat the same procedure for each SNR until we have over 50 frames with at least one bit error in each. Having gathered this information we can then estimate the bit error rate (BER) of the code at each SNR value

Chapter 3

Analog Decoding

In a traditional digital decoder, messages passed back and forth in the decoder are quantized and represented in terms of bits and all the decoding functions are implemented using digital circuitry. In an analog decoder these messages are mapped into voltages and currents. The decoding functions are carried out, exploiting the physical characteristics of semiconductors and electrostatics. There are a number of advantages and disadvantages in employing analog circuits. We will briefly point out the main issues before we start the discussion on analog decoding circuitry. Messages passed in a digital decoder are generally represented by word lengths of 4 to 8 bits [29]. This means that in a parallel implementation of a decoder, each edge connection of the graph requires 4 to 16 wires depending on a unidirectional or bidirectional implementation of the edge connections. With a simple back-of-the-envelope calculation for a length 1000 $(3, 6)$-regular LDPC code, we need 24000 wires just for the edge connections. The routing congestion results in a decoder with low area utilization [30]. Also due to the large number of wires, the average length of wires can be on the order of millimeters [31]. Long wires tend to increase the RC delays and become power consuming due to the switching of their capacitance. Therefore a fully parallel digital implementation of these codes is very challenging and usually not practiced. High speed synchronous decoders require clocks at very high speeds which generate noise and consume a significant amount of power.

Analog decoders have an easier time dealing with message passing. One wire or at most two for the differential case is enough for passing values between nodes. Some functions like summation are available at negligible hardware cost. The voltage current relationships in a decoder provide some of the required complicated function such as the hyperbolic tangent. Due to the smaller number of connections, parallel implementation of codes is feasible.

In a digital decoder, power consumption is a function of input signal to noise ratio; at low SNR where many bits have wrong values, the switching activity of the bits is very high [32]. Capacitors are charged and discharged at each iteration. The high switching activity wastes a lot of power. Analog decoders on the other hand gradually converge to their final output values. Therefore, they almost have a steady power consumption.

All these advantages have made implemented analog prototypes very promising. However, despite all the great advantages, they have not been commercially used yet. Just like every other analog circuit, imperfections such as mismatch, process variations and temperature variations affect the performance of the decoder [33]. Even moderate decoder size implementations, spread over a large wafer area. Hence the transistors experience huge process and temperature variations. These problems are less profound due to the iterative error correcting nature of the algorithm compared to other analog circuits. The other problem facing analog decoder designers is the lack of circuit tools for simulating the full decoder. Even for a small code, simulations take on the order of hours. Therefore there is no way we can perform a statistically significant monte-carlo simulation on a full decoder to get bit error rates even down to 10^{-3} or perform optimizations on the whole decoder. Most of these decoders run in the sub-threshold and moderate inversion regions. The models, tend to lose accuracy in these regions, especially for smaller processes. Testing these circuits is another big issue. Recently some promising work has been done to assist with testing [34]. The I/O interface of the analog decoders reported so far are very big with comparable complexity to the decoder and problematic in many cases [14]. Just like any other analog circuit, going from one process to another demands a huge amount of design time.

A digital decoder is typically robust with a small time to market cost. The simulation results are very close to actual measurements. The testing is easy and dealing with thousands of transistors is a common practice in digital design. Whether or not the analog or digital decoders will dominate is still under debate. This chapter covers the essential basic structures for analog decoders. The first section talks about the properties of MOS transistors in weak inversion. The subsequent section presents the basic circuit topologies that are commonly used in analog decoders and the last section concludes the chapter.

3.1 MOSFETs in Weak Inversion

The metal oxide semiconductor field effect transistor (*MOSFET*) is the most common transistor used today. Bulk silicon MOSFETs are four terminal devices. The terminals are called Gate, Source, Drain and Bulk. The current through the Drain is controlled by the voltages applied to the MOSFET. The two common MOSFET types are *nMOS* and *pMOS*. Fig. 3.1 shows the physical structure of an nMOS transistor. The substrate is p-type doped silicon. The source and drain regions are heavily doped with n-type donors and the gate is usually fabricated from polysilicon. Applying a voltage to the gate of the MOSFET creates ionized acceptor atoms. This phenomenon is called depletion. A further increase in the gate to source voltage creates a noticeable number of n-type carriers in the surface of the p-type substrate. At a certain gate to source voltage, the number of electrons exceeds the number of holes in the channel, producing surface inversion. Therefore the surface of the p-type substrate becomes inverted into an n-type material. While the density of electrons is less than the density of holes in the original p-type material, we are in the weak inversion mode of operation. Applying a voltage to the drain terminal attracts electrons and reduces the charge density at the drain. The difference in charge density creates a small current due to diffusion of charges. This current is the *weak inversion* current of the MOS transistor, also known as the sub-threshold current. If we continue increasing the voltage on the gate above its threshold voltage, the density of electrons becomes much greater than the density of holes in the original p-type material. As the charge

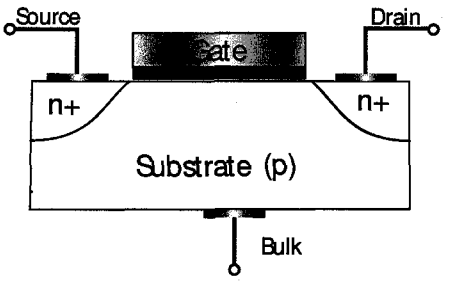

Figure 3.1: Cross section of an n-type MOSFET

in the surface of the device increases, the current includes a big drift component. This current is known as the *strong inversion* current. The transition between the weak and the strong regions of operation is gradual and through the *moderate inversion* region. In the context of this thesis we are interested in the sub-threshold behaviour of MOSFETs. The current in this region is an exponential function of gate voltage. This exponential behaviour can be exploited to implement the basic functions in the sum-product algorithm. Equation (3.1) shows the sub-threshold current in an n-type MOSFET:

$$I_{ds} = I_0 \left[1 - \exp\left(-\frac{V_{ds}}{V_T}\right)\right] \times \exp\left(\frac{V_{gs} - V_t - V_{off}}{nV_T}\right), \tag{3.1}$$

and

$$I_0 = \mu \frac{W}{L} \sqrt{\frac{q\varepsilon_{si} N_{substrate}}{2\Phi_s}} V_T^2, \tag{3.2}$$

where V_{ds} is the drain source voltage of the MOSFET, $V_T = \frac{K_B T}{q}$ is the thermal voltage around $25.9mV$ at room temperature, K_B is the Boltzmann constant and q is the magnitude of the electrical charge (in coulombs) on the electron. V_{gs} is the gate to source voltage, V_t is the threshold voltage and V_{off} determines the current at $V_{gs} = 0$. n is the sub-threshold swing parameter and is a function of the length of the channel and the interface state density. W and L are the width and length of the transistor. ε_{si} is the permittivity of silicon, $N_{substrate}$ is the substrate doping concentration and Φ_s is the surface potential. At high drain voltages the $\exp(-V_{ds}/V_T)$ term becomes

negligible compared to 1 and is usually omitted for simplicity. The area near the gate oxide and between the drain and the source terminals in a MOS device is called the *channel*. In a long channel MOSFET we can assume as long as V_{ds} is large, for a certain device, variations in I_{ds} are only due to variations in V_{gs}. Hence we can rewrite (3.1) as:

$$I_{ds} = I_s \exp\left(\frac{V_{gs}}{nV_T}\right), \tag{3.3}$$

where I_s is solely dependent on sizing and fabrication properties of the transistors as well as the temperature. These assumptions do not entirely hold in short channel devices. We will study these effects and their impact on analog decoder performance, in more detail in the next chapter.

3.2 Basic Circuits

Implementation of the sum-product (SP) algorithm requires three sets of operations: (1) Addition of LLRs (2.20) (2) Soft XOR of the probabilities (2.22) and (3) Conversion from probabilities to LLRs and vice versa (2.14). In this section we introduce circuit architectures based on CMOS transistors for each of these modules. The core circuit for the first two functions is the current vector multiplier. We first analyze this circuit in detail and then we show how we can use it to implement the first two modules.

3.2.1 Core Circuit for Current Multiplication

There are many different circuits that can perform current multiplication. Out of these many choices one circuit has become very popular and widely used in analog decoding. Gilbert proposed the concept of translinear circuits and proposed a multiplier based on bipolar transistors [35] [36] [37]. These multipliers are highly tolerant to temperature variations. A two by two multiplier requires only four transistors. The outputs are scaled and it can be implemented in CMOS with low power consumption. Signal processing modules in a decoder have to be small and low power because we

have to replicate a block many times on the order of several thousands for a moderate size code. The CMOS based Gilbert Multiplier works in the sub-threshold region; therefore smaller currents behave closer to ideal behaviour. This is because with small currents the circuit stays in subthreshold region and far from moderate or strong inversion modes. The Gilbert Multiplier seems to be an excellent choice for all these reasons. As mentioned in the previous section, The MOSFET has an exponential behaviour in the sub-threshold region and the currents are small, which is ideal for low power implementation. Assuming, all transistors have the same size and the transistor have and ideal behaviour according to (3.3), we can derive the equations for the Gilbert multiplier. Fig. 3.2 shows the circuit for a two by two Gilbert multiplier. Going through voltage loops, such as the one specified with the arrows, we get the following relations between the currents:

$$I_{x_1y_1} = \frac{I_{x_1}.I_{y_1}}{I_{y_1} + I_{y_2}} \tag{3.4}$$

$$I_{x_1y_2} = \frac{I_{x_1}.I_{y_2}}{I_{y_1} + I_{y_2}} \tag{3.5}$$

$$I_{x_2y_1} = \frac{I_{x_2}.I_{y_1}}{I_{y_1} + I_{y_2}} \tag{3.6}$$

$$I_{x_2y_2} = \frac{I_{x_2}.I_{y_2}}{I_{y_1} + I_{y_2}} \tag{3.7}$$

3.2.2 Representing Probabilities and LLRs

In a typical sum-product analog decoder, probabilities are naturally represented by currents and LLRs are represented in the voltage domain. Other forms of signaling scheme for analog decoders have also been used [16] [19]. We need a circuit to implement equation (2.14). An ideal differential pair operating in the sub-threshold region can easily implement the function. Fig. 3.3 shows such a circuit. By applying the differential voltage $\Delta_V = V_+ - V_-$ to the differential pair, and using (3.3)we have:

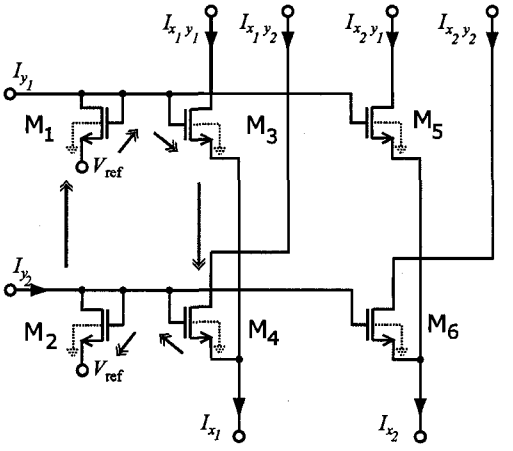

Figure 3.2: 2 by 2 Gilbert Current Multiplier

$$I_+ = I_s . \exp \frac{V_+ - V_s}{nV_T}, \tag{3.8}$$

$$I_- = I_s . \exp \frac{V_- - V_s}{nV_T}. \tag{3.9}$$

Using (3.8), the differential voltage can be written in terms of I_+ and I_-:

$$\Delta_V = nV_T \ln \frac{I_+}{I_-}, \tag{3.10}$$

where Δ_V is the differential voltage applied to the differential pair. Fig. 3.3 shows the differential pair circuit. The above relationship provides us with the tools to go from the LLRs domain to probabilities. This circuit also has one other advantage: the current source tail normalizes the differential currents. The probabilities of zeros and ones have to add up to one. The current source forces the total current to be equal to some normalizing value.

$$I_+ + I_- = I_u \tag{3.11}$$

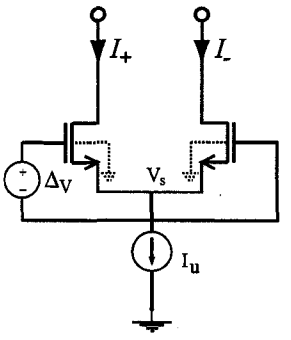

Figure 3.3: Differential pair operating in the sub-threshold region

This value is arbitrary but bounded and is an important design parameter. The power consumption of the decoder is proportional to this value. It also effects the speed of the circuit. The upper limited is defined by the sub-threshold region of operation and the lower limit is bounded by the required dynamic range and speed. These issues are studied in more details in the following chapter.

3.2.3 Soft XOR Implementation of Check Nodes

The Check node is responsible for calculating the probability of zero or one based on inputs from variable nodes. Looking at (2.22) we can see that the check node performs a soft XOR computation that requires multiplication and addition of probabilities, as shown below.

$$P_z(0) = P_x(0)P_y(0) + P_x(1)P_y(1), \tag{3.12}$$

$$\tag{3.13}$$

and

$$P_z(1) = P_x(0)P_y(1) + P_x(1)P_y(0). \tag{3.14}$$

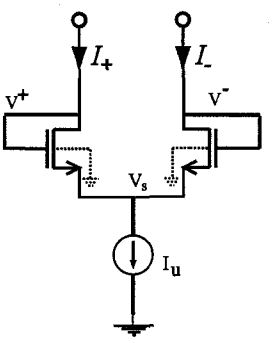

Figure 3.4: Diode connected differential pair

These blocks have been discussed in more detail in the literature [6] [38]. The notation $P_x(i) \equiv P(x = i)$ is used in the above equations. We can use the Gilbert multiplier to produce the product terms and tie the outputs together to perform the summation.

3.2.4 Summation of LLRs

The variable node in the sum-product algorithm is in charge of summing the LLRs from the channel to the LLRs from the check nodes connected to it (2.20). Instead of adding the LLRs we will multiply the probabilities:

$$\text{LLR}_x + \text{LLR}_y = \ln \frac{P_x(0)}{P_x(1)} + \ln \frac{P_y(0)}{P_y(1)} = \ln \frac{P_x(0).P_y(0)}{P_x(1).P_y(1)}$$

So given the input probabilities We need to calculate:

$$P_z(0) = P_x(0)P_y(0), \tag{3.15}$$

$$\tag{3.16}$$

and

$$P_z(1) = P_x(1)P_y(1) \tag{3.17}$$

Once again the multiplier can handle the multiplication operation and we can use

a modified version of the differential pair to convert the probabilities into the LLR domain. The circuit is shown in Fig. 3.4 and works under same principles as the differential pair we presented earlier, as follows:

$$\Delta V_{diff} = nV_T \ln \frac{I_+}{I_-}$$

3.2.5 Connecting Blocks

A large decoder requires a large number of variable and check node building blocks to be connected. Messages passed between blocks can be of current or voltage nature. One other issue is the loss of current as we connect these circuits together. Each time we use the Gilbert cell as the variable or check node we only use two outputs out of four and some of the current is lost to the power supply without being mirrored onto other blocks. Therefore we need normalizing stages as well. The easiest way to connect these building blocks is using current input and outputs. In its native form, the Gilbert cell works on a current in current out basis. Another option is conversion to voltage and back to current at each stage. This method has a transistor overhead but provides a degree of freedom for reshaping the voltages using voltage level shifters and transistor sizing. The downside of this approach is the temperature dependance of the these blocks. Current to voltage conversion blocks rely on the thermal voltage V_T which can introduce errors when the temperature of the chip varies from one point to another. Fig. 3.5 shows the normalizing block that can be used for the nMOS Gilbert cell. I^+, I^- are outputs from Gilbert multiplier and I_n^+, I_n^- are the normalized output currents.

3.3 Conclusion

In this chapter we studied the basic circuit blocks for building a sum-product analog decoder. Throughout this chapter we assumed ideal behaviour for MOSFET transistors in weak inversion. In order to build a functional decoders we need a full understanding of the device behaviour in nano scales. In the next chapter we will study the detailed characteristics of the CMOS transistors in nano-scale regime.

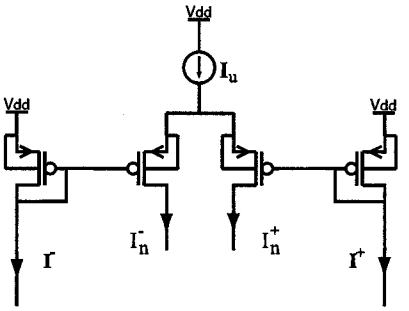

Figure 3.5: Current Normalizing Block

Chapter 4

Theoretical Limitations of CMOS Technologies

Semiconductor devices are scaled aggressively for high performance and integration. Scaled transistors tend to operate faster but their behaviour deviates from long channel models and undesired effects arise. The gate loses its control over the channel while the drain becomes more prominent. The off state current, known as the leakage current increases, and the gate insulator starts conducting current. The design of functional analog circuits without considering these issues is impossible. A pioneering study on sub-threshold analog decoders was done previously in [39]. In this paper the effect of body voltage in the sub-threshold transistors is studied. The analysis are based on MM9 circuit models from Philips [40]. The MM9 is physics based analytical model for electrical circuit simulation. They also propose a certain ratio between the width and length of the transistors for precise subthreshold operation. Here we take the analysis one step further and discuss the effect of transistor sizing on the transistor behaviour in nano-scale transistors. Our analysis are based on BSIM4 CMOS models [41] . In this chapter we will first introduce these non-idealities. Then we will address the limitations they force on the design of analog decoders and introduce guidelines to overcome some of these issues. We propose transistor sizing strategies to mitigate the short channel effects and extend the threshold voltage. For the first

time, the issue of leakage currents and threshold mismatch in analog decoders is discussed in this chapter. We have also proposed a model that represents the behaviour of variable node in the presence of leakage currents and threshold mismatch. These new models provide the means for predicting the performance of the decoder in the presence of imperfections.

4.1 Short Channel Effects in CMOS

The long channel models in CMOS technologies are derived using one-dimensional analysis and a gradual-channel assumption [42]. These assumptions fail to hold as the transistor size shrinks. The electric field lines have a considerable component along the channel which will lead to many undesirable short channel effects (*SCE*) [42]. For the purposes of this thesis we are interested in the way the threshold voltage, the sub-threshold swing (n) and the leakage currents are affected. The variations in threshold voltage can dramatically change the current in the sub-threshold region of operation. This is because the current has an exponential relationship with the threshold voltage.

4.1.1 Threshold Voltage

The threshold voltage of transistors plays an important role in analog decoders. The threshold voltage determines the boundary between the sub-threshold mode of operation and strong inversion. In the long channel MOSFET equations, the threshold voltage is only a function of process parameters and the source to bulk voltage. As the channel length of the MOSFET is scaled into the nano regime, the threshold becomes a complicated function of the applied voltages, transistor sizing as well as the process parameters. Here we will introduce these relationships and how we can exploit them to change the sub-threshold bounds. The classic equation for the threshold voltage in a MOS transistor is [42]:

$$V_t = V_{FB} + \Phi_s + \gamma\sqrt{\Phi_s + V_{sb}} = V_{FB} + \Phi_s - \frac{Q'_B}{C'_{ox}}, \tag{4.1}$$

where V_{FB} is the flat band voltage, Φ_s is the surface potential and the last term represents the voltage across the depletion region. The parameter γ in the last term is the body bias coefficient, given by

$$\gamma = \frac{\sqrt{2q\varepsilon_{si}N_{substrate}}}{C_{ox}}, \tag{4.2}$$

where $N_{substrate}$ is the doping level of the substrate. The flat band voltage depends on the substrate, gate work functions and the ion implantation in the channel, which are all process parameters. The surface potential is a function of channel doping and the temperature. In long channel MOS devices, these values are determined by the process and are independent of transistor sizing. In short channel transistors, the channel doping is no longer uniform. The variations in channel doping are both vertical and lateral. The vertical variations of doping concentration can make the threshold a strong function of the gate to source drive. This extra dependance is not necessarily present in any CMOS process. As long as the edge of depletion is beyond the ion implantation edge this extra dependance is not present. This can be easily verified through simulation of weak inversion current versus the gate to source drive. The lateral doping change is due to a process called Pocket (Halo) implant. This fabrication step is very popular in short channel transistor fabrication [43]. During this process, the doping near the source and drain regions is increased and it is necessary to combat some serious short channel effects [43]. Therefore, as the channel becomes shorter, the average effective concentration becomes higher and increases the surface potential. Thus, the threshold voltage becomes a function of channel length. This dependency is modelled by an additional term added to the classic threshold value. Assuming no body effect the threshold voltage in the presence of pocket implants is [41] [44]:

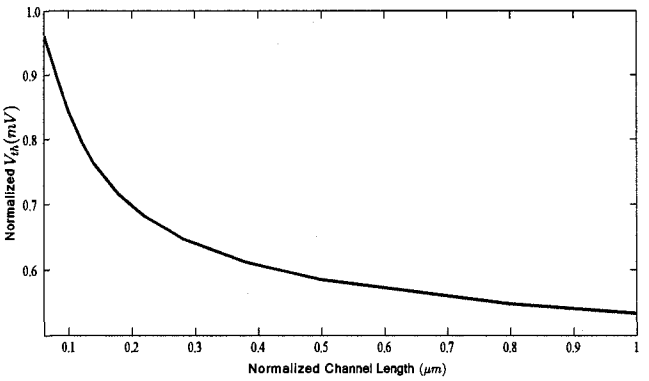

Figure 4.1: Variations in Threshold Voltage due to change in Channel Length

$$V_t = V_{FB} + \Phi_s - \frac{Q'_B}{C'_{ox}} + K_1 \left(\sqrt{1 + \frac{LPE0}{L_{eff}}} - 1 \right) \sqrt{\Phi_s}, \tag{4.3}$$

where K_1 and $LPE0$ are process parameters. Thus threshold voltage is a decreasing function of effective length. Therefore lower channel lengths result in higher threshold values and hence larger weak inversion range of operation. This higher threshold voltage is desirable for subthreshold applications as it provides a wider range of accurate operation. Fig. 4.1 shows this relationship in a 65nm process. The simulation data are from the Cadence software using a typical 65nm CMOS process data.

The width of a transistor also plays a role in its effective threshold. In a long channel MOS transistor, the source and drain regions are far away from the channel. Therefore, the fields can be assumed to be vertical in most of the channel. In short channel devices, with channel lengths less than $0.35\mu m$, the fabrication process follows a pattern known as *Shallow Trench Isolation (STI)* [45] . This isolation technique is used to prevent leakage currents between adjacent devices. As a result of this process, a big portion of gate electric fields terminate on source and drain ends. These non vertical fields are called fringing fields. Therefore part of the depletion charge is due to the fringing fields. In order to calculate the exact threshold value, these charges

have to be taken into consideration. The new threshold voltage in an STI process, assuming no body effect is:

$$V_t = V_{FB} + \Phi_s - \frac{Q_B}{C'_{ox}.WL + 2C_F}, \tag{4.4}$$

where C_F is fringing field capacitor and is only a function of L. Since Q_B is a function of $W.L$, the threshold voltage increases with device width. It is now clear that the threshold voltages in modern devices are very sensitive to transistor sizing, especially in minimum sized transistors. Therefore, due to undesired uncertainty in fabrication, known as *mismatch*, choosing minimum size transistors might result in significant deviation of results from simulator prediction. The other important point to notice is the sub-threshold range of operation. Larger threshold values provide a wider range of sub-threshold operation given a constant sub-threshold swing. Fig 4.2 plots the relationship between the threshold voltage and the width of a transistor. The data are from a BSIM4 typical 65nm CMOS process based on the Cadence simulation tool.

Based on these observations, we offer the following *design recommendation:*

- Use transistor sizes that are at least three times larger than the process minimum sizing. Hence the threshold variations due to mismatch effects in L will be smaller.

- Expand the region of sub-threshold operation by increasing the transistor widths.

4.1.2 Drain Induced Barrier Lowering

In the previous section, it was mentioned that the transistor threshold voltage is a function of surface potential (4.3). In long channel transistors the drain can only affect a small part of the channel. As the channel shrinks, field lines from the drain can influence the charge and potential throughout the channel and force the threshold voltage to be a function of drain voltage. The term Drain Induced Barrier Lowering

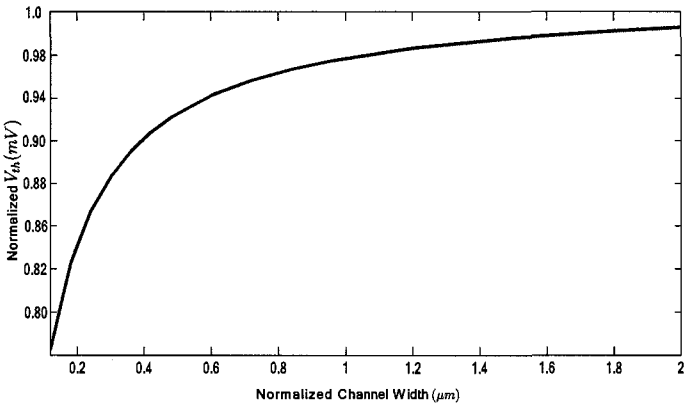

Figure 4.2: Threshold voltage versus channel Width in 65nm CMOS process

(*DIBL*) is used to describe this phenomenon. Drain voltage increases the surface potential and hence lowers the potential energy for electrons [42]. Normally the gate voltage is responsible for increasing the surface potential and driving the channel into inversion mode. Therefore, in the presence of the DIBL effect, we would need less gate voltage to achieve the same surface potential. This relationship between the threshold voltage and the drain voltage is nonlinear and very complicated, but if the source and bulk of the transistor are connected together we can approximate it with a linear equation and the DIBL effect becomes milder under this condition. The bulk to source voltage increases the DIBL effect by adding another term to the linear DIBL parameter. The latter term is a linear function of bulk to source voltage. In the context of analog decoding in sub-100nm CMOS technologies, the drain to source voltage of transistors usually varies between $50mV$ to $450mV$ and the DIBL effect can be modeled with a first order linear term added to the threshold. This added term is not a function of transistor width as the DIBL effect is independent of the width. The drain current considering DIBL is [46]:

$$I_{DIBL} = I_i.e^{\eta V_{ds}}, \tag{4.5}$$

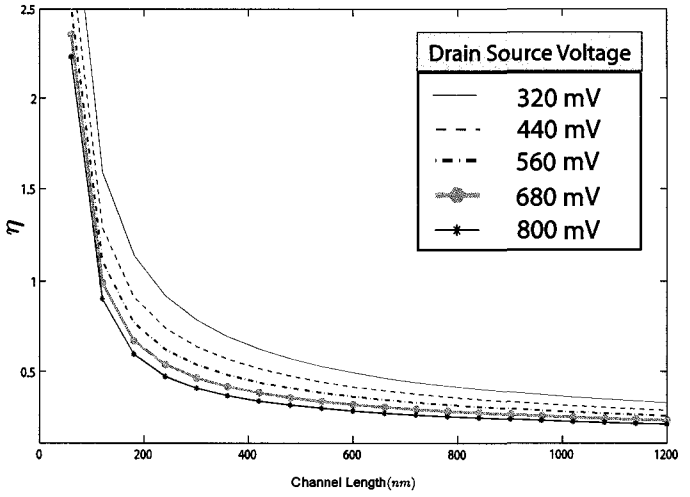

Figure 4.3: DIBL parameter versus channel length for different drain source voltages

where I_i is the sub-threshold current from (3.1). The parameter η has a $1/L$ relationship with the channel length. Looking at Fig. 4.3, we can see that the value of η for small channel L, is between 2 and 6 and between 0.2 and 1 for $L > 4 \times$ minimum length. These values are from a 65nm CMOS processes and assuming $V_{bs} = 0$. The simulation data are generated for an nMOS transistor in a typical 65nm CMOS process using the Cadence software using BSIM 4 models. The pMOS and the nMOS transistors tend to have different values for η yet these values are close and depend on the channel doping in each case [47]. Keeping these in mind when designing the decoder, we can heavily reduce the non-linearities due to the DIBL effect in circuits. Channel lengths larger than four times the minimum size tend to have a small value of η, which is very desirable for eliminating the variation due to the drain source voltage.

4.1.3 Sub-Threshold Swing

The Sub-threshold swing parameter, n in (3.1) when multiplied by the thermal voltage V_T is the inverse of the slope of the natural logarithm of I_{ds} versus V_{gs}. As we saw earlier, this number plays an important role in accurate functioning of our Gilbert

multipliers. Also, conversion from LLRs to probabilities is directly proportional to this value. Ideally we want this number to be constant throughout the weak inversion region. The best estimate of this number can be found through the following formula [41]:

$$n = 1 + NFACTOR.\frac{C_{dep}}{C_{ox}} + \frac{Cds_{term}}{C_{ox}}, \tag{4.6}$$

where $NFACTOR$ is a fitting parameter very close to one, $C_{dep} = \frac{\varepsilon_{si}}{W_{dep}}$ is the depletion capacitance in silicon and W_{dep} is its width. The second term, $\frac{Cds_{Term}}{C_{ox}}$ in (4.6) is usually smaller than the first part and contains the short channel effects. This part links the sub-threshold swing to the drain source voltage, the bulk source voltage and the effective length. Cds_{term} is given by:

$$Cds_{term} = (\alpha_1 + \alpha_2.V_{ds} + \alpha_3.V_{bseff}).\frac{0.5}{\cosh(DVT1.L_{eff}/l_t) - 1}. \tag{4.7}$$

The α_i terms are fitting parameters, $DVT1$ is a short channel effect parameter, L_{eff} is the effective length. The effective length is very close but smaller than the actual length (L) and l_t is called the characteristic length. If these terms are substituted by actual fabrication data we notice that n increases with smaller channel lengths. This is due to the $\cosh(L_{eff}) - 1$ expression in the denominator. We can also notice that the value of n is highly sensitive to L_{eff}, when L_{eff} is small. Fig. 4.4 shows the variations in n for different transistor sizing at different gate to source voltages. The sub-threshold swing increases with with smaller channel lengths. In small channel lengths the gate control over the surface potential decreases. Therefore, the variations in the surface potential due to variations in the gate potential is smaller [48]. This relationship indicates a bigger subthreshold swing. The simulation result also shows that the variations in the sub-threshold swing with regards to the length tend to decreases with longer channel lengths. Therefore, longer transistor lengths are recommended. This is because due to the imperfections in fabrication processes, the size of the fabricated transistors are different from the implemented size. Therefore,

using longer transistors reduces the uncertainty in the sub-threshold swing. The data for the plot are from a typical 65nm CMOS process generated for an nMOS transistor with no body effect using the Cadence software. The assumed temperature for the simulation is 300 degrees Kelvin.

Here we would like discuss the relationship between the subthreshold swing parameter n and the source to bulk voltage. The sub-threshold swing parameter n, is a function of the depletion width W_{dep}

$$W_{dep} = \sqrt{\frac{2\varepsilon_{si}(\Phi_s + V_{sb})}{qN_{substrate}}}, \tag{4.8}$$

The surface potential is a very weak function of the gate voltage and is almost constant throughout the sub-threshold region. Therefore, W_{dep} mainly changes with the source to bulk voltage. The higher the source to bulk voltage (body effect) the bigger the depletion width and hence n is smaller. The source to bulk voltage also changes the threshold voltage. Therefore, the gate voltage where the weak inversion region starts/ends shifts towards higher gate voltages. Aside from these effects, the source to bulk voltage might play another important role in the value of n. As we just mentioned the variations of surface potential in the sub-threshold region is a very weak function of the gate voltage. Therefore, the change in the depletion width due to the change in the gate voltage is very minor. However even this minor change can be very problematic if the edge of the depletion region is within the channel ion implant width at the beginning of the sub-threshold region. In small transistors, the channel doping concentration is increased in order to adjust the threshold voltage. Therefore the channel has a different doping than the substrate. Hence the $N_{substrate}$ in W_{dep} varies as the depletion width moves beyond the ion implantation region and n would no longer be a constant. This only happens, if the edge of the depletion moves within regions with two different doping levels.

Such cases can be problematic for analog decoders. The starting position of the depletion in the sub-threshold region is processes dependant. Therefore, the case has to be verified through simulations. In case the problem exist, the source to bulk voltage can move the starting point of the depletion edge beyond the implant edge

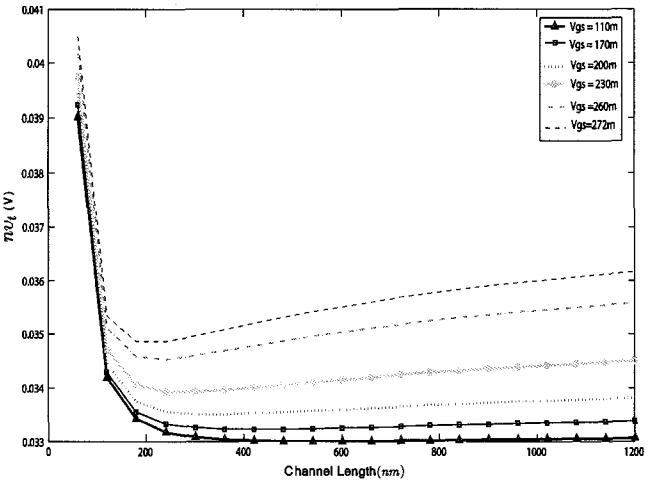

Figure 4.4: Sub-threshold swing versus channel length for different gate source voltages

and solve the problem. In our process, simulations show that even with a zero V_{sb} voltage, variations in n are acceptable.

4.2 Leakage Currents & Limitations

The currents in a real transistor can never be fully turned off. As we decrease the gate to source voltage of a transistor, we ideally expect all the currents to go to zero. In a real fabricated transistor however, these currents hit a lower bound that is non-zero and we cannot decrease them further. This lower limit is known as the leakage current. Depending on the context, leakage currents are often interpreted differently. Digital circuit designers refer to leakage as any current flowing when gate to source voltage is less than the threshold voltage, $V_{gs} < V_t$. It is obvious that in the context of analog decoding we have a different definition for leakage currents. Under the condition of a zero source to bulk voltage, if we reduce the gate voltage into negative values, the depletion region will disappear. Therefore, the sub-threshold rules no longer apply. The channel turns into a p-type silicon and we expect all the currents in the device to vanish but as we further reduce the gate voltage into negative

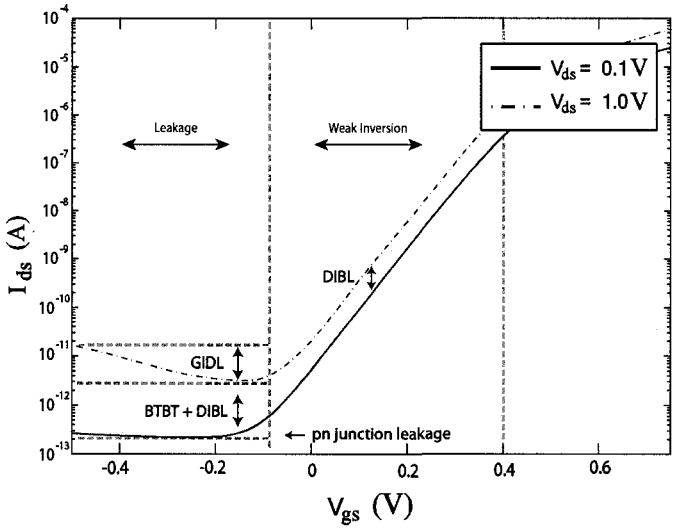

Figure 4.5: Leakage currents in 65nm CMOS with two different drain voltages

values these currents remain or increase in size. Fig. 4.5 shows the drain current versus gate source voltage for two different drain voltages. The figure shows different leakage mechanisms affecting the drain leakage. In this section we will talk about the physical source of these unwanted currents and how they affect the performance of the decoder.

4.2.1 PN Junction Leakage

The pn junction leakage is the classic leakage current in MOS transistors. The source of this current is the reverse bias voltage applied to the pn junction formed by drain, n-type and substrate, p-type in an nMOS transistor. The reverse bias will create a region with very few carriers in between n and p regions, known as the depletion region. The current in this region is due to the minority carriers and electron-hole generation near the edge of the depletion region. These carriers are swept by the strong reverse bias field. This is a well studied phenomenon in long channel MOS transistors [49]. Short channel transistors have non-uniform drain and source doping levels. As it was mentioned earlier, in order to eliminate some fatal short channel

effects, the regions near the source and drain extension are highly doped, which causes band-to-band tunneling (BTBT) . This current is much larger than the reverse bias current and dominates the junction leakage in small transistors [46]. The BTBT is the effect of electrons tunneling from the valence band in the p-region to the conduction band of the n-region. The high voltage on the drain side causes the conduction band of the n-side to move down to the same level as the valence band in the p-region. The high doping concentration increases the probability of tunneling and results in large leakage currents. Current I_1 in Fig. 4.6 shows this effect. An analytical expression for these currents can be found in [46] [50]. In terms of transistor sizing, this current is a weak function of the junction width and large transistor width tend to increase the leakage. The length of the transistor does not affect this type of leakage [51].

4.2.2 Gate Induced Drain Leakage

At negative gate voltages, the depletion layer in the substrate disappears and we gradually enter the accumulation region. As we go deeper into the accumulation region, more holes enter the p-region and the p-region near drain and source acts like a heavily doped p-region. The overlap region between the drain and the gate now has a higher concentration of holes so the depletion width gets narrower. The narrow depletion in the substrate and the wider depleted n region in the presence of the gate voltage introduce strong fields which enables BTBT. Tunneling between the depleted region in n and the heavily doped substrate occurs. This tunneling current is a strong function of drain voltage and the negative gate voltage and causes the drain current to increase as the gate voltage becomes more negative. We refer to this leakage current as the Gate Induced Drain Leakage (GIDL) . The I_2 in Fig. 4.6 shows this current.

In the context of analog decoders, GIDL is very undesirable since the drain current increases with negative gate source drive. Unfortunately, in the course of decoder operation some transistors experience the negative gate source drive. Proper biasing of the input voltages can help the situation but not fully resolve the issue. The tunneling currents we presented, BTBT and GIDL both highly depend on the drain

voltage. Therefore, by controlling the drain source voltage where there is chance of negative drive, we can heavily reduce these unwanted effects. Unlike the long channel decoder design rules, very high drain source voltages are not desirable in nano-scale transistors and they should be avoided.

4.2.3 Gate Leakage

In classic MOS theory the gate of a MOS transistor is purely capacitive and does not conduct any current. Scaling transistors into the nano regime has introduced very thin gate oxides. On the other hand the electric field strength has increased [52]. The high electric field enables the electrons to tunnel through the thin oxide barrier or directly tunnel and enter the gate region from substrate. The size of this current is a function of the thickness, height and structure of the barrier. The tunneling in short channel transistors is mainly direct tunneling. In an nMOS type of transistor, tunneling from conduction band (ECB) forms the major leakage where as in pMOS, holes from valance band (HVB) play the main role. These two experience different barrier heights: around $(4.5eV)$ for HVB and $(3.1eV)$ for ECB. Therefore, the gate leakage in p-type transistors is much smaller (around one tenth) [53]. The gate leakage current starts becoming noticeable as we scale the oxide thickness and can become a serious issue for very thin oxide thicknesses. In the context of analog decoders however, the gate leakage is not very severe. The reason is that in the sub-threshold mode of operation the voltage drop across the oxide can be found using:

$$V_{ox} = V_{gs} - V_{FB} - \Phi_s - V_{poly},$$

where V_{ox} is the voltage drop across the oxide, V_{FB} is the flat band voltage, Φ_s is the surface potential and V_{poly} is the voltage drop across the depletion region. Also, from the classic MOS theory we know that the threshold voltage is

$$V_t = V_{FB} + \Phi_s + V_{poly}.$$

As long as we are operating in weak inversion, $V_{gs} < V_t$, the voltage across the oxide

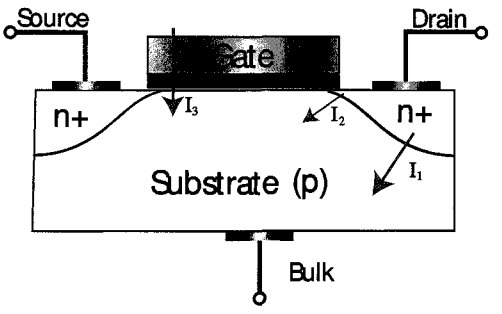

Figure 4.6: Three Sources of Leakage Currents: pn junction leakage (I_1), GIDL leakage (I_2) and gate tunnelling (I_3)

is small and the gate leakage is negligible but grows almost exponentially as the gate voltage increases. The above argument shows that in the gate leakage in subthreshold region is very small. Analog decoders can tolerate small gate leakages. The bipolar decoders that have been previously reported [11] are a good example of this fact. The base current in a bipolar transistor is larger than the MOSFET subthreshold gate leakage.

4.2.4 Summary

Transistor sizing is very important for nano-scale analog design. Several different parameters are affected by changing the transistor dimensions. Each of the above effects, forces a limitation on the transistor sizing. As the result it is not recommended to choose very long or short channel lengths. The above sections can be summarized in the following tables. Table. 4.1 shows the reasons why it is not recommended to use short length, width or small drain to source voltages. Table. 4.2 summarizes why very long transistor sizes or high drain to source voltages are not recommended

Table 4.1: Lower limit on design recommendations for nano-scale analog decoders

Design Variable	**Why small values are not recommended**
Width W	Excessive threshold variations (Fig. 4.2) Reduction in sub-threshold bound (Fig. 4.2)
Length L	Excessive threshold variations (Fig. 4.1) Excessive variations in n (Fig. 4.4)
V_{ds}	Sub-threshold current dependance on drain voltage (3.1) Excessive DIBL effect (Fig. 4.3)

Table 4.2: Upper limit on design recommendations for nano-scale analog decoders

Design Variable	**Why large values are not recommended**
Width W	Excessive Leakage Currents (PN leakage section) Slow signal propagation (Large junction capacitors)
Length L	Reduction in sub-threshold bound (Fig. 4.1) Slow signal propagation (Large junction capacitors)
V_{ds}	Excessive Leakage Currents (GIDL and BTBT leakage currents)

4.3 Leakage Current and Theoretical Limits

In the previous chapter we saw that currents in analog decoders based on Gilbert multipliers represent probabilities and that in an actual fabricated transistor with non zero drain to source voltage, the currents never go to zero. It can already be seen that we might run into a problem when dealing with very small probabilities. In this section we take our variable node analysis one step further and include a constant leakage current in each transistor. We first carry out the analysis for a differential pair and then use the result in the Gilbert multiplier which will lead us to the variable node behaviour in the presence of leakage currents. As we saw earlier, using a longer transistors and with drain to source voltages above 200mV, the role of drain in the sub-threshold region is very minor. Therefore, in this analysis we ignore the variations in drain voltage. Fig. 4.7 depicts the variable node architecture we are analysing.

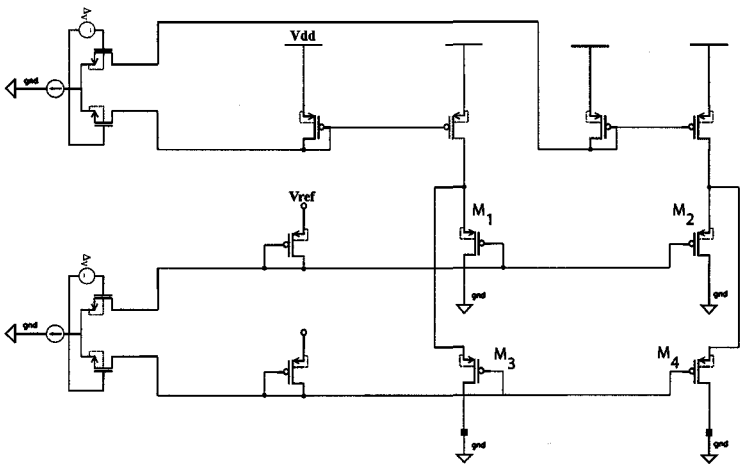

Figure 4.7: Variable node circuit for maximum LLR clipping analysis

4.3.1 The Differential Pair

The current in each branch of the differential pair consists of two components: the sub-threshold and the leakage current. We did not include the leakage in the normalizing current (I_U). This is an acceptable assumption since the normalizing current is large and constant. Also, a common value $I_{Leakage}$ for all leakage currents will be assumed,

$$I = I_{sub-threshold} + I_{Leakage} \tag{4.9}$$

The currents in Fig. 4.8 follow these equations:

$$I_+ + I_- = I_U \tag{4.10}$$

$$I_+ = I_0 \exp\left(\frac{V_+ - V_S}{nV_T}\right) + I_{Leakage}, \tag{4.11}$$

$$I_- = I_0 \exp\left(\frac{V_- - V_S}{nV_T}\right) + I_{Leakage}, \tag{4.12}$$

where I_0 is the sub-threshold current at zero gate source drive. Using these equations we can rewrite the current in terms of the differential voltage drive (Δ_V), leakage

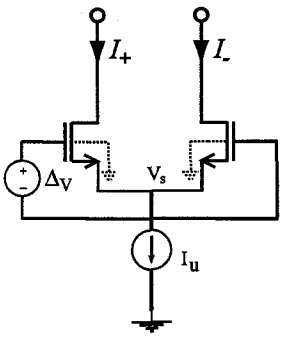

Figure 4.8: Differential pair

current and the normalizing current,

$$I_+ = \frac{\exp(\Delta_V/nV_T)(I_U - I_{Leakage}) + I_{Leakage}}{1 + \exp(\Delta_V/nV_T)}, \tag{4.13}$$

and

$$I_- = I_U - I_+. \tag{4.14}$$

4.3.2 Gilbert Multiplier Analysis

Using the Gilbert Multiplier in Fig. 3.2 we can derive the output currents assuming each transistor has a constant leakage of $I_{Leakage}$.

$$I_{Drain} = I_0 \exp\left(\frac{V(M_i)}{nV_T}\right) + I_{Leakage},$$

where I_0 is the sub-threshold current at zero gate source drive and $V(M_i)$ is the gate source voltage of the M_i. Using a voltage loop with M_2, M_6, M_5 and M_1 we have:

$$\frac{I_{M_2} - I_{Leakage}}{I_{M_6} - I_{Leakage}} = \frac{I_{M_1} - I_{Leakage}}{I_{M_5} - I_{Leakage}}. \tag{4.15}$$

By Kirchhoff's current law we have:

$$I_{M_5} = I_{x_2} - I_{M_6}.$$

Then using the above equations:

$$I_{M_6} = \frac{I_{M_2} I_{x_2} - I_{Leakage}(I_{M_2} - I_{M_1} + I_{x_2})}{I_{M_1} + I_{M_2} - 2I_{Leakage}}. \tag{4.16}$$

The voltage loop with M_1, M_3, M_4 and M_2 can be used to calculate I_{M_3}. Using the same steps as above:

$$I_{M_3} = \frac{I_{M_1} I_{x_1} - I_{Leakage}(I_{M_1} - I_{M_2} + I_{x_1})}{I_{M_1} + I_{M_2} - 2I_{Leakage}}. \tag{4.17}$$

From (4.17),(4.16), The output LLR of the variable node is:

$$\text{LLR}_{out} = \ln\left(\frac{I_{M_2} I_{x_2} - I_{Leakage}(I_{M_2} - I_{M_1} + I_{x_2})}{I_{M_1} I_{x_1} - I_{Leakage}(I_{M_1} - I_{M_2} + I_{x_1})}\right). \tag{4.18}$$

The source of the input currents such as $I_{M_1}, I_{M_2}, I_{x_1}$ and I_{x_2} are the differential pair we studied earlier. In this step we will substitute the results from (4.13) and derive the output LLR in terms of the differential input voltage and the normalizing current. I_U. The input LLRs are of the form $LLR_i = \Delta_{V_i}/nV_T = \lambda_i$ and $\beta = I_{Leakage}/I_U$ is the ratio between the leakage current and the normalizing current.

$$\lambda_{out} = \ln\left(\frac{[1 + \exp(\lambda_2) - 2\exp(\lambda_1 + \lambda_2)] \times \beta + \exp(\lambda_1 + \lambda_2)}{1 + [\exp(\lambda_2 + \lambda_1) + \exp(\lambda_1) - 2] \times \beta}\right) \tag{4.19}$$

If we have no leakage current, $\beta = 0$ and $\lambda_{out} = \lambda_1 + \lambda_2$, which is the ideal case. In the presence of leakage currents as (4.19) shows for large input LLRs, ($\lambda_i > 7$), the output LLR approaches:

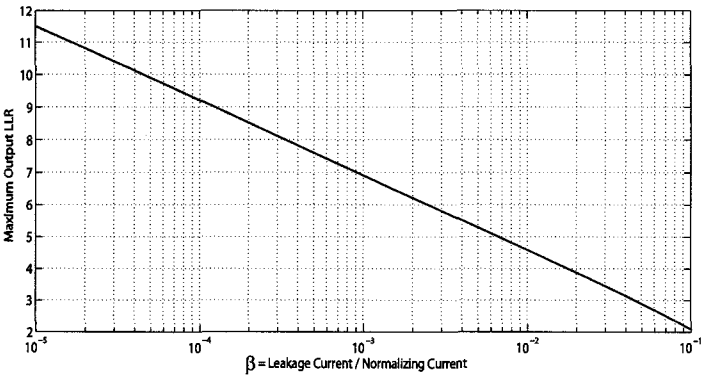

Figure 4.9: Output LLR clipping based on the leakage to normalizing current ratio

$$\lambda_{out} = \ln\left(\frac{1-2\beta}{\beta}\right). \tag{4.20}$$

Fig. 4.9 shows this upper bound on maximum possible LLR value in the circuit. The performance loss and system-level analysis due to LLR clipping are discussed in the last chapter. Larger values of β represent deep sub-micron CMOS technologies. As it was presented earlier the leakage currents increase in these technologies. One way to avoid large β values is to use a higher normalizing current source. The higher values of normalizing current, I_U leads to higher power consumption. Also very big currents force the circuit to move from the sub-threshold region to strong inversion. The latter reduces the accuracy of our circuits.

Here we will compare the results with circuit simulations from a $65nm$ CMOS process to verify the accuracy of the simulations. In the leakage analysis we assumed a direct connection between the differential pairs and the multiplier. The circuit architecture uses nMOS differential pairs and a pMOS Gilbert multiplier. This way the current directions in the differential pair and Gilbert multiplier match and we can connect the two directly without any current mirrors. An ideal 1μA current source was used as the normalizing current. In order to achieve higher leakage currents in the circuit level we have artificially added current sources between the drain and bulk terminal

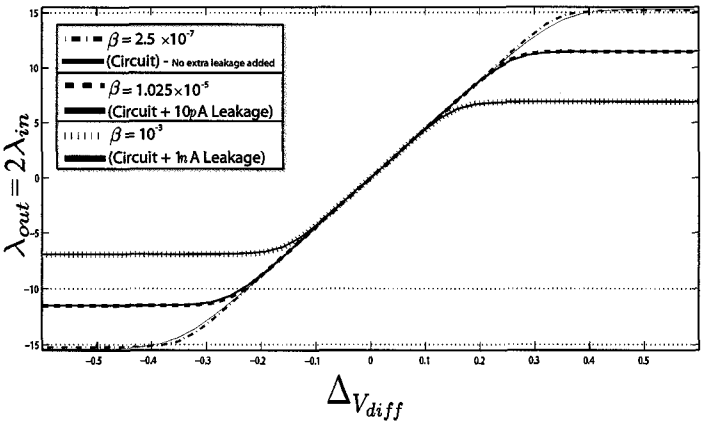

Figure 4.10: Circuit simulations and analytical plots of output LLR versus differential input voltage for different leakage current values

of the transistors. We applied the same voltage to both inputs and calculated the natural logarithm of the ratio of the currents form transistor $M4$ to $M1$ in Fig. 4.7. For conversion between voltages and LLRs we used a value of subthreshold swing which best matched the circuit results. The simulation results are presented in Fig. 4.10. The figure shows the close match between the circuit simulations from Cadence and the analytical results for three different values of β. It is clear that there is a good match between the analytical and the circuit results in this case.

4.4 Mismatch Analysis

Modern CMOS processes are highly sensitive to process variations. Phenomena such as local V_t variations can make results deviate from circuit simulators. Mismatch can be caused due to many different variations in the fabrication process. The fabricated width and length of the transistors can be different from the implemented dimensions and the channel doping varies in local areas. Also the thickness of the oxide varies across the die. As a result two matched transistors placed next to each other can experience different threshold voltages or channel lengths. These fluctuations lead to error in the threshold voltage of the transistor. In long channel transistors, ion implantation dose and dielectric thickness were the major factors behind threshold

variations. In sub-100nm CMOS processes other factors are added to the picture: the effective length, the effect of halo implants and charge sharing due to DIBL effect. It is not quite obvious how each of these short channel effects contribute to the threshold mismatch as these factors are correlated. But it has been shown through many different measurement that the mismatch factor A_Δ which will be introduced in this section, increases rapidly with shorter channel lengths [54]. The threshold of the transistor is a function of the effective length, L_{eff} and the width of the transistor. The effect of these two combined in addition to surface charge variations specifies the change in threshold voltage. These variations represent the local threshold variations. Transistors on a die also experience global threshold variations. The global variations are location depended are a function of the distance between transistors. Therefore for the case of Gilbert multipliers, these variations are negligible. The local variations assumed to have a Gaussian distribution with variance $\sigma^2_{V_t}$ [55] [56]:

$$\sigma_{V_t} = \frac{A_\Delta}{\sqrt{WL}}, \tag{4.21}$$

where A_Δ is a process parameter with $mV.\mu m$ units. The value of this parameter depends on the fabrication process and can be found in the process documentation. The value of A_Δ decreases with oxide thickness and hence, scales with technology [57]. This decrease starts to saturate for sub-100nm processes and in some cases even the reverse effect is experienced and A_Δ increases with scaling [48]. Mismatch is also present in mobility, C_{ox}, width and the length of a transistor. The net effect of these variations on current factor, $\beta = \mu C_{ox} W/L$, also has a gaussian distribution and affects the drain current mismatch. The statistic of current factor mismatch is given by (4.22)

$$\sigma(\frac{\Delta_\beta}{\beta}) = \frac{A_\beta}{\sqrt{WL}}, \tag{4.22}$$

In the sub-threshold region of operation we are dealing with small currents, the A_β is a small number on the order of 1% and the current has an exponential dependance on

the threshold voltage. In addition, the mismatch variations in the sub-threshold swing parameter, n, is negligible [58]. Therefore, the threshold variation is the dominant mismatch causing effect [48]. Hence, in this section our focus is on threshold variations and their effects on sub-threshold current mismatch. The typical value for A_Δ in nano-scale processes is around $4mV\mu m$. Using this value the total estimated threshold mismatch for a minimum sized transistor in a typical 65nm CMOS technology is $\sigma_t = 35mV$. As we show, this amount of variation has severe effects on the performance of analog decoders.

Some early work has been done on mismatch analysis for analog decoders [33] [59]. Winstead used the current mismatch model:

$$I_{Actual} = I_{Ideal}.\delta,$$

where δ is a Gaussian variable with mean, $\mu = 1$ and variance $= \sigma_\delta^2$ [33] but, As we mentioned earlier, the Gaussian current mismatch is small compared to threshold variations. Hence, here we are using a more accurate mismatch model than the one used previously. Using this model the LLR variation density function of a variable node is derived to calculate the performance loss of the decoder.

For the purposes of our mismatch analysis, the drain current model of each transistor given by (4.23):

$$I_{D,Actual} = I_{D,Ideal}.\exp\left(-\frac{\Delta_{V_t}}{nV_T}\right), \tag{4.23}$$

where Δ_{V_t} is a Gaussian variable with zero mean and variance $= \sigma_\Delta^2$. Now recall the Gilbert multiplier in Fig. 3.2. The actual currents in the multiplier are:

$$I_{y_1,a} = I_{y_1}.\exp\left(-\frac{\Delta_{V_{t1}}}{nV_T}\right) = I_{y_1}/\varepsilon_1, \tag{4.24}$$

$$I_{y_2,a} = I_{y_2}.\exp\left(-\frac{\Delta_{V_{t2}}}{nV_T}\right) = I_{y_2}/\varepsilon_2, \tag{4.25}$$

$$I_{x_1y_1,a} = I_{x_1y_1}.\exp\left(-\frac{\Delta_{V_{t3}}}{nV_T}\right) = I_{x_1y_1}/\varepsilon_3, \tag{4.26}$$

$$I_{x_1y_2,a} = I_{x_1y_2}.\exp\left(-\frac{\Delta_{V_{t4}}}{nV_T}\right) = I_{x_1y_2}/\varepsilon_4, \tag{4.27}$$

$$I_{x_2y_1,a} = I_{x_2y_1}.\exp\left(-\frac{\Delta_{V_{t5}}}{nV_T}\right) = I_{x_2y_1}/\varepsilon_5, \tag{4.28}$$

$$I_{x_2y_2,a} = I_{x_2y_2}.\exp\left(-\frac{\Delta_{V_{t6}}}{nV_T}\right) = I_{x_2y_2}/\varepsilon_6, \tag{4.29}$$

where a index indicates actual current in the circuit and $\varepsilon_i = \exp\left(\frac{\Delta_{V_{t(i)}}}{nV_T}\right)$. The ε_i has a log-normal distribution and its parameters depend on A_Δ, W and L. A voltage loop leaves us with the following relationship between the currents:

$$\frac{I_{y_1}.\varepsilon_1}{I_{x_1y_1}.\varepsilon_3} = \frac{I_{y_2}.\varepsilon_2}{(I_{x_1} - I_{x_1y_1}).\varepsilon_4},$$

$$\frac{I_{y_1}.\varepsilon_1}{(I_{x_2} - I_{x_2y_2}).\varepsilon_5} = \frac{I_{y_2}.\varepsilon_2}{I_{x_2y_2}.\varepsilon_6}.$$

Using the above relationship, the actual output currents of the multiplier are:

$$I_{x_1y_1,a} = \frac{I_{x_1}I_{y_1}.\varepsilon_1.\varepsilon_4}{\varepsilon_1.\varepsilon_4 I_{y_1} + \varepsilon_2.\varepsilon_3 I_{y_2}}, \tag{4.30}$$

$$I_{x_2y_2,a} = \frac{I_{x_2}I_{y_2}.\varepsilon_2.\varepsilon_5}{\varepsilon_1.\varepsilon_6 I_{y_1} + \varepsilon_2.\varepsilon_5 I_{y_2}}. \tag{4.31}$$

$$\tag{4.32}$$

The output likelihood value is the natural log of the ratio between the above two currents:

$$\lambda_a = \lambda_i + \ln \frac{I_{y_1}(\varepsilon_1^2\varepsilon_4\varepsilon_6) + I_{y_2}(\varepsilon_1\varepsilon_2\varepsilon_4\varepsilon_5)}{I_{y_2}(\varepsilon_2^2\varepsilon_3\varepsilon_5) + I_{y_1}(\varepsilon_1\varepsilon_2\varepsilon_4\varepsilon_5)}. \tag{4.33}$$

where λ_a is the actual LLR in the circuit and λ_i is the ideal LLR. Using (3.11) we can rewrite everything in terms of input probability $P_{y0} = I_{y_1}/I_u$, where I_u is the normalizing current. Therefore the mismatch part is:

$$\lambda_{mismatch} = \ln \frac{1 + P_{y0}(e^{\frac{\Delta V_{t1}}{nV_T} + \frac{\Delta V_{t6}}{nV_T} - \frac{\Delta V_{t2}}{nV_T} - \frac{\Delta V_{t5}}{nV_T}} - 1)}{1 + P_{y1}(e^{\frac{\Delta V_{t2}}{nV_T} + \frac{\Delta V_{t3}}{nV_T} - \frac{\Delta V_{t1}}{nV_T} - \frac{\Delta V_{t4}}{nV_T}} - 1)} \tag{4.34}$$

New variables can be defined as, $\alpha = (\frac{\Delta V_{t1}}{nV_T} - \frac{\Delta V_{t2}}{nV_T})$, $\beta = (\frac{\Delta V_{t5}}{nV_T} - \frac{\Delta V_{t6}}{nV_T})$, $\theta = (\frac{\Delta V_{t3}}{nV_T} - \frac{\Delta V_{t4}}{nV_T})$. These new variables will also be Gaussian with $\sigma^2_{new} = 2 \times \sigma^2_{old}$ and mean of zero. Re-writing the mismatch in the terms of the new variables we have:

$$\lambda_{mismatch} = \ln \frac{1 + P_{y0}(e^{\alpha-\beta} - 1)}{1 + P_{y1}(e^{\theta-\alpha} - 1)} \tag{4.35}$$

Unfortunately the probability density function of P_{y0} is unknown. Hence, obtaining a closed form is impossible. The two options we have to accurately identify the effect of mismatch on our decoder are Monte-Carlo simulations of the decoder using the above expression or density evolution (DE). DE becomes highly computationally intensive. A C-Mex implementation of the joint pdf with more than 6bits = 64 bins has prohibitively long runtime. On the other hand running the density evolution with 6 bits already heavily degrades the results. According to our DE simulations, running DE on a (3,6)-regular code with 6 bits degrades the performance by more than $2.8dB$. For our simulations, we ran DE with 12 bits = 4096 points and the LLR mismatch calculation code with 6 bits = 64 points on a smaller LLR range of $[-5, 5]$ and the main DE clips the values at $[-10, 10]$. Due to the limited accuracy the results may not be very reliable, nevertheless it is worth considering. The following table shows the required E_b/N_0 for different values of σ_t. The code ensemble used is

a (3,6)-regular code.

Table 4.3: Density evolution considering mismatch on a (3,6)-regular code

$\sigma_t(mV\mu m)$	$E_b/N_0(dB)$
0.00	1.1
2.00	1.6
3.00	2.3

We can apply the same models and derive the output of the check node in the presence of mismatch. Following the same procedure we obtain:

$$\mathrm{LLR}_{check} = \ln\left(\frac{\varepsilon_1^2\varepsilon_4\varepsilon_6\,\mathrm{P}_{x0}\,\mathrm{P}_{y0}^2 + \varepsilon_1\varepsilon_2\varepsilon_4\varepsilon_5\,\mathrm{P}_{y0}\,\mathrm{P}_{y1} + \varepsilon_2^2\varepsilon_3\varepsilon_5\,\mathrm{P}_{x1}\,\mathrm{P}_{y1}^2}{\varepsilon_1^2\varepsilon_4\varepsilon_6\,\mathrm{P}_{x1}\,\mathrm{P}_{y0}^2 + \varepsilon_1\varepsilon_2\varepsilon_3\varepsilon_6\,\mathrm{P}_{y0}\,\mathrm{P}_{y1} + \varepsilon_2^2\varepsilon_3\varepsilon_5\,\mathrm{P}_{x0}\,\mathrm{P}_{y1}^2}\right) \tag{4.36}$$

In an analog decoder, the output of the check node is normalized before passing to the next node for processing. This normalizing block can be enforced by making sure that the output current probabilities always add up to the normalizing current. In the next system level mismatch decoder simulations we present, this normalizing step is enforced.

To better verify the results, Monte-Carlo simulations were used for a (3,6)-regular code with $n = 816, k = 408$. Fig. 4.11 shows the decoder Bit Error Rate (BER) results. The ideal node is simulated assuming a perfect multiplier. The rest of the curves are extracted using direct data from the Cadence circuit simulator based on the variable node designed in a 65nm CMOS technology. The term "SPC" refers to this circuit based simulations. The decoder used 50 iterations. For generating the BER curves we searched for 50 frame errors in small threshold mismatch values, 70 for moderate and 90 for the higher bit error rates. The choice of these numbers was based on the statistics reports from the decoder simulator to ensure accuracy. The simulations use different threshold mismatch variances for the variable node and check nodes. This is because the decoder is far more sensitive to mismatch in the check node than the variable node. This is an important observation since it will

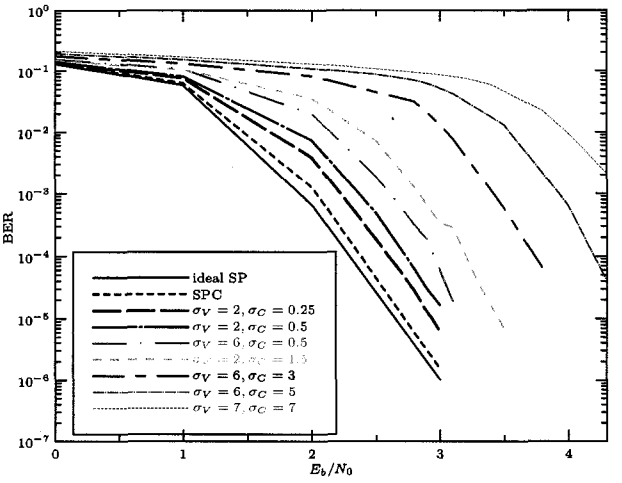

Figure 4.11: Bit Error rate versus E_b/N_0 for different threshold variations, assuming the transistor width is twice its length: $W = 2 \times L$.

allow for savings in area for the variable node.

In order to have a better feeling about what these standard deviation values mean, table 4.4 maps the threshold standard deviation values to the transistor sizing. For the threshold mismatch mapping, we assumed that $W = 2 \times L$. Using the fabrication parameters, $A_\Delta = 4mV\mu m$ the mismatch standard deviation was calculated for different transistor sizes.

Table 4.4: Mapping of the threshold standard deviation to transistor sizing

σ	**W** (μm)	**L**(μm)	**W** $\times$ **L** $(\mu m)^2$
0.25	22.64	11.32	256.3
0.50	11.32	5.66	64.1
1.50	3.78	1.89	7.1
2.00	2.84	1.42	4.0
3.00	1.90	0.95	1.8
5.00	1.14	0.57	0.6
6.00	0.96	0.48	0.5
7.00	0.82	0.41	0.3
9.00	0.64	0.32	0.2

4.5 Electrical Noise

Noise in the MOSFET can degrade the accuracy of the circuit and analog decoders are not an exception. There has been some study on the effect of noise on analog decoders by J. Dai [60]. Unfortunately this study uses a wrong noise model. The spectral density function of the drain current noise in weak inversion is given by (4.37) [42]:

$$\sigma_{I_n}^2 = 2qI_D\left(1 + e^{-V_{DS}/V_T}\right)\Delta f \tag{4.37}$$

In order to calculate the maximum standard deviation of the noise we need to know the current I_D and the bandwidth Δf. The maximum current in the decoder, I_D is the normalizing current. A reasonable value for the normalizing current that we used throughout this thesis is $1\mu m$. Using the average convergence speed of the designed nodes that we will present in the next chapter, the average bandwidth is 20MHz. From these values the standard deviation of noise current is 2.53nA. Such a change in current can be caused by a threshold variation with $\sigma = 0.063$ which according to mismatch analysis is negligible. Therefore we can ignore the effect of thermal noise.

Chapter 5

Implementation of variable nodes for LDPC decoder

This chapter describes the implementation of analog decoder variable nodes in 90nm and 65nm CMOS technologies. The implementation of the variable nodes is based on the Gilbert multipliers and we used two different circuit topologies. Only the details from one of the circuit topologies are presented here[1]. Another possibility that we consider here is the use of pMOS versus nMOS transistors for the Gilbert multipliers. Typically, Gilbert multiplier based decoders are designed using nMOS transistors [61]. In this chapter, we will present the arguments supporting each decision. Therefore, we implemented four different variable nodes for each technology. In each case the guidelines from the fourth chapter were employed to achieve good accuracy. Yet we used a wide range of transistor sizes. This variety enables us to test the fabricated chips and confirm the significance of our mismatch theory and the accuracy of circuit simulations. This chapter reports various post-layout simulation results. The energy consumption and speed of the circuits is compared for different situations. In addition, we present the accuracy of each circuit for the set of all useful inputs.
The following is an outline of the chapter. The first section discusses the advantages

[1]The implementation of the fundamental LDPC decoder nodes was accomplished in close collaboration with Jorge Perez, Cyril Lahuec, Fabrice Seguin and Michel Jezequel from TELECOM Bretagne - PRACOM - France. Here the circuits related to this thesis are presented.

and disadvantages of designing Gilbert multipliers with pMOS versus nMOS transistors. Section 5.2 presents the topology, design methodology and simulation results for the implemented nodes in 90nm and 65nm CMOS technologies. The third section is a new proposed variable node design. The advantage of this proposed topology is its inherent symmetry. Therefore, unlike the previous structures, this topology treats both input signals in the same way. These input signals are the input LLRs from the channel and or the check nodes. The fourth section briefly discusses the testing strategy for the fabricated chips. Finally the last section summarizes the chapter. All the circuit simulation data in this chapter are based on BSIM4 post-layout simulations from the Cadence kit.

5.1 Variable node design: pMOS versus nMOS

The design of variable nodes can be accomplished using either nMOS or pMOS transistors. To the best of our knowledge, all the reported variable nodes for analog decoders use nMOS transistors. There are several advantages and disadvantages in choosing a pMOS based structure over an nMOS based one. The argument supporting the use of pMOS transistors over the nMOS transistors was first brought up in [39]. The main advantage of a pMOS transistor over an nMOS is the ease of applying the voltage to the bulk terminal. We will explain shorty why this would be an advantage in analog decoders. In a typical CMOS process, pMOS transistors are fabricated in an n-well. The well acts as an isolation mask and separates the bulk terminal of the pMOS transistors. The nMOS transistors share a common bulk voltage. This voltage is usually the lowest voltage in the circuit. Many modern processes have a triple well (deep n-well) option that allows the bulk of the nMOS transistors to be isolated from the die substrate. The deep n-well provides the freedom to connect the bulk terminal of an nMOS transistor to a desired voltage. Yet it does not affect the DC characteristics of the transistor [62]. The big disadvantage of using deep n-well is its huge cost in terms of space. In our 65nm CMOS process, each group of nMOS transistors with a common bulk voltage requires a minimum of $25(\mu m)^2$ space. This is 29 times bigger than the required space for a nMOS transistor with $L = 4 \times L_{min}$

and $W = 4 \times W_{min}$. According to [39] connecting the bulk to the source terminal results in better matching between the simulation models and the actual transistor behaviour. Also it was mentioned that the variations due to the drain to source voltage will be reduced. This is a direct effect of the milder DIBL effect under $V_{bs} = 0$ condition that is mentioned in the previous chapter. The bulk source voltage also appears directly in the threshold voltage equation [41]:

$$V_t = V_{t0} + \gamma(\sqrt{\Phi_s - V_{bs}} - \sqrt{\Phi_s}), \tag{5.1}$$

where V_{t0} is the threshold voltage assuming zero source to bulk voltage, γ is the body effect coefficient and V_{bs} is the bulk to source voltage. Connecting the bulk terminal to a fixed voltage can change the threshold voltage when the source voltage changes. This is an undesirable effect for analog decoders. By using pMOS transistors we can easily avoid this situation. Another advantage of pMOS transistors is their lower mismatch variations. In many technologies, pMOS transistors experience less threshold mismatch variations compared to their nMOS counter-parts. This difference is fabrication technology based and varies from one location to another. The A_Δ in (4.22) for pMOS can be almost half the value of an nMOS counterpart. The disadvantage of using pMOS transistors is the larger size. pMOS transistors have a smaller mobility factor [42]. Therefore, a pMOS transistor requires a larger width for the same current performance as an nMOS transistor. The gate bulk capacitor of a transistor is a direct function of its dimensions [42]. Hence the larger width introduces larger parasitic capacitance that makes the circuit slower and has the potential to increase its power consumption. This fact becomes clearer when we present the energy consumption and the speed results in the next section. The next issue is the larger area consumption. Each bulk to source connected pMOS transistor in a Gilbert multiplier carries a separate n-well. Due to the restriction on the minimum spacing between the n-wells the circuit with pMOS transistors consumes more area on the chip. Therefore the choice between pMOS and nMOS transistors is not a trivial choice. Achieving accuracy is easier with a pMOS circuit but the cost is the larger

area and lower speed.

5.2 Variable nodes implemented in 90nm and 65nm CMOS

Over the past decade, several analog decoders have been designed, fabricated and tested. Yet there is no record of any analog decoder implementation in a sub-100nm CMOS process. In this section, we present the post-layout simulation results of the variable nodes that we designed in 90nm and 65nm CMOS processes. These prototypes have been sent for fabrication and will be tested upon delivery. The circuit topology is shown in Fig. 5.1. In the simulations presented in this chapter, we used voltage signals. This makes it easier to test the chip.

Here we explain the circuit in more detail. The circuit topology shown in Fig. 5.1 uses a separate but identical differential pair for each of the inputs. The differential pair circuit (M_5, M_6) is shown on the left and the Gilbert multiplier is on the right. In the input differential pair circuit, the transistors on the top (M_7, M_8) serve as a current mirror. The differential pair converts the input voltages into currents and sends them to the Gilbert multiplier. Transistor M_1 through M_4 are the main Gilbert multiplier transistors. Voltage V_{ref} controls the drain to source voltage of bottom transistors. In order to ensure that there is enough voltage headroom for the transistors on the bottom this voltage has to be higher than $0.3V$. Lower voltage values require minor modification to the circuit [63]. To convert the signals back to the voltage domain, the currents are routed to an output stage shown in Fig. 5.2. In this circuit, transistors M_2 to M_5 are current mirrors and transfer the output currents I_{out+} and I_{out-} from the multiplier to the differential output stage. Transistors M_1 and M_6 are dummy transistors and are connected to I_{dummy_1} and I_{dummy_2} from the Gilbert multiplier. These transistors help the symmetry condition of the Gilbert multiplier transistors. Without the drain connected dummies, the two main outputs will see the impedance of the diode-connected transistors between their drain and the voltage source, while the dummy signals will be directly connected to the voltage source. Therefore the

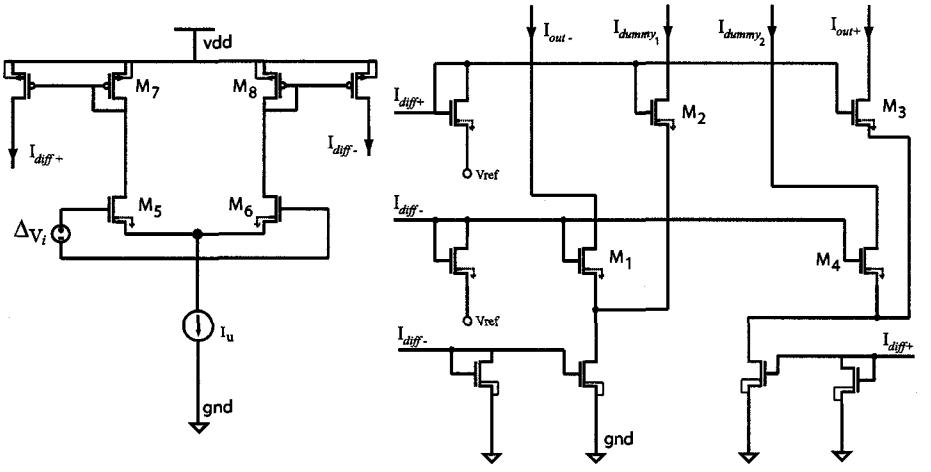

Figure 5.1: Variable node topology common in CMOS based analog decoders.

Gilbert multiplier transistors will experience different drain to source voltages. Hence using the dummy transistors provides better symmetry for the circuit. Designers tend to use current as the input and output signals when they use this structure. In such settings, the input differential pairs are removed and the output stage is replaced by a normalizing current stage (*see* Fig. 3.5). In this thesis we use voltage input/output signals for this topology. In the previous section, it was pointed out that in the sub-threshold region, pMOS transistors can provide better accuracy than nMOS transistors. The pMOS version of these circuits is the dual of the circuit with all the nMOS transistors replaced by pMOS and vice versa. The rest of this section explains our strategy for transistor sizing and presents the simulation results for four different variable nodes in 90nm and 65nm CMOS processes.

5.2.1 Design Strategy for 90nm CMOS Variable Nodes

The first set of variable nodes that we designed were fabricated in a 90nm CMOS process. Both the pMOS version and the nMOS version were implemented. In the design of the variable nodes we used the guidelines in the fourth chapter as well as the proposed strategies in [39]. Here we will review some of these recommendations. Accurate behavior of our variable nodes requires the gate to source drive to be far

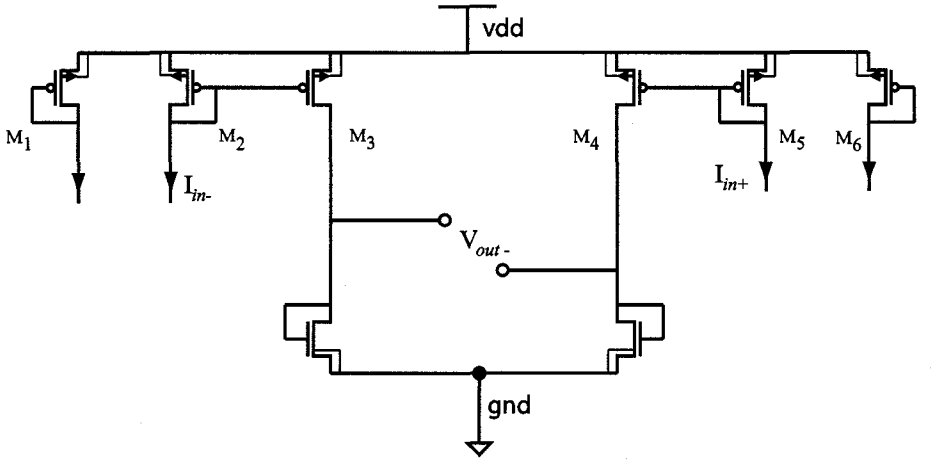

Figure 5.2: Output stage for converting the currents in into a differential voltage

away from the threshold voltage. The exponential overdrive term in the current model can be an indicator of how deep in the sub-threshold region the circuit is operating. In the literature, the term inversion coefficient is used for this metric and it shows the level of inversion in the channel. In the EKV CMOS models, the inversion coefficient in the sub-threshold region is [64]:

$$I_{D,Weak} = 2n\mu C_{ox} V_T^2 (W/L) \left(e^{(V_{gs}-V_t)/nV_T}\right), \tag{5.2}$$

$$I_C = \left(e^{(V_{gs}-V_t)/nV_T}\right), \tag{5.3}$$

and

$$I_{C_{max}} = \frac{I_{D,Weak(max)}}{2n\mu C_{ox} V_T^2 \left(\frac{W}{L}\right)}, \tag{5.4}$$

where $I_{D,Weak}$ is the weak inversion current, I_C is the inversion coefficient and $I_{C_{max}}$ shows the maximum inversion coefficient for a given maximum drain current. In the design rules presented in [39], the authors suggest an inversion coefficient much less than one. Then the required W/L ratio that guarantees this inversion coefficient can be calculated:

$$\frac{W}{L} = \frac{I_{D,Weak(max)}}{2n\mu C_{ox} V_T^2 I_{C_{max}}} \tag{5.5}$$

In these calculations, it is assumed that the transistor sizing has no effect on the threshold voltage. The inversion coefficient assures a certain distance between the maximum gate to source drive and the threshold as long as the threshold is constant. Looking back at the design guidelines proposed in the previous chapters we can see the more complete picture. In reality, the threshold changes with the W/L ratio. In CMOS processes smaller than 0.35μm, the short channel effects are helpful. The fourth chapter shows that a large W/L ratio pushes the threshold voltage to bigger values. Therefore, we have a larger margin and a bigger voltage swing. The inversion coefficient also determines the maximum voltage swing for a given drain current. The maximum possible voltage swing is important and it can be problematic if we limit the swing too much. Therefore, very small inversion coefficients are also undesirable.

5.2.2 Variable Nodes in 90nm CMOS

In the design of these variable nodes, we used double the minimum size length for the main Gilbert multiplier transistors. We also used W/L ratios larger than one. The transistor sizing of each part of the circuit is shown in the Table. 5.1 below. The single input sweep figures for the variable nodes are shown in Figs. 5.4 and 5.3. The pMOS circuit behaves very close to the ideal curve. The pMOS circuit also has a wide dynamic range. Figs. 5.6 and 5.5 show the output LLR when both inputs are swept. Also, Fig. 5.4 depicts the close behaviour of the two inputs. In order to perform the conversion from voltage to LLR domain we used the data from the input differential pair. In Fig. 4.4 we observed that the sub-threshold swing is not constant throughout the operation and varies with the gate to source drive as well as the transistor sizing. Therefore, the best way to perform the conversion is to directly use the input differential pair voltage to current ratio. Using (3.10), the subthreshold swing is:

$$nV_T = \frac{\Delta V_{diff}}{\ln I_+/I_-} = \frac{\Delta V_{diff}}{\text{LLR}}. \tag{5.6}$$

The differential voltages were mapped to the appropriate LLR value using the above equation for each circuit. In all the DC sweep graphs, the average of the swing (5.6) over the input differential voltage range was used for conversion between the voltages and LLR values. So far, we have presented the accuracy and the maximum range of operation of the two designs. The next important factor is a power and speed estimation. The latter is not an easy task since both the power consumption and the speed are a function of the differential input values. For a fair comparison, ten different boundary conditions were chosen and we calculated the required energy and time delay for each transition. In addition, the average value of these numbers is presented as a single number for comparison purposes. It is important to notice that extracting the speed of the decoder from the delay values is not a straightforward task. Unlike digital decoders where we wait for the output of each node to converge to their final values before passing them around with each clock, in analog decoders the transfer of information happens in real time. Nevertheless, these numbers can clearly compare the energy consumption and speed performance of these nodes. The ten different test settings are described in terms of three different conditions on the variable node inputs, as follows:

1. Zero (Z) = Zero differential drive

2. High (H) = The minimum positive differential drive that clips the output

3. Low (L) = The minimum negative differential drive that clips the output

In order to find the delay value, we calculated the time it takes for the output to reach 95% of its final value. This final voltage is between 0 and 700mV and depends on the differential inputs. The term In1 in the table stands for the first input and In2 indicates the second input. Table .5.2 provides the energy consumption and the delay of the 90nm CMOS nodes under the ten test setups we mentioned earlier. In

Fig. 5.7, the percentage of error for both designs is plotted. The pMOS based circuit has negligible error for small differential drives. This error increases in size as the input drive becomes larger. In order to calculate this error we took the average of the absolute value of the error between the two inputs when one input is sweeping and the other one has a zero differential drive. So far, we can see that each one of these implemented circuits has an advantage and a disadvantage over the other. A functional decoder node requires small error, large dynamic range, low power consumption and fast response. Judging the accuracy of the nodes is not an easy job. Therefore, for a fair comparison between these nodes, we need to use the simulation data in a system-level decoder. To achieve this goal, a full characterization of the node for all possible combinations of the inputs is required. The author used the Ocean scripting tools of the Cadence to generate a full 1201 by 1201 matrix of input differential drives between -600mV and 600mV. An input of 600mV is the maximum possible drive for all of these implemented nodes. The step size is 1mV, which translates to LLR step size between 0.018 and 0.026, depending on the transistors, the circuit and the differential drive. The voltages were then translated into LLRs. The exact voltage to LLR conversion was used for this step. Therefore, each voltage was mapped to the correct LLR using its unique sub-threshold swing. To construct the surface contour we converted the input voltages and the voltage error into their corresponding LLRs. This way the full table was converted into LLR representation. This matrix is used to generate the contour of the absolute error in the LLR values for the two pMOS designs in 90nm CMOS and all of our implemented circuits in 65nm CMOS. These simulation data points are used in the next chapter to accurately characterize the behaviour of the decoder under circuit imperfections. Here we present these graphs to provide a visual indication of the error in the variable nodes. Figs. 5.8 shows the absolute error contours for the pMOS version of the implemented variable node in 90nm CMOS. The color bar on the right of these graphs indicates the mapping between the colors and the error in the LLR values. The next section introduces the variable nodes designed in a 65nm CMOS process.

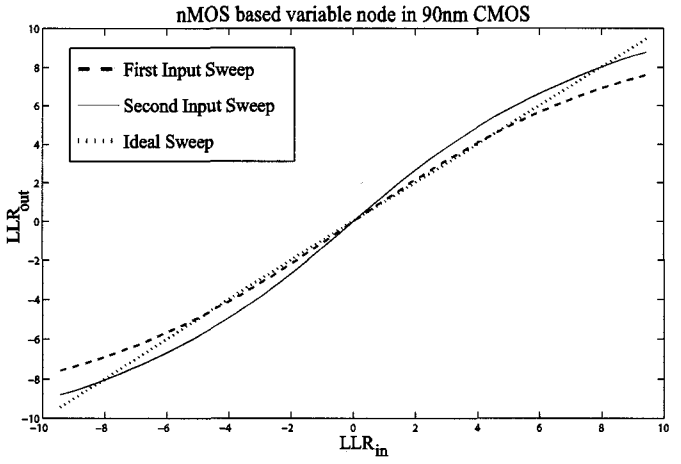

Figure 5.3: DC sweep of **one** terminal in **nMOS** variable node in 90nm CMOS

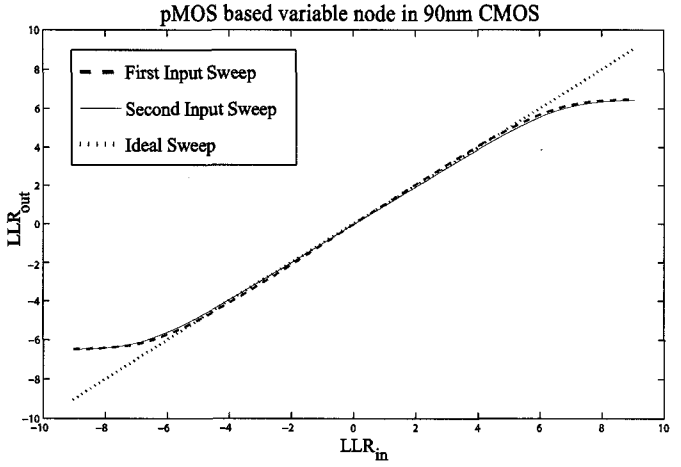

Figure 5.4: DC sweep of **one** terminal in **pMOS** variable node in 90nm CMOS

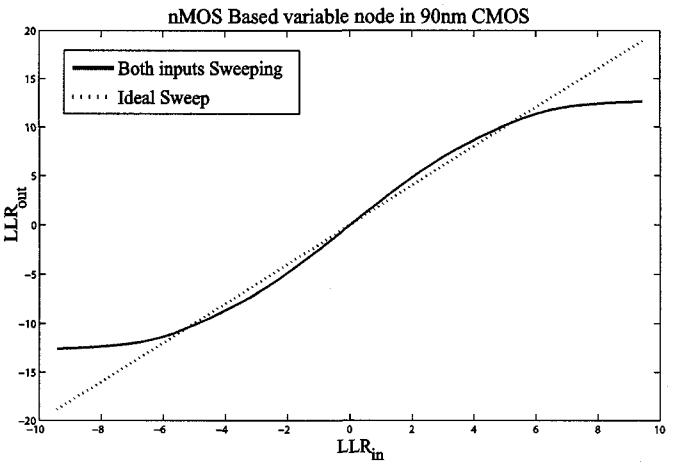

Figure 5.5: DC sweep of **both** terminals in **nMOS** variable node in 90nm CMOS

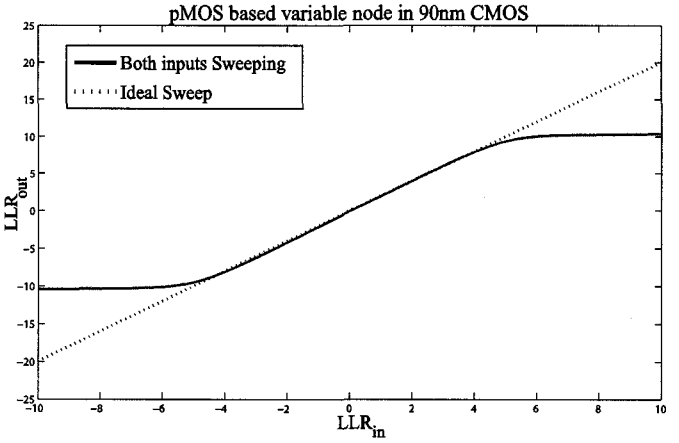

Figure 5.6: DC sweep of **both** terminals in **pMOS** variable node in 90nm CMOS

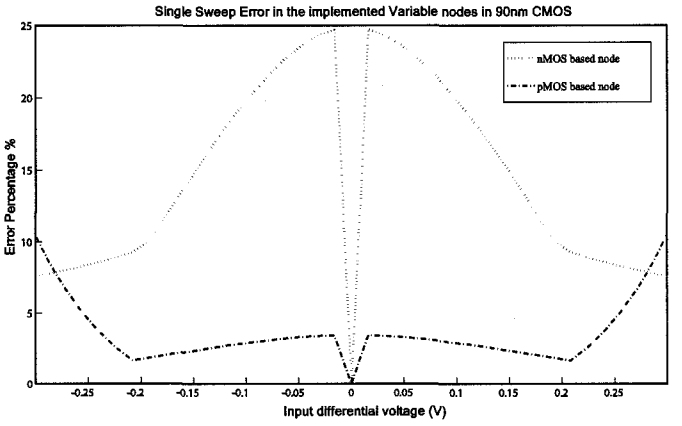

Figure 5.7: Percentage of error when sweeping one input in the 90nm CMOS nodes

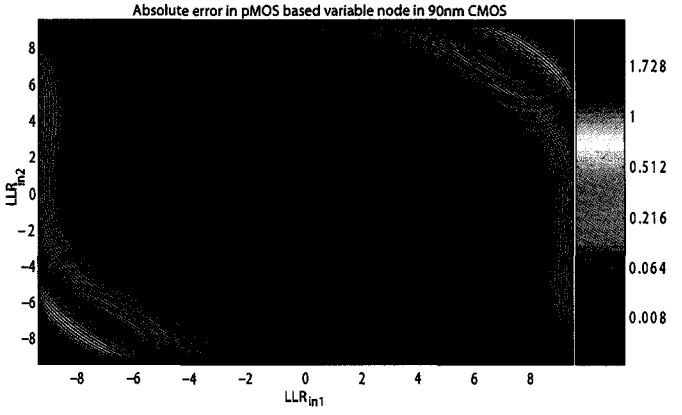

Figure 5.8: Absolute **error** in the **pMOS** variable node in 90nm CMOS

Table 5.1: Transistor sizing for 90nm CMOS variable node

Gilbert Multiplier Transistors		
	$nMOS(\mu m)$	$pMOS(\mu m)$
Width	0.70	2.00
Length	0.20	0.20
Input Differential Pair Transistors		
Width	1.20	0.60
Length	0.10	0.22
Output Stage pMOS Transistors		
Width	0.30	0.12
Length	0.10	0.10
Output Stage nMOS Transistors		
Width	0.12	0.28
Length	0.80	0.15
Differential Pair Current Mirrors		
Width	0.22	0.20
Length	0.10	0.10
Diode-connected Transistors		
Width	0.30	0.40
Length	0.10	0.10

5.2.3 Design Strategy for 65nm CMOS Variable Nodes

The 90nm CMOS simulation results are promising and have motivated our research towards applying the same rules in a smaller process. The 65nm CMOS process is currently the latest accessible CMOS technology that is available through CMC Microsystems [65]. In the design of the 65nm CMOS variable nodes, we mostly used the same guidelines as for our 90nm CMOS circuits. We also ran the design through an optimizer. The Cadence tool can be programmed to run optimization routines on the design. The optimization routines require design constraints, initial values and design goals. The optimizer searches the design space for the first values that match the design goals within a certain error margin. Therefore, we enforced a couple of constraints on the minimum transistor length and width, the range of

Table 5.2: Energy consumption and delay values for different test settings in 90nm CMOS circuit

Test Setting	nMOS version		pMOS version	
90nm CMOS	Energy (pJ)	Speed (μs)	Energy (pJ)	Speed (μs)
In1: $Z \rightarrow H$, In2: Z	0.2961	0.024	0.484	0.060
In1: $H \rightarrow Z$, In2: Z	0.2509	0.020	0.093	0.012
In1: Z, In2: $Z \rightarrow H$	0.2223	0.018	0.657	0.084
In1: Z, In2: $H \rightarrow Z$	0.1668	0.013	0.094	0.015
In1, In2: $Z \rightarrow H$	0.6222	0.050	1.836	0.210
In1, In2: $H \rightarrow Z$	0.1607	0.013	0.107	0.016
In1: $Z \rightarrow H$, In2: $Z \rightarrow L$	0.4317	0.035	0.296	0.040
In1: $H \rightarrow Z$, In2: $L \rightarrow Z$	0.1621	0.013	0.090	0.016
In1: $Z \rightarrow L$, In2: $Z \rightarrow H$	0.1135	0.009	0.104	0.012
In1: $L \rightarrow Z$, In2: $H \rightarrow Z$	0.1367	0.012	0.105	0.018
Average of the Column	0.2563	0.021	0.387	0.048

possible biasing voltages and the normalizing current. The optimizer performed the last set of simulations for the final design parameters. Table. 5.3 shows the transistor sizing chosen for the circuit.

In the design of the 65nm CMOS nodes, we tried to follow all the design recommendations from the fourth chapter. The smallest transistor length in the nMOS version is $0.3\mu m$ which is about five times larger than the minimum transistor length. The pMOS version also has long transistor lengths except for the output stage. This is because the output capacitance of the node has a major effect on the speed of the node. This output capacitor has to be charged and discharged during the node operation. Therefore, we tried to keep the area of the output transistors as small as possible. In addition, the smaller width decreases the pn leakage current of the output. The rest of the transistors in the design have large widths and lengths. The larger width helps to increase the threshold voltage and the larger length mitigates the performance loss due to mismatch. All the transistors in the Gilbert multiplier have an area (Width $\times$ Length) larger than 2 $(\mu m)^2$. This guarantees that the threshold variations have little effect on the performance of the cell. On the downside, the large transistor size slows

Table 5.3: Transistor Sizing for 65nm CMOS variable node

Gilbert Multiplier Transistors		
	$nMOS(\mu m)$	$pMOS(\mu m)$
Width	1.69	1.86
Length	2.30	4.53
Input Differential Pair Transistors		
Width	0.12	6.87
Length	1.01	1.99
Output Stage pMOS Transistors		
Width	1.35	0.16
Length	2.40	0.06
Output Stage nMOS Transistors		
Width	0.17	0.45
Length	3.88	3.62
Differential Pair Current Mirrors		
Width	4.32	0.90
Length	0.30	3.17
Diode-connected Transistors		
Width	1.04	0.62
Length	1.80	1.49

the node and increases the power consumption. The nodes have a near-perfect performance. They have a wider range of operation compared to the 90nm CMOS circuits we presented earlier and perform closer to the ideal curve. The Figs. 5.9, 5.10, 5.11 and 5.12 show the behaviour of the variable nodes we designed in 65nm CMOS. The conversion from voltages to the LLR domain uses the average sub-threshold swing of the node. The graph in Fig. 5.9 shows the output LLR in the 65nm CMOS nMOS variable node when one of the inputs is swept and the other one is driven with a zero differential voltage. Fig. 5.10 shows the same test setup for the pMOS version of the node. These graphs are followed by Fig. 5.11 which shows the simulation results for the nMOS node. In this simulation, both inputs are swept. The last graph in Fig. 5.12 illustrates the output LLR of the node when both inputs are swept in a pMOS version of the node. Both designs are very close to the ideal behaviour with less than

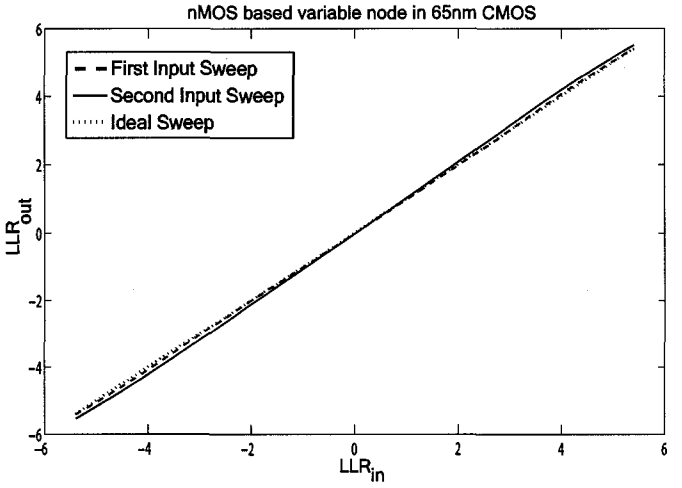

Figure 5.9: DC sweep of **one** terminal in the 65nm **nMOS** variable node

$\Delta_{LLR} = 1.5$ error in the LLR range of -10 to 10. The next important metrics are the energy consumption and delay, shown in Table. 5.4. The pMOS version of the node is very slow and consumes almost twice the energy of the nMOS version.
The above simulations can only reveal a small subset of all possible input values. The best way to judge the accuracy of the variable node is to look at the error contour introduced earlier. The contour surface covers the whole subset of meaningful inputs. The Fig. 5.14 shows the error contour surface for the curren65nm nMOS node and Fig. 5.13 illustrates this error for the pMOS version of the node.

5.3 Forced symmetry variable node

The previous section covered the implementation of different variable nodes in 90nm and 65nm CMOS technologies. The simulation of the implemented nodes is very close to the ideal behaviour but all of them shared one imperfection. The two inputs of the node did not behave completely similarly. For example note the difference between the dashed and solid line in Fig. 5.3. This fact can be deduced by looking at the topology of these nodes. In this section, the author proposes a new node topology that guarantees the symmetry between the two inputs of the variable node. The

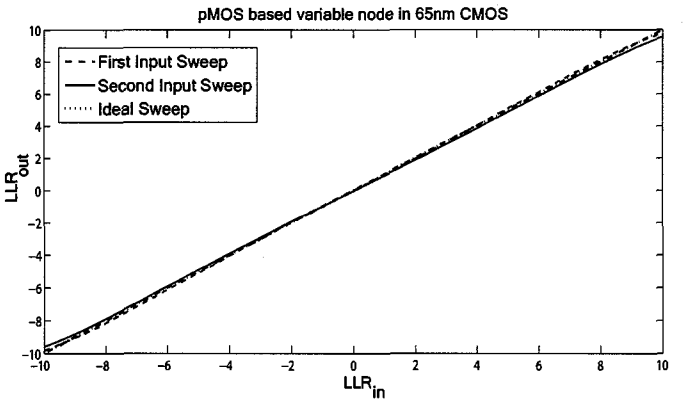

Figure 5.10: DC sweep of **one** terminal in the 65nm **pMOS** variable node

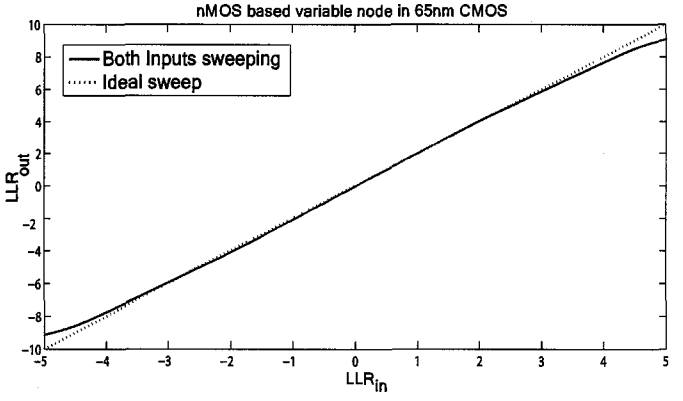

Figure 5.11: DC sweep of **both** terminals in the 65nm **nMOS** variable node

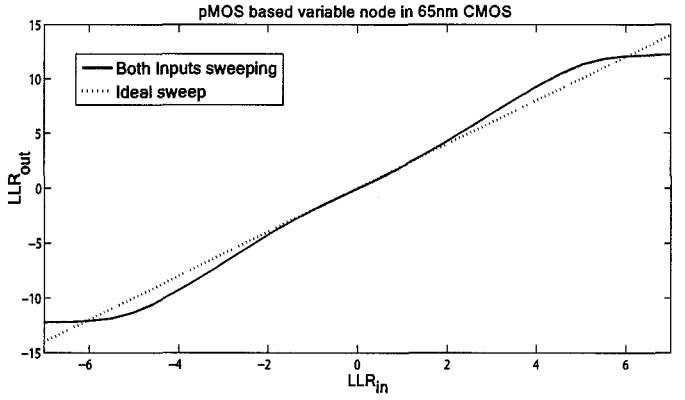

Figure 5.12: DC sweep of **both** terminals in the 65nm **pMOS** variable node

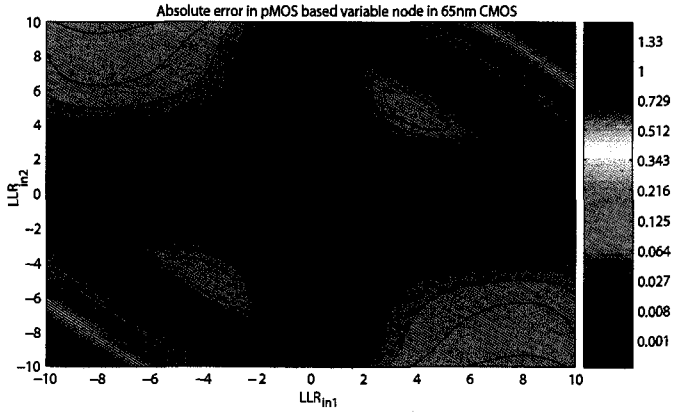

Figure 5.13: Absolute error in the **pMOS** variable node in 65nm CMOS

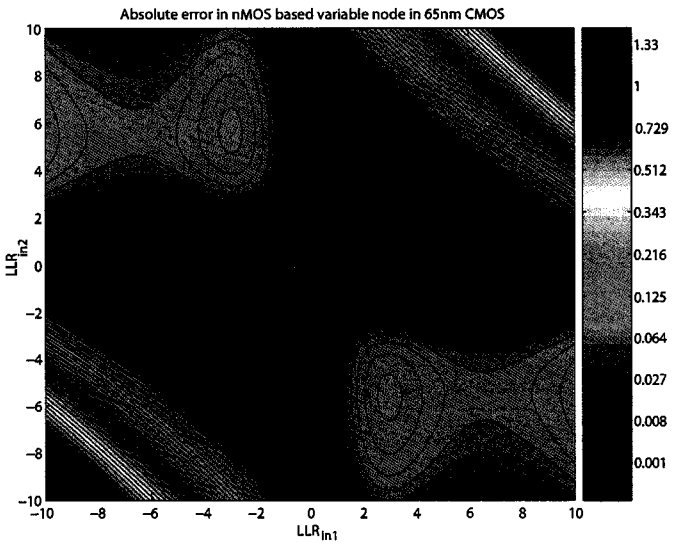

Figure 5.14: Absolute error in the **nMOS** variable node in 65nm CMOS

disadvantage of the new proposed topology is its larger size. The symmetry of the two inputs can be very important for correct functioning of the decoder. The inputs of the variable node are connected to different check nodes and a non-symmetric variable node might prefer one check node to another and this preference will be hard coded in the decoder. The processes would not be random anymore and is likely to degrade the performance of the decoder. The idea behind the symmetric node is very simple. The non-symmetry in the node is due to the non-symmetry between the two inputs of Gilbert multiplier. In order to overcome this problem we have added an extra Gilbert multiplier to the node. The extra Gilbert multiplier is fed through the same differential pairs but this time in the reverse order. This leaves us with two sets of output currents, each pair representing one desired output. Each input of the output differential pair is supplied with the two currents, each from a different multiplier. Therefore, each input receives almost twice the current. The node still operates normally because it works based on the ratio of the two input currents. Fig. 5.15 shows the block diagram for the two multipliers and how they share the input and output signals.

The forced symmetry node has good accuracy and the two inputs act identical, which

Table 5.4: Energy consumption and delay values for different test settings in 65nm CMOS circuit

Test Setting	nMOS version		pMOS version	
65nm CMOS	Energy (pJ)	Speed (μs)	Energy (pJ)	Speed (μs)
In1: $Z \rightarrow H$, In2: Z	0.740	0.10	5.007	1.11
In1: $H \rightarrow Z$, In2: Z	0.664	0.09	0.936	0.21
In1: Z, In2: $Z \rightarrow H$	0.667	0.09	5.005	1.10
In1: Z, In2: $H \rightarrow Z$	0.659	0.09	0.883	0.21
In1, In2: $Z \rightarrow H$	1.512	0.20	17.360	3.51
In1, In2: $H \rightarrow Z$	0.663	0.09	0.909	0.21
In1: $Z \rightarrow H$, In2: $Z \rightarrow L$	0.510	0.07	1.792	0.41
In1: $H \rightarrow Z$, In2: $L \rightarrow Z$	0.359	0.05	0.847	0.21
In1: $Z \rightarrow L$, In2: $Z \rightarrow H$	0.510	0.07	1.792	0.41
In1: $L \rightarrow Z$, In2: $H \rightarrow Z$	0.358	0.05	0.847	0.21
Average of the Column	0.664	0.09	3.538	0.759

Table 5.5: Average energy and delay values for variable nodes

Test Setting	nMOS version		pMOS version	
Topology	Energy (pJ)	Speed (μs)	Energy (pJ)	Speed (μs)
90nm	0.256	0.020	0.386	0.048
65nm	1.785	0.090	3.538	0.759

is a useful property. A disadvantage of this design is the increase in the leakage currents. The higher leakage currents reduce the LLR range of the node. Therefore the designed variable node has a functional LLR range of -9.5 to 9.5. We optimized the transistor sizing of this node for the best performance in 65nm CMOS. The result of this optimization is summarized in Table. 5.6. We used pMOS transistor for constructing the Gilbert multiplier in the symmetric node.

Fig. 5.16 shows the error surface generated for this node. The symmetry of the results is clear in this graph. The energy and delay of this node was also calculated based on the same testing strategy we used for the other nodes. Table. 5.8 has the power simulation results for the proposed forced symmetry node. A comparison between this node and the rest of the implemented nodes in the 65nm CMOS makes this node

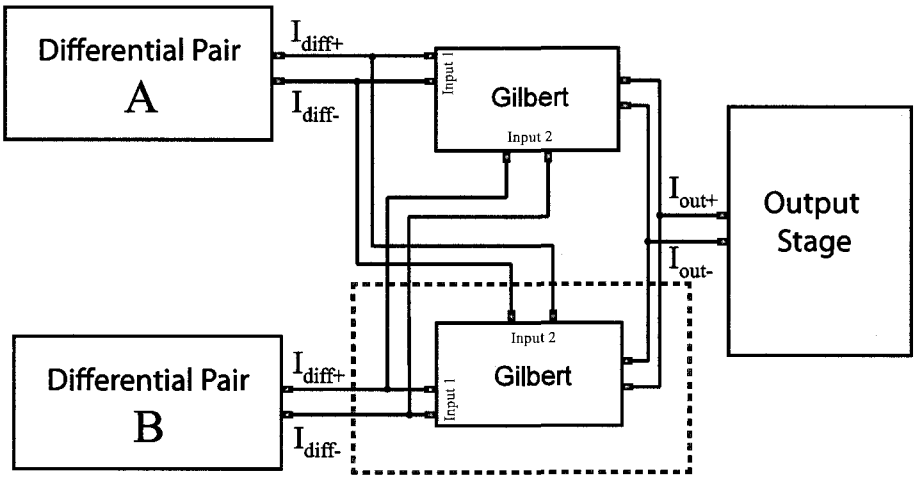

Figure 5.15: Forced symmetry variable node

a serious candidate for a decoder. The main drawback from using this node is its larger area consumption. The problem can be reduced using a smart layout. The next table shows the size of all the implemented nodes. In drawing the layout for these nodes, we used extra space to achieve less capacitance and resistance. Hence, they layouts are not very compact. Nevertheless Table. 5.7 shows the comparison between the implemented 65nm CMOS nodes in terms of their physical layout.

5.4 Testing strategy

In the previous section, we presented our circuits for 90nm and 65nm CMOS technologies. In this section, we briefly present the ASIC die and our test strategy. The implemented 90nm ASIC contains four different implementations of the variable nodes. In addition to that there is a single nMOS transistor, a single pMOS transistor and a triple WELL nMOS transistor for characterization. The die microphotograph of the 90nm CMOS chip is shown in Fig. 5.17. For accurate testing of the implemented nodes, we did not use the I/O ring. The provided I/O pads have large leakage currents. Because in our circuits we are dealing with small currents these extra leakage currents could affect our measurement results. In addition, the implemented nodes

Table 5.6: Transistor sizing for forced symmetry variable node

Gilbert Multiplier Transistors	
	$pMOS(\mu m)$
Width	0.215
Length	2.100
Input Differential Pair Transistors	
Width	0.610
Length	2.050
Output Stage pMOS Transistors	
Width	0.640
Length	8.200
Output Stage nMOS Transistors	
Width	1.400
Length	10.00
Differential Pair Current Mirrors	
Width	1.160
Length	1.100
Diode-connected Transistors	
Width	0.500
Length	10.000

were designed for driving small capacitances on the order of tens of femto Farads. This is because the goal of these nodes is to drive the input of the next stage, which has a small capacitance and resistance. Therefore, normal probes are not suitable, as they would heavily load the output. The solution was is to use pads on the die. The recommended pitch for easy testing using pads is $150\mu m$ and a width of $65\mu m$. The pads we use have a pitch of $165\mu m$ and a width of $50\mu m$. The 65nm CMOS chip is still in the fabrication process, hence the die photo is not available. The 65nm CMOS test chip contains the four different variable node circuits we studied earlier. We used the same pad method for reading the signals off the chip. Testing of the chip will be carried out using a set of fabricated probes. These probes were fabricated specifically for testing this chip.

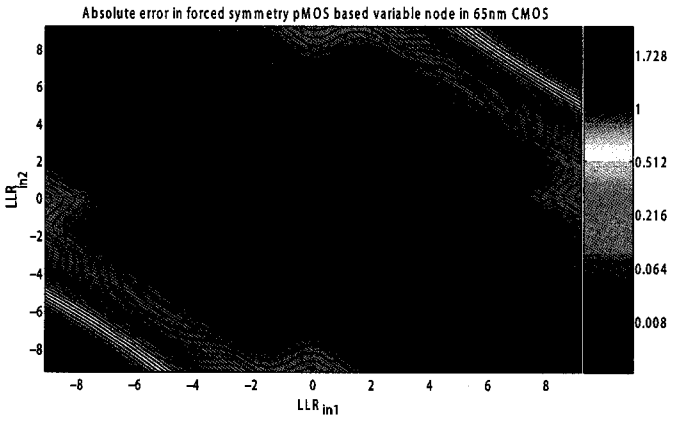

Figure 5.16: Absolute error in the forced symmetry **pMOS** variable node in 65nm CMOS

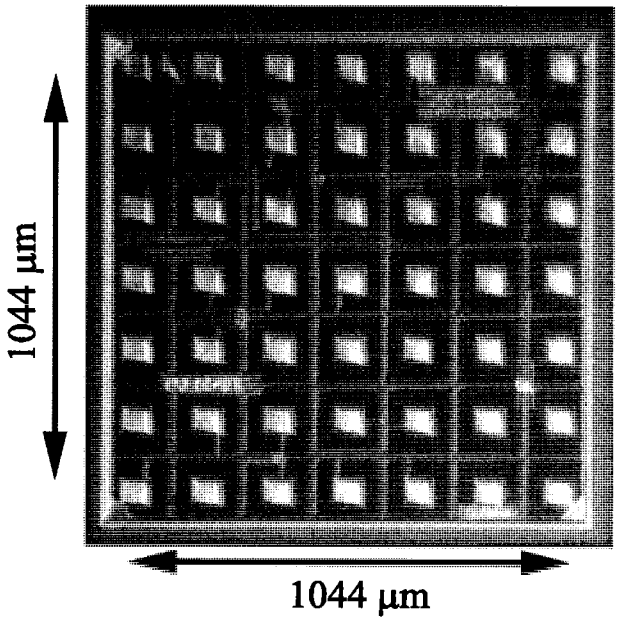

Figure 5.17: Die photo of the 90nm CMOS test chip

Table 5.7: Implemented layout size of the variable nodes in 65nm CMOS

Dimensions	nMOS node	pMOS node	Symmetric pMOS
Width (μm)	25	23	22
Length(μm)	28	42	47
Area(um^2)	700	966	1034

Table 5.8: Energy consumption and delay values for different test settings in 65nm CMOS symmetric node circuit

Test Setting	pMOS version	
65nm CMOS	Energy (pJ)	Speed (μs)
In1: $Z \rightarrow H$, In2: Z	0.947	0.150
In1: $H \rightarrow Z$, In2: Z	1.345	0.210
In1: Z, In2: $Z \rightarrow H$	0.947	0.150
In1: Z, In2: $H \rightarrow Z$	1.345	0.210
In1, In2: $Z \rightarrow H$	3.922	0.630
In1, In2: $H \rightarrow Z$	1.981	0.310
In1: $Z \rightarrow H$, In2: $Z \rightarrow L$	0.006	0.00
In1: $H \rightarrow Z$, In2: $L \rightarrow Z$	0.005	0.01
In1: $Z \rightarrow L$, In2: $Z \rightarrow H$	0.006	0.00
In1: $L \rightarrow Z$, In2: $H \rightarrow Z$	0.005	0.00
Average of the Column	1.051	0.17

5.5 Conclusion and summary

In this chapter we have presented the simulation results for eight different implemented variable nodes. The energy consumption, accuracy, speed and size of these implemented nodes were compared using post-layout simulation results. Based on the simulation results, the performance of the nodes is promising. Also we note that using large transistor sizes in order to overcome the mismatch problem has a big penalty in terms of speed and power. Therefore there is a tradeoff between the accuracy and power/speed of the nodes that depends on the design constraints. Now the questions that still remain is how accurate the nodes have to be for the decoder to perform without major performance loss and how different imperfections can affect

the performance of the overall decoder. This is the question we will try to answer in the next chapter.

Chapter 6

System Level Decoder Analysis

In the past five chapters, we have presented many different variable node circuits. Most of these nodes behave close to the ideal variable node, yet they carry different imperfections due to the nature of the analog circuits. One of the consequences of these limitations is the upper bound on the maximum possible LLR values in the circuit. In addition to this limitation, the output of the circuit has an error associated with it and this error is not constant throughout the operation and varies for every input. The last problem is the non-symmetry between the two inputs in the circuit. In an ideal node, both inputs are the same but in the previous chapter, we saw that this is not the case for our analog nodes. How each of these imperfections can affect the performance of the decoder is not obvious. The most accurate way of addressing this question is to simulate the decoder at the circuit level. Unfortunately, due to the high computational complexity, such simulations would take months or even years before we could obtain meaningful results. The next best way is to abstract the actual behaviour of the nodes and use it in a system-level Monte-carlo simulation. These simulations are performed in C++/ MATLAB within a reasonable amount of time. In addition to the Monte-Carlo type simulations, density evolution is a great tool for analyzing the statistical behaviour of a decoder under different situations such as LLR clipping. In this chapter, we will first target the effect of LLR clipping in the circuit and then we present the performance of the decoder using the nodes we designed in the previous chapter.

6.1 The Effect of Clipping on Sum-Product Algorithm

By definition, LLR messages in the sum-product algorithm are the natural logarithm of the ratio of the probabilities of each bit. Therefore, these messages are unbounded numbers that can take any real value. The hardware implementation of the sum-product algorithm forces an upper bound on the magnitude of the LLR values. This new constraint can affect the performance of the algorithm. In order to study how this limitation changes our result, we have included a clipping procedure into our simulation programs. This additional clipping function only affects the variable node results.

Clipping does not occur in check nodes. Equation (2.19) shows the function of the check node. The input LLRs are fed to a hyperbolic tangent function, which is always bounded by one, as follows:

$$\left|\prod_{i=2}^{4}\tanh(\frac{\mathrm{LLR}[y_i]}{2})\right| < min\left|\tanh(\frac{\mathrm{LLR}[y_i]}{2})\right|. \tag{6.1}$$

The magnitude of the inverse hyperbolic tangent function increases with magnitude of the argument. Hence we have:

$$\left|\mathrm{atanh}\left(\prod_{i=2}^{4}\tanh(\frac{\mathrm{LLR}[y_i]}{2})\right)\right| < min\left|\mathrm{LLR}[y_i]\right|. \tag{6.2}$$

Therefore, the magnitude of the output LLR in a check node is never clipped. Hence we only have to worry about the variable node. The variable node clipping function can be implemented in many different ways. The structure of the implemented variable node adds two inputs at a time. Therefore, the clipping might happen after every sum of two LLRs. In our simulations, we implemented two different types of clipping. The first type clips the output every time two LLR values are summed.

This clipping function is consistent with the actual implemented circuits. The second function clips the output at the end after all the LLRs are added. The second implementation shows how much better the decoder would perform using a multi-input variable node design. The implementation of the first clipping function is not a straightforward task. The order, which the channel input and check node outputs are added together, is important. In our implementation of the clipping function we first add the input from the channel to the first check node output then we add the result to the next check node output until the last connected check node is added. Our simulations showed that this implementation has a better performance compared to the other possible implementations. This is because when a bit is wrong and the check nodes are able to correct the bit. The check nodes would have a different LLR sign than the channel input. In addition to this, most of the check nodes would have the same LLR sign as they are agreeing with each other and opposing to the channel value. Therefore, the best method would be to add the channel values to the check node in the first round and adding the result to the next check node.

The following example better clarifies the situation. Assume we are clipping the outputs at a maximum LLR value of ten, the input LLR from the channel is $+7$, and the three connected check nodes have LLRs of $-8, -7$ and $+4$ at their output. Using our implemented routine we would first add the channel input to the first LLR, $+7-8=-1$. Then the next two check node values are added. Therefore the next two summations are, $-1-7=-8$ and $-8+4=-4$. Hence, we end up with a negative LLR value. Now if we first add the check node outputs together, the result of the first addition is $-8-7=-15$. This number is clipped to -10 and then is added to $+4$ which makes it -6. In the last two steps, we add the channel input to the result and the output of the variable node becomes $+1$. The result has a positive LLR and is wrong.

In order to study the effect of clipping on large codes we used density evolution. DE is a useful tool for studying the statistical behaviour of the decoder under LLR limitations. The original DE tool uses a maximum LLR range of -25 to 25. Simulations have proved that extending the LLR range beyond 25 does not improve the performance. In this section, DE is modified to predict the performance of the decoder

under limited LLR constraints. We used both types of proposed clipping functions for the DE test. The results of the first type are referred to as "Clipping at each addition" and the title "Clipping at the end" points to the second type of clipping. The modified DE was run on eight different code ensembles. For each code ensemble, we clipped the internal LLRs at three different values. These values were chosen according to a leakage current to normalizing current ratio (β) of $0.001, 0.005$ and 0.01. These ratios correspond to maximum LLR value of $6.90, 5.29$ and 4.6. When the internal LLRs are clipped, we cannot allow large input LLRs. This is because, in such situation the check nodes might never be able to correct a data bit with large input LLR. Therefore, we also clipped the input LLRs from the channel. The upper bound on the LLRs from the channel was optimized for best performance in each case. The simulations show that the optimized bound is very similar in both (at each/at the end) clipping situations. We can also see that the optimized channel clipping value moves closer to the internal clipping bound as we further clip the internal LLRs. For example in the $(3, 6)$-regular code the best value for input clipping is 7 when the internal LLRs are clipped at 6.9 and 5.3 for an internal clipping of 5.29. In order to find the threshold we let the simulation to go trough 2000 iterations and the target bit error rate was 10^{-8}. The first set of simulations show the clipping effect when the clipping is enforced after every summation of two numbers. The result of this simulation is shown in Table. 6.1.

Table 6.1: DE analysis of clipping at each addition in LDPC codes

Code ensemble /Threshold	IBound: 6.9		IBound: 5.29		IBound: 4.6	
	CBound	Threshold	CBound	Threshold	CBound	Threshold
$(5, 10) - 2.0077$	7.00	2.0077	5.30	2.028	4.7	2.090
$(4, 08) - 1.5384$	6.95	1.5384	6.00	1.557	4.6	1.6149
$(3, 04) - 0.9568$	8.00	0.9568	5.29	0.9598	4.65	0.975
$(4, 10) - 1.7288$	7.00	1.7295	5.29	1.769	—	—
$(3, 05) - 0.8886$	7.80	0.8887	5.35	0.9035	—	—
$(3, 06) - 1.1015$	7.00	1.1021	5.30	1.135	—	—
$(3, 09) - 1.7474$	7.00	1.7520	—	—	—	—
$(3, 15) - 2.5857$	—	—	—	—	—	—

The term CBound shows the maximum LLR limit we enforced on the inputs from the channel and IBound is upper bound for internal clipping. The "—" sign shows inability of convergence under given conditions. This table clearly shows that clipping has a different effect on different code ensembles.

If we take a closer look, we can find a pattern in the table. The codes with the most variable node connections are more tolerant to clipping. If the two codes have same number of variable node connections, then the number of check node connections is the tiebreaker. Fewer check node connections result in better tolerance to clipping. A large number of check node connections can be problematic. For example the (4,10)-regular code is more affected by clipping than the (3,4)-regular code even though it has more variable node connections. The explanation is simple. With internal clipping enforced, the output of the check nodes is very small. This output is a function of the number of connections to the check node and becomes smaller as the number of connections grows. For example if the decoder is clipping the internal LLRs at 4.6, then the maximum output of a degree six check node is 2.81 and is 2.30 for a degree ten node. Therefore, high number of check node connections reduces the output of the check nodes to small values. Small check node outputs fail to correct large input LLRs from the channel unless there is enough number of check nodes connected to the variable node. The number of variable node and check node connections determines the rate of the code. Therefore, the lower rate codes tend to have a better immunity against clipping. Another issue with more connections is the increase in circuit complexity and power consumption. We performed the same set of simulations with a second type of clipping function. This time we clipped the LLRs after all inputs to the variable node are added together. Table. 6.2 shows the result of these simulations.

The date in Tables 6.1 and 6.2 shows that clipping the LLRs at the end of the variable node summations allows for a wider range of input LLR values. The performance also seems to be improved in some cases. Therefore without doubt clipping at the end is a better option in terms of performance but this small improvement comes with a big price. Designing a multi input multiplier requires large voltage headroom. Therefore, the complexity of the design in a nano-scale technology with small voltage drives

Table 6.2: DE analysis of clipping at the end of variable node additions

Code ensemble /Threshold	IBound: 6.9		IBound: 5.29		IBound: 4.6	
	CBound	Threshold	CBound	Threshold	CBound	Threshold
$(5,10)-2.0077$	8.00	2.0077	5.90	2.015	6.5	2.055
$(4,08)-1.5384$	8.00	1.5384	8.50	1.549	4.6	1.6149
$(3,04)-0.9568$	10.0	0.9568	5.50	0.9595	4.8	0.975
$(4,10)-1.7288$	8.00	1.7289	5.29	1.769	—	—
$(3,05)-0.8886$	8.00	0.8886	5.65	0.9005	—	—
$(3,06)-1.1015$	8.00	1.1019	5.30	1.135	—	—
$(3,09)-1.7474$	7.80	1.7510	—	—	—	—
$(3,15)-2.5857$	—	—	—	—	—	—

ranges from hard to impossible and it is not recommended, especially considering the negligible effect, it has on the performance. The last set of DE analysis is shown in Table. 6.3. In these simulations, we tried to lower the LLR limit for both internal LLRs and the data from the channel until it affected the performance so that the code threshold was $0.001dB$ higher than its original threshold value.

Table 6.3: DE analysis of clipping

Code ensemble	Minimum LLR limit with less than $0.001dB$ performance loss
$(3,04)$	5.7
$(5,10)$	6.4
$(4,08)$	6.5
$(4,10)$	5.9
$(3,05)$	6.3
$(3,06)$	6.8
$(3,09)$	7.6
$(3,15)$	8.4

To further verify the results in a limited size code, Monte-Carlo simulations were carried out under different clipping scenarios. A moderate size $n = 816, k = 408$ code was used as our target LDPC code. We used regular code ensembles (3,6) and (5,10). The codes were obtained from Mackay's website [66]. The Monte-Carlo simulation waits for a minimum of 50 frame errors before calculating the bit error rate.

We increased this error limit to 70 or 90 when it was necessary. The decoder goes through 50 iterations for decoding. This number was chosen so that further iterations would not significantly change the error rate results. We had to increase the number to 100 for some few cases where the LLR clippings was severe. In the first set of simulations, we clipped the internal LLRs. The input LLRs from the channel were also clipped according to Table. 6.1. The results confirm the outcome from the DE analysis. All the simulations are based on the clipping after each summation method. The simulation results for a (3,6)-regular code with clipping are presented in Fig. 6.1. The term UBound is the LLR limit were we cut the internal LLRs. The graph shows the simulation for three different clipping limits as well as the normal decoder behaviour. The graph shows the bit error rate curves. The loss for clipping the LLRs at 6.9 is very small and acceptable. But as we push the limit and enforce tighter bounds on the LLRs, the performance degrades significantly and at an LLR limit of 4.6, the decoder hits an early error floor and loses over 3dB in performance. The situation is a little better for the (5,10)-regular code, as expected. The LLR limitation has a little effect on the curve but once again clipping at an LLR value of 4.6 causes the curve to hit an early error floor. The simulation results for the (5,10)-regular code are presented in Fig. 6.3. In chapter four we found out that the clipping effect on LLR values due to leakage currents is somewhat gradual. Therefore for the next set of simulations we used the equation from (4.19) and substituted the three different values of $\beta = 0.001, 0.005$ and 0.01 into the equation. These values of β map to the same LLR bounds we had earlier. The Fig. 6.2 shows the simulations results for the (3,6)-regular code. The leakage-included curves show worse performance compared to simple clipping of the LLRs. Nevertheless, there is common behaviour in both simulations. The effect of clipping is more prominent as we move towards higher input energies and even the curve for the smallest β tends to move away from the ideal curve and diverge from the waterfall behaviour.

The experience in the (5,10)-regular code is rather interesting. The simple clipping of LLRs has a very small impact on the performance of the code just as was predicted by the density evolution but implementing the full leakage effect form (4.19) strongly

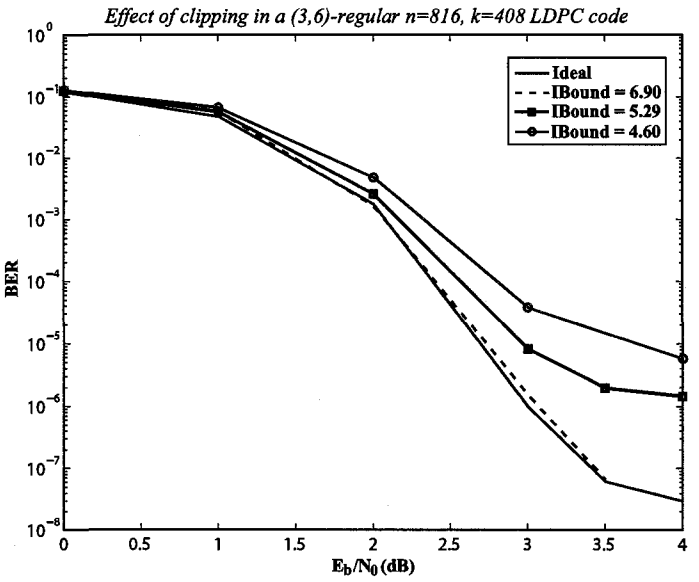

Figure 6.1: The effect of clipping the internal LLRs at different magnitudes for a (3,6)-regular code

lowers the performance and the (5,10)-regular code behaves no better than the (3,6)-regular code. A graph including the leakage effects is plotted in Fig. 6.4 and can be used to compare the results with Fig. 6.3 which only includes a simple clipping. An explanation for this behaviour is that since the (5,10)-regular code requires a greater number of summation operations, the error in the variable nodes accumulates to a bigger value compared to a (3,6)-regular code. Therefore you see a larger degradation compared to simple clipping in this code.

6.2 Decoder Performance of the Implemented Nodes

In this section, we investigate the performance of the variable nodes in a system-level decoder Monte-Carlo simulation setup. These simulations do not exactly predict the behaviour of an analog decoder but they give a good estimate of what we to expect and how much imperfection the decoder can tolerate. In the previous chapter we compared these different designs in terms of energy consumption, speed, accuracy,

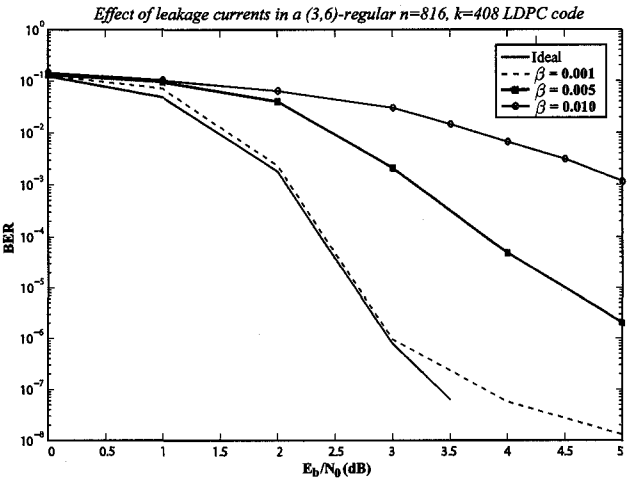

Figure 6.2: The effect of leakage currents in a (3,6)-regular code

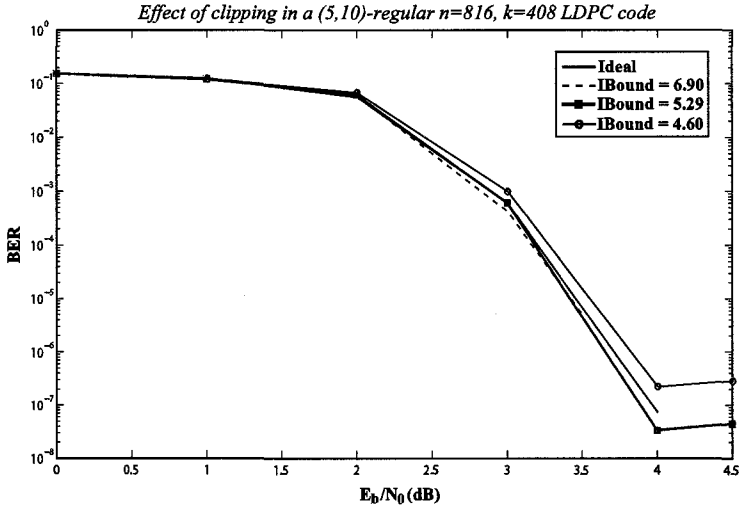

Figure 6.3: The effect of clipping currents in a (5,10)-regular code

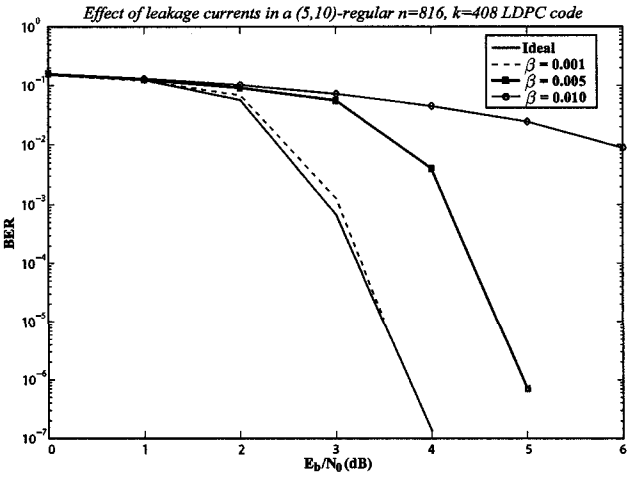

Figure 6.4: The effect of leakage currents in a (5,10)-regular code

reliability and size. Using Cadence circuit simulation we built a huge matrix of all possible combination of input and outputs with a $1mV$ resolution that was represented in terms of LLRs. In this section this matrix is fed into a decoder to generate the bit error rate curves. The simulation assumes AWGN channel using antipodal signaling. These simulations go through a maximum of 100 iterations before giving up and search for a minimum of 50 to 90 frame errors before calculating the error statistics. The code we used here is the same (3,6)-regular n=816, k=408 code we used previously. The five curves in Fig. 6.5 represent two of the implemented nodes in 65nm CMOS and one implementation from the 90nm CMOS chip. The overall performance of all these nodes is very close to the ideal behaviour. The matrix for the red curve is plotted in Fig. 5.14, the black line uses the error matrix from Fig. 5.13, and finally the brown curve uses the matrix from Fig. 5.8. Note that these pictures show the absolute error and ignore the sign, but we considered the sign when we used it in the decoder simulations. As you can notice, the orange curve performs very close to ideal floating point sum-product algorithm although the other two circuits had a more accurate response. This result is not very unlikely. The sum-product algorithm is not an optimal algorithm in graphs with cycles. Similar behaviour has been observed in analog decoders [67].

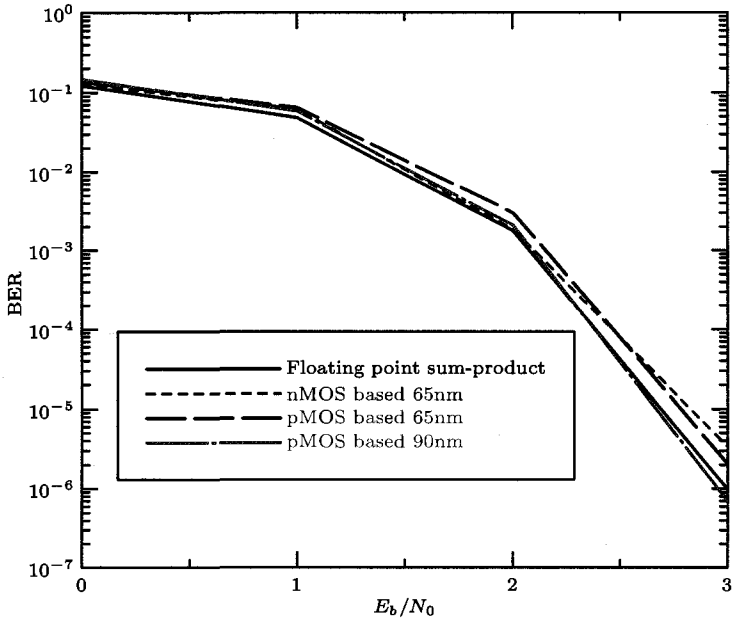

Figure 6.5: The performance of the different implemented nodes under system-level Monte-Carlo decoder simulation

The system-level simulation of the implemented nodes shows an acceptable performance. Node implementations with different imperfections were all very close to the ideal variable node implementation with less than 0.1 dB loss in performance up to BER of 10^{-6}. Therefore, the real tradeoff is between the mismatch effect and speed or power performance of the nodes.

6.3 Conclusion

In this chapter we studied the effect of LLR clipping on different code ensembles. The simulations show that the minimum acceptable LLR range without noticeable performance loss code-dependent and lower rate codes are less affected by clipping. We also used the gradual clipping of LLRs from equation (4.19). The simulations show that as soon as we consider the gradual clipping of LLRs as it happens in the circuit the code dependance becomes less important. Therefore, all codes are affected by the LLR limitation is a similar way. We also performed decoder simulations using the implemented nodes and noticed that the decoder can tolerate the inaccuracy in the variable nodes. These simulation results provide us with sufficient confidence that analog decoders in sub-100nm CMOS are feasible as long as we account for mismatch in our design and the choose the right normalizing current that results in an acceptable value of β.

Chapter 7

Conclusions and Future Directions

In this chapter we present the thesis contributions and conclusions. Section 7.2 proposes possible future directions for this work.

7.1 Thesis Contributions

In this thesis we studied the possibility of building decoders in nano-scale CMOS technologies. The system-level C simulations of the decoder was performed based on BSIM4 post-layout simulation results from Cadence. The simulations were performed down to bit error rates of 10^{-6} for a length 816, (3,6)-regular LDPC code and the results showed less than 0.1dB deviation from the floating point BER curve. These results have shown that the analog decoder are still able to meet system-level performance specifications in nano-scale CMOS technologies.

For the first time we have presented a detailed study on the effect of leakage currents in analog decoders. These leakage currents tend to increase as we move towards smaller processes. The leakage currents limit the maximum magnitude of LLRs in the circuit. The results in Fig. 6.2 indicate that high leakage currents can seriously hurt the performance of the decoder. Based on (4.20) we can predict the normalizing current required to assure a desired LLR bound. Therefore we can control the LLR bound for satisfactory decoder behaviour.

The effect of mismatch on nano-scale decoders was presented for the first time. We showed that mismatch can have a strong impact on the functionality of decoders in nano-scale technologies. The mismatch theory revealed that using minimum sized transistors in nano-scale analog decoders critically affects the decoder performance in large codes. There exists a lower limit on the transistor area where mismatch effects are negligible. In addition to the restriction on the area, the minimum transistor length also has to be bigger than a value that we can determine. Following these guidelines can mitigate the effect of mismatch on the decoder.

We also implemented several different decoder computational nodes and proposed a symmetric design where both inputs in the variable node behave identically. These fabricated nodes are currently under testing. The system-level simulations of the decoder revealed that mismatch has the strongest effect on the performance of the decoder. Also we can see that not only do small circuit non-idealities not severely hurt the error correcting capability of the decoder, but in some cases they can improve the performance of algorithm.

7.2 Future Directions

In this thesis we have shown that implementation of functional analog LDPC decoders in sub-100nm CMOS processes is possible. Mismatch analysis showed that there is a lower bound on the decoder area. Therefore, regardless of the CMOS technology, the size of the decoder can not shrink and the power consumption and speed will not improve significantly. Digital decoders on the other hand can benefit from scaling to achieve lower power consumption and higher speeds. In addition the decoder size scales with the technology. Whether or not the analog or digital decoders will dominate is still hard to answer. A mixed signal approach might be the solution. A mixed decoder consisting of an inaccurate analog decoder followed by an accurate digital decoder can take advantage of both decoders and still consume low power.

The dynamics of analog decoders indicates that analog decoders converge faster at lower signal to noise ratios and therefore consume less power [68] in the low SNR region compared to high SNR region. The activity and power consumption of a digital decoder on the other hand is lower in the high SNR region [32]. The analog decoder can primary correct some of the received bits and pass the results to the digital decoder to clean up the work. Analog implementation of LDPC decoders have proved that the decoding algorithms can tolerate a great deal of imperfections and they are very insensitive to timing and synchronization. These allow for a very relaxed digital implementation of these decoders. Therefore, more investigation on the algorithm can help build more efficient low power decoders.

Appendix A

Layout of the implemented nodes

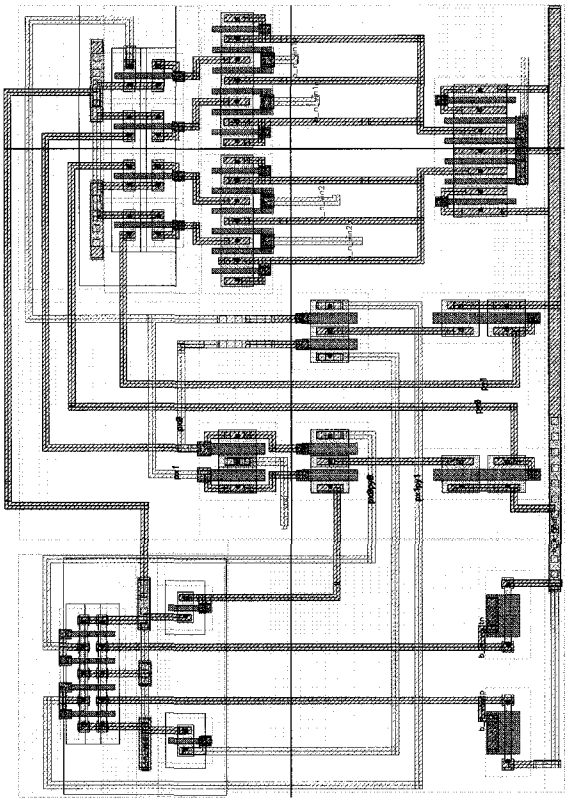

Figure A.1: Layout of the nMOS based circuit in 90nm

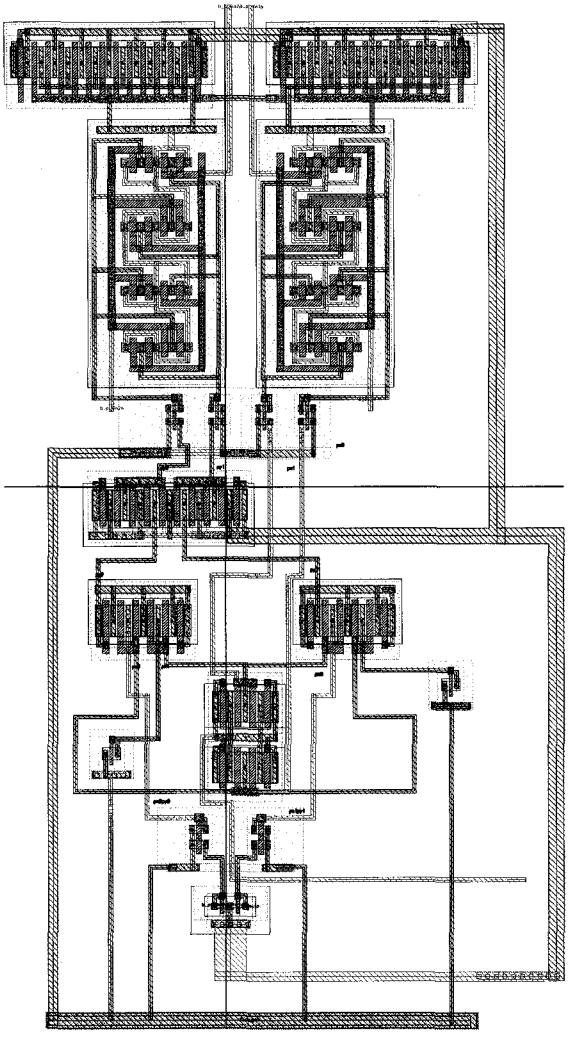

Figure A.2: Layout of the pMOS based circuit in 90nm

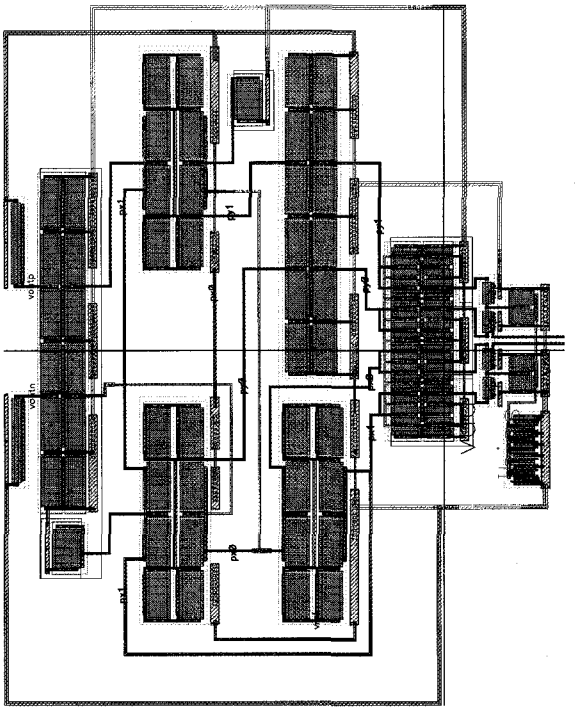

Figure A.3: Layout of the nMOS based circuit in 65nm

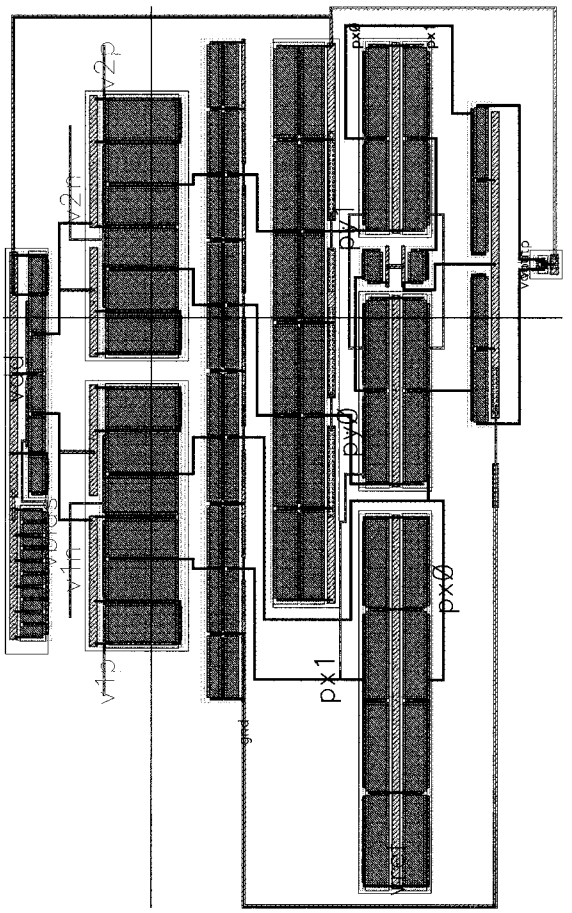

Figure A.4: Layout of the pMOS based circuit in 65nm

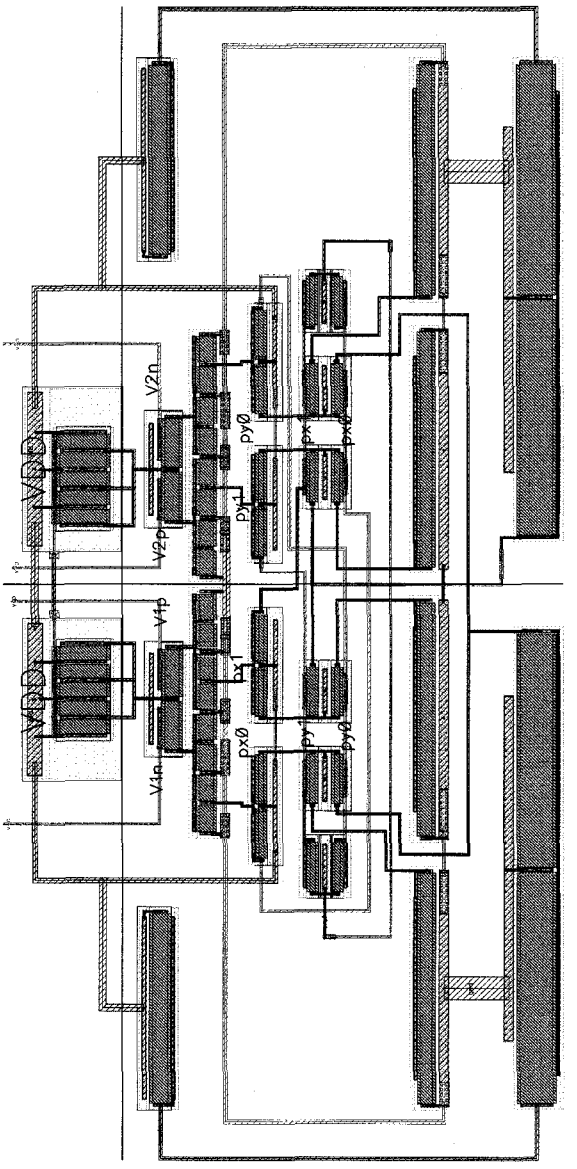

Figure A.5: Layout of the symmetric circuit in 65nm

Bibliography

[1] C. Schlegel, L. Perez, *Trellis and Turbo Coding.* IEEE/Wiley, 2004.

[2] T. Osaka, M. Datta, *Energy Storage Systems for Electronics.* Springer, 2000.

[3] Z. Wang, S. Mai, C. Zhang, "Power Issues on Circuit Design for Cochlear Implants," in *IEEE International Symposium on Electronic Design, Test and Applications*, January 2008, pp. 163–166.

[4] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, "Wireless sensor networks: a survey," in *Computer Networks: The International Journal of Computer and Telecommunications Networking*, vol. 38, no. 4, December 2001, pp. 393–422.

[5] P. Gerrish, E. Herrmann, L. Tyler, K. Walsh, "Challenges and constraints in designing implantable medical ICs," in *IEEE Transactions on Device and Materials Reliability*, vol. 5, no. 3, September 2005, pp. 435–444.

[6] H. Loeliger, F. Lustenberger, M. Helfenstein, F. Tarky, "Probability Propagation and Decoding in Analog VLSI," in *IEEE Transactions on Information Theory*, vol. 46, no. 2, February 2001, pp. 837–843.

[7] R. Sarpeshkar, "Ultra low power electronics for medicine," in *International Workshop on Wearable and Implantable Body Sensor Networks*, April 2006.

[8] P. D. T.M.Massengill, D.M.Wilson, "Empirical Comparison of Analog and Digital Auditory Preprocessing for Automatic Speech Recognition," in *IEEE International Symposium on Circuits and Systems*, vol. 5, May 2002, pp. 77 – 80.

[9] A. Geus, *Key Characteristics for Success.* Synopsys, 2006. [Online]. Available: http://www.synopsys.com/news/pubs/compiler/2006/artlead_3ps-may06.html

[10] S.Y. Chung, G.D. Forney Jr., T.J. Richardson andf R. Urbanke, "On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit," in *IEEE Communications Letters*, vol. 5, no. 2, February 2001, pp. 58–60.

[11] M. Moerz, T. Gabara, R. Yan and J. Hagenauer, "An analog 0.25μm BiCMOS tailbiting MAP decoder," in *IEEE International Solid-State Circuits Conference: Digest of Technical Papers*, February 2000, pp. 356–357.

[12] C. Winstead, J. Dai, S. Yu, C. Myers, R.R. Harrison and C. Schlegel, "CMOS analog MAP decoder for (8,4) Hamming code," in *IEEE Journal of Solid-State Circuits*, vol. 39, no. 1, January 2004, pp. 122–131.

[13] M. Mansour, N. Shanbhag, "A 640-Mb/s 2048-Bit Programmable LDPC Decoder Chip," in *IEEE Journal of Solid-State Circuits*, vol. 41, no. 3, March 2006, pp. 684–689.

[14] D. Vogrig, A. Gerosa, A. Neviani, A.Gi. Amat, G. Montorsi and S. Benedetto, "A 0.35-μm CMOS analog turbo decoder for the 40-bit rate 1/3 UMTS channel code," in *IEEE Journal of Solid-State Circuits*, vol. 40, no. 3, March 2005, pp. 753–762.

[15] S. Yang, D. Li and Y. Qiu, "The decoder of trellis code implemented by CMOS analog circuits," in *IEEE 7th International Conference on ASIC*, October 2007, pp. 898–901.

[16] V.C. Gaudet and P.G. Gulak, "A 13.3-Mb/s 0.35-μm CMOS analog turbo decoder IC with a configurable interleaver," in *IEEE Journal of Solid-State Circuits*, vol. 38, no. 11, November 2003, pp. 2010–2015.

[17] M. Arzel, C. Lahuec, F. Seguin, D. Gnaedigtt and M. Jézéquel, "Semi-iterative analog turbo decoding," in *IEEE Transactions on Circuits and Systems - I:Regular Papers*, vol. 54, no. 6, June 2007, pp. 1305–1316.

[18] F. Lustenberger, M. Helfenstein, G. S. Moschytz, H. A. Loeliger, F. Tarkoy, "All analog decoder for (18,9,5) tail-biting trellis code," in *IEEE European Solid-State Circuits Conference*, September 1999, pp. 362–365.

[19] S. Hemati, A.H. Banihashemi and C. Plett, " A 0.18μm CMOS analog min-sum iterative decoder for a (32,8) low-density parity-check (LDPC) code," in *IEEE Journal of Solid-State Circuits*, vol. 41, no. 11, November 2006, pp. 2531–2540.

[20] A. Schaefer, M. Moerz, J. Hagenauer, A. Sridharan, D. J. Costello, "Analog rotating ring decoder for an LDPC convolutional code," in *IEEE Information Theory Workshop*, April 2003, pp. 226–229.

[21] C. E. Shannon, "A mathematical theory of communication," in *Bell System Technical Journal*, vol. 27, July and Oct. 1948, pp. 379–423, 623–656.

[22] R. G.Gallager, *Low Density Parity Check Codes.* MIT Press,Cambridge, 1963.

[23] D. MacKay, R. M. Neal, "Near Shannon-limit performance of lowdensity parity-check codes," in *IEE Electronics Letters*, vol. 32, no. 18, August 1996, pp. 1645–1646.

[24] F.R. Kschischang, B.J. Frey and H.A. Loeliger, "Factor graphs and the sum-product algorithm," in *IEEE Transactions on Information Theory*, vol. 47, no. 2, February 2001, pp. 498–519.

[25] R.M. Tanner, "A recursive approach to low complexity codes," in *IEEE Transactions on Information Theory*, vol. 27, no. 5, September 1981, pp. 533–547.

[26] Amin Shokrollahi, "LDPC Codes: An Introduction," 2003. [Online]. Available: http://www.ics.uci.edu/~welling/teaching/ICS279/LPCD.pdf

[27] T. Richardson and R. Urbanke, "The capacity of low-density parity check codes under message-passing decoding," in *IEEE Transactions on Information Theory*, vol. 47, no. 2, February 2001, pp. 599–618.

[28] T. Richardson, A. Shokrollahi and R. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," in *IEEE Transactions on Information Theory*, vol. 47, no. 2, February 2001, pp. 619–637.

[29] S. Sivakumar, "VLSI implementation of encoder and decoder for low density parity check codes," in *Master of Science Thesis, Texas A & M University*, December 2001.

[30] A. Darabiha, A. Chan Carusone and F. R. Kschischang, "A 3.3Gbps bit-serial block-interlaced LDPC decoder in 0.13μm CMOS," in *IEEE Custom Integrated Circuits Conference*, September 2007, pp. 459–462.

[31] M.M. Mansour, N.R. Shanbhag, "A 640-Mb/s 2048-Bit Programmable LDPC Decoder Chip," in *IEEE Journal of Solid-State Circuits*, vol. 41, March 2006, pp. 684–698.

[32] V. Gaudet,C. Schlegel, R. Dodd, "LDPC Decoder Message Formatting Based on Activity Factor Minimization Using Differential Density Evolution," in *IEEE Information Theory Workshop*, September 2007, pp. 571–576.

[33] C. Winstead and C. Schlegel, "Density evolution analysis of device mismatch in analog decoders," in *IEEE International Symposium on Information Theory*, June 2004, p. 293.

[34] M. Yiu, V.C. Gaudet, C. Schlegel, C. Winstead, "Digital Built-in Self-Test of CMOS Analog Iterative Decoders," in *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 54, no. 8, August 2007, pp. 675–679.

[35] B. Gilbert, " A new wide-band amplifier technique," in *IEEE Journal of Solid-State Circuits*, vol. 3, no. 4, December 1968, pp. 353–365.

[36] ——, " A precise four-quadrant multiplier with subnanosecond response," in *IEEE Journal of Solid-State Circuits*, vol. 3, no. 4, December 1968, pp. 365–373.

[37] ——, " Translinear circuits: a proposed classification," in *IEE Electronics Letters*, vol. 11, no. 1, January 1975, pp. 14–16.

[38] J. Hagenauer, "Decoding of binary codes with analog networks," in *Information Theory Workshop, San Diego*, February 1998, pp. 13–14.

[39] J. Pérez Chamorro, C. Lahuec, F. Seguin, M. Jézéquel, "Design rules for sub-threshold MOS circuits," in *5th Analogue Decoding Workshop*, 2006.

[40] Royal Philips Electronics, "MOS model 9," 2005. [Online]. Available: http://www.semiconductors.philips.com/Philips_Models/mos_models/model9

[41] M.V. Dunga, W. Yang, X.Xi, J. He, W. Liu, Kanyu, M. Cao, X. Jin, J. Ou, M. Chan, A. Niknejad and C. Hu, *BSIM4.6.1 MOSFET Model Users Manual*, Department of EECS University of California, Berkeley, 2007.

[42] Y. Tsividis, *Operation and Modeling of The MOS Transistor, 2nd ed.* NY: Oxford University Press, 2003.

[43] K. Cao, W. Liu, X. Jin, V. Green, K. Krick, J. Vrotsos and T. Chenming Hu, "Modeling of pocket implanted MOSFETs for anomalous analog behavior," in *International IEDM Technical Digest on Electron Devices Meeting*, December 1999, pp. 171–174.

[44] T. Ytterdal, Y. Cheng and T.A. Fjeldly, *Device Modeling for Analog and RF CMOS Circuit Design.* John Wiley and Sons, 2003.

[45] Clarycon, *http://www.clarycon.com/shallowtrenchisa.html*, January 2008.

[46] K. Roy, S. Mukhopadhyay and H. Mahmoodi-Meimand, "Leakage current mechanisms and leakage reduction techniques in deep-submicrometer CMOS circuits," in *Proceedings of the IEEE*, vol. 91, no. 2, February 2003, pp. 305–327.

[47] M. Fadlallaha, G. Ghibaudo, J. Jomaaha, M. Zoaeterb, G. Guegan, "Static and low frequency noise characterization of surface- and buried-mode 0.1 μm P and N MOSFETs," in *Microelectronics Reliability*, vol. 42, no. 1, August 2001, pp. 42–46.

[48] J.A. Croon, W.M. Sansen, H.E. Maes, *Matching Properties of Deep Sub-Micron MOS Transistors.* Springer, 2005.

[49] S. Chih-Tang, R.N. Noyce and W. Shockley, "Carrier generation and recombination in P-N junctions and P-N junction characteristics," in *Proceedings of the IRE*, vol. 45, no. 9, September 1957, pp. 1228–1243.

[50] S. Mukhopadhyay, A. Raychowdhury and K. Roy, " Accurate estimation of total leakage in nanometer-scale bulk CMOS circuits based on device geometry and doping profile," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 3, March 2005, pp. 363–381.

[51] Y. Taur, T. H. Ning, *Fundamentals of Modern VLSI Devices.* Cambridge Univ. Press, 1998.

[52] Y. Taur, D. Buchanan, W. Chen, D. Frank, K. Ismail, S.-H. Lo, G. Sai-Halasz, R. Viswanathan, H.-J. Wann, S. Wind, H.-S. Wong, "CMOS scaling into the nanometer regime," in *Proceedings of the IEEE*, vol. 85, no. 4, April 1997, pp. 486–504.

[53] Y. C. Yeo, Q. Lu, W. C. Lee, T.-J. King, C. Hu, X. Wang, X. Guo, T. Ma, "Direct tunneling gate leakage current in transistors with ultrathin silicon nitride gate dielectric," in *IEEE Electron Device Letters*, vol. 21, no. 11, November 2000, pp. 540–542.

[54] R. Difrenza, P. Llinares and G. Ghibaudo, "The impact of short channel and quantum effects on the MOS transistor mismatch," in *Elsevier Solid-State Electronics*, vol. 47, no. 7, July 2003, pp. 1161–1165.

[55] M.J.M. Pelgrom, "Matching properties of MOS transistors," in *IEEE Journal of Solid-State Circuits*, vol. 24, no. 5, October 1989, pp. 1433–1440.

[56] J.P. de Gyvez and H.P. Tuinhout, "Threshold voltage mismatch and intra-die leakage current in digital CMOS circuits," in *IEEE Journal of Solid-State Circuits*, vol. 39, no. 1, January 2004, pp. 157–168.

[57] R.J. van de Plassche, J.H. Huijsing and W.M. Sansen, *Analog Circuit Design: High-Speed Analog-to-Digital Converters, Mixed Signal Design; PLLs and Synthesizers.* Springer, 2000.

[58] A. Pavasovic, A.G. Andreou and C.R. Westgate, "Characterization of subthreshold MOS mismatch in transistors for VLSI systems," in *Journal of VLSI Signal Processing*, vol. 8, no. 1, February 1994, pp. 75–85.

[59] M. Frey, H.A. Loeliger, F. Lustenberger, P. Merkli and P. Strebel, "Analog-decoder experiments with subthreshold CMOS soft-gates," in *IEEE International Symposium on Circuits and Systems*, May 2003, pp. 85–88.

[60] J. Dai, "Design methodology for analog VLSI implementations of error control decoders," Ph.D. dissertation, University of Utah.

[61] F. Lustenberg, "On the design of analog iterative VLSI decoders," in *PhD dissertation, Swiss Federal Institute of Technology*, 2000.

[62] K. Chew, J. Zhang,K. Shao, W. Boon Loh, S. Chu, "Impact of Deep N-well Implantation on Substrate Noise Coupling and RF Transistor Performance for Systems-on-a-Chip Integration," in *Proceeding of the 32nd European Solid-State Device Research Conference*, September 2002, pp. 251–254.

[63] C. Winstead, N. Nguyen, V.C. Gaudet, C. Schlegel, "Low-voltage CMOS circuits for analog iterative decoders," in *IEEE Transactions on Circuits and Systems I:Regular papers*, vol. 53, no. 4, April 2006, pp. 829–841.

[64] D. M. Binkley, B. J. Blalock, J. M. Rochelle, "Optimizing Drain Current, Inversion Level, and Channel Length in Analog CMOS Design," in *Analog Integrated Circuits and Signal Processing*, vol. 47, no. 2. Springer, March 2006, pp. 137–163.

[65] CMC, "Canadian Microelectronics Corporation," 2008. [Online]. Available: http://www.cmc.ca

[66] D. MacKay. (2007) Encyclopedia of sparse graph codes. [Online]. Available: http://www.inference.phy.cam.ac.uk/mackay/codes/data.html

[67] S. Hemati, A. H. Banihashemi, "Dynamics and Performance Analysis of Analog Iterative Decoding for Low-Density Parity-Check (LDPC) Codes," in *IEEE Transactions on Communications*, vol. 54, no. 1.

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