An Analog Decoder for Turbo-Structured Low-Density Parity-Check Codes

Ali Reza Rabbani Abolfazli

A thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Doctor of Philosophy (Electrical Enginnering) at

Concordia University

Montreal, Quebec, Canada

December 2012

CONCORDIA UNIVERSITY
SCHOOL OF GRADUATE STUDIES

This is to certify that the thesis prepared

By: **Ali Reza Rabbani Abolfazli**

Entitled: **An Analog Decoder for Turbo-Structured Low-Density Parity-Check Codes**

and submitted in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY (Electrical and Computer Engineering)

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Chair
Dr. A. Ben Hamza

_____ External Examiner
Dr. V. Gaudet

_____ External to Program
Dr. R. Wuthrich

_____ Examiner
Dr. M.R. Soleymani

_____ Examiner
Dr. C. Wang

_____ Thesis Co-Supervisor
Dr. G. Cowan

_____ Thesis Co-Supervisor
Dr. Y.R. Shayan

Approved by _____
Dr. J.X. Zhang, Graduate Program Director

December 12, 2012

_____
Dr. Robin Drew, Dean
Faculty of Engineering and Computer Science

ABSTRACT

An Analog Decoder for Turbo-Structured Low-Density Parity-Check Codes

Ali Reza Rabbani Abolfazli, Ph.D.

Concordia University, 2012

In this work, we consider a class of structured regular LDPC codes, called Turbo-Structured LDPC (TS-LDPC). TS-LDPC codes outperform random LDPC codes and have much lower error floor at high Signal-to-Noise Ratio (SNR). In this thesis, Min-Sum (MS) algorithms are adopted in the decoding of TS-LDPC codes due to their low complexity in the implementation. We show that the error performance of the MS-based TS-LDPC decoder is comparable with the Sum-Product (SP) based decoder and the error floor property of TS-LDPC codes is preserved.

The TS-LDPC decoding algorithms can be performed by analog or digital circuitry. Analog decoders are preferred in many communication systems due to their potential for higher speed, lower power dissipation and smaller chip area compared to their digital counterparts. In this work, implementation of the (120, 75) MS-based TS-LDPC analog decoder is considered.

The decoder chip consists of an analog decoder heart, digital input and digital output blocks. These digital blocks are required to deliver the received signal to the analog decoder heart and transfer the estimated codewords to the off-chip module. The analog decoder heart is an analog processor performing decoding on the Tanner graph of the code. Variable and check nodes are the main building blocks of analog decoder which are designed and evaluated. The check node is the most complicated

unit in MS-based decoders. The minimizer circuit, the fundamental block of a check node, is designed to have a good trade-off between speed and accuracy. In addition, the structure of a high degree minimizer is proposed considering the accuracy, speed, power consumption and robustness against mismatch of the check node unit.

The measurement results demonstrate that the error performance of the chip is comparable with theory. The SNR loss at Bit-Error-Rate of 10^{-5} is only 0.2dB compared to the theory while information throughput is $750Mb/s$ and the energy efficiency of the decoder chip is $17pJ/b$. It is shown that the proposed decoder outperforms the analog decoders that have been fabricated to date in the sense of error performance, throughput and energy efficiency. This decoder is the first analog decoder that has ever been implemented in a sub 100-nm technology and it improves the throughput of analog decoders by a factor of 56. This decoder sets a new state-of-the-art in analog decoding.

Acknowledgements

I would like to express my deepest gratitude to my supervisor Dr. Yousef R. Shayan for his valuable guidance, advice, encouragements and support. Without him, I could not pursue Masters and Ph.D. degrees in Concordia University. Thank you Dr. Shayan for always believing in my capabilities.

My sincere appreciation goes to my supervisor Dr. Glenn Cowan for his great ideas, help, generously answering my questions, inspiring discussions and providing useful information regarding my thesis.

I wish to acknowledge my external examiner Dr. Vincent Gaudet, one of the pioneers in analog decoding, for his detailed and helpful comments.

I also would like to thank Mr. Tadeusz Obuchowicz for his technical support. I wish to thank all my friends and colleagues for their encouragements and support.

I am lucky to have parents that gave me the opportunity to pursue my dream. Mom and Dad thanks for your encouragements, help, emotional and financial support. Without your sacrifices I could never become the person whom I am today. Also, many thanks to my sisters, Gita and Ara, for their love and encouragements.

I want to thank Nasrin and Mehdi Akhavan for their help and support, without their encouragement I would never ended in Montreal.

Last but not least, I am really indebted to my beloved wife, Sanaz, for all her sacrifices during the past 7 years. Thank you Sanaz for your endless love, patience, understanding, continuous support and encouragements.

To my parents and Sanaz

Contents

Contents vii

List of Tables x

List of Figures xi

1 Introduction 1
- 1.1 Literature Survey and Motivations . . . 2
- 1.2 Research Objectives and Contributions . . . 7
- 1.3 Thesis Organization . . . 9

2 LDPC Codes 11
- 2.1 Characteristic of LDPC Codes . . . 12
  - 2.1.1 Block and Convolutional LDPC Codes . . . 12
  - 2.1.2 Binary and Non-binary LDPC Codes . . . 13
  - 2.1.3 Regular and Irregular LDPC Codes . . . 14
- 2.2 Representations of LDPC Codes . . . 15
  - 2.2.1 Matrix Representations of LDPC Codes . . . 15
  - 2.2.2 Graphical Representations of LDPC Codes . . . 17
- 2.3 Construction of LDPC Codes . . . 19
  - 2.3.1 Random Construction . . . 19
  - 2.3.2 Structured Construction . . . 22
- 2.4 Encoding of LDPC Codes . . . 23
  - 2.4.1 Systematic Encoding . . . 23
- 2.5 Decoding of LDPC Codes . . . 25
  - 2.5.1 Iterative Decoding Algorithms . . . 27
- 2.6 Summary . . . 29

3 Analog Decoders based on the Sum-Product Algorithm 31
- 3.1 Sum-Product Algorithm . . . 31
  - 3.1.1 Sum-Product Algorithm with LLR Representation . . . 33

3.1.2 Sum-Product Algorithm with probability Representation . . . 36
3.2 Basic Circuits for Sum-Product Analog Decoding . . . 38
3.2.1 MOSFET Devices Operation Region . . . 38
3.2.2 Translinear Principle . . . 41
3.2.3 The Canonical Sum-Product Modules . . . 43
3.3 Summary . . . 48

4 Performance Evaluation of TS-LDPC Codes 49
4.1 TS-LDPC Codes . . . 50
4.2 Min-Sum Algorithms . . . 55
4.3 Simulation Results . . . 58
4.4 Conclusion . . . 60

5 Min-Sum algorithm suitability for analog TS-LDPC decoders 63
5.1 The Speed of the TS-LDPC MS Decoder . . . 64
5.2 Analog Circuits Impairments . . . 75
5.2.1 Mismatch in Between Transistors . . . 75
5.2.2 Offset Current . . . 79
5.2.3 Noise . . . 82
5.3 Design Procedure for Analog Decoder Chip Implementation . . . 85
5.4 Conclusion . . . 87

6 TS-LDPC Analog Decoder Chip and Input-Output Stages 90
6.1 TS-LDPC Analog Decoder Top Level Building Blocks . . . 91
6.2 Input-Stage of the Analog Decoder . . . 98
6.2.1 Memory Block . . . 98
6.2.2 Latch-DAC-MSCU Block . . . 100
6.2.3 Design and Schematic of a Latch-DAC-MSCU Cell . . . 103
6.3 Output Stage of Analog Decoder . . . 112
6.4 Conclusion . . . 114

7 Design and Implementation of TS-LDPC Analog Decoder 115
7.1 Architecture and Design of the Variable Node . . . 116
7.2 Architecture and Design of the Check Node . . . 125
7.2.1 Magnitude and Sign Extractor Unit (MSEU) . . . 133
7.2.2 Magnitude and Sign Combiner Unit (MSCU) . . . 135
7.2.3 Minimizer . . . 137
7.3 Conclusion . . . 151

8 Measurement Results of TS-LDPC Analog Decoder Chip 153
8.1 Teradyne Flex Tester 153
8.2 Test Board 156
8.3 Timing Scheme of the Decoder Chip 157
8.4 The Fabricated (120,75) TS-LDPC Analog Decoder Chip 161
8.5 Chip Measurement Results and Discussion 168
8.6 Comparison Between the Analog Decoders 174
8.7 Conclusion 176

9 Conclusion and Future Work 177
9.1 Future Work 180

Bibliography 182

List of Tables

6.1 Transistors vs their threshold voltage 108
6.2 Spectre simulation results for DAC unit without mismatch while the output of the DAC is connected to a P-type current mirror 108
6.3 Mismatch simulation results for DAC unit 108

7.1 Mismatch simulation results for Variable node unit 125
7.2 Comparison of transient responses and frequency responses of the designed minimizer circuits 146
7.3 Comparison of dissipated current, power consumption and estimated gate area, when inputs changes from $0 - 1.2\mu A$ in the designed minimizer circuits 146
7.4 Comparison of mismatch, energy efficiency/each check node and accuracy in the designed minimizer circuits 147
7.5 Step response simulation results for check node proposed in Fig. 7.14 while T_r and T_f are representing the rise time and fall time, respectively. 148
7.6 Mismatch simulation results for check node module. 150

8.1 Compasion table of recent analog and digital decoders 175

List of Figures

2.1 $(8, 2, 4)$ Regular Parity-Check Matrix 14
2.2 Representations of a $(8, 2, 4)$ regular LDPC code 17
2.3 Example of Gallager $(20, 3, 4)$ LDPC code matrix 22
2.4 A snap shot of information flow during the message updating procedure of Iterative decoding 28

3.1 Block diagram of a sample variable (equality) node/check node 32
3.2 A three terminal MOSFET 38
3.3 A simple translinear loop 43
3.4 A standard variable (equality) node design 46
3.5 A standard check node design 47

4.1 The general form of TS-LDPC 52
4.2 Auxiliary nodes in a TS-LDPC code with $j = 3$, $k = 4$ and $h = 4$ . . 54
4.3 The Tanner graph of TS-LDPC code with $j = 3$, $k = 4$, $h = 4$, $N = 28$ and $g = 6$ 56
4.4 BER vs Eb/No(dB) for $(28, 7)$ TS-LDPC code with $g = 6$ using SP, MS and MS with correction factor decoding algorithms 60
4.5 BER vs Eb/No(dB) for $(172, 43)$ TS-LDPC code with $g = 8$ using SP, MS and MS with correction factor decoding algorithms 61
4.6 BER vs Eb/No(dB) for $(1036, 259)$ TS-LDPC code with $g = 10$ using SP, MS and MS with correction factor decoding algorithms 61
4.7 BER vs Eb/No(dB) for $(120, 75)$ TS-LDPC code with $g = 6$ using SP, MS and MS with correction factor decoding algorithms 62

5.1 Average number of iteration versus Eb/No for $N = 28$ 66
5.2 Average number of iteration versus Eb/No for $N = 172$ 67
5.3 Average number of iteration versus Eb/No for $N = 120$ 67
5.4 Density of number of iterations for $N = 28$ 68
5.5 Density of number of iterations for $N = 172$ 69
5.6 Density of number of iterations for $N = 120$ 69

5.7 Bit Error Rate comparison for (28,7) TS-LDPC code using SP algorithm with $IT_{max} = 100$ and $IT_{max} = 20$ 70
5.8 Bit Error Rate comparison for (28,7) TS-LDPC code using MS algorithm with $IT_{max} = 100$ and $IT_{max} = 20$ 70
5.9 Bit Error Rate comparison for (28,7) TS-LDPC code using MS with correction factor algorithm with $IT_{max} = 100$ and $IT_{max} = 20$ 71
5.10 Bit Error Rate comparison for (172,43) TS-LDPC code using SP algorithm with $IT_{max} = 100$ and $IT_{max} = 20$ 71
5.11 Bit Error Rate comparison for (172,43) TS-LDPC code using MS algorithm with $IT_{max} = 100$ and $IT_{max} = 20$ 72
5.12 Bit Error Rate comparison for (172,43) TS-LDPC code using MS with correction factor algorithm with $IT_{max} = 100$ and $IT_{max} = 20$ 72
5.13 Bit Error Rate comparison for (120,75) TS-LDPC code using SP algorithm with $IT_{max} = 100$ and $IT_{max} = 20$ 73
5.14 Bit Error Rate comparison for (120,75) TS-LDPC code using MS algorithm with $IT_{max} = 100$ and $IT_{max} = 20$ 73
5.15 Bit Error Rate comparison for (120,75) TS-LDPC code using MS with correction factor algorithm with $IT_{max} = 100$ and $IT_{max} = 20$ 74
5.16 The block diagram of a latched input-output Tanner graph implemented by analog VLSI circuitry 74
5.17 The error performance of (28,7) TS-LDPC code using MS decoding algorithm in ideal case and with 10% and 20% mismatch 77
5.18 The error performance of (172,43) TS-LDPC code using MS decoding algorithm in ideal case and with 10% and 20% mismatch 78
5.19 The error performance of (120,75) TS-LDPC code using MS decoding algorithm in ideal case and with 10% and 20% mismatch 78
5.20 A simple current buffer without mismatch (a) and suffering from mismatch (b) & (c) 80
5.21 Bit Error Rate performance for (28,7) TS-LDPC code using MS algorithm in ideal case and with offset 81
5.22 Bit Error Rate performance for (172,43) TS-LDPC code using MS algorithm in ideal case and with offset 81
5.23 Bit Error Rate performance for (120,75) TS-LDPC code using MS algorithm in ideal case and with offset 82
5.24 Bit Error Rate performance for (28,4) TS-LDPC code using MS algorithm in ideal case and with noise 10% and 20% 84
5.25 Bit Error Rate performance for (172,43) TS-LDPC code using MS algorithm in ideal case and with noise 10% and 20% 84
5.26 Bit Error Rate performance for (120,75) TS-LDPC code using MS algorithm in ideal case and with noise 10% and 20% 85

5.27 The design procedure 88

6.1 Main building blocks of the TS-LDPC analog decoder 91
6.2 Error Performance of (120,75) TS-LDPC code when clipping is applied 95
6.3 Error Performance of (120,75) TS-LDPC code when LLR values are clipped to seven and quantized to three to nine bits 95
6.4 Tanner graph associated to (120,75) TS-LDPC code 97
6.5 A Snap shot of variable node 1 with complete connections 97
6.6 Input-Stage block of the decoder 99
6.7 A memory bank 99
6.8 720-cell memory 100
6.9 Latch-DAC-MSCU cell 101
6.10 10-cell Latch-DAC-MSCU block 102
6.11 Architecture of Latch-DAC-MSCU 104
6.12 Architecture of a Latch-DAC-MSCU cell 105
6.13 Design of a 6-bit latch 105
6.14 Binary weighted DACs structure 106
6.15 Binary weighted DACs structure 107
6.16 Current direction convention 109
6.17 Current direction in current mirrors 110
6.18 Cascode current mirrors (a) N-type (b) P-type 111
6.19 MSCU block's output based on the sign bit and DAC current 111
6.20 Magnitude and Sign Combiner Unit schematic 112
6.21 Output stage block 113
6.22 120-bit parallel latch block 114
6.23 120-bit serial/parallel latch block 114

7.1 Snapshot of Variable node 1 117
7.2 Implementation of bidirectional connections for variable node 1 117
7.3 The circuit which performs equation 7.2 118
7.4 Architecture of variable node 1 119
7.5 Architecture of a current buffer 120
7.6 One-input one-output (1:1) current buffer 121
7.7 Circuitry level scheme of: (a) one-input two-output (1:2) current buffer (b) one-input four-output (1:4) current buffer 121
7.8 Circuit level structure of variable node 1 122
7.9 Decision making scenario 123
7.10 DC analysis on variable node 1-snapshot 1 124
7.11 DC analysis on variable node 1-snapshot 2 124
7.12 Transient analysis on variable node 125
7.13 Architecture of the circuit performing equation 7.7 131

7.14 Architecture of the check node 1 132
7.15 One-input seven-output (1:7) P-type current mirror 132
7.16 Design of a 7-input XNOR gate 133
7.17 Architecture of the Sign and Magnitude Extractor Unit (MSEU) . . . 134
7.18 Sign extractor block 135
7.19 Sign and Magnitude Extractor Unit (MSEU) 136
7.20 Simulation results for Magnitude and Sign Extractor Unit (MSEU) . 137
7.21 Schematic of the Magnitude and Sign Combiner Unit (MSCU) 138
7.22 General architecture of an LTA designed based on a WTA 139
7.23 Loser-Take-All (LTA) circuit 140
7.24 2-input minimizer performance 142
7.25 DC analysis for 2-input LTA 142
7.26 Behaviour of an ideal check node performing SP algorithm 143
7.27 Behaviour of a check node performing MS algorithm using analog minimizer 144
7.28 Error between an SP algorithm check node and digital MS algorithm check node 144
7.29 Error between an SP algorithm check node and analog MS algorithm check node 145
7.30 Step response of the check node when the minimum input changes from 200nA to 400nA. 148
7.31 Step response of the check node including the input current mirrors and estimation of wiring cap when one of the inputs changes from 200nA to 700nA while the other input is 400nA. 148
7.32 Performance of the check node when one input changes from $-1.2\mu A$ to $1.2\mu A$ while the other input is $500nA$ 149
7.33 Performance of the modified check node when one input changes from $-1.2\mu A$ to $1.2\mu A$ while the other input is $500nA$ 149
7.34 Mismatch analysis for the check node 150

8.1 Exmaple describing timing specification of different signals in Teradyne Flex tester 156
8.2 Test bench schematic 158
8.3 Test bench schematic 159
8.4 Timing diagram for scenario 1 that LLR bits are loaded into the chip serially. 160
8.5 Timing diagram for the case that LLR bits are loaded into the chip semi-serially. 161
8.6 Timing diagram for the scenario 3 162
8.7 clk-Serial-in and clk-Latch-Data-in signals are presented. 163

8.8 clk-Latch-Data-in and V_{SW} signals are presented. 163
8.9 V_{SW} and clk-load-Ro1 signals are presented. 164
8.10 clk-Ro2 signal is presented. 164
8.11 clk-load-Ro1 and clk-Ro2 signals are presented. 165
8.12 Latch-mode-Ro2 and clk-Ro2 signals are presented. 165
8.13 The die micrograph of the (120, 75) TS-LDPC analog CMOS decoder. 166
8.14 Teradyne Flex tester while the (120,75) TS-LDPC analog decoder IC is under test. 167
8.15 Snapshot of the (120,75) TS-LDPC analog decoder IC in the CQFP-44 pin package. 167
8.16 Snapshot of the decimal based decoder chip's output illustrating Early Termination Signal and Serial-data-out. 169
8.17 The chip measurement results with different I_{ref} and I_{adj} currents . . 171
8.18 Comparison of the chip measurements with different decoding times and Matlab simulation results 172
8.19 Snapshot showing V_{sw} and Early Termination Signal at low SNR. The Early Termination line is heavily loaded. 173
8.20 The histogram of the average required decoding time 173

List of Acronyms

ADC	Analog-to-Digital Converter
APP	a Posteriori Probability
ASIC	Application-Specific Integrated Circuit
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
BP	Belief Propagation
clk	clock
d_{min}	Minimum Hamming Distance
DAC	Digital-to-Analog Converter
FF	Flip Flop
GF	Galois fields
HVT	High threshold voltage
KCL	Kirchhoff's Current Law
KVL	Kirchhoff's Voltage Law
LDPC	Low-Density Parity-Check
LLR	Log-Likelihood-Radio
LR	Likelihood Ratio
LSB	Least Significant Bit
LTA	Loser-Take-All
LVT	Low threshold voltage
MOSFET	Metal Oxide Semiconductor Field Effect Transistor
MP	Message Passing
MS	Min-Sum
MSB	Most Significant Bit
MSCU	Magnitude and Sign Combiner Unit
MSEU	Magnitude and Sign Extractor Unit
PCB	Printed Circuit Board

SNR	Signal-to-Noise Ratio
SP	Sum-Product
SVT	Standard threshold voltage
TG	Tanner Graph
TS-LDPC	Turbo-Structured Low-Density Parity Check
VBT	Visual Basic for Test
VLSI	Very Large Scale integration
WTA	Winner-Take-All

Chapter 1

Introduction

In today's life style, portable communication devices are becoming a non-separable part of a day to day life. As of today, there are six billion cellular phone registered customers [1]. Communication giants compete constantly to control the market by introducing smaller, lighter and faster devices which have several capabilities. Increase in the quantity of devices raises the expectation of the quality of service. The demand for high speed and more reliable communication systems is increasing every day. There are different communication modules that contribute to a higher quality of voice and data transmission.

Each communication system consists of two basic blocks: transmitter and receiver. The transmitter's task is to convert information bits to signals that can be transmitted over the channel. The receiver recovers the information bits with the lowest possible probability of error. To minimize the number of errors at the output of the receiver and consequently having more reliable communication over noisy channels, error control codes are introduced. The presence of an error control code in a communication system implies that an encoder and a decoder should be utilized

at the transmitter and receiver sides, respectively.

Increasing demand for using wireless communication systems and expecting good quality of service have shifted researchers to focus towards designing more efficient codes with less complexity. One of the most promising channel codes is called a *Low-Density Parity Check* (LDPC) code. Using this type of code in communication systems, the Shannon limit [2] (maximum coding efficiency) can be approached.

1.1 Literature Survey and Motivations

Low-Density Parity Check codes are a class of linear block codes that have sparse *parity check matrices* (**H**) . LDPC codes were first introduced by Gallager in his thesis in 1962 [3]. For almost 30 years, LDPC codes were forgotten due to their high decoding complexity. In the 1990s, with the advance of *Application-Specific Integrated Circuit* (ASIC) technology these LDPC codes became attractive and were rediscovered by Mackay [4]. Afterwards, LDPC codes have been studied vastly due to their linear decoding complexity and superior error performance. It has been shown that the *Bit Error Rate* (BER) performance of LDPC codes using iterative decoding algorithms is near the Shannon limit [5]. LDPC codes have been adopted in the latest communication and storage systems. LDPC code applications include but are not limited to, Digital Video Broadcasting satellite (DVB-S2 and DVB-T2) [6, 7], 10Gigabit Ethernet (10GBASE-T) [8], broadband wireless access (WiMax) [9], wireless local area network (WiFi) [10], deep-space communications [11] and magnetic storage in hard disk drives [12].

The graphical representation of LDPC codes using bipartite graphs is known as *Tanner* graph and was introduced by Tanner [13] in 1980s. Tanner graph is a great

approach to understand the behaviour of the iterative decoding of LDPC codes. The Tanner graph associated with the parity check matrix is comprised of two disjoint sets of nodes, known as variable nodes and check nodes, corresponding to the columns and rows of the **H** matrix, respectively. Variable nodes are connected to parity check nodes through the edges. The connection structure is defined by the parity check matrix. Each variable node represents a coded bit and each check node represents a parity check equation. The girth of a LDPC code is defined as the length of the shortest cycle in the associated Tanner graph.

There are many different ways for constructing parity check matrices and designing LDPC codes. These methods are generally defined as random and structured construction. Structured codes are more interesting due to their regularity. In this thesis, we consider a class of structured regular LDPC codes, called *Turbo-structured LDPC* (TS-LDPC) codes [14, 15]. TS-LDPC codes can be designed with any arbitrarily chosen column weight, row weight and girth. There are many advantages of TS-LDPC codes as compared with other kinds of LDPC codes. The most important advantage of TS-LDPC codes is that they can be designed to have a large girth (shortest cycle) which is a crucial parameter in designing LDPC codes. Since the dependencies of the messages that are propagating through the graph are introduced by cycles, iterative decoding algorithms are optimal when the Tanner graph is cycle-free (infinite girth). Hence, iterative decoding algorithms are sub-optimal in the presence of cycles. Increasing the length of the shortest cycle (girth) in a graph improves the effectiveness of the iterative decoding. Thus, TS-LDPC codes with inherently large girths would enjoy a more efficient iterative decoding compared to other LDPC codes. In addition, large girth guarantees large minimum Hamming distance (d_{min}) of the

code, resulting in lower error floor for TS-LDPC codes at high *Signal-to-Noise Ratio* (SNR) [13]. Moreover, the Parity check matrix of TS-LDPC codes can be constructed from a much smaller shift matrix $\mathbf{S}$. The simulation results show that TS-LDPC codes outperform the randomly generated LDPC codes at high SNR and have much lower error floor [14, 15]. Therefore, a TS-LDPC code is a good candidate to be adopted in most communication applications.

Iterative decoding algorithms can be used in decoding TS-LDPC codes. There are a number of iterative decoding algorithms with different trade offs between the complexity and error performance. Iterative decoding algorithms are performed by iteratively exchanging messages between check nodes and variable nodes bidirectionally through the edges. Depending on the decoding algorithm and type of messages, specific computations have to be executed in the check nodes and variable nodes of the Tanner graph. There are two types of iterative decoding algorithms: hard decision and soft decision. Hard decision algorithms deal with 0s and 1s. The advantage of hard decision iterative algorithms is their low complexity which makes them attractive for high data rate communication applications by trading off the error performance. Two major hard decision algorithms are Majority decoding algorithm [16] and Gallager algorithm [3].

The performance of soft decision algorithms is better than the hard decision algorithms but they have higher complexity. The *Sum-Product* (SP) or *Belief Propagation* (BP) algorithm is the best performing decoding algorithm while its implementation complexity is high [3, 17]. Another algorithm with nearly the same performance but less complexity is known as the *Min-Sum* (MS) algorithm [13, 18–23].

Generally, decoders have been implemented by digital *Very Large Scale integration*

(VLSI) circuits. The fundamental factors in implementing channel decoders are including but not limited to speed, power consumption and chip area. Digital decoders can be implemented serially or in parallel. In serially implemented decoders each bit of information is processed serially based on the decoding procedure. Therefore, these types of decoders cannot provide the required throughput for today's high-speed communication standards. High speed decoding is achievable using parallel implementation of decoders. However, large parallel implementations of decoders result in larger chip area, higher complexity of the wiring and consequently increase the fabrication cost. In addition, decoders based on iterative decoding algorithms such as, Turbo and LDPC decoders, are very complex.

To resolve the complexity issue some researches have considered decoding of these codes using analog circuits. Recently, implementation of analog decoders has been targeted as a research direction by many VLSI research groups [24–32]. Potentially, complex computations can be performed by simple analog circuits. Moreover, soft-decision digital decoders require *Analog-to-Digital Converters* (ADC) with multi-bit resolution. Analog decoders can directly process analog channel samples, eliminating these ADCs, replacing them with 1-bit ADCs at the output of the decoder. Therefore, analog decoders potentially enjoy higher speed, lower power dissipation and smaller chip area compared to their digital counterpart.

For instance, most of iterative decoding algorithms such as SP use multiplication as a basic block. However implementing SP algorithm in log domain eliminates the use of multiplication block. Multiplication can be done easily using non-linear analog circuits. These circuits are biased in weak inversion region where the transistor is

almost off, therefore the transistor current is small. This leads to lower power consumption. On the other hand, biasing a transistor in weak inversion results in lower speed. To increase the speed the power should be raised which results in increasing the power consumption. Therefore, there are numbers of challenges in the design of analog decoders.

The first analog decoders were designed to perform the Viterbi decoding algorithm [33,34]. The designed analog Viterbi decoders outperformed the digital ones by a wide margin [35, 36]. Afterwards analog decoders using iterative decoding algorithms such as, Turbo codes, LDPC codes and Block Product codes were proposed [37–39]. Some of the proposed analog decoders have been designed and implemented. The first fabricated chip was reported in 1999 by Lustenberger et. al. [40].

Analog decoders are suitable for the decoding algorithms using iterative schemes such as SP and MS algorithms. However, MS algorithms are preferable since biasing transistors to operate in either strong or moderate inversion is less complex than biasing in weak inversion [21, 28, 41].

In [28], Hemati has designed and implemented a short length LDPC analog decoder based on MS algorithm. This is the only MS-based analog LDPC decoder chip that has been implemented. It is shown that the implemented chip under-performs the theory by roughly 2dB of $\frac{E_b}{No}$ at 10^{-5}. The BER was measured while the throughput of the decoder is 6 Mb/s. It should be noted that the reported throughput is calculated based on the settling time.

In practical wireless communication systems depending on the application, BER of 10^{-5} to 10^{-7} is of interest. Given the BER shape in [28], the penalty would be more than 2dB as BER tends towards 10^{-7}. Therefore, an energy efficient analog decoder

which achieves the promising error performance needs to be designed and fabricated.

1.2 Research Objectives and Contributions

The main objective of this thesis is to implement and fabricate the best-in-class analog decoder. The proposed analog decoder is one of the longest length decoders and outperforms all analog decoders in the sense of error performance, energy efficiency, throughput and die area. In this work, TS-LDPC codes are considered due to their superior error performance at high SNR and especially lower error floor which makes them attractive for communication standards. The Min-Sum algorithm is utilized in TS-LDPC decoder context to reduce the complexity of the decoder. To show the scalability of analog decoders the code length of 120 is considered. An analog decoder chip is designed and fabricated. The measurement results are presented and compared to the analog and digital decoders. Contributions are detailed in point form as follows:

- TS-LDPC codes are selected as a candidate of analog decoder due to their superior error performance and structural design compared to randomly generated LDPC codes.

- A number of TS-LDPC codes with different girths and code rates have been designed. For code rate of $\frac{1}{4}$, (28,7), (172,43), and (1036,259) TS-LDPC codes are designed corresponding to girths of 6, 8 and 10, respectively. Moreover, (120,75) TS-LDPC code with code rate of $\frac{5}{8}$ is designed and implemented.

- To reduce the complexity of the decoder, for the first time the Min-Sum algorithm is adopted as the iterative decoding algorithm for TS-LDPC codes. The

error performance of MS-based TS-LDPC decoder is evaluated and compared to the decoder using the SP algorithm. It is shown that the error performance of the MS-based decoder is comparable with the SP-based decoder with less complexity.

- The Min-Sum algorithm suitability for the TS-LDPC analog decoders is studied. The TS-LDPC analog decoder is simulated by adding analog impairments to the digital decoder (simulation). It is shown that the MS-based TS-LDPC analog decoder is fairly robust against introduced analog impairments.

- The top level architecture of the Min-Sum based (120,75) TS-LDPC analog decoder chip is proposed. The discussed decoder consists of three main building blocks: input-stage, analog-decoder and output-stage.

- A minimizer circuit is selected and modified to be used in the check node structure. The accuracy, speed and matching property of the minimizer are considered. The structure of a high degree minimizer is studied. Using the minimizer circuit, the check node architecture and circuit is introduced and evaluated.

- Based on the Tanner graph of (120,75) TS-LDPC code and variable and check node structures, analog decoder performing Min-Sum algorithm is proposed, implemented and tested.

- This analog decoder is the fist analog decoder that is successfully implemented in sub 100-nm technology.

- The measurement shows that the proposed analog decoder has the highest

throughput and the most energy efficiency compared to previous analog decoders. This decoder narrows the gap between the analog and digital decoders. This analog decoder sets a new state-of-the-art in analog decoding.

1.3 Thesis Organization

The organization of this thesis is as follows. In Chapter 2, the necessary background for LDPC codes including characteristics, representations, encoding and decoding is presented. In Chapter 3 the Sum-Product algorithm is explained in *Log-Likelihood-Radio*(LLR) and probability domain representations. In this chapter, the necessary background required to design the SP algorithm modules is discussed as well.

In Chapter 4, TS-LDPC codes are presented and various TS-LDPC codes are designed. The Min-Sum algorithms are adopted as the TS-LDPC iterative decoding algorithm. The error performance of TS-LDPC codes decoded utilizing MS algorithms is evaluated for the first time.

Chapter 5 is focused on implementation issues of the MS based TS-LDPC decoder. In this chapter the sensitivity of the TS-LDPC analog decoder to some analog impairments such as mismatch, noise and offset current is illustrated. Moreover, the tolerability margin of the decoder with respect to each analog impairment is tested.

In Chapter 6, the architecture of the (120,75) TS-LDPC MS-based analog decoder is proposed. The required input and output modules are designed and implemented. Chapter 7 describes the analog heart of the decoder. The architecture and circuit level design of the variable node and the check node are recommended. Based on the Tanner graph of the (120,75) TS-LDPC code, the analog decoder is designed.

The fabricated analog decoder chip is tested and the measurement results are

summarized in Chapter 8. In this chapter, the proposed analog decoder is compared with the most recent analog and digital decoders. Finally, in Chapter 9 the conclusion of the thesis is presented and some possible future works are introduced.

Chapter 2

LDPC Codes

In this chapter the necessary background for LDPC codes is presented. In the following the characteristics, representation, encoding and decoding methods of LDPC codes will be discussed.

The organization of this chapter is as follows. In Section 2.1 the characteristics of LDPC codes are reviewed. Block and convolutional, binary and non-binary and regular LDPC codes are discussed. In Section 2.2, Matrix and Tanner graph representations of an LDPC code are illustrated. Two main structures of LDPC codes, random and structured construction, are explained in Section 2.3. Encoding of LDPC codes is discussed in Section 2.4 and systematic encoding is explained as well. Section 2.5.1 explains the decoding of LDPC codes. Iterative decoding of LDPC codes are discussed in 2.5.1.

2.1 Characteristic of LDPC Codes

Low-Density Parity-Check Codes (LDPC) are a class of linear block codes. One of the advantages of LDPC codes compared to other classes of linear block codes is the sparseness of their parity check matrix. This means that their parity check matrix has a small number of non-zero elements. Another attractive feature of LDPC codes is their efficient iterative decoding. The performance of LDPC codes is very close to the Shannon limit [2] when iterative decoding is used. It has been shown that the performance of a rate $R = \frac{1}{2}$ LDPC code with a block length of 10^7 bits at BER of 10^{-6} is within 0.04 dB of the Shannon limit for binary input *Additive White Gaussian Noise* (AWGN) channel [42]. Also it has been suggested that optimal LDPC codes under sum-product decoding for AWGN channels may approach the Shannon limit asymptotically as the block length tends to infinity [42].

The complexity of iterative decoding depends on the number of non-zero elements in the parity check matrix which creates the connection between check nodes and variable nodes. The complexity of iterative decoding is low for LDPC codes since they have a sparse parity check matrix. Some important characteristics of LDPC codes are explained in following sections.

2.1.1 Block and Convolutional LDPC Codes

LDPC block codes have attracted many researchers due to their great performance that approaches to the Shannon limit and their relatively simple iterative decoding structure. Another category of LDPC codes are Low-density parity-check convolutional codes. LDPC convolutional codes have been proposed by Felstrom and Zigangirov in the late 1990s [43]. LDPC convolutional codes have also very good error

performance and their decoding algorithms are similar to LDPC block codes. The main advantage of LDPC convolutional codes compared to the block codes is the ability to work with an arbitrary length of information bits without any need to break them into blocks. A single LDPC convolutional code can be employed to construct a family of codes of varying frame length. This is an advantage in terms of flexibility compared to LDPC block codes, where a new code must be constructed each time a new transmission frame length is required. Otherwise the transmitter divides the data packets into blocks, if block codes are employed. In this work, LDPC block codes are considered due to their employment in the communication standards and simpler iterative decoding procedure.

2.1.2 Binary and Non-binary LDPC Codes

A great deal research has been done on LDPC codes over *Galois fields* $GF(2)$ (binary LDPC codes), however they can be extended to $GF(q)$ by considering the non-zero weights over $GF(q)$, forming the parity-check matrix [44]. It has been shown that especially for moderate code length, LDPC codes over $GF(q)$ have a better error performance [45]. However the higher performance is gained at the expense of increased decoding complexity.

In this work, binary LDPC codes are considered where the elements of the parity check matrix are either 0 or 1. An (N, j, k) LDPC code is a block code where N is the length of the block, j and k are the numbers of ones in each row and in each column of the parity check matrix $\mathbf{H}$, respectively. This code can be presented using block code notation as an (N, K) LDPC code. The rate of the code $R = \frac{K}{N}$ where K is the length of information bits and N is the codeword length. This means that

$$H = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

Figure 2.1: $(8, 2, 4)$ Regular Parity-Check Matrix

$(N - K)$ redundant bits have been added to the message to determine and possibly correct the errors.

2.1.3 Regular and Irregular LDPC Codes

LDPC codes can be categorized in two groups, regular and irregular. An LDPC code is called regular if all the columns of the parity check matrix have the same number of ones and all the rows have the same number of ones as well. Fig. 2.1 shows a sample regular parity check matrix.

The number of ones in each column is called the degree of the column or the degree of variable nodes denoted by d_v. In the same way the number of ones in each row is called the degree of the row or the degree of check nodes represented by d_c. For the regular LDPC codes $N \cdot d_v = M \cdot d_c$ where N is the number of columns and $M = N - K$ is the number of rows in the parity check matrix. In regular LDPC codes degrees of all variable nodes are the same $d_v = j$ as well as degrees of all check nodes $d_c = k$.

LDPC codes are called irregular if the number of ones in each column or row are not constant. An irregular LDPC code cannot be represented by the degree parameters $d_v = j$ and $d_c = k$.

If the parity check matrix ($\mathbf{H}$) of the code is full rank, then $R = \frac{K}{N} = 1 - \frac{M}{N} = 1 - \frac{j}{k}$.

The parity check matrix of a regular $(8, 2, 4)$ LDPC code is shown in Fig. 2.1. For

this specific LDPC code, the block length $N = 8$, the degree of columns $d_v = 2$ and the degree of rows $d_c = 4$ and therefore $R = \frac{1}{2}$.

2.2 Representations of LDPC Codes

For each matrix representation of an LDPC code, there is a corresponding graphical representation. Considering the graph of an LDPC code, the element h_{ij} of the $M \times N$ parity check matrix is 1 if and only if the ith check node is connected to the jth variable node in the graph. In the same way, a graph between N variable nodes (messages) and M check nodes (parity checks) can be obtained using the $M \times N$ parity check matrix $\mathbf{H}$. A connection is assigned between any check node and variable node of the graph that has entry 1 in the parity-check matrix. The graph obtained using the parity check matrix is a bipartite graph. A bipartite graph is a special graph where the set of vertices can be divided into two disjoint sets $\mathbf{V}$ and $\mathbf{C}$ such that every edge has one end-point in $\mathbf{V}$ and one end-point in $\mathbf{C}$ and no edge exists between vertices at the same set. The graphical representation of the LDPC codes is known as a Tanner graph and was introduced in 1980s by Tanner [13]. In the following, two fundamental these ways for representing LDPC codes are reviewed in more detail.

2.2.1 Matrix Representations of LDPC Codes

Since LDPC codes form a special class of linear block codes, they follow all of the definitions of block codes which already exist [46]. K bits of the message vector $\mathbf{m} = m_1, m_2, ..., m_k$ is mapped to N bits of the codeword $\mathbf{c} = c_1, c_2, ..., c_n$ as follows,

$$\begin{aligned} c_1 &= m_1 g_{1,1} + m_2 g_{1,2} + ... + m_K g_{1,K} \\ \vdots &= \qquad\qquad \vdots \\ c_N &= m_1 g_{N,1} + m_2 g_{N,2} + ... + m_K g_{N,K} \end{aligned} \tag{2.1}$$

where + is GF(2) summation.

Equation 2.1 can be presented in matrix form as,

$$\mathbf{c} = \mathbf{m} \cdot \mathbf{G} \tag{2.2}$$

where $\mathbf{G}$ is,

$$\mathbf{G} = \begin{bmatrix} g_{1,1} & \cdots & g_{1,K} \\ \cdots & \ddots & \cdots \\ g_{N,1} & \cdots & g_{N,K} \end{bmatrix}. \tag{2.3}$$

$\mathbf{G}$ is a $K \times N$ full rank matrix which is called the *generator matrix*. The $(N - K) \times N$ parity check matrix $\mathbf{H}$ can be obtained from the G matrix, using

$$\mathbf{G}\mathbf{H}^T = \underline{0}. \tag{2.4}$$

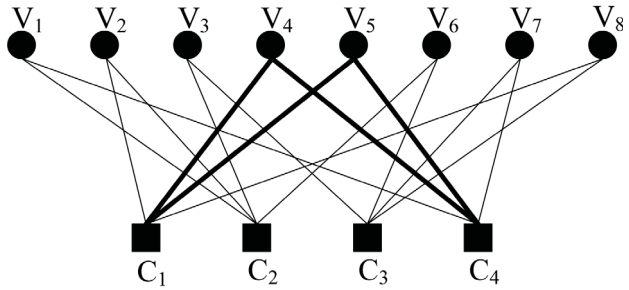
This implies that any row of $\mathbf{H}$ is orthogonal to all rows of $\mathbf{G}$. Using equations 2.2 and 2.4,

$$\mathbf{c}\mathbf{H}^T = (\mathbf{m}\mathbf{G})\mathbf{H}^T = \mathbf{m}(\mathbf{G}\mathbf{H}^T) = \mathbf{m}\underline{0} = \underline{0}. \tag{2.5}$$

Based on equation 2.5, the parity-check matrix $\mathbf{H}$ can be designed based on the

$$H = \begin{array}{c} \begin{matrix} v1 & v2 & v3 & v4 & v5 & v6 & v7 & v8 \end{matrix} \\ \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix} \begin{matrix} c1 \\ c2 \\ c3 \\ c4 \end{matrix} \end{array}$$

(a) Matrix representation



(b) Tanner Graph

Figure 2.2: Representations of a $(8, 2, 4)$ regular LDPC code

degree distribution of the nodes (degree of columns and rows).

2.2.2 Graphical Representations of LDPC Codes

The *Tanner graph* (TG) of an LDPC code is equivalent to the trellis for a convolutional code. In the same way it provides us with a better idea about the structure of the decoding algorithm and gives a complete representation of the code. Tanner graphs of LDPC codes are bipartite graphs, hence there is no connection or edge between any two nodes of the same set and the edges only connect two nodes of different sets. There are two types of nodes in a Tanner graph, called variable nodes and check nodes. Number of check nodes and variable nodes are equal to the number of rows (M) and columns (N) of the **H** matrix, respectively.

As an illustrative example an (8,2,4) LDPC code is considered in Fig. 2.2. This figure shows two types of representation of an LDPC code. In Fig. 2.2(b) the Tanner

graph is represented while variable nodes are shown by circles and check nodes by squares. The connections are based on the 0s and 1s in the parity check matrix. Variable node v_i, $i = 1, ..., N$ is connected to check node c_j, $j = 1, ..., M = N - K$, if and only if element h_{ij} in the **H** matrix has the value of 1.

Fig. 2.2(b) shows that there are $M = N - K = 4$ check nodes and $N = 8$ variable nodes. On the other hand as shown in Fig. 2.2(a), $M = 4$ rows of **H** represent the 4 check nodes and $N = 8$ columns of **H** represent the 8 variable nodes of the Tanner graph. As an example, c_1 is connected to v_2, v_4, v_5 and v_8 in the graph while in the first row of the corresponding **H** matrix h_{12}, h_{14}, h_{15} and h_{18} are equal to 1 and $h_{11} = h_{13} = h_{16} = h_{17} = 0$. The Tanner graph in 2.2(b) is regular and 2 edges are connected to each variable node, which implies that the degree of variable nodes is 2. Each check node is connected through 4 edges and holds degree of 4. In the **H** matrix of Fig. 2.2(a), the degree of columns is 2 and the degree of rows is 4.

One of the important characteristics of a Tanner graph is its *cycle* length. A cycle of length t in a Tanner graph is a path comprising of t edges which starts from one node and ends in the very same node. In Fig. 2.2(b) a cycle of length of 4 is specified with bold solid lines. The *girth* of a Tanner graph is defined as the shortest cycle of the graph. Therefore, the girth of the Tanner graph in Fig. 2.2 is 4. The cycles present themselves in an **H** matrix with ones on the corners of polygons. The shape of polygon depends on the length of the cycle. For example a cycle with length of 4 manifests itself as 4 ones on the corners of a rectangular. Presence of cycles degrades the performance of iterative decoding algorithms used for LDPC codes. Iterative decoding procedure of LDPC codes is optimal when a cycle-free graph is used (graph with the shortest cycle of infinite length) [13]. Therefore, increasing the length of the

shortest cycle (girth) in a graph improves the effectiveness of the iterative decoding.

2.3 Construction of LDPC Codes

Construction of an LDPC code is defined as designing the parity check matrix **H**. To design an **H** matrix all of the parameters of LDPC codes should be considered including, the desired block length, degree distribution, code rate and other characteristics. The length of the girth of LDPC codes is one of the crucial parameters in designing the **H** matrix, which is required to be as high as possible. This will be discussed in the following chapter. On the other hand the sparseness of the **H** matrix should be high to maximize the minimum distance. Different design criteria for the **H** matrix such as efficient encoding and decoding, near capacity performance and low error-rate floor are targeted by different design approaches. Considering these factors, there are many different ways for constructing the parity check matrices and designing LDPC codes. These methods are generally defined as random and structured construction.

2.3.1 Random Construction

The first method used to construct LDPC codes is random construction. Random construction does not have many constraints and can be easily applied for different combinations of parameters. On the other hand randomly generated LDPC codes do not guarantee a large girth. Albeit for a desired girth, random construction with some added constraints can be used. Two methods for random designing of LDPC codes, Mackay and Gallager random construction, are discussed in this section.

Mackay Construction

Mackay discovered the advantages of designing block codes with sparse parity check matrices. He was the first one who showed that LDPC codes can perform near the Shannon limit [4]. In [5] Mackay represented different random construction methods. He has designed and archived a large number of LDPC codes on his web page targeting data communication and storage [47]. Most of his designed codes are regular LDPC codes based on bipartite graphs. The major drawback of his designed codes is the lack of structure which results in high complexity encoding. A few of Mackay's algorithms are reviewed in the following.

- Construction 1A

  An $M \times N$ $\mathbf{H}$ matrix is generated with fixed number of ones in each column and as high as possible with uniform row weights. In this type of construction, overlap between any two columns of the created matrix is not more than 1 and this leads to a matrix which does not have cycles of length 4. Since the graph of the code is bipartite, the length of cycles should be even. Consequently the shortest cycle in the graph will be 6. This means the girth of the code is 6.

- Construction 2A

  In $M \times N$ $\mathbf{H}$ matrix, $\frac{M}{2}$ of the columns are designed with weight 2 and there is no overlap between any two columns. The remaining columns are generated randomly with weight 3. Weights of the rows are as uniform as possible and the overlap between any two columns of the entire matrix is not greater than one. It is clear that the resultant LDPC code is irregular.

- Construction 1B and 2B

A matrix is produced using methods 1A or 2A. Then a small number of columns of this matrix are chosen and deleted randomly. Hence, the bipartite graph corresponding to the matrix has no short cycles of less than some length l. By reducing the possibility of having short cycles in the matrix the performance of the code is improved.

Gallager Construction

Gallager in his thesis designed a group of LDPC codes, then added some constraints to design the matrix free of cycles of length 4 [3].

To construct the matrix, first we define an (N, j, k) parity check matrix, a matrix with N columns that has j ones in each column and k ones in each row and zero in other locations. For constructing an ensemble of (N, j, k) matrices, an example from Gallager's dissertation has been taken with $N = 20$, $j = 3$ and $k = 4$. The matrix is divided into j submatrices and each submatrix contains a single 1 in each column. The first division of these submatrices are constructed such that the ith row contains ones in column $(i - 1)k + 1$ up to column ik as it is shown in Fig. 2.3. The other submatrices are just column permutations of the first division. Gallager defines the ensemble of (N, j, k) codes as the ensemble resulting from random permutations of the columns of each of the bottom $(j - 1)$ submatrices of a matrix with equal probability is assigned to each permutation [3].

There are other random constructions used in literature like Bit filling construction [48] and Average girth distribution based construction [49]. [48] provides a heuristic method called "bit-filling" to search for LDPC codes with large girth. [49] searches for a good LDPC code based on the average of the girth distribution of the code.

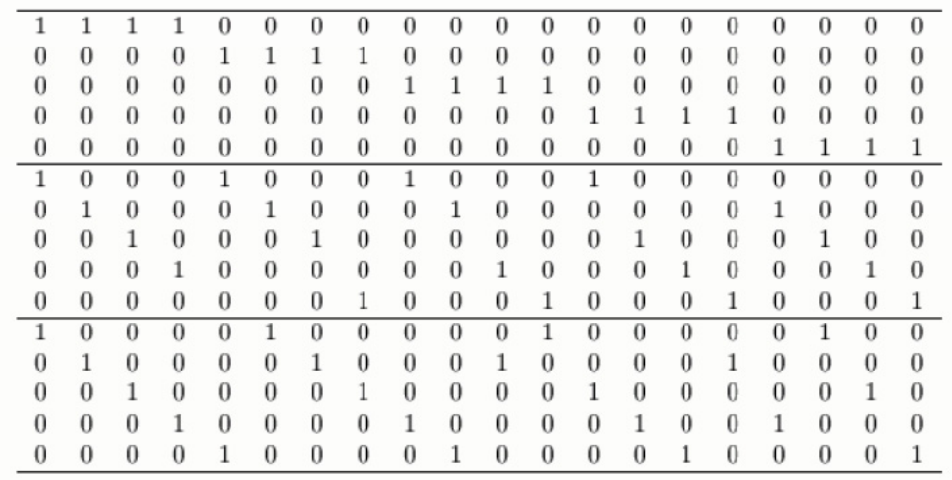


Figure 2.3: Example of Gallager $(20, 3, 4)$ LDPC code matrix

2.3.2 Structured Construction

For randomly constructed regular and irregular LDPC codes, the required memory to store and manipulate the nonzero elements of the parity check matrix is a significant burden for hardware implementation. Therefore, construction of structured LDPC codes is desired in many applications to reduce the hardware cost and simplify the encoding/decoding systems.

Considerable research has been done on regular structured LDPC codes. Two examples of such codes are cyclic and quasi-cyclic LDPC codes [50,51] which have been recently targeted by researchers. These codes can be encoded with low complexity and constructed algebraically. There are many ways to construct quasi-cyclic codes; three typical methods are based on finite geometries [52], [53], Balanced Incomplete Block Design (BIBD) [54–60] and disjoint different sets [61]. One of the most recent proposed structured LDPC codes is Partition-and-Shift (PS)-LDPC code [62–64]. This code is adopted by the IEEE 802.16e standard [65].

2.4 Encoding of LDPC Codes

The weakness of LDPC codes is their high encoding complexity. As a result many different encoding schemes have been suggested for LDPC codes to reduce the complexity [66, 67]. In this work, systematic encoding is used and described in details.

2.4.1 Systematic Encoding

After designing the **H** matrix, the generator matrix **G** can be found from **H**. The conventional way to encode LDPC codes is called systematic encoding. For systematic encoding the data blocks **m** are multiplied by the generator matrix **G**. It means that the codeword is $\mathbf{c} = \mathbf{mG}$. Although the parity check matrix **H** for LDPC codes is sparse, the associated generator matrix **G** is not. The encoding complexity of LDPC codes is $o(N^2)$ where N is the code block length of the LDPC code [68]. As the code block length is high, the complexity of the encoder significantly affects the application of LDPC codes. For short block lengths, systematic encoding is preferable because the size of the matrix is not big and the method results in low complex decoding. A systematic Generator matrix **G** can be computed from **H** through the following steps:

1. Choose the **H** matrix (regular or irregular).
2. Reorder the columns of **H** in a way that the first $M = N - K$ columns of new **H** matrix ($\mathbf{H}_{new}$) have all ones in the diagonal of matrix.
3. Apply Gaussian Elimination to **H** in order to get the systematic **H** matrix.
4. The generator matrix **G** is the transpose of the systematic **H** matrix and can be found easily.

Now assume that the parity check matrix has the following format:

$$\mathbf{H} = [\mathbf{H}_1 | \mathbf{H}_2] \tag{2.6}$$

$\mathbf{H}_1$ and $\mathbf{H}_2$ are sparse matrices and the rows in $\mathbf{H}$ are linearly independent. Matrix $\mathbf{H}_1$ is a square $(N-K)\times(N-K)$ invertible matrix and $\mathbf{H}_2$ is a rectangular $(N-K)k$ matrix. We can reorder the columns of the $\mathbf{H}$ matrix, in order to get invertible $\mathbf{H}_1$. The equivalent parity-check matrix is,

$$\mathbf{H}_{new} = \mathbf{H}_1^{-1}\mathbf{H} = \mathbf{H}_1^{-1}\left[\mathbf{H}_1 | \mathbf{H}_2\right] = \left[\mathbf{I}_{(N-K)} | \mathbf{P}\right] \tag{2.7}$$

where

$$\mathbf{I}_{(N-K)} = \mathbf{H}_1^{-1}\mathbf{H}_1 \quad \& \quad \mathbf{P} = \mathbf{H}_1^{-1}\mathbf{H}_2 \tag{2.8}$$

After deriving the parity matrix $\mathbf{P}$, the generator matrix $\mathbf{G}$ can be formed as follows,

$$\mathbf{G}_{K\times(N-K)} = \left[\mathbf{P}_{N-K} | \mathbf{I}_K\right]. \tag{2.9}$$

The codeword $\mathbf{c}$ can be obtained using the following equation,

$$\mathbf{c} = \mathbf{m}\mathbf{G}. \tag{2.10}$$

where $\mathbf{c}$ is the codeword, $\mathbf{m}$ is the information vector and $\mathbf{G}$ is the generator matrix. From,

$$\mathbf{G} \times \mathbf{H}^T = \mathbf{G} \times \mathbf{H}_{new}^T = \mathbf{0} \tag{2.11}$$

it can be understood that the systematic Generator matrix $\mathbf{G}$ can be applied at the transmitter to encode the information bits and sparse parity-check matrix $\mathbf{H}$ can be used in the receiver to decode the received codeword. The reason for using the sparse parity-check matrix is reducing the complexity of the decoder.

2.5 Decoding of LDPC Codes

Consider an (N, K) linear block code in its general format. Assume that the codeword $\mathbf{c} = c_1, ..., c_n$ has been transmitted over an Additive White Gaussian Noise channel. Then, the received vector $\mathbf{r}$ at the receiver is,

$$\mathbf{r} = \mathbf{c} + \mathbf{n} \tag{2.12}$$

where $\mathbf{n}$ is additive white Gaussian noise. The codeword $\mathbf{c}$ can be written as,

$$\mathbf{c} = \mathbf{mG} \tag{2.13}$$

and $\mathbf{m} = m_1, ..., m_K$ is the information bits vector and $\mathbf{G}$ is the generator matrix. Equation 2.12 suggests that the received vector is a mixture of the transmitted vector and noise. Since the decoder does not have any information on the signal and noise patterns, therefore the the decoder should find the most likely message vector $\widehat{\mathbf{m}}$ which was transmitted.

At first, the decoder determines if any error has occurred. Then the decoder will take a proper action to locate the errors and possibly correct them. The decoder

computes the following vector

$$\mathbf{S} = \mathbf{r}\mathbf{H}^T = s_1, ..., s_{N-K} \tag{2.14}$$

which is called *syndrome* of $\mathbf{r}$. If $\mathbf{S} = \mathbf{0}$ then $\mathbf{r}$ is a codeword and we accept it as a transmitted codeword. If $\mathbf{S} \neq \mathbf{0}$ then an error has occurred during the transmission of the codeword and $\mathbf{r}$ is not a valid codeword. The syndrome vector is independent of the transmitted codeword and it is only dependent on the channel noise and parity check matrix, since

$$\mathbf{S} = \mathbf{r}\mathbf{H}^T = (\mathbf{c} + \mathbf{n})\mathbf{H}^T = \mathbf{c}\mathbf{H}^T + \mathbf{n}\mathbf{H}^T \tag{2.15}$$

where $\mathbf{c}\mathbf{H}^T = \mathbf{m}\mathbf{G}\mathbf{H}^T = \underline{0}$ and therefore $\mathbf{S} = \mathbf{n}\mathbf{H}^T$. Having $\mathbf{S}$, and the parity check matrix, the noise vector can be computed. In that case an estimation of the transmitted vector can be obtain using $\mathbf{c} = \mathbf{r} - \mathbf{n}$. But finding the noise vector is not an easy job, therefore many decoding algorithms have been proposed to overcome this problem.

For decoding LDPC codes various decoding algorithms can be used such as, Majority-Logic (MLG) decoding, Bit-Flipping (BF) decoding, weighted BF decoding and *a Posteriori Probability* (APP) decoding.

Iterative decoding algorithms are another type of LDPC decoding algorithm which depending on their context are called: the Sum-Product (SP) algorithm, the Belief Propagation (BP) algorithm and the *Message Passing* (MP) algorithm. Usually all iterative decoding algorithms are known as message passing algorithms.

Decoding of LDPC codes has a crucial roll in the error performance of the LDPC

codes. It has been shown that long LDPC codes using soft decision iterative decoding can perform within 0.0045 dB of the Shannon limit in AWGN channels. It is worth to mention that the complexity of iterative decoding algorithms grow linearly with the code block length.

2.5.1 Iterative Decoding Algorithms

In the decoding procedure, we are interested in finding the a posteriori probability $Pr(c_i = 1|\mathbf{r})$ of a specific bit (c_i) in the codeword being equal to $1(c_i = 1)$ given the received vector $\mathbf{r}$.

The APP can be written as a posteriori probability ratio or *Likelihood Ratio* (LR) which is defined as,

$$l(c_i) \triangleq \frac{Pr(c_i = 0|\mathbf{r})}{Pr(c_i = 1|\mathbf{r})} \tag{2.16}$$

In some implementations of iterative algorithms *Log Likelihood Ratio* (LLR) is utilized which is computationally more efficient and is presented as

$$L(c_i) \triangleq \log\left(\frac{Pr(c_i = 0|\mathbf{r})}{Pr(c_i = 1|\mathbf{r})}\right). \tag{2.17}$$

where log is natural base logarithm.

The message passing algorithm for computing each of the APP, LR or LLR is an iterative algorithm based on the Tanner graph of the code. In each iteration the channel observation (intrinsic information) and the information from the previous iteration received from the adjacent neighbours (extrinsic information) are used to get a better estimation of the transmitted message. To represent this the LDPC code

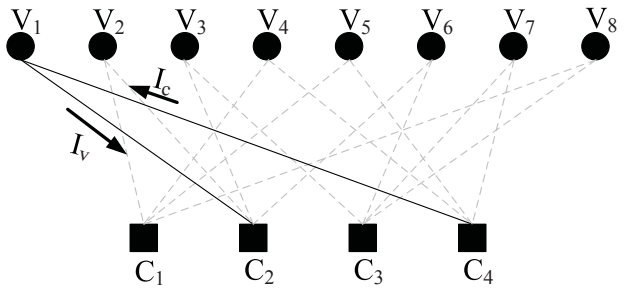


Figure 2.4: A snap shot of information flow during the message updating procedure of Iterative decoding

of Fig. 2.2 is considered and a snap shot of the information flow in its Tanner graph as illustrated in Fig. 2.4.

All of the iterative decoding algorithms share the same procedure. The only difference between those procedures is the method of computation in the check nodes.

Generally, iterative decoding algorithms follow 4 steps. Algorithms start with an initialization step, where an initial or local message is assigned to the variable nodes. The initial value is computed based on the channel observation. After the initialization step, the messages (I_v) are passed from variable nodes to check nodes through edges of the Tanner graph as shown in Fig.2.4 . In the third step, the information received from the variable nodes are processed in the check nodes based on the decoding algorithm and the new calculated information (I_c) is sent back to the variable nodes. In the fourth step the variable nodes update their states based on the current value and the received information from the check nodes. At this step, based on the new value of variable nodes, the algorithm estimates the codeword from the probabilistic information and makes a hard decision. The algorithm stops if $\widehat{\mathbf{c}}\mathbf{H}^T = \mathbf{0}$ or the maximum number of iterations reached, otherwise the algorithm continues and jumps to step 2.

Depending on the decoding algorithm and type of messages, specific computations are executed in the check nodes and variable nodes of the Tanner graph. The Sum-Product (SP) algorithm is the best performing while its complexity is high for digital decoders. The SP algorithm works well when short cycles are not present in the Tanner graph and it is optimal when the TG is cycle-free. There are some approximations for the SP algorithm such as the Min-Sum (MS) algorithm and the *Min-sum with correction factor* algorithm which will be discussed in Chapter 4. The complexity of MS algorithms is less for digital decoders compared to the SP algorithm but they have nearly the same performance. It is believed that MS algorithms are preferable in analog decoders as well since in this algorithm transistors can be designed to work in any operation region while in the SP algorithm transistors are designed to work only in weak inversion. The SP algorithm is discussed in detail in Chapter 3.

2.6 Summary

In this chapter the necessary background for LDPC codes is discussed. Characteristics, different type of representations, construction methods, encoding and decoding algorithms are presented. It is discussed that, iterative decoding algorithms are the best performing decoding algorithms for LDPC codes. Also, the general procedure of the iterative decoding is shown.

According to the knowledge provided in this chapter, it can be concluded that regular LDPC code is one of the most promising channel codes while accompanied by one of the soft decision iterative algorithms. However the construction and decoding of the LDPC code may be complicated. Therefore a method to construct a regular

structural LDPC code is presented in Chapter 4 and the matching decoder structure is proposed in Chapters 6-8.

Chapter 3

Analog Decoders based on the Sum-Product Algorithm

In this chapter the implementation of an LDPC analog decoder based on the Sum-Product (SP) algorithm is considered. The SP algorithm is the most promising decoding algorithm for LDPC codes in the sense of error performance but suffers from high complexity. The Sum-Product algorithm is presented in Section 3.1. The SP algorithm is discussed in detail using probability and Log Likelihood representations.

Basic circuit level concepts are reviewed. The necessary background required to design building modules of the SP algorithm is discussed. One of the main building blocks of the SP algorithm, variable (equality) node, is studied.

3.1 Sum-Product Algorithm

As it is already discussed the decoding algorithm can be performed based on one of the three different message formats, APP, LR or LLR. Due to implementation

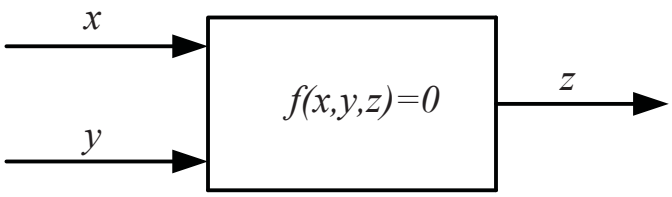


Figure 3.1: Block diagram of a sample variable (equality) node/check node

limitations, variable (equality) nodes and check nodes are designed with two inputs and one output. Therefore nodes with higher inputs, can be realized by cascading two-input blocks. Fig. 3.1 shows a 3-edge node with two inputs and one output while the local constraint is $f(x, y, z) = 0$.

Assume P_0 and P_1 are the probabilities of zero and one for every edge of the block, respectively. In the probability format, the block works as a variable (equality) node if,

$$P_{z0} = \eta P_{x0} P_{y0} \tag{3.1}$$

and

$$P_{z1} = \eta P_{x1} P_{y1} \tag{3.2}$$

where η is a positive correction factor for which $\eta = P_{z0} + P_{z1}$. Therefore $\eta = \frac{1}{P_{x0}P_{y0}+P_{x1}P_{y1}}$.

For the check node, we have

$$P_{z0} = P_{x0} P_{y0} + P_{x1} P_{y1} \tag{3.3}$$

and

$$P_{z1} = P_{x1} P_{y0} + P_{x0} P_{y1}. \tag{3.4}$$

For LR representation, $\frac{P_0}{P_1}$ is considered. Using equations 3.1 to 3.4, the variable (equality) node and check node have the following outputs respectively,

$$Y_z = Y_x \cdot Y_y \tag{3.5}$$

and

$$Y_z = \frac{1 + Y_x \cdot Y_y}{Y_x + Y_y}. \tag{3.6}$$

Similarly to represent the variable in LLR format, $\ln\left(\frac{P_0}{P_1}\right)$ is considered. Therefore the outputs of variable (equality) node and check node are,

$$X_z = X_x + X_y \tag{3.7}$$

and

$$X_z = 2\tanh^{-1}\left(\tanh\left(\frac{X_x}{2}\right) \cdot \tanh\left(\frac{X_y}{2}\right)\right). \tag{3.8}$$

In the following, the SP algorithm using LLR representation is explained. To have an understanding of the SP algorithm with probability representation, the updating rules are presented in the probability domain.

3.1.1 Sum-Product Algorithm with LLR Representation

Before explaining the SP decoding algorithm, notations used in the decoding procedure are presented.

- $r_{ji}(b)$ is the message that is sent from the check node c_j to variable node v_i. $r_{ji}(0)$ and $r_{ji}(1)$ are representing the probabilities of receiving 0 or 1 respectively.

- $q_{ij}(b)$ is the message that is sent from the variable node v_i to check node c_j. $q_{ij}(0)$ and $q_{ij}(1)$ are representing the probabilities of receiving 0 or 1 respectively.
- V_j is the set of variable nodes connected to check node j.
- $V_{j\backslash i}$ is the set of variable nodes connected to check node j excluding variable node i.
- C_i is the set check nodes connected to variable node i.
- $C_{i\backslash j}$ is the set of check nodes connected to variable node i excluding check node j.

Consider BPSK modulation where +1 is sent representing binary bit 0 and -1 is sent representing binary bit 1. It should be recalled that,

$$\begin{aligned} L(c_i) &= \log\left(\frac{Pr(x_i = +1|y_i)}{Pr(x_i = -1|y_i)}\right) \\ L(r_{ji}) &= \log\left(\frac{r_{ji}(0)}{r_{ji}(1)}\right) \\ L(q_{ij}) &= \log\left(\frac{q_{ij}(0)}{q_{ij}(1)}\right) \\ L(Q_i) &= \log\left(\frac{Q_{ij}(0)}{Q_{ij}(1)}\right). \end{aligned} \tag{3.9}$$

Following steps are performed for Sum-Product algorithm in Log Likelihood domain [3], [17]:

Step 1: At the initialization step, each variable node sends out an LLR value along its edges based on the channel observation. Assuming BPSK modulation and AWGN channel,

$$L(q_{ij}) = L(c_i) = \frac{2y_i}{\sigma^2} \tag{3.10}$$

where y_i is the channel observation i and σ^2 is the AWG noise variance.

Step 2: Check nodes compute their response message according to the check node updating rule,

$$L(r_{ji}) = \left[\prod_{\acute{i}\in V_{j\setminus i}} \alpha_{\acute{i}j}\right] \cdot \phi\left[\sum_{\acute{i}\in V_{j\setminus i}} \phi(\beta_{\acute{i}j})\right], \tag{3.11}$$

where

$$\begin{aligned} \alpha_{ij} &= sign(L(q_{ij})) \quad \& \quad \beta_{ij} = |L(q_{ij})| \\ \phi(x) &= -\log\left[\tanh(\frac{x}{2})\right] = \log\frac{e^x+1}{e^x-1} \end{aligned} \tag{3.12}$$

and *log* is the natural base logarithm.

Step 3: Variable nodes update their response message to check nodes according to variable node updating rule.

$$L(q_{ij}) = L(c_i) + \sum_{\acute{j}\in C_{i\setminus j}} L(r_{\acute{j}i}) \tag{3.13}$$

Step 4: At this stage all of the variable nodes update their current state based on,

$$L(Q_i) = L(c_i) + \sum_{j\in C_i} L(r_{ji}). \tag{3.14}$$

Now to obtain the estimated codeword a hard decision is performed for each variable node,

$$\forall i, \quad \hat{c}_i = \begin{cases} 1 & \text{if} \quad L(Q_i) < 0 \\ 0 & \text{else} \end{cases} \tag{3.15}$$

At this point if the estimated codeword satisfies the parity check matrix ($\hat{\mathbf{c}}_i\mathbf{H}^T = \mathbf{0}$) or maximum number of iterations reached, then the algorithm stops. Otherwise the algorithms goes back to step 2 and next iteration starts.

3.1.2 Sum-Product Algorithm with probability Representation

In this part, the updating rules of SP algorithm described in the previous sub-section are presented in the probability domain.

All the definitions previously provided stand for this sub-section. The decoding procedure is the same as the one described for SP algorithm with LLR representation. The only difference is the updating rule where instead of using LLR values, two probabilities representing the probability of the value being 0 or 1 are utilized. Consequently the decoding procedure is,

Step1:

$$\begin{aligned} q_{ij}(0) &= 1 - P_i = Pr(x_i = +1|y_i) = \frac{1}{1 + e^{-\frac{2y_i}{\sigma^2}}} \\ q_{ij}(1) &= P_i = Pr(x_i = -1|y_i) = \frac{1}{1 + e^{\frac{2y_i}{\sigma^2}}} \end{aligned} \tag{3.16}$$

Step2:

$$\begin{aligned} r_{ji}(0) &= \frac{1}{2} + \frac{1}{2} \prod_{\acute{i} \in V_{j \setminus i}} \left(1 - 2q_{\acute{i}j(1)}\right) \\ r_{ji}(1) &= 1 - r_{ji}(0) \end{aligned} \tag{3.17}$$

Step3:

$$\begin{aligned} q_{ij}(0) &= K_{ij}(1-P_i)\prod_{\acute{j}\in C_{i\setminus j}} r_{\acute{j}i}(0) \\ q_{ij}(1) &= K_{ij}P_i\prod_{\acute{j}\in C_{i\setminus j}} r_{\acute{j}i}(1) \end{aligned} \tag{3.18}$$

The normalization constant K_{ij} are chosen in order to have $q_{ij}(0)+q_{ij}(1)=1$.

Step4:

$$\begin{aligned} Q_i(0) &= K_i(1-P_i)\prod_{j\in C_i} r_{\acute{j}i}(0) \\ Q_i(1) &= K_iP_i\prod_{j\in C_i} r_{\acute{j}i}(1) \end{aligned} \tag{3.19}$$

The normalization constant K_i are chosen in order to have $Q_i(0)+Q_i(1)=1$.

Now to obtain the estimated codeword, a hard decision is performed or each variable node,

$$\forall i, \quad \hat{c}_i = \begin{cases} 1 & \text{if} \quad Q_i(1) > 0.5 \\ 0 & \text{else} \end{cases} \tag{3.20}$$

At this point if the estimated codeword satisfies the parity check matrix ($\hat{\mathbf{c}}_i\mathbf{H}^T = \mathbf{0}$) or maximum number of iterations reached, then the algorithm stops. Otherwise the algorithm goes back to step 2 and the next iteration starts.

3.2 Basic Circuits for Sum-Product Analog Decoding

In this section, implementation of LDPC decoder using analog VLSI circuits is briefly discussed. In the literature LDPC decoders utilizing primarily the SP algorithm are implemented, therefore in this section the necessary circuitry to implement the SP algorithm is presented. Analog implementation of LDPC codes are designed based on the Tanner Graph of the code. Therefore, variable (equality) nodes and check nodes are the basic blocks of any analog implementation of an LDPC decoder. In order to have a well performing LDPC decoder, the speed and accuracy of the basic blocks should be as high as possible. In this section, some essential knowledge required to design the basic blocks is presented.

3.2.1 MOSFET Devices Operation Region

A three terminal *Metal Oxide Semiconductor Field Effect Transistor* (MOSFET) transistor is shown in Fig. 3.2. A MOSFET can operate in three different regions called, *strong inversion*, *moderate inversion* and *weak inversion* or *sub-threshold* inversion regions.

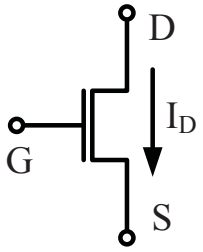


Figure 3.2: A three terminal MOSFET

A MOSFET is in strong inversion if the gate-source voltage is greater than the threshold voltage by some margin, in other words, $V_{GS} > V_{th} + 100mV$. In this case a channel between the drain and source is made and the transistor is on. The transistor is working in *saturation* if $V_{DS} > V_{GS} - V_{th}$. If a transistor is on and working in the saturation region, then the drain-source current can be found in from the,

$$I_D = \frac{1}{2}\mu C_{OX}\frac{W}{L}(V_{GS} - V_{th})^2(1 + \lambda V_{DS}) \tag{3.21}$$

where μ is the mobility of carriers, C_{OX} is the oxide capacitance, λ is the channel length modulation parameter, W and L are the width and length of the channel respectively. In many cases the channel length effect is neglected, therefore $\lambda = 0$. If $V_{DS} < V_{GS} - V_{th}$ then the transistor is not in the saturation region anymore and the transistor is working in a region called *triode*. In the triode region the transistor works as a voltage controlled resistor which is linear for small V_{DS}. If a MOS transistor works in the triode region then the drain current can be obtained by,

$$I_D = \mu C_{OX}\frac{W}{L}\left[(V_{GS} - V_{th})V_{DS} - \frac{V_{DS}^2}{2}\right]. \tag{3.22}$$

If the gate-source voltage is less than the threshold, then the transistor works either in moderate inversion or weak inversion. Moderate and weak inversion working regions are distinguished by the transistor current as,

$$If \begin{cases} I < \frac{1}{10}I_S & \text{Transistor is in weak inversion} \\ \frac{1}{10}I_S < I < 10I_S & \text{Transistor is in moderate inversion} \end{cases} \tag{3.23}$$

where I is the device's current and I_S is the device's *specific current*. I_S is defined as,

$$I_S = \frac{2\mu C_{OX} V_T^2 W}{\kappa L} \cdot exp\left(-\frac{\kappa V_{T0}}{V_T}\right) \tag{3.24}$$

where $V_T \approx 25mV$ is the thermal voltage, κ and V_{T0} are constants of the fabrication process. It should be noted that $0.5 < \kappa < 0.99$ and typically is 0.7. Depending on the fabrication process $0.3V < V_{T0} < 1V$ [39].

There is another definition which identifies different inversion regions based on the gate-source voltage such as [69],

$$If \begin{cases} V_{GS} - V_{th} < -100mV & \text{Transistor is in weak inversion} \\ -100mV < V_{GS} - V_{th} < 100mV & \text{Transistor is in moderate inversion} \\ V_{GS} - V_{th} > 100mV & \text{Transistor is in strong inversion} \end{cases} \cdot \tag{3.25}$$

We can conclude that if $V_{GS} \ll V_{th}$, then the transistor is deeply in weak inversion and the current is resultant of the diffusion. If $V_{GS} \approx V_{th}$, then transistor is in moderate inversion. It should be mentioned that there is no exact boundary for weak and moderate inversion regions.

If the transistor is in weak inversion (subthreshold), then the drain current follows the following equation [39],

$$I_D \approx I_0 exp\left(\frac{\kappa \cdot V_{GS}}{V_T}\right)\left(1 - exp\left(-\frac{\kappa \cdot V_{DS}}{V_T}\right)\right) \tag{3.26}$$

where I_0 is a small constant current based on I_S. In equation 3.26, n is the *subthreshold slope factor* and is defined as $n = \frac{C_{OX}+C_{dep}}{C_{OX}}$ where C_{dep} is representing the depletion capacitance.

When V_{DS} is large enough ($V_{DS} > 4V_T \approx 100mV$), the transistor is in saturation and equation 3.26 reduces to,

$$I_D \approx I_0 exp\left(\frac{\kappa \cdot V_{GS}}{nV_T}\right). \tag{3.27}$$

For small V_{DS} the transistor is not in saturation anymore and works in a region called *unsaturated* weak inversion. Equation 3.26 can be rewritten as,

$$\begin{aligned} I_D &\approx I_0 exp\left(\frac{\kappa \cdot V_{GS}}{nV_T}\right)\left(1 - exp\left(-\frac{\kappa \cdot V_{DS}}{nV_T}\right)\right) \\ &= I_0\left(exp\left(\frac{\kappa \cdot V_{GS}}{nV_T}\right) - exp\left(\frac{\kappa \cdot V_{GS}}{nV_T}\right) exp\left(-\frac{\kappa \cdot V_{DS}}{nV_T}\right)\right) \\ &= I_0\left(exp\left(\frac{\kappa \cdot V_{GS}}{nV_T}\right) - exp\left(-\frac{\kappa \cdot (V_{GS} - V_{DS})}{nV_T}\right)\right) \\ &= I_0\left(exp\left(\frac{\kappa \cdot V_{GS}}{nV_T}\right) - exp\left(-\frac{\kappa \cdot (V_{GD})}{nV_T}\right)\right) \\ &= I_f - I_r \end{aligned} \tag{3.28}$$

Therefore, in this case I_D has two components, one is the forward current which is the desired one and the other is the reverse current.

3.2.2 Translinear Principle

Translinear devices are used in designing the basic blocks of decoders utilizing the SP algorithm. The translinear principle uses the non-linear exponential relationship between the current and the voltage in the semiconductor devices. The translinear principle was originally defined for Bipolar transistors, but has been extended to MOS transistors biased in the subthreshold region [70], where the device's current follows

equation 3.27. Translinear devices can be used to perform multiplication since they can be arranged in *translinear loops*. A translinear loop is a Kirchhoff voltage loop which contains the V_{GS} of an even number of translinear devices. To be precise, there needs to be an equal number of voltage rises and voltage drops. It should be noted that there is a voltage-translinear principle for the case that all the saturated MOS transistors are biased in strong inversion [70]. Fig. 3.3 shows a simple translinear loop. By following the closed loop in the direction shown and writing *Kirchhoff's Voltage Law* (KVL), we have

$$-V_{GS1} + V_{GS2} - V_{GS3} + V_{GS4} = 0 \tag{3.29}$$

Consider the case that all of the transistors are biased in saturated-weak inversion, then using equation 3.27 we can rewrite equation 3.29 as,

$$-\frac{nV_T}{\kappa}\ln\left(\frac{I_1}{I_0}\right) + \frac{nV_T}{\kappa}\ln\left(\frac{I_2}{I_0}\right) - \frac{nV_T}{\kappa}\ln\left(\frac{I_3}{I_0}\right) + \frac{nV_T}{\kappa}\ln\left(\frac{I_1}{I_0}\right) = 0 \tag{3.30}$$

Therefore,

$$I_1 I_3 = I_2 I_4 \tag{3.31}$$

Equation 3.31 shows the translinear principle in a translinear loop with translinear MOS devices. Currents which are in the direction of loop is called *clockwise currents*, otherwise they are called *counter-clockwise currents*. In the Fig. 3.3, I_2 and I_4 are clockwise currents and I_1 and I_3 are counter-clockwise currents. Based on this definition, the translinear principle is: In a translinear loop, the product of clockwise currents is equal to the product of counter-clockwise currents.

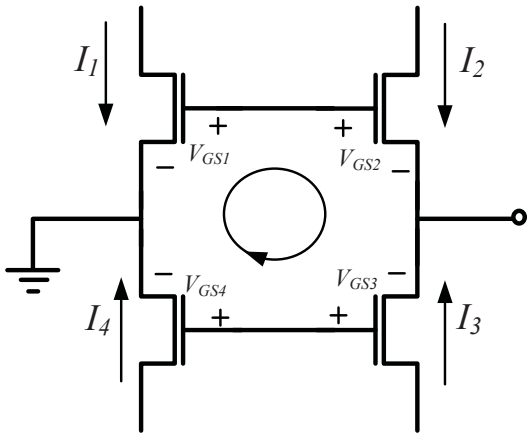


Figure 3.3: A simple translinear loop

3.2.3 The Canonical Sum-Product Modules

Analog decoders based on the SP algorithm use the probability representation of the variables [71, 72]. Each signal represents the probability of a variable being either 1 or 0. The structures of the canonical variable (equality) node and check node are illustrated in Figs. 3.4 and 3.5, respectively. Each circuit has two inputs and one output which are in the current mode. For each of the inputs and the output, there are two terminals representing probabilities of 0 and 1. Therefore there are a total of 4 input and 2 output terminals.

It can be observed that the circuits depicted in Figs. 3.4 and 3.5 have been designed based on Gilbert cells [71,72] which contain several translinear loops. Based on the translinear principle the currents I_{Z0} and I_{Z1} shown in Fig. 3.4 can be obtained by,

$$I_{Z0} = \frac{I_{x0} I_{y0}}{I_{x0} I_{x1}} \tag{3.32}$$

and

$$I_{Z1} = \frac{I_{x1} I_{y1}}{I_{x0} I_{x1}}. \tag{3.33}$$

In the same way, I_{Z0} and I_{Z1} of the check node are as follows,

$$I_{Z0} = \frac{I_{x0}I_{y0} + I_{x1}I_{y1}}{I_{x0}I_{x1}} \tag{3.34}$$

$$I_{Z1} = \frac{I_{x1}I_{y0} + I_{x0}I_{y1}}{I_{x0}I_{x1}} \tag{3.35}$$

At the top of each module, a renormalization circuit is used. Since the renormalization circuit is a translinear circuit, the outputs of the variable (equality) nodes and check nodes can be derived using the following equations,

$$I_{out0} = \frac{I_{z0}I_U}{I_{z0} + I_{z1}} \tag{3.36}$$

$$I_{out1} = \frac{I_{z1}I_U}{I_{z0} + I_{z1}}. \tag{3.37}$$

I_U is globally used in all variable (equality) nodes and check nodes to boost the attenuated I_{z0} and I_{z1}. This makes it possible to calibrate the output current and change it from 0 to I_U (0 for zero output current and I_U for the maximum output current). The renormalization is referred to this calibration. The output currents of the variable (equality) and check nodes based on the input currents can be found by substituting equations 3.32 to 3.37 and assuming $I_{x0} + I_{x1} = I_{y0} + I_{y1} = I_U$. The output currents of the module can be obtained by,

$$I_{out0} = \frac{I_{x0}I_{y0}}{I_{x0}I_{y0} + I_{x1}I_{y1}}I_U \tag{3.38}$$

$$I_{out1} = \frac{I_{x1}I_{y1}}{I_{x0}I_{y0} + I_{x1}I_{y1}}I_U \tag{3.39}$$

and

$$I_{out0} = \frac{I_{x0}I_{y0} + I_{x1}I_{y1}}{I_U} \tag{3.40}$$

$$I_{out1} = \frac{I_{x1}I_{y0} + I_{x0}I_{y1}}{I_U}. \tag{3.41}$$

The essential condition to be able to describe the behaviour of the circuits shown in Figs. 3.4 and 3.5 with equations 3.38 to 3.41, is that the transistors should work in weak inversion and should be in the saturation region as well [39]. Therefore, two voltages $V_{ref}(N)$ and $V_{ref}(P)$ are used to satisfy these conditions. These two voltages are used to keep transistors M_1 and M_{15} in saturation. The drain voltage of M_1 can be controlled by utilizing $V_{ref}(N)$ and in the same way the the drain voltage of M_{15} can be controlled by $V_{ref}(p)$.

In [39], Winstead showed that the minimum required bias voltage for the variable (equality) and check nodes to work properly can be obtained using the following equation.

$$V_{DD} \geqslant 0.42V + V_{ref} + V_{TOP} + \frac{V_T}{\kappa} \ln\left(\frac{I_U}{100nA}\right) \tag{3.42}$$

where $V_{ref} = V_{DD} - V_{ref}(P)$, V_T is the thermal voltage, κ and V_{TOP} are process dependent parameters. Equation 3.42 was formulated in $0.18\mu m$ technology. It was discussed that for this technology $V_{TOP} = 0.39V$. To satisfy the saturation condition for all of the transistors, we should have $V_{DS} \geqslant 4V_T$.

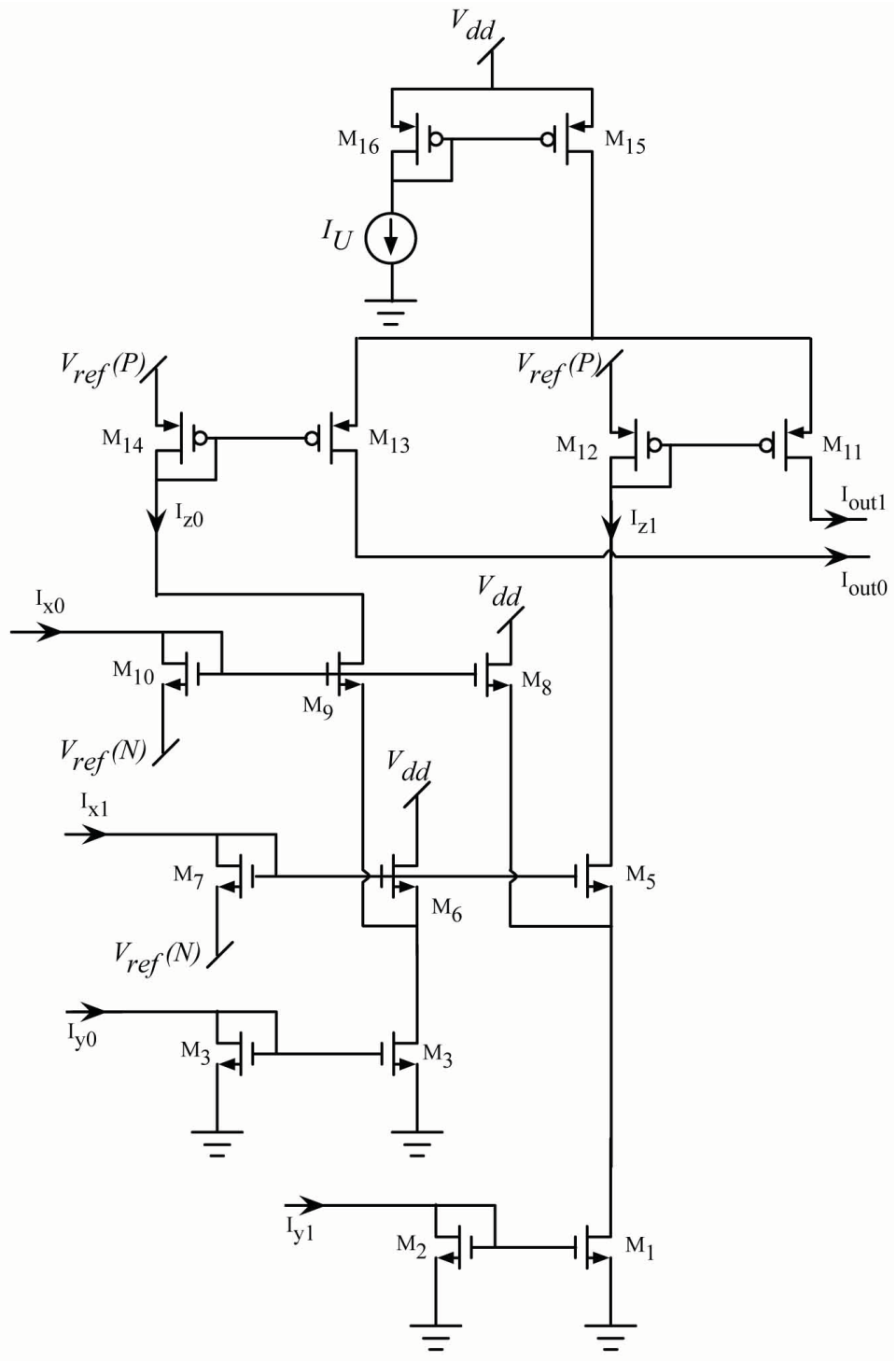


Figure 3.4: A standard variable (equality) node design

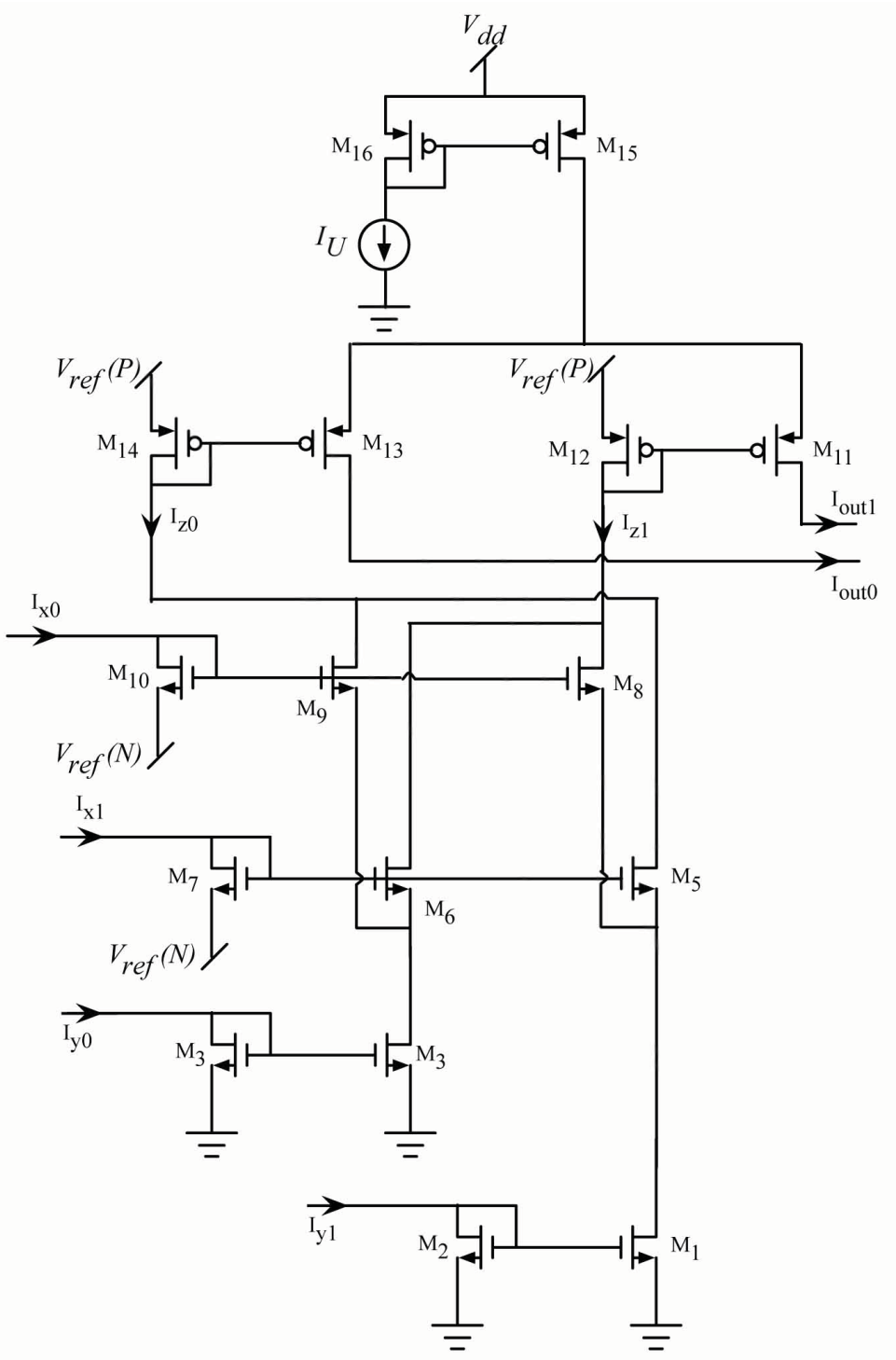


Figure 3.5: A standard check node design

3.3 Summary

In this chapter, the Sum-Product algorithm is discussed in Log-Likelihood-Radio and probability presentation. Some analog VLSI circuit background has been presented. It is described that the implementation of LDPC decoder in both digital and analog realization is based on Tanner graphs. Two major building blocks for implementing LDPC decoders are represented, which are variable (equality) nodes and check nodes. It has been discussed that the variable (equality) and check nodes work based on translinear principle and Gilbert multiplier concept. The design of these two basic block has been illustrated and discussed.

Chapter 4

Performance Evaluation of TS-LDPC Codes

In this chapter we consider a class of structured regular LDPC codes, called Turbo-structured LDPC (TS-LDPC) codes. There are many advantages of TS-LDPC codes as compared with other kinds of LDPC codes.

The most important advantage of TS-LDPC codes is that they can be designed to have a large girth which is a crucial parameter in designing LDPC codes. The iterative decoding procedure of LDPC codes is optimal when a cycle-free graph is used (graph with the shortest cycle being of infinite length). Therefore, increasing the length of the shortest cycle (girth) in a graph improves the effectiveness of iterative decoding. Thus, TS-LDPC codes with inherently large girths would enjoy a more efficient iterative decoding compared to other types of LDPC codes. Moreover, as shown by Tanner, large girth guarantees large minimum Hamming distance (d_{min}) of the code, resulting in lower error floor at high SNR. The simulation results show that TS-LDPC codes outperform randomly generated LDPC codes at higher SNR and

have much lower error floor [15,73]. Therefore, TS-LDPC codes are a good candidate to be adopted in most communication applications.

Due to above mentioned advantages of TS-LDPC codes and the MS algorithm, we have adapted the MS algorithm in the decoding of TS-LDPC codes and evaluated the decoding performance to verify that the property of low error floor is maintained when the MS algorithms are used. We have designed a number of examples of TS-LDPC codes and used SP, MS and MS with correction factor decoding algorithms to evaluate the error performance of the TS-LDPC codes. It is shown that TS-LDPC codes with MS algorithms are well suited for analog implementations due to their performance and complexity.

In Section 4.1 of this chapter, TS-DPC codes are explained and a sample TS-LDPC code is designed. Section 4.2 describes Min-Sum algorithm adopted to be used in analog decoder. Min-Sum with correction factor algorithm which is a variation of Min-Sum algorithm is reviewed. In Section 4.3 for the fist time MS and MS with correction factor decoding algorithms have been applied to sample TS-LDPC codes. Simulation results for error performance have been presented in this section. Finally, Section 4.4 provides the conclusion.

4.1 TS-LDPC Codes

In this section we briefly describe TS-LDPC codes. The Tanner graph of a TS-LDPC code is shown in Fig. 4.1 and consists of three parts; two height balanced sub-trees T_U and T_L and an interleaver. Sub-tree T_U is the upper sub-tree whose leaves are variable nodes (circles) and T_L is the lower sub-tree whose leaves are check nodes (squares). The number of tiers in each sub-tree (T_U or T_L) is called the height of the

sub-tree and represented by h. The upper sub-tree starts with a check node as its root and in the same way the root of the lower sub-tree is always a variable node. The leaf nodes of T_U are connected to the leaf nodes of T_L in the turbo-like manner, through the interleaver. In the design of TS-LDPC codes, structured regular LDPC codes are targeted. Therefore all the variable nodes have the same degree j and all the check nodes have the same degree k. As it is shown in Fig. 4.1, by the dashed line, the root of the lower sub-tree (V*) is connected to the root of the upper sub-tree (C*). This guarantees the same degree for all the variable nodes and check nodes and consequently the constructed LDPC code is regular. The height of the sub-trees h or simply the number of tiers in both sub-trees, should be even. This guarantees that the leaf nodes of the upper sub-tree are variable nodes and the leaf nodes of the lower sub-tree are check nodes. Fig. 4.1 shows an example of a TS-LDPC code where $j = 3$, $k = 4$ and $h = 4$. It can be shown that the code rate for LDPC codes with full rank parity check matrix $\mathbf{H}$ is $r = 1 - \frac{j}{k}$. Therefore to design a code with a specific code rate r, the degree of variable nodes j and the degree of check nodes k should be chosen carefully. For example, in Fig. 4.1, we have $j = 3$ and $k = 4$, hence the expected code rate is $\frac{1}{4}$. In the selection of j and k, implementation limitations should be considered. Designing high degree variable or check nodes increases the implementation complexity in VLSI circuity.

One of the main advantages of TS-LDPC is the capability of designing codes with large girth. As shown in Fig. 4.1, there is no cycle in each of the sub-trees in isolation. Cycles are produced by the introduction of the interleaver. Therefore a careful design of the interleaver is crucial for the performance of the code.

Due to the special structure of TS-LDPC codes there are two types of cycles. Type

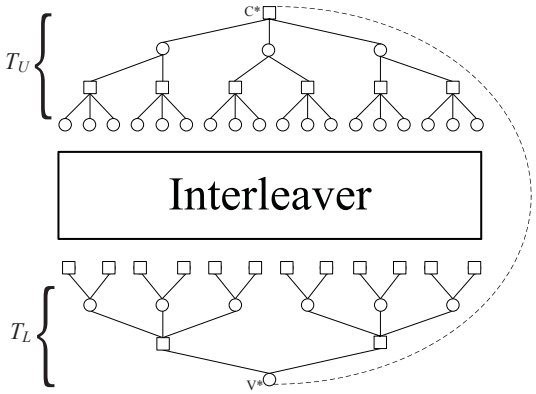


Figure 4.1: The general form of TS-LDPC

I cycles which start in either sub-tree and can be terminated in the interleaver or the complement sub-tree. Type II cycles are those which start in the interleaver and end in the interleaver. In type II cycles, it is possible that the cycle passes through each of the sub-trees and ends again in the interleaver.

The interleaver for TS-LDPC codes is not a one to one mapping. For example in Fig. 4.1, it can be observed that each leaf node (variable node) in the last tier of the upper sub-tree should be connected to $j - 1 = 2$ leaf nodes (check nodes) in the last tier of the lower sub-tree. Therefore conventional techniques cannot be used in the design of the interleaver. Since in conventional interleavers the same number of inputs are connected randomly to the same number of outputs. This random connections assure that two adjacent inputs are connected to two outputs as far as possible. Whereas, in a TS-LDPC interleaver the number of inputs and outputs are different. Moreover, the connection are made to increase the length of closed loops.

To design the interleaver for TS-LDPC codes, auxiliary nodes are introduced for both sub-trees such that the resultant interleaver has one to one connections. Fig. 4.2 shows the auxiliary nodes for the upper and lower sub-tress of a TS-LDPC code

with $j = 3$, $k = 4$ and $h = 4$. As it can be observed the auxiliary nodes are added to the last tier of the sub-tree. Since each variable node is connected to $j - 1$ leaf nodes of the lower sub-tree, $j - 1$ auxiliary nodes should be defined for each variable node. In the same way each of the leaf nodes in the last tier of the lower sub-tree should be connected to $k - 1$ auxiliary nodes.

After the introduction of auxiliary nodes and having a one to one interleaver, the connection should be designed to achieve the specific girth. The following steps should be taken to remove type I and type II cycles for the defined girth. In the following paragraphs we briefly introduce the steps which are required for the design of the interleaver [14, 15, 73, 74].

Auxiliary nodes in both sub-trees are numbered from zero to $[(k-1)(j-1)]^{\frac{h}{2}} - 1$ from left to right. The indices of the upper sub-tree are represented in X_{p-q} format while in the lower sub-tree the indices are shown in X_{q-p} format. Each index contains h digits as its coordinates. The digits take values up to p and q values alternatively where $p = k - 1$ and $q = j - 1$. For a TS-LDPC code having $h = 4$,

$$X_{p-q} = a_1 a_2 a_3 a_4 = (a_1 \times pq^2 + a_2 \times pq + a_3 \times q + a_4) \tag{4.1}$$

which $0 \leqslant a_1 \leqslant p - 1$, $0 \leqslant a_2 \leqslant q - 1$, $0 \leqslant a_3 \leqslant p - 1$ and $0 \leqslant a_4 \leqslant q - 1$. Each auxiliary node in the upper sub-tree with X_{p-q} index is connected to the auxiliary node in the lower sub-tree having X_{q-p} index. X_{q-p} is the symbol-wise reversal of X_{p-q} defined as, $\pi_s(a_1 a_2 a_3 a_4) = a_4 a_3 a_2 a_1$.

To remove any type I cycles each auxiliary node having X_{p-q} decimal index should be connected to its correspondent X_{q-p} index.

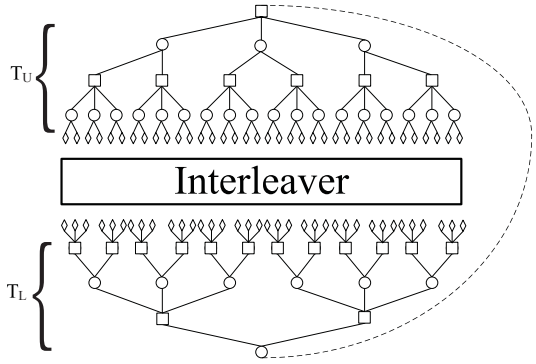


Figure 4.2: Auxiliary nodes in a TS-LDPC code with $j = 3$, $k = 4$ and $h = 4$

$$X_{p-q} \longrightarrow X_{q-p} = \pi_s(X_{p-q}) \tag{4.2}$$

For our TS-LDPC example design of Fig. 4.2 with $j = 3$, $k = 4$ and $h = 4$, there are 36 auxiliary nodes in each sub-tree. Using the index assignment described in the previous paragraph, e.g. auxiliary node 0 in T_U is connected to auxiliary node 0 in T_L, node 1 in T_U is connected to 18 in T_L and so on. This connection scheme guarantees that the length of any type I cycle is not less than $2h$ [14]. To maximize the length of type II cycles, auxiliary nodes in T_U and T_L are partitioned into G_U and G_L groups respectively. To ensure that the length of any type II cycle is not less than g, the required number of groups in T_U and T_L are,

$$\begin{aligned} G_U &\geqslant [(k-1)(j-1)]^{\lfloor \frac{(g-2)/4}{2} \rfloor}(k-1)^{\lfloor \frac{(g-2)}{2} \rfloor} \\ G_L &\geqslant [(k-1)(j-1)]^{\lfloor \frac{(g-2)/4}{2} \rfloor}(j-1)^{\lfloor \frac{(g-2)}{2} \rfloor}. \end{aligned} \tag{4.3}$$

Each auxiliary node of T_U in group α is connected to an auxiliary node of T_L in group β with the shift value of $S_{\alpha,\beta}$. This means each auxiliary node in T_U with

X_{p-q} index should be connected to the auxiliary node in T_L with $\pi_s(X_{p-q})\dot{+}S_{q-p(\alpha,\beta)}$ indexing. The summation is a digit-wise addition. Let $\pi_s(X_{p-q}) = a_1a_2a_3a_4$ and $S_{\alpha,\beta} = s_1s_2s_3s_4$, then

$$\pi_s(X_{p-q})\dot{+}S_{\alpha,\beta} = y_1y_2y_3y_4 \tag{4.4}$$

where $y_i = mod(a_i + s_i, d_i)$, which $d_i = p$ for even i and $d_i = q$ for odd i. Shifted values ($S_{\alpha,\beta}$) are elements of a G_U by G_L matrix which is called shift matrix $\mathbf{S}$. For details of choosing $S_{\alpha,\beta}$ and consequently shift matrix $\mathbf{S}$ refer to [14, 15, 73, 74].

As an example, Fig. 4.3 shows the Tanner graph of a TS-LDPC code with $j = 3$, $k = 4$, $h = 4$, $N = 28$ and $g = 6$. The required number of groups in the upper tree $G_U = 3$ and in the lower tree $G_L = 2$. The thick dashed lines illustrate a type I cycle while thick solid lines depict a type II cycle. It is clear that there is no type I cycle less than $2h = 8$ and there is no type II cycle with length of less than 6 which is the girth.

It should be noted that the code block length N in TS-LDPC codes is a function of the column weight j (the degree of variable nodes), the row weight k (the degree of check nodes) and the girth g. For small girths [14]:

$$N = \frac{k\{[(k-1)(j-1)]^{\frac{(g-2)}{2}} - 1\}}{[(k-1)(j-1)] - 1}. \tag{4.5}$$

4.2 Min-Sum Algorithms

The Min-Sum algorithm [13, 18–21] is an approximation of the Belief Propagation or Sum-Product algorithm [13, 17, 18].

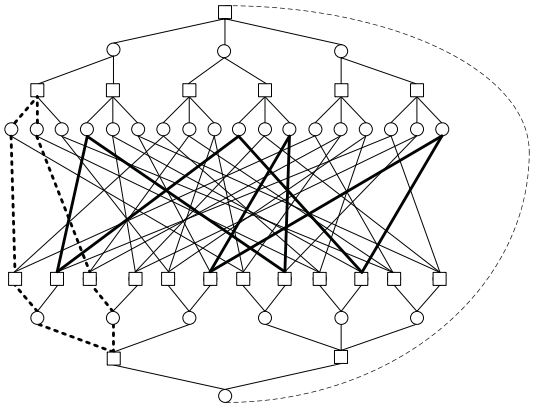


Figure 4.3: The Tanner graph of TS-LDPC code with $j = 3$, $k = 4$, $h = 4$, $N = 28$ and $g = 6$

Assume that the coded data block $\mathbf{c}$ is,

$$\mathbf{c} = [c_0, c_1, ..., c_N] \tag{4.6}$$

where N is the code block length. If BPSK modulation is utilized, each coded bit maps to ± 1. If $c_i = 0$ then $s_i = 1$ and if $c_i = 1$ then $s_i = -1$, where s_i is the symbol transmitted through the channel. Considering AWGN channel, at the input of the receiver we have $\mathbf{r} = \mathbf{s} + \mathbf{n}$, where $\mathbf{n}$ is the noise. Each sample of noise n_i has zero mean and variance σ_n^2. For a coded communication system $\sigma_n^2 = \frac{No}{2 \cdot R \cdot E_b}$, where $\frac{No}{2}$ is the power spectral density of AWGN, R is the code rate and E_b is the average energy per each information bit. It should be mentioned that MS algorithms work in the Log-Likelihood Ratio domain. A message along the edge connecting variable node i to check node j reflects the probability of c_i being "0" or "1" in the form of LLR.

Focusing on the decoding of bit c_i, the variable node associated with c_i is initially updated by M_i^0. For the described communication system, $M_i^0 = \frac{2r_i}{\sigma_n^2}$. Following the initialization step, three steps are performed.

Step 1: check nodes are updated using:

$$M_{CH_j \to V_i} = \left[\prod_{i' \in R_{j \setminus i}} sign(V_{i'} \to CH_j) \right] \cdot \min_{i' \in R_{j \setminus i}} \{ |V_{i'} \to CH_j| \} \tag{4.7}$$

where CH_j is the check node j, V_i is the variable node i and $R_{j \setminus i}$ is the set of all variable nodes connected to the check node j excluding the variable node i.

Step 2: After the check node updating step, variable nodes get updated using,

$$M_{V_i \to CH_j} = M^0_{V_i} + \sum_{j' \in C_{i \setminus j}} M_{CH_{j'} \to V_i} \tag{4.8}$$

where $C_{i \setminus j}$ is the set of all check nodes connected to the variable node i excluding check node j.

Step 3: At this point, variable nodes update their current estimate by utilizing,

$$M_{V_i} = M^0_{V_i} + \sum_{j \in C_i} M_{CH_j \to V_i} \tag{4.9}$$

where C_i is the set of all check nodes connected to variable node i. At this stage, we may use a hard decision to decide on the estimated coded bits (codewords).

$$\hat{c} = \begin{cases} 1 & \text{if} \quad M_{V_i} < 0, \\ 0 & \text{else.} \end{cases} \tag{4.10}$$

Now, if the estimated codeword satisfies the parity check equation ($\hat{\mathbf{c}}\mathbf{H}^T = \mathbf{0}$) or the maximum number of iterations are reached, then the algorithm stops, otherwise it goes to step 1.

In general using the MS algorithm for decoding results in a few tenth of a decibel

loss in error performance compared to the SP algorithm. Therefore some modifications have been proposed for the MS algorithm in order to make its performance as close as possible to the performance of SP algorithm [20], [75]. It has been shown that adding a correction term to the equation (4.7) in step 2 closes the gap between MS algorithm and SP algorithm. For the sake of illustration, equation (4.7) considering two-variable case can be rewritten as,

$$M_{CH_j \to V_i} = [sign(AB)] \cdot \min\{|A|, |B|\} + ln\left(\frac{1+\exp(-|A+B|)}{1+\exp(-|A-B|)}\right). \tag{4.11}$$

The correction term can be approximated by,

$$ln\left(\frac{1+\exp(-|A+B|)}{1+\exp(-|A-B|)}\right) \approx \begin{cases} c, & |A+B| < 2 \ \& \ |A-B| > 2|A+B| \\ -c, & |A-B| < 2 \ \& \ |A+B| > 2|A-B| \\ 0, & else \end{cases} \tag{4.12}$$

where c is a constant value in the order of 0.5.

4.3 Simulation Results

It was discussed before that TS-LDPC codes outperform other types of LDPC codes from performance and complexity points of view as long as the length of the code is the same. In this section, we simulate TS-LDPC codes and compare their error performance using SP, MS and MS with correction factor decoding algorithms. We have designed some examples of TS-LDPC codes with different girths. For the degree of variable node nodes $j = 3$, $k = 4$ and $r = \frac{1}{4}$, the code block lengths corresponding to the selected girths of 6, 8 and 10 are $n = 28$, $n = 172$ and $n = 1036$ respectively. Also $j = 3$ and $k = 8$ are considered to have a higher code rate $r = \frac{5}{8}$. Assuming

a girth of 6, the code block length is 120 bits. Fig. 4.3 shows the Tanner graph of TS-LDPC code with code block length 28 and girth 6. The girth of TS-LDPC was evaluated and confirmed using the algorithm proposed in [76].

The Tanner graph of TS-LDPC code can be modified to be used for the encoding process with linear complexity [14] [77]. However, in this work we use the systematic encoding where the generator matrix $\mathbf{G}$ of the code can be found by performing Gauss-Jordan elimination on the $\mathbf{H}$ matrix.

The simulations for $(N = 28, K = 7)$, $(172, 43)$, $(1036, 256)$ and $(120, 75)$ TS-LDPC codes using BPSK modulation scheme are performed. The signal to noise ratio was normalized using $SNR = 10\log[\frac{E_b}{2r\sigma_n^2}]$ where r is the code rate and $\sigma_n^2 = \frac{N_o}{2}$. The maximum number of 100 iterations was considered for the TS-LDPC decoder. Three different iterative decoding algorithms have been applied for the decoding. It should be noted that throughout this work each BER is calculated based on a minimum of 100 errors. Simulation results are illustrated in Figs. 4.4-4.7 using SP, MS and MS with correction factor algorithms for $N = 28$, $N = 172$, $N = 1036$ and $N = 120$, respectively. The constant c for the MS with correction factor is considered to be 0.5. As shown in Figs. 4.4-4.7 all three decoding algorithms perform approximately the same for $N = 28$ and $N = 120$, while for code block lengths of $N = 172$ and $N = 1036$ the SP algorithm outperformed the MS algorithm by about 0.1dB at a BER of 10^{-5}. However MS with correction factor algorithm performs the same as SP algorithm for $N = 172$ and $N = 1036$. It can be observed in Fig. 4.7 that for the higher rate TS-LDPC code with block length of 120 all three decoding algorithms result in the same error performance. This is consistent with other types of LDPC codes where by increasing block length of the code N, performance improves. Figures

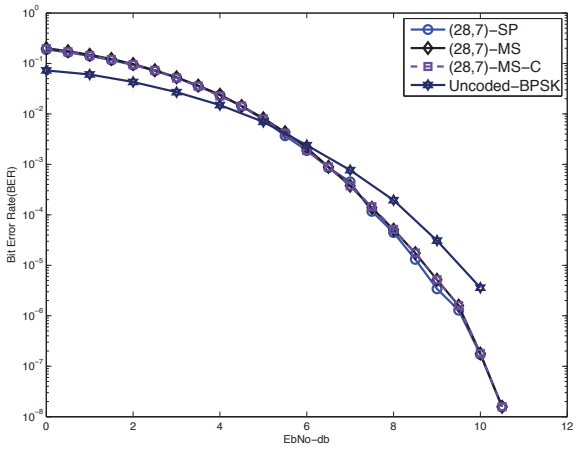


Figure 4.4: BER vs Eb/No(dB) for $(28, 7)$ TS-LDPC code with $g = 6$ using SP, MS and MS with correction factor decoding algorithms

also indicate that the error floors have not been achieved even at bit error rate of 10^{-8}. This demonstrates the suitability of MS algorithm in decoding of TS-LDPC codes.

Based on the above simulation results, it is clear that TS-LDPC codes can be decoded using the MS algorithm without any significant degradation in the performance. For long codes, degradation in performance using MS algorithm may become large and in that case MS with correction factor algorithm can be used with virtually identical performance to the SP algorithm.

4.4 Conclusion

In this work TS-LDPC codes were considered for analog VLSI implementation. It has been shown that this class of LDPC codes can be designed in order to have a large girth and consequently a lower error floor at high SNR compared to other classes of LDPC codes. Since MS algorithms have low complexity in the implementation of

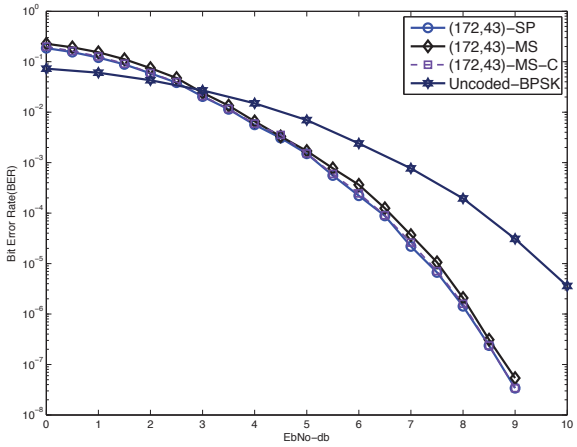


Figure 4.5: BER vs Eb/No(dB) for $(172, 43)$ TS-LDPC code with $g = 8$ using SP, MS and MS with correction factor decoding algorithms

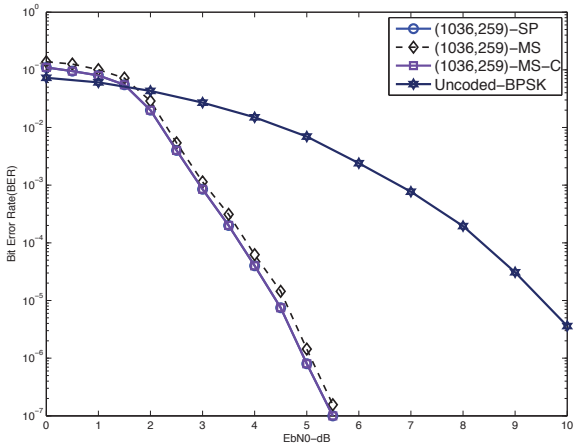


Figure 4.6: BER vs Eb/No(dB) for $(1036, 259)$ TS-LDPC code with $g = 10$ using SP, MS and MS with correction factor decoding algorithms

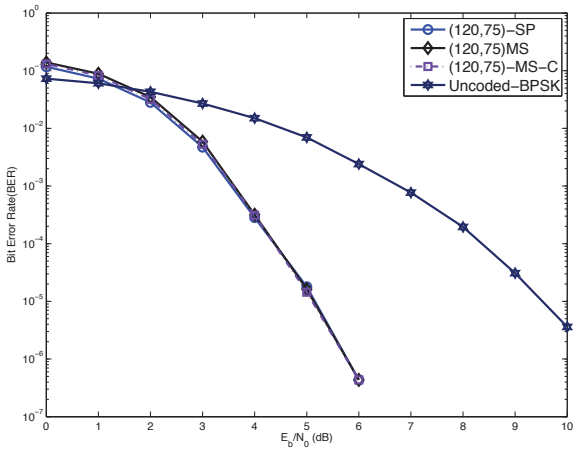


Figure 4.7: BER vs Eb/No(dB) for $(120, 75)$ TS-LDPC code with $g = 6$ using SP, MS and MS with correction factor decoding algorithms

analog decoders, MS and MS with correction factor algorithms have been considered with TS-LDPC codes.

Three different iterative decoding schemes, SP, MS and MS with correction factor, have been used to evaluate the error performance of the TS-LDPC codes. Simulation results have shown that the error performance of the MS algorithm is comparable with SP algorithm, for the same block length. Since MS algorithm can be implemented in analog circuitry with very low complexity compared to SP algorithm, we are going to use this algorithm for future implementation of TS-LDPC analog decoder. This implementation will result in high speed, low power consumption and small chip area.

Chapter 5

Min-Sum algorithm suitability for analog TS-LDPC decoders

In Chapter 4, it was shown that MS algorithms are suitable to be used as the decoding algorithm of TS-LDPC codes. This chapter is focused on implementation issues of the MS based TS-LDPC decoder. It is illustrated that the decoder using MS algorithm is robust to analog impairments such as mismatch, offset current and noise.

In [28], an MS based $(32, 8)$ LDPC analog decoder in 0.18μm CMOS technology has been designed and fabricated. The LDPC code used in [28] is a random LDPC code. As it has been discussed in the previous chapter, random LDPC codes have a higher error floor compared to TS-LDPC codes. In [28], it has been shown that the designed MS based LDPC decoder performs poorly at high SNR. To be more accurate at around 5dB SNR and approximately 10^{-3} BER, the error performance of the analog decoder deteriorates. In practical wireless communication systems depending on the application, BER of 10^{-5} to 10^{-7} is of interest. Therefore, to achieve a BER of 10^{-7} by using the decoder proposed in [28], a large SNR penalty is needed. This loss in the

error performance of the decoder chip presented in [28] is mainly due to mismatch imperfection.

Due to the reasons given in Chapter 4 we proposed to use MS algorithm for TS-LDPC codes. In this chapter necessary steps are taken in order to assure that analog decoder's building blocks (check nodes and variable nodes) are designed to meet the requirements. In other words, the blocks should be designed to provide the best performance.

In order to predict and avoid undesired affects of using analog circuitry on the error performance of TS-LDPC decoder in this chapter sensitivity to analog impairments of TS-LDPC analog decoder are studied. The sensitivity of the decoding algorithm to analog imperfections is simulated. In Chapter 6 and 7 the analog building blocks of the decoder are designed and modified based on the sensitivity results shown in this chapter.

In this chapter preliminary concept design and simulations are performed on the TS-LDPC modelled decoder to make sure accuracy and speed limitations imposed by analog circuitry do not degrade the performance of the decoder.

5.1 The Speed of the TS-LDPC MS Decoder

Speed is one of the crucial parameters of every decoder. The speed of the decoder depends on different factors, such as speed of computations performed in the nodes, Analog-to-Digital Converter (ADC), *Digital-to-Analog Converter* (DAC) and interconnections of the decoders' blocks as well as limitations imposed by the layout structure.

In digital implementation a counter is responsible for the number of iterations.

After a certain number of iterations the decoder converges. In other words, there is no more improvement in the reliability of messages through iterations. At this stage the decision is made. In contrast with digital decoders, there is neither a counter nor an actual iteration defined for their analog counterpart. Variable nodes and check nodes are asynchronous analog circuits. These analog modules process their inputs similar to their digital counter part. Thus, iterations are vanished and replaced by continuous-time transition of messages. The iterations of digital decoders can be interpreted as settling time in analog decoders. Here we define settling time as a time during which the error rate performance of an analog decoder improves. Despite the fact that dynamics of digital and analog decoders are different [78], a direct relation between the number of iterations in digital decoders and setting time in analog decoders can be observed. This means, if one algorithm requires more iterations to converge than another algorithm in digital decoders, it will likely have longer settling time in analog decoders.

As it has been illustrated in Chapter 4, the error performance of TS-LDPC codes using MS decoding algorithm is comparable with decoders utilizing SP algorithm. In other words, the error performance is not sacrificed in return for less complexity. The number of iterations required for messages to converge is depicted in Figs. 5.1 -5.3. The maximum number of iteration is set to be 100 which is usually used as maximum number of iterations. According to the algorithm discussed in Chapters 2 and 4, the iterations stop if the maximum number of iterations reached or the parity check equation is satisfied. From Fig. 5.1, it can be observed that for the code length of 28, the MS algorithm needs fewer iterations compared to the SP algorithm. Fig. 5.2 shows that the number of required iterations for the MS algorithm is more than

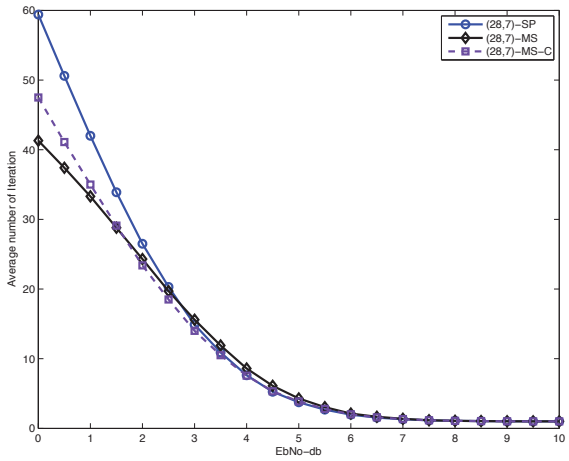


Figure 5.1: Average number of iteration versus Eb/No for $N = 28$

the SP algorithm if the block length increases. This is in agreement with Fig. 5.3 that at lower SNR, MS algorithm requires more iteration to converge compared to SP algorithm. In all three cases the number of required iterations for the MS with correction factor algorithm is between SP and MS algorithms. It can be noted that for all three algorithms, the number of required iterations converge at higher SNR. This implies that if the design target is the applications working at BER of 10^{-5} and lower, the required number of iterations for the different algorithms are the same while MS algorithm can be implemented using blocks with less computational complexity.

In this work, TS-LDPC codes with code length of $N = 28$, $N = 172$ and $N = 120$ are considered. For these sample codes, the density of the number of iterations is illustrated in Figs. 5.4 - 5.6. It can be observed that the number of iterations is much less than 100 which is a typical value of the maximum number of iterations. Comparing Figs. 5.4 - 5.6 shows that the required number of iteration increases with the code block length. In other words, for successful decoding of a TS-LDPC code

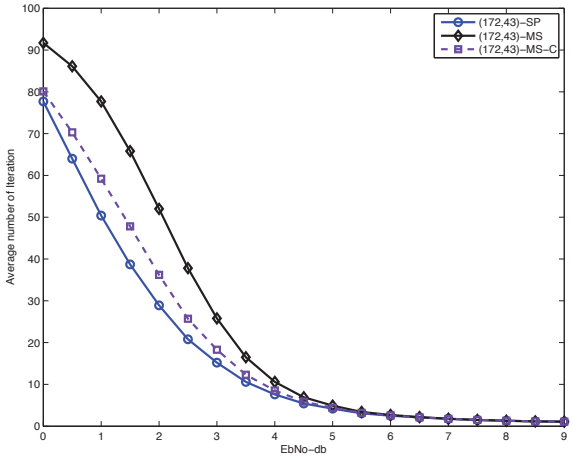


Figure 5.2: Average number of iteration versus Eb/No for $N = 172$

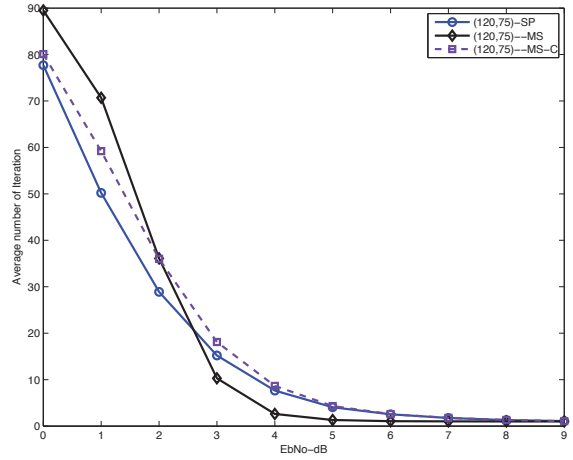


Figure 5.3: Average number of iteration versus Eb/No for $N = 120$

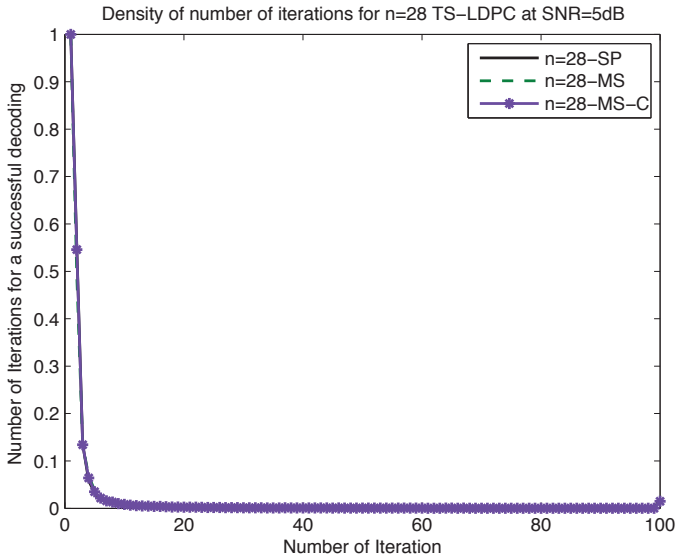


Figure 5.4: Density of number of iterations for $N = 28$

with length of $N = 172$ more iterations are needed compared to the codes with length of $N = 28$ and $N = 120$. This leads to more settling time in the continuous-time domain.

The effect of decoding with fewer iterations on the errors performance of TS-LDPC codes is shown in Figs. 5.7 to 5.15. Figures illustrate that the error performance of the TS-LDPC code with $IT_{max} = 100$ differs with approximately 0.1 dB from the error performance of the TS-LDPC code with $IT_{max} = 20$. Such a scenario is depicted in Fig. 5.16, when the settling time is dictated by the delay between clock 1 and clock 2. If the delay reduces, the available settling time in the analog decoder circuits reduces and consequently the error performance degrades. Therefore there is a trade off between the speed and the error performance.

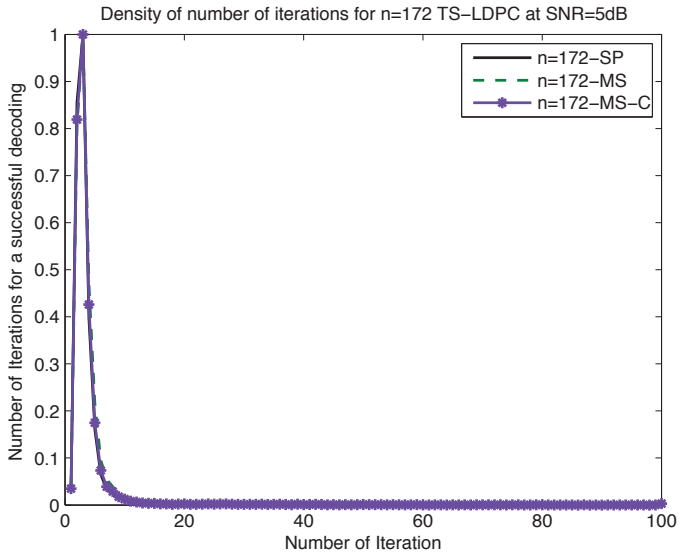


Figure 5.5: Density of number of iterations for $N = 172$

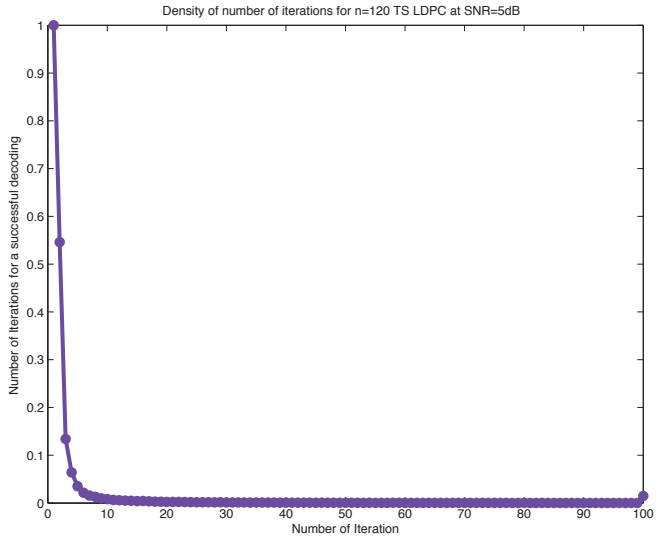


Figure 5.6: Density of number of iterations for $N = 120$

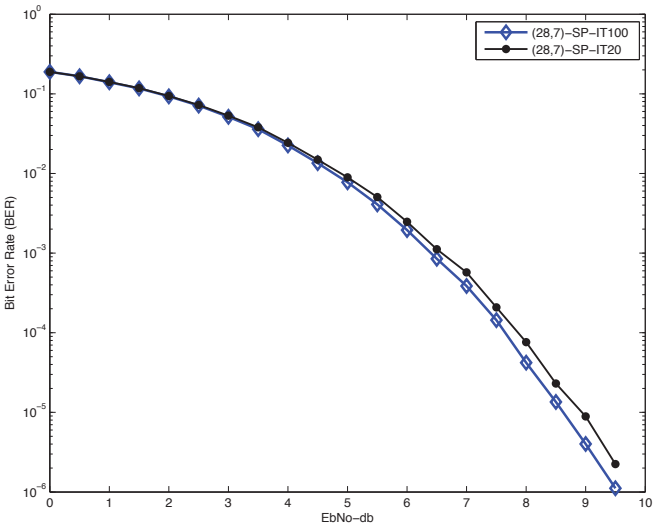


Figure 5.7: Bit Error Rate comparison for (28,7) TS-LDPC code using SP algorithm with $IT_{max} = 100$ and $IT_{max} = 20$

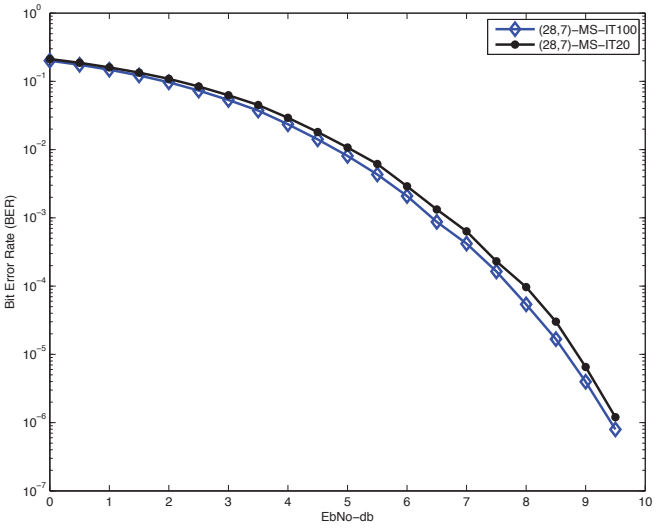


Figure 5.8: Bit Error Rate comparison for (28,7) TS-LDPC code using MS algorithm with $IT_{max} = 100$ and $IT_{max} = 20$

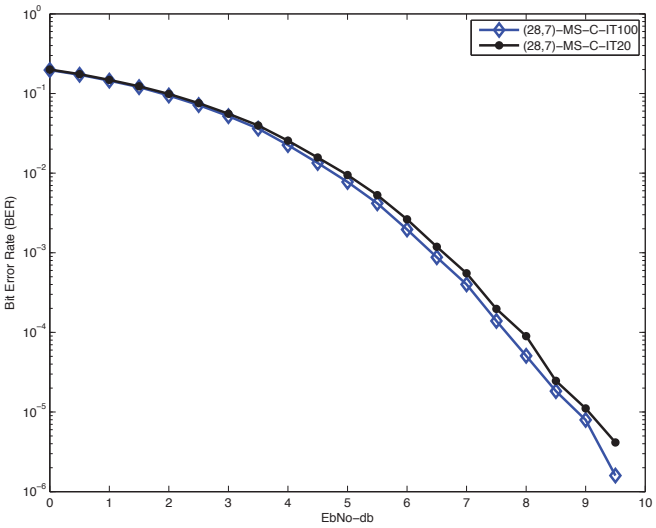


Figure 5.9: Bit Error Rate comparison for (28,7) TS-LDPC code using MS with correction factor algorithm with $IT_{max} = 100$ and $IT_{max} = 20$

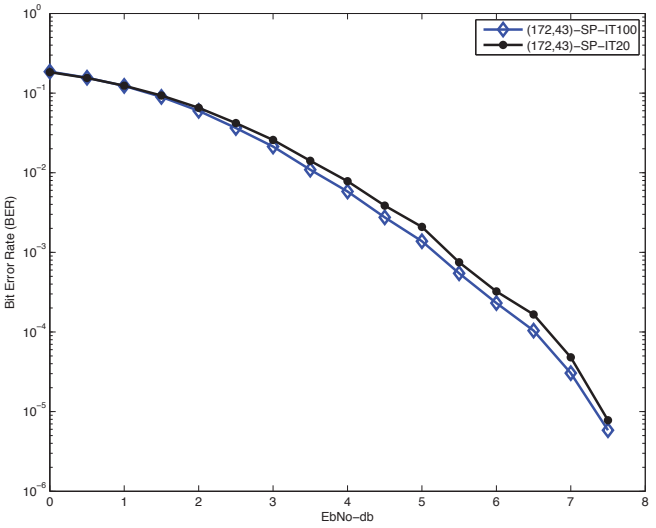


Figure 5.10: Bit Error Rate comparison for (172,43) TS-LDPC code using SP algorithm with $IT_{max} = 100$ and $IT_{max} = 20$

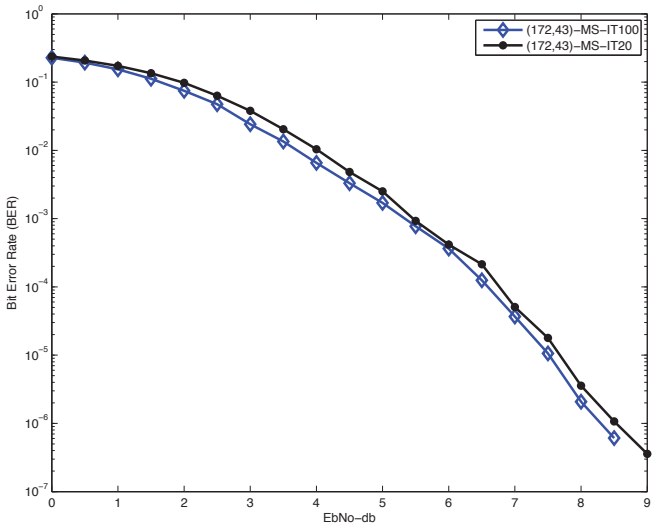


Figure 5.11: Bit Error Rate comparison for (172,43) TS-LDPC code using MS algorithm with $IT_{max} = 100$ and $IT_{max} = 20$

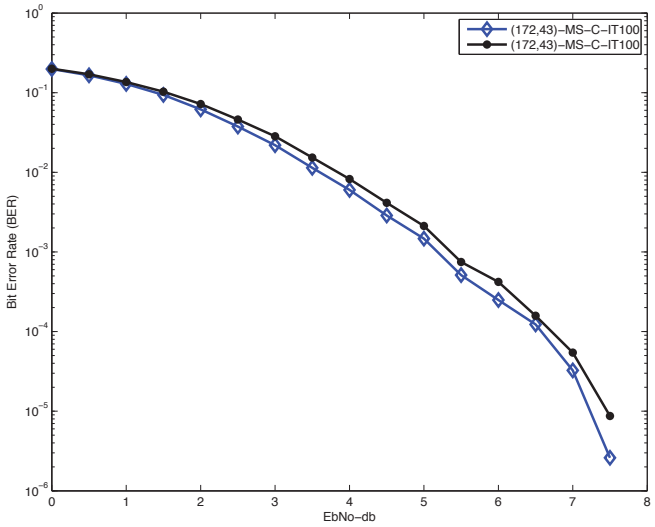


Figure 5.12: Bit Error Rate comparison for (172,43) TS-LDPC code using MS with correction factor algorithm with $IT_{max} = 100$ and $IT_{max} = 20$

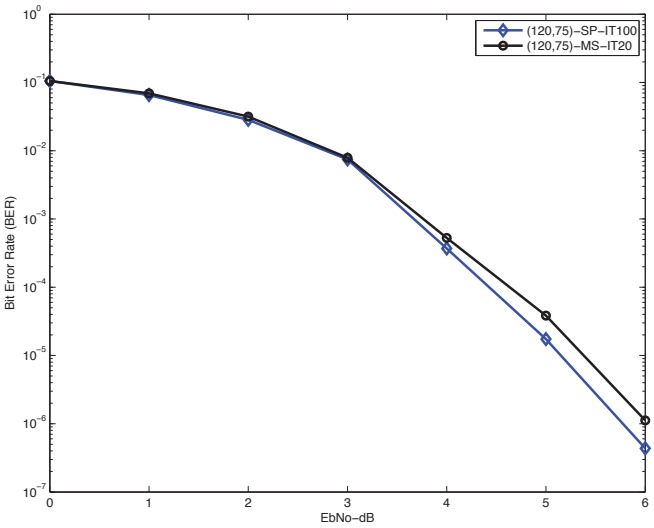


Figure 5.13: Bit Error Rate comparison for (120,75) TS-LDPC code using SP algorithm with $IT_{max} = 100$ and $IT_{max} = 20$

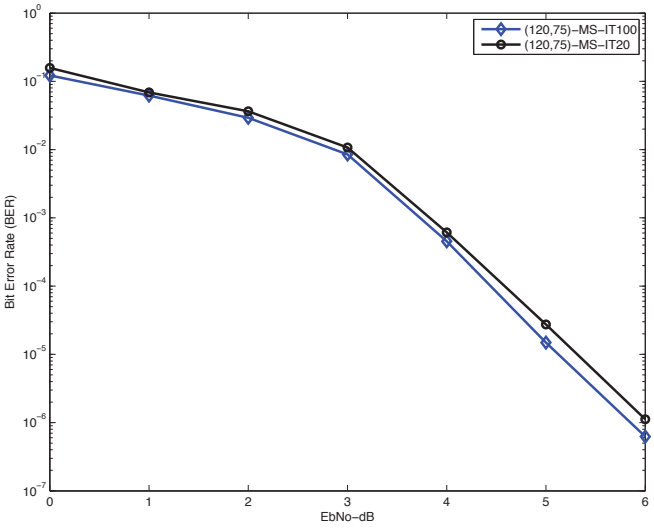


Figure 5.14: Bit Error Rate comparison for (120,75) TS-LDPC code using MS algorithm with $IT_{max} = 100$ and $IT_{max} = 20$

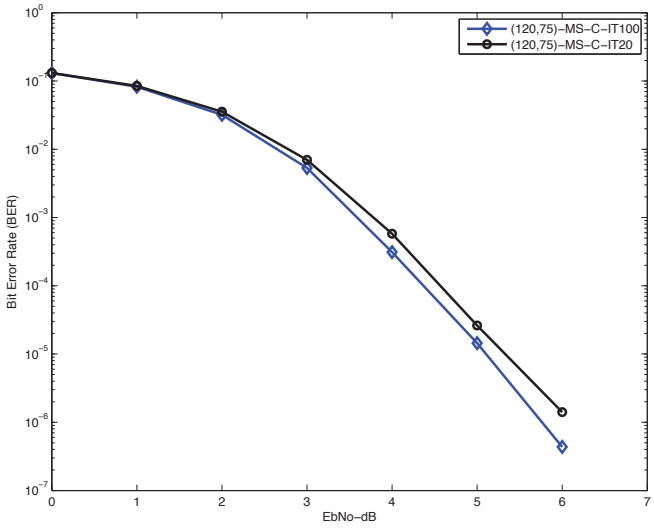


Figure 5.15: Bit Error Rate comparison for (120,75) TS-LDPC code using MS with correction factor algorithm with $IT_{max} = 100$ and $IT_{max} = 20$

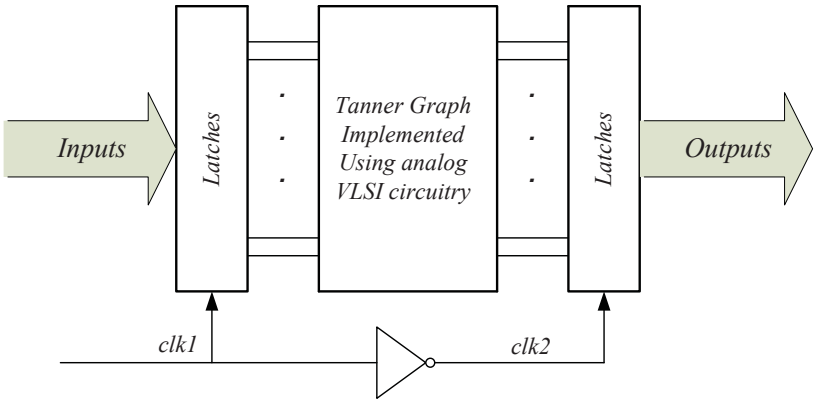


Figure 5.16: The block diagram of a latched input-output Tanner graph implemented by analog VLSI circuitry

5.2 Analog Circuits Impairments

In this section, some known analog impairments are discussed. The goal of this part is to show the robustness of utilizing the MS algorithm for decoding TS-LDPC codes against analog imperfections. The most important analog circuits impairments are mismatch, offset and noise.

5.2.1 Mismatch in Between Transistors

Ideally we assume that all transistors of equal dimensions exhibit the same properties. However in practice identical devices suffer from a mismatch due to variations in length, width and/or doping level. It is well known that one of the main factors of the accuracy of analog circuits is mismatch [79, 80]. Mismatch contributes to inaccuracy in computational blocks of analog decoders [39] [79–82].

MOSFET threshold voltage mismatch is caused by the contrast of the doping level of the channel and gate. It has been shown that the threshold voltage mismatch is the most dominant mismatch in the weak inversion working region [83]. This results in current matching error. It has been shown in [79] that the variance of threshold voltage mismatch is,

$$\sigma^2(\Delta V_{th}) = \frac{A^2_{V_{th}}}{W \cdot L} \tag{5.1}$$

where $A_{V_{th}}$ is the threshold voltage matching parameter. This parameter is technology dependent and in $90nm$ technology $A_{V_{th}}$ and can be estimated as $5mV\mu m$ by using [84]. It can be written that:

$$\Delta I = g_m \cdot \Delta V_{th} \tag{5.2}$$

where ΔI is generated by the threshold voltage mismatch for identically biased transistors. In the weak inversion region we have,

$$\Delta I = \frac{I_D}{nV_T}\Delta V_{th} \tag{5.3}$$

and V_T is the thermal voltage. Therefore,

$$\frac{\Delta I}{I_D} = \frac{\Delta V_{th}}{nV_T} = \frac{\frac{A_{V_{th}}}{\sqrt{WL}}}{nV_T}. \tag{5.4}$$

Equation (5.4) can be rewritten as,

$$\frac{\Delta I}{I_D} = \frac{g_m}{I_D}\Delta V_{th}. \tag{5.5}$$

Equations (5.1) and (5.5) show that by increasing W and L of the transistors for a specific drain current, the current variation due to mismatch can be reduced.

The effect of the mismatch on the current mirrors which are the basic building block of analog decoders is studied in [79]. Mismatch in these modules contributes significantly to the overall mismatch. It has been shown that for $0.13\mu m$ CMOS technology and $0.25\mu m$ BiCMOS technology the difference between the reference current ($10\mu A$) and the output is less than 1% for large size transistors. For minimum size transistor this difference changes to 16% for $0.13\mu m$ CMOS technology and 24% for $0.25\mu m$ BiCMOS technology. In these technologies, if the size of transistors is five times the minimum size transistor, the variation of the current due to mismatch is about 5% to 10%. Therefore, for the targeted technology of $90nm$, we may assume

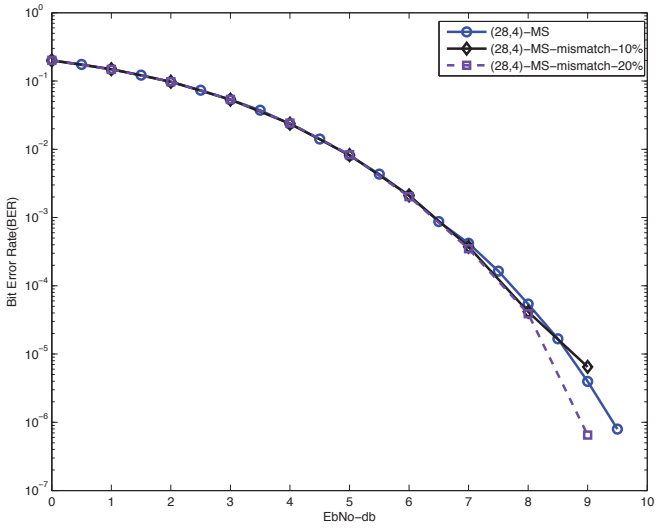


Figure 5.17: The error performance of (28,7) TS-LDPC code using MS decoding algorithm in ideal case and with 10% and 20% mismatch

that the mismatch in fabrication process is between 10% to 20% of the ideal value (without mismatch). If the size of the transistors used in current mirrors is 10 times of the minimum size transistor in $90nm$ technology, based on equation (5.4) mismatch of 20% is expected which is in agreement with our assumption.

To verify the effect of mismatch in the error performance of the analog decoder of TS-LDPC code, we include the mismatch in the decoder model. The modelled mismatch can be represented by multiplying the outgoing message by a normal random variable with mean of 1 and standard deviation σ, $(N(1, \sigma))$, where $\sigma = 0.1$ represents 10% variation and $\sigma = 0.2$ represents 20% variation. The simulation results are shown in Figs. 5.17 to 5.19. It can be observed that the error performance of the TS-LDPC codes considering the effect of mismatch is almost the same as the ideal case.

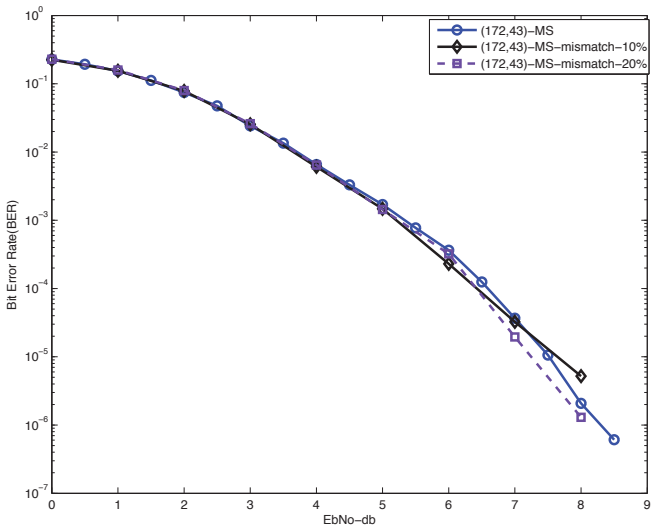


Figure 5.18: The error performance of (172,43) TS-LDPC code using MS decoding algorithm in ideal case and with 10% and 20% mismatch

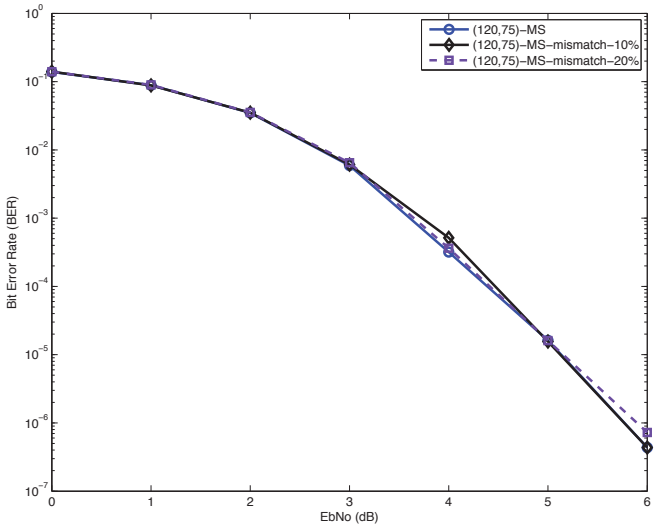


Figure 5.19: The error performance of (120,75) TS-LDPC code using MS decoding algorithm in ideal case and with 10% and 20% mismatch

5.2.2 Offset Current

Current buffers are the most common blocks in implementation of the MS algorithm. Offset current for current buffers is introduced by mismatch. Consider Fig. 5.20(a), without mismatch, offset current flows from voltage supply to ground and does not affect the output. In Fig. 5.20(b) & (c), mismatch makes the current buffers unbalanced. Therefore, the offset current produces an input dependent offset current at the output. Assuming an unbalanced current buffer where x% mismatch changes $1:1$ current buffer to $1:1+|x|$ and $1:1-|x|$ buffers as shown in Fig. 5.20(b) & (c). Therefore, based on the magnitude of the input current, offset current and percentage of mismatch, the magnitude and the direction of the output current can be changed. This may result in changing of the sign or reliability of messages depending on the application of the current buffer.

Since mismatch contributes to offset current in current mirrors, in this work the offset current is modelled with a random variable with Gaussian distribution. To observe the effect of the offset current on the error performance of the TS-LDPC codes, a random variable with standard deviation of $x\%$ is added at the output of the blocks. Moreover, the sign of the random variable is changed randomly with the same standard deviation. It is assumed that current buffer blocks suffer from 10% mismatch. The simulation results are depicted in Figs. 5.21 to 5.23. The simulation results show that the error performance of the TS-LDPC codes considering the offset imperfection of analog decoders is almost the same as the ideal case.

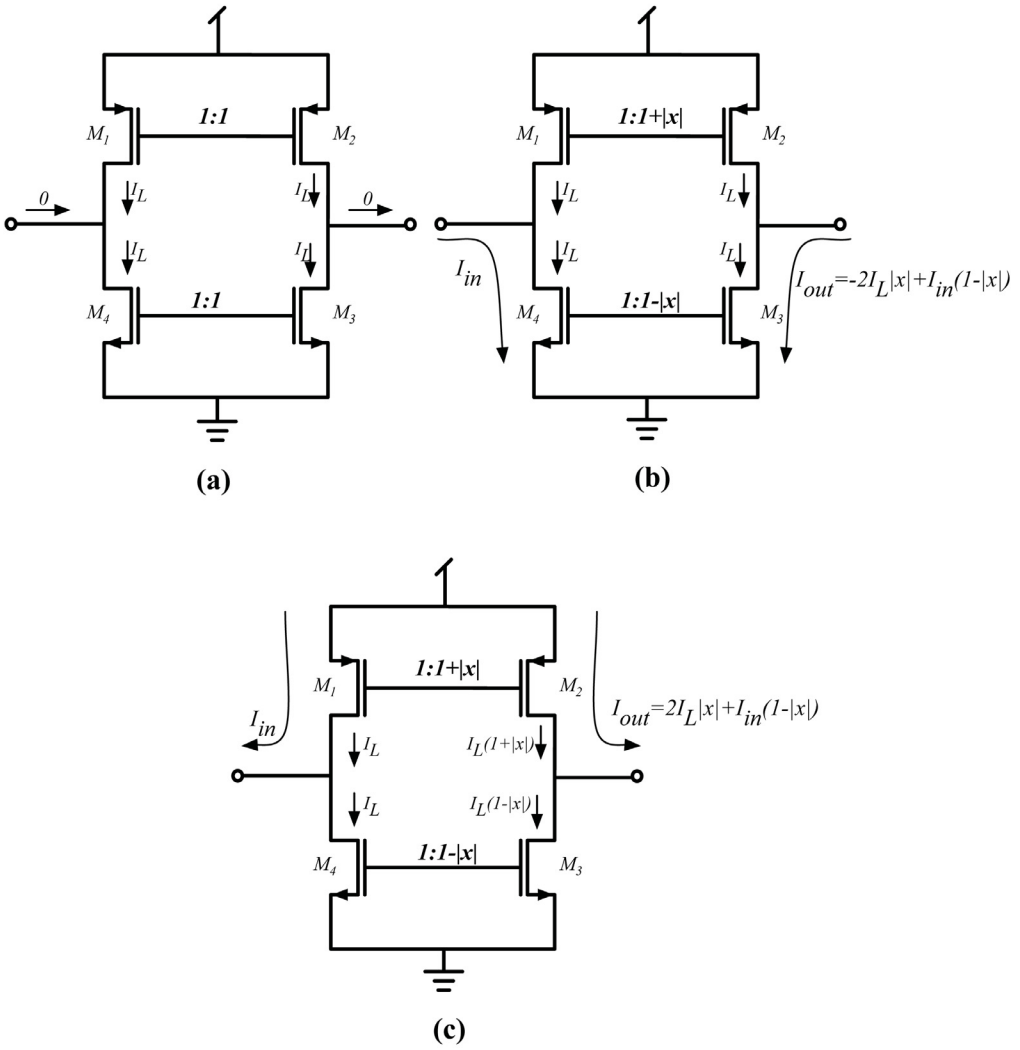


Figure 5.20: A simple current buffer without mismatch (a) and suffering from mismatch (b) & (c)

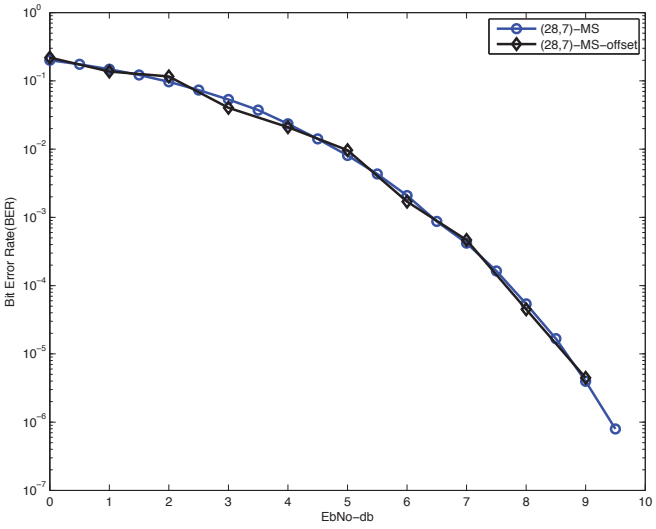


Figure 5.21: Bit Error Rate performance for (28,7) TS-LDPC code using MS algorithm in ideal case and with offset

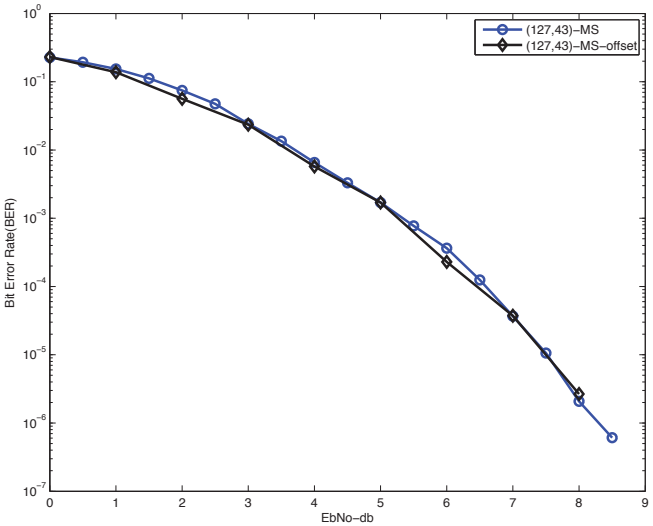


Figure 5.22: Bit Error Rate performance for (172,43) TS-LDPC code using MS algorithm in ideal case and with offset

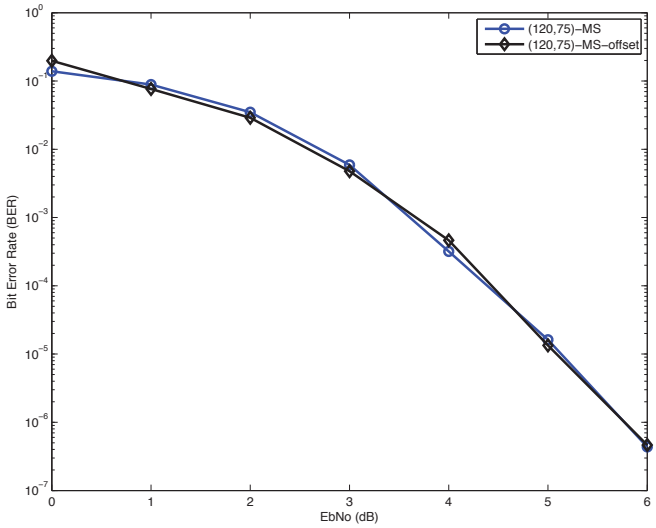


Figure 5.23: Bit Error Rate performance for (120,75) TS-LDPC code using MS algorithm in ideal case and with offset

5.2.3 Noise

In analog VLSI circuits the minimum applicable signal is limited by noise [85]. There are two types of noise in analog circuits which are inherent noise and interference noise. Inherent noise is a random signal generated by the device. This type of noise can be reduced through a careful design but cannot be eliminated completely. Interference noise is any undesired signal produced by other parts of the circuit or coming from the surrounding world. Interference noise can be minimized by a careful design, wiring and layout [86].

The main sources of inherent noise are flicker and thermal noise of the MOS transistor and thermal noise of the distributed resistors over the chip. The power spectral density of flicker noise is proportional to $\frac{1}{f}$. In high frequency circuits flicker noise can be ignored. Since in analog decoders the frequency is not very high, flicker

noise must be taken into account. The power spectral density of the thermal noise is white and includes all the frequencies. The power spectral density of the thermal noise depends on the resistance of the distributed resistors and channel resistor of the MOS transistors [85].

The other important source of interference noise is the switching noise of the logic gates and test board noise. Test board noise is mainly due to ground coupling of analog design and digital design.

In this work, the effect of the additive white Gaussian noise (AWGN) on the error performance of TS-LDPC code is considered. The AWGN is modelled with a random Gaussian variable added at the output of each block. This random variable has zero mean and standard deviation σ. In our simulation AWG noises with $\sigma = 0.1$ and $\sigma = 0.2$ are considered. The simulation results are shown in Figs. 5.24 to 5.26. It can be observed that the noise does not change the error performance of TS-LDPC codes significantly.

It should be mentioned that in all of the simulations, random variables that are representing the impairments are kept constant in each iteration.

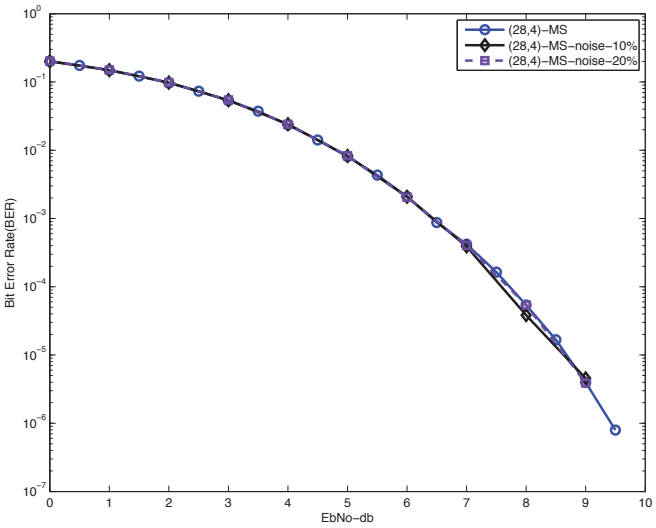


Figure 5.24: Bit Error Rate performance for (28,4) TS-LDPC code using MS algorithm in ideal case and with noise 10% and 20%

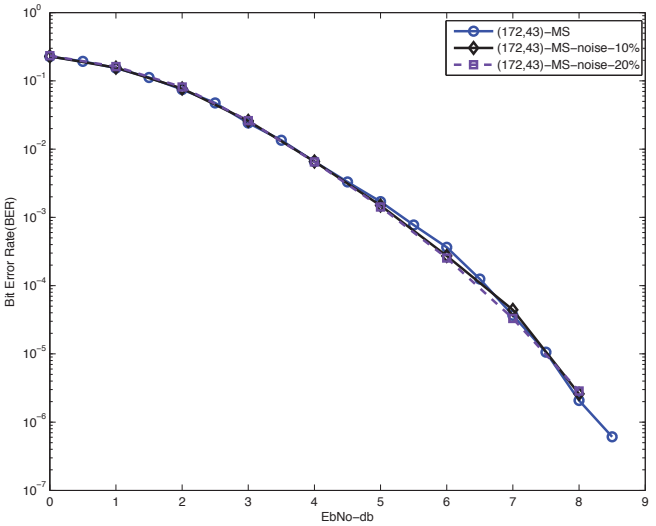


Figure 5.25: Bit Error Rate performance for (172,43) TS-LDPC code using MS algorithm in ideal case and with noise 10% and 20%

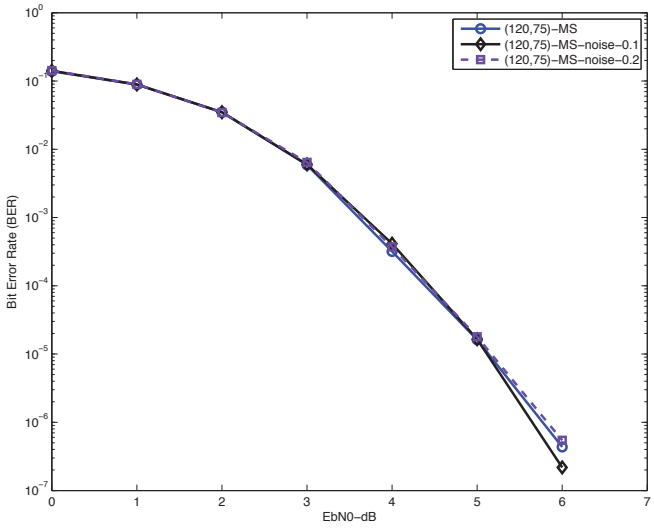


Figure 5.26: Bit Error Rate performance for (120,75) TS-LDPC code using MS algorithm in ideal case and with noise 10% and 20%

5.3 Design Procedure for Analog Decoder Chip Implementation

Based on the simulation results in the previous sections the design procedure for the TS-LDPC analog decoder is discussed here. Designing the basic blocks of the Tanner graph of the TS-LDPC code, check nodes and variable nodes, is the first step toward the design of the analog decoder chip. Utilizing the modelled analog impairments in the error performance evaluation of the TS-LDPC decoders shows the effect of imperfections on the performance of the code. It can be observed that the typical 10% to 20% variation of the ideal case does not contribute to the degradation of the error performance. This gives us a scope in designing check nodes and equality nodes. It should be noted that a more precise model of analog impairments can be achieved after the design of building blocks. The most important building block

for implementing MS algorithm is the check node. The designed check node should precisely implement the equation (4.7). This equation can be divided in two parts, the sign multiplier and the magnitude minimizer. Design of the magnitude minimizer is more crucial than the sign multiplier. As it has been mentioned previously, based on the design of the minimizer the effect of mismatch on the error performance of the decoder at higher SNR can be severe. Therefore, re-designing of the minimizer is the most important step. The re-designed minimizer module is evaluated based on the speed, power consumption and accuracy. According to the evaluations, the minimizer module is modified to meet the requirements. It is worth mentioning that there are always trade-offs between the speed, power consumption and accuracy. Therefore the most suitable trade-off is chosen. Monte-Carlo simulations are performed on the minimizer module to obtain the standard deviation of the mismatch. Based on the mismatch statistics, error performance of the TS-LDPC decoder is evaluated through Matlab simulations. According to the outcome of the simulations the minimizer module is modified.

Equation (4.8) shows that the updating rule of the variable nodes is a simple summation. Therefore, *Kirchhoff's current Law* (KCL) can be used for the variable nodes of the Tanner graph. Having the building blocks, Tanner Graph of the designed (120,75) TS-LDPC code is constructed. The speed, power consumption and the accuracy of the analog decoder is evaluated. The error performance of the decoder cannot be tested using Cadence tools (Spectre). However based on the estimated imperfections of the analog decoder, the Matlab model of the TS-LDPC code can be modified and simulated. A comparison based on accuracy, speed and power consumption is made between the fabricated analog TS-LDPC decoder chip and the available MS

algorithm based random LDPC decoder and its digital counterpart.

Power consumption is one of the important issues in any analog VLSI circuit. According to the discussion in previous sections, at higher SNR the number of iterations required for messages to converge is less than the number of required iterations at low SNR. This leads to a shorter convergence time in the analog decoder. However, the designed decoder should accommodate the worst case (longer convergence time at low SNR). Therefore, at high SNR; the decoder needs less time to converge, the decoder circuit can be shut down for a period of time and waken up with the next codeword.

The design procedure of the analog decoder is summarized in Fig. 5.27.

5.4 Conclusion

In this chapter it was assumed that there is a one to one relation between the iteration numbers in digital iterative decoders and settling time of analog iterative decoders. It was shown that an MS based TS-LDPC decoder does not many more iterations compared to the decoder using the SP algorithm. Based on the relation between number of iterations in digital decoders and settling time in analog decoders it can be concluded that an MS based analog TS-LDPC decoder does not require more settling time compared to SP-based decoders. Moreover it was observed that at high SNR the number of iterations for MS algorithms and the SP algorithm are the same. This means the same settling time in their analog counterparts. The effect of decoding with smaller number of iterations was also considered for TS-LDPC decoders. It was shown that if the number of iterations are limited to 20, the loss in error performance is about 0.1dB. This shows a trade-off between the speed and accuracy of the decoder.

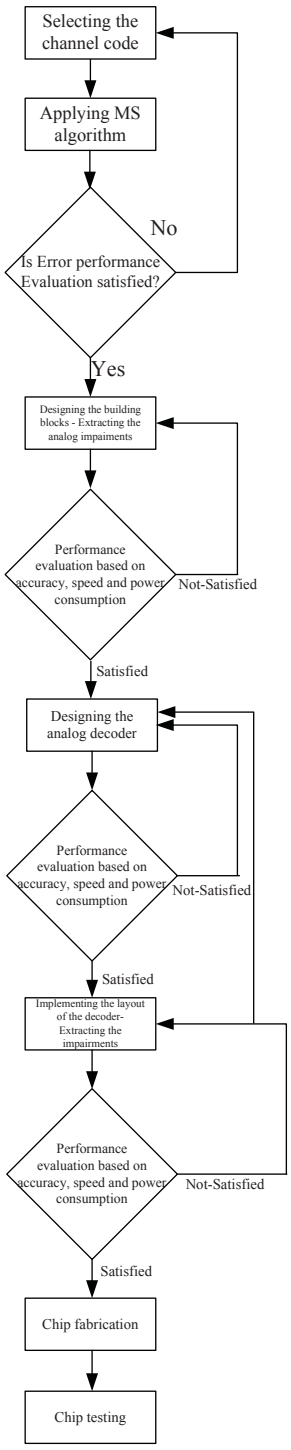


Figure 5.27: The design procedure

It was discussed that the error performance of analog decoders can be affected by analog impairments such as mismatch, offset current and noise. Therefore these imperfections should be tested in the context of TS-LDPC decoders to observe any undesired impact on the error performance. For each analog imperfection case, a model for simulation has been used. In each case, simulation results showed that the degradation of error performance of TS-LDPC codes due to analog impairments is negligible. Therefore, it can be concluded that the analog decoder of TS-LDPC code using MS algorithm is robust against analog imperfections.

It has been discussed that analog impairments may result in significant loss in error performance of analog decoders [28]. To evaluate required specification of the MS algorithm building blocks (check nodes and variable nodes), tolerability of decoding algorithm was simulated when analog impairments were changing the output of MS algorithm building blocks. It was illustrated that the error performance of the decoder is unchanged if impairments vary the output of the major blocks by 10% to 20%. This gives us the safe margin in designing the variable nodes and check nodes. Based on the tolerability margin of the simulated decoder to these imperfections, the analog building blocks of the decoder are designed and modified as it will be shown in Chapters 6 and 7. The design procedure was briefly discussed at the end of the chapter.

Chapter 6

TS-LDPC Analog Decoder Chip and Input-Output Stages

In this chapter the architecture of the TS-LDPC analog decoder is proposed. For the first time a TS-LDPC analog decoder is studied, designed and fabricated. In this design a TS-LDPC code with length of 120 bits is considered. To the best of our knowledge, this analog decoder is the longest code length analog decoder designed and fabricated so far. This sets a new state-of-the-art for analog decoding.

It should be noted that a top to bottom approach is used to discuss the architecture and design of the proposed TS-LDPC analog decoder. The chapter starts with the architecture of the proposed design followed by top level building blocks of the decoder. Input and output stage blocks are discussed in detail.

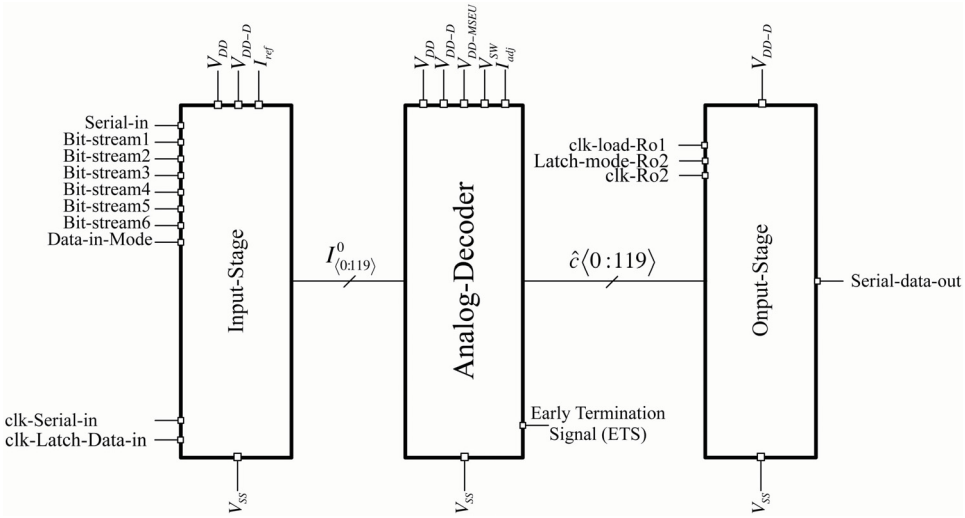


Figure 6.1: Main building blocks of the TS-LDPC analog decoder

6.1 TS-LDPC Analog Decoder Top Level Building Blocks

In this section the building blocks of the decoder chip are discussed as shown in Fig. 6.1 which consists of an Input-Stage, an Analog-Decoder and an Output-Stage. In total, 22 different I/O pins are used in this chip as shown in Fig. 6.1. Description of each pin is as follows:

- $\mathbf{V_{DD}}$ (voltage): Analog power supply of 1.1V.
- $\mathbf{V_{DD-D}}$(voltage): Digital power supply of 1.2V.
- $\mathbf{V_{DD-MSEU}}$ (voltage): Analog power supply of 0.9V.
- $\mathbf{I_{ref}}$ (current): Analog current reference of 140nA.
- $\mathbf{I_{adj}}$ (current): Analog current reference of 100nA.

- $\mathbf{V_{SW}}$ (voltage): Switch for controlling the decoding procedure and decoding time. This signal controls the bus connection between check-nodes and variable-nodes. Before the decoding starts, this switch should be off (0V) to cut the connection between variable-nodes and check-nodes. When the switch is connected (1V) the decoding procedure starts.

- $\mathbf{V_{SS}}$: Ground.

- **Data-in-Mode** (voltage): Chip can receive data serially or through the bit-stream pins. Data-in-Mode pin selects which method is used.

  - If Data-in-Mode=1V: Serial input.

  - If Data-in-Mode=0V: Semi-Serial input.

- **Serial-in**(voltage): This pin is utilized to transfer initial condition bits serially.

- **Bit-stream** $< 1 : 6 >$ (voltage): Input data bits are sent semi-serially to the decoder through Bit-stream1 to Bit-stream6 buses. Each data bus carries 120 bits corresponding to 20 channel symbols. Therefore bus 1 carries channel symbols 1 to 20 and consequently bus 6 carries channels symbols 100 to 120 while each symbol is represented by 6 bits. Semi-serial mode is considered to speed up the transferring rate by a factor of 6.

- **clk-serial-in** (voltage): Data are latched into input memory by the positive edge of clk-serial-in.

- **clk-Latch-Data-in** (voltage): With the positive edge of this clock pulse, data are captured at the latches connected to the output of memory cells.

- **clk-load-Ro1** (voltage): This clocking signal is used to save data into the first latch at the output.
- **Latch-mode-Ro2** (voltage): The second latch at the output has the ability to capture data in parallel and then serially output the saved data.
  - If Latch-mode-Ro2=1V : Latch works in parallel input mode.
  - If Latch-mode-Ro2=0V : Latch works in serial output mode.
- **clk-Ro2** (voltage): This clock pulse is used to load the data into the second latch and serially output the data.
- **Serial-data-out** (voltage): Serial data are transmitted on this pin.
- **Early Termination Signal (ETS)** (voltage): If the decoder comes to a decision in less than the nominal decoding time, this signal changes from 0 to 1V. This signal may be used to power off the analog decoding modules. Using this signal power dissipation of the chip can be improved. The other use of this signal is for hand shaking. In this case, when ETS goes high, the output can be captured at the output latch and new initial LLR values can be loaded into the the chip.

In order to start decoding correctly, initial LLR values should be received by the TS-LDPC analog decoder chip. In general, these data are in continuous-time mode which can be categorized as analog values as given in equation (3.10). In this design, due to the testing constraints, the initial LLR values are transmitted into the chip in digital form. These digital LLR values are then converted to analog values utilizing an Digital-to-Analog Converters (DAC) to be used in the analog decoder.

The Input-Stage block is responsible for converting the received digital initial LLR values to analog current mode signals. For 120-bit long decoder chip, 120 initial condition values are required. Quantization of analog LLR values is one of the important issues for any decoder which works with digital LLR values. During the quantization process each analog value should be clipped to a level c_{th} and then the clipped analog value should be coded with some number of bits.

In order to find the appropriate clipping level c_{th}, the analog LLR values of the (120,75) TS-LDPC code are clipped to values between two and ten. Simulation results are shown in Fig. 6.2. It can be observed that if LLR values are clipped to six or lower, then the error performance of the decoder degrades significantly. If LLR values are clipped to seven and higher the error performance is even improved at higher SNR. Hence, in analog to digital conversion, the LLR values are clipped to seven. After clipping the LLR values, each value should be quantized using some number of bits. In Fig. 6.3, we have simulated the TS-LDPC decoder while the LLR values are clipped to seven and then quantized from three to nine bits. Simulation results show that if LLR values are mapped to between six and nine bits the loss in error performance is negligible. Therefore, in this work 6-bit quantization of LLR values is used to reduce the memory size and save the chip area.

It is assumed that the magnitude of each initial value can be represented by five bits and the sign by the sixth bit. Therefore, six bits are used to represent each initial value. In total, 720 bits should be transferred to the decoder chip in order to have a complete set of initial condition values. As it was discussed, these bits can be sent to the decoder serially or semi-serially through pins Serial-in or Bit-stream $< 1 : 6 >$ respectively. Pin Data-in-Mode selects the data transfer. Semi-serial mode

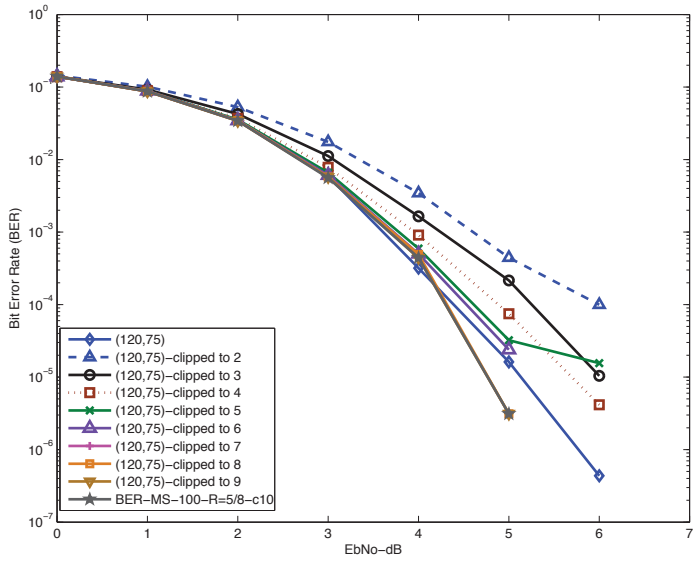


Figure 6.2: Error Performance of (120,75) TS-LDPC code when clipping is applied

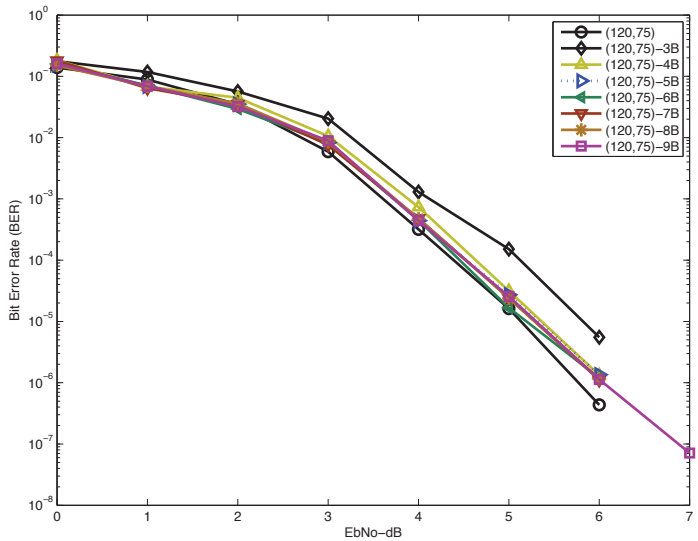


Figure 6.3: Error Performance of (120,75) TS-LDPC code when LLR values are clipped to seven and quantized to three to nine bits

is considered to speed up the transferring process by a factor of 6. In this case, bits 1-120 corresponding to initial conditions for variable nodes 1-20 are sent to Bit-stream1, bits 121-240 corresponding to initial conditions for variable nodes 21-40 are sent to Bit-stream2 and so on. Data are loaded into memory cells by the positive edges of the clock pulse clk-serial-in. When all 720 bits are stored into memory cells, with positive edge of clk-Latch-Data-in stored data are captured by a latch designed to keep a valid initial condition for one complete decoding cycle. The outputs of this block (120 outputs) are analog current mode signals which are connected to the inputs of the next block.

The Analog-Decoder block is the analog heart of the decoder chip. This block is responsible for performing the Min-Sum algorithm on the received channel observations (initial values) and report the results to the Output-Stage. As mentioned before, the Inputs of the analog block are current mode signals provided by the Input-Stage block. If the decoding is successfully done or a certain settling time passes, outputs are captured by the Output-Stage block.

The Analog-Decoder block is a physical representation of the (120,75) TS-LDPC code's Tanner graph. The Tanner graph associated with the designed TS-LDPC code is shown in Fig. 6.4. To have a better realization of the graph, the interleaver part is shown briefly and only a few sample connections are depicted. In this figure, only variable nodes, check nodes and connections between these set of nodes are presented. A snapshot of variable node 1 is illustrated in Fig. 6.5. As it can be seen the connections between initial conditions (obtained by using equation (3.10)) and variable nodes are one directional while the connections between variable nodes and check nodes are bidirectional. As discussed in Chapter 3, when the decision has been

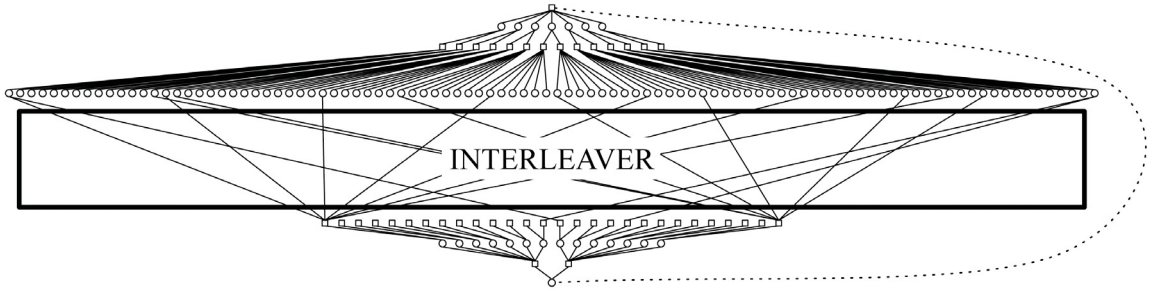


Figure 6.4: Tanner graph associated to (120,75) TS-LDPC code

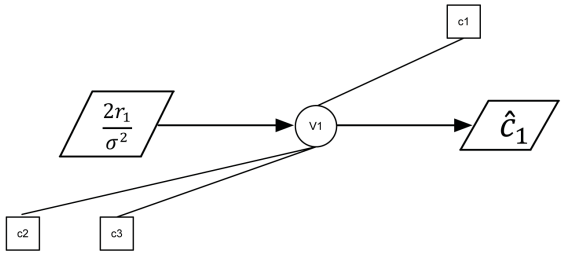


Figure 6.5: A Snap shot of variable node 1 with complete connections

made, $\widehat{c_i}$ is the sign of the summation of signals received by the variable node i. This decoded bit is stored in a memory cell located in the output stage.

If the decoding is done successfully before the maximum settling time is reached then the Early Termination Signal (ETS) is activated. Upon activation of this signal, the analog decoder block may be shut-down using off-chip circuitry to reduce the total power consumption of the chip. It should be noted that the shut-down process starts after storing the decoded bits at the output memory and the decoder restarts operating when new received data are available for decoding.

The Output-Stage block consists of two latches: one is a parallel latch R1 and the other one is a parallel-in/serial-out latch R2. When the analog decoder has concluded the decoding, the decoded bits are captured by the Output-Stage block. This occurs at the positive edge of clk-load-Ro1. At this stage the decoded bits are saved into the

first register R1. To transfer decoded data out of the chip, the bits loaded into the register R1 should be saved into the second register R2. If $Latch-mode-Ro2 = 1V$ and $clk-load-Ro2$ changes from low to high, data are loaded into register R2. The stored decoded codeword in R2, can be sent to the off-chip module serially with the positive edge of $clk-load-Ro2$ when $Latch-mode-Ro2 = 0V$.

It should be noted that to minimize the power supply noise of the digital part on the analog circuitry, two separate voltage sources are considered. The voltage supplies V_{DD} and V_{DD-D} are used for analog and digital circuits, respectively.

6.2 Input-Stage of the Analog Decoder

In this section, the Input-Stage block is discussed in details. Architecture, building blocks and circuit schematics are presented as well. The Input-Stage is comprised of two blocks: Memory block and Latch-DAC-MSCU block converter as shown in Fig. 6.6.

6.2.1 Memory Block

As discussed in Section 6.1, 720 initial bits are transferred to the memory block serially or semi-serially and stored in 720 memory cells. Memory cells are organized in six banks of 120 cells. Each bank of 120-cell memory is shown in Fig. 6.7, which consists of 120 *Flip Flops* (FF). Flip flops in this figure are D-type FF and are chosen from the standard cell library of TSMC 90nm under name of DFQD1. The first FF is integrated with a 2:1 multiplxer and therefore, each memory bank is accepting inputs from two different sources: from the output of the previous register (in serial mode)

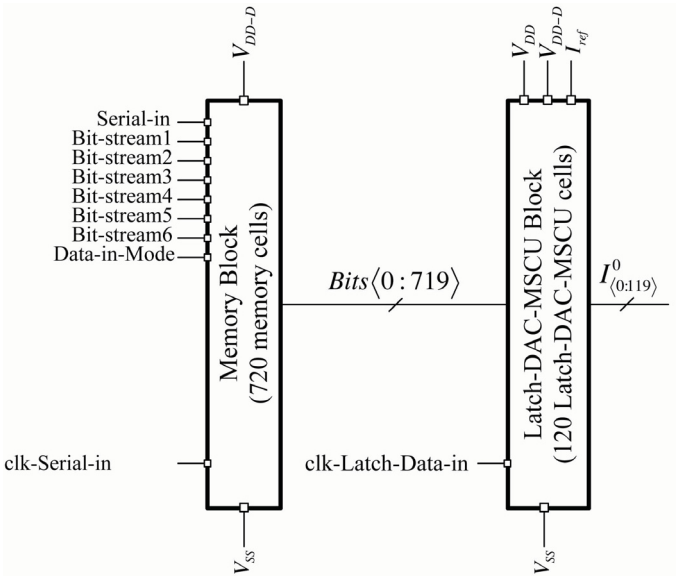


Figure 6.6: Input-Stage block of the decoder

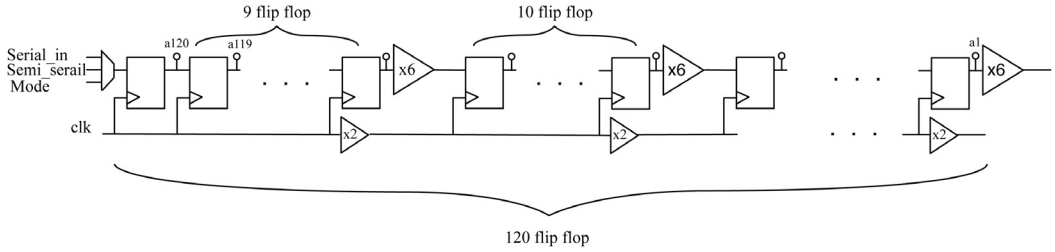


Figure 6.7: A memory bank

or from the off-chip source (in semi-serial mode). Also, it can be seen that after each 10 FFs, six delay components are used in series on the data line and two delay elements are used in the clock line. In this way we assure that we have valid data to be captured by the next FF. The delay elements on the data line and the clock line are the NOT gates which are selected from standard cell library under names of INVD0 and INVD2, respectively.

Fig. 6.8 shows a 720-cell memory architecture. This memory consists of six 120-register banks. Some delay elements are used in this design. Delay elements on data

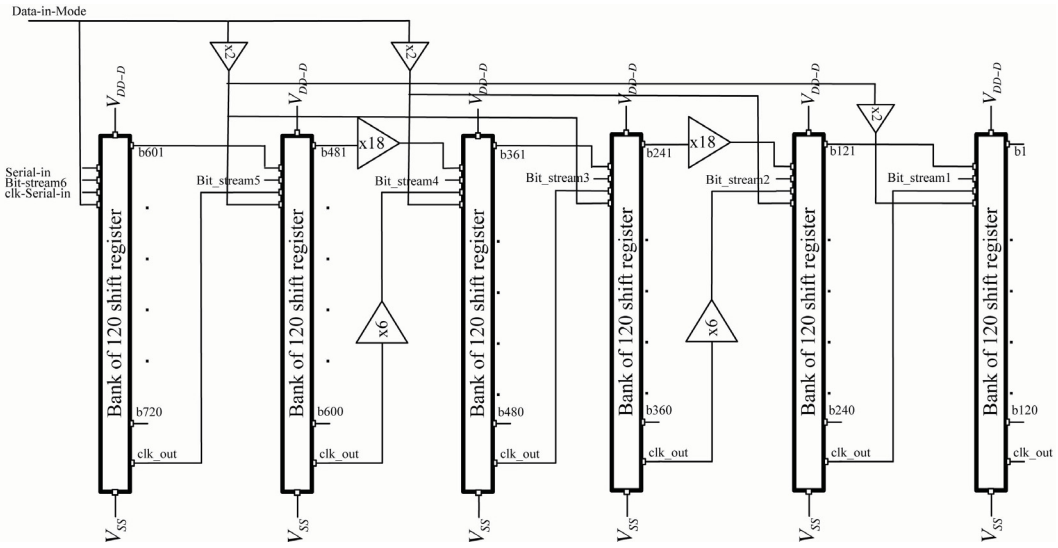


Figure 6.8: 720-cell memory

lines are utilized to maintain valid set-up time, while elements on the clock and mode lines are used to deliver valid signals. It should be noted that delay elements are employed to compensate the effects of long wires in the chip layout. As previously mentioned, the delay elements on the data and the clock lines are the NOT gates which are available in the standard cell library under names of INVD0 and INVD8, respectively.

6.2.2 Latch-DAC-MSCU Block

In this part, the architecture and design of the Latch-DAC-MSCU block shown in Fig. 6.6 are discussed. This block is responsible to convert 720 digital bits corresponding to initial conditions ($Bits < 0 : 719 >$) to 120 current mode signals ($I^0_{<0:119>}$). As mentioned earlier, to represent each initial condition signal, six bits are required. Therefore each cell of Latch-DAC-MSCU deals with six bits and one output signal is produced. Consequently, Latch-DAC-MSCU block is comprised of

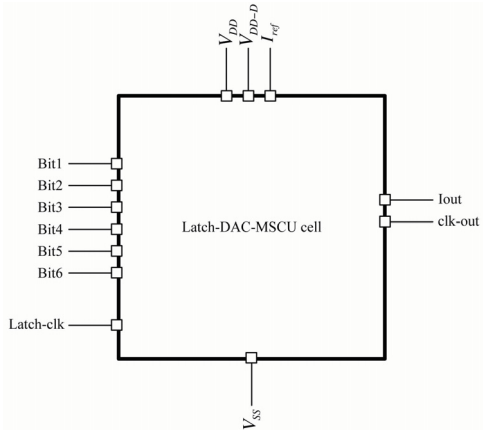


Figure 6.9: Latch-DAC-MSCU cell

120 cells. One Latch-DAC-MSCU cell is shown in Fig 6.9. Each cell requires a reference current (I_{ref}). This current is provided by an off-chip current I_{ref} as shown in the pin arrangement of the decoder chip in Fig. 6.1. To deliver equal I_{ref} current to all Latch-DAC-MSCU cells, the off-chip current should be mirrored through a 1:120 current mirror. This current mirror is designed using a 1:12 N-type and twelve 1:10 P-type current mirrors.

To facilitate chip layout, a block consisting of 10 Latch-DAC-MSCU cells is considered. Fig. 6.10 shows the architecture of this block. In Fig. 6.10, a 1:10 P-type current mirror is responsible for distributing I_{ref} current to the 10 Latch-DAC-MSCU cells in the block.

Complete architecture of the Latch-DAC-MSCU block is shown in Fig. 6.11. The required 120 Latch-DAC-MSCU cells are implemented using 12 blocks of 10-cell Latch-DAC-MSCU as represented in Fig. 6.10. In this figure, there are 720 inputs and 120 outputs. A 1:12 N-type current mirror is considered to distribute the same reference current to each block. As discussed before, each current will be mirrored again by a 1:10 P-type current mirror to provide almost the same reference current

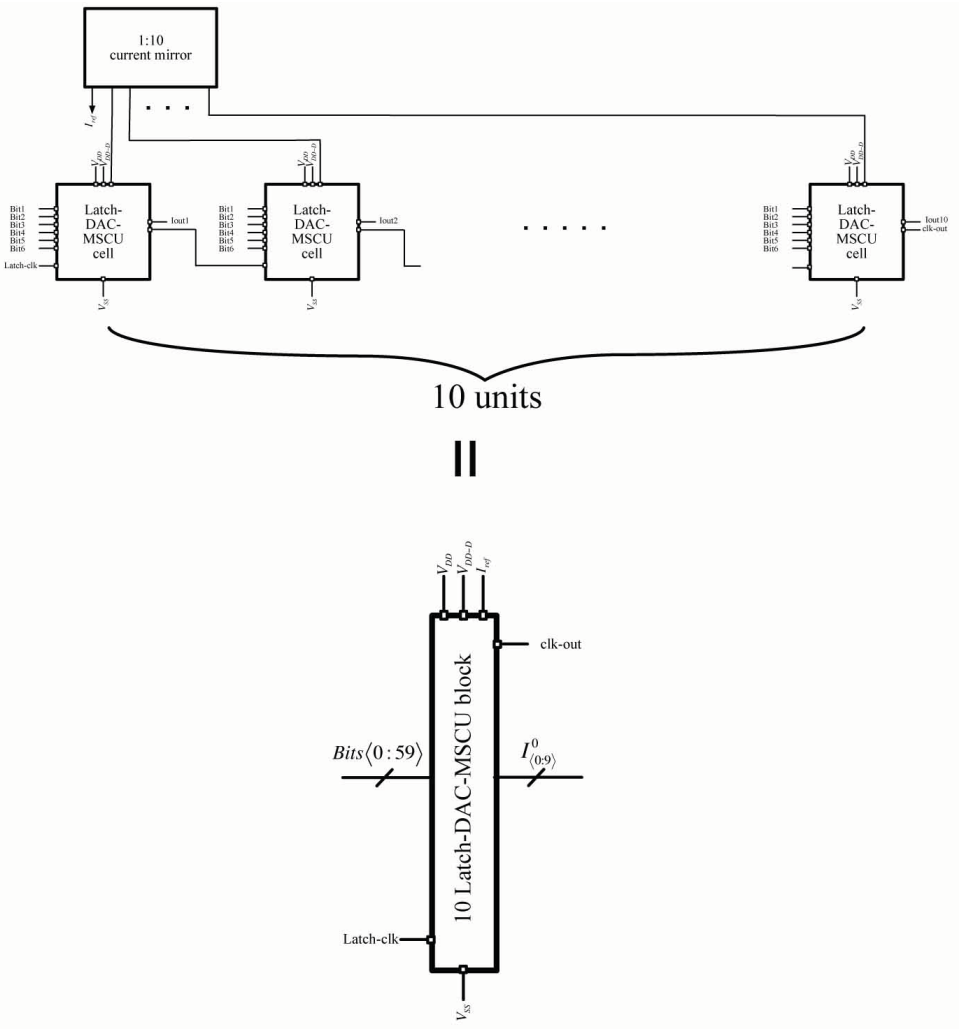


Figure 6.10: 10-cell Latch-DAC-MSCU block

to each Latch-DAC-MSCU cell. Moreover, the buffers shown in this design are used to compensate the effects of wiring in the layout floor plan.

6.2.3 Design and Schematic of a Latch-DAC-MSCU Cell

Fig. 6.9 shows a Latch-DAC-MSCU cell. In this block a 6-bit latch, one Digital-to-Analog converter (DAC) and one *Magnitude and Sign Combiner Unit* (MSCU) are utilized. The architecture of a Latch-DAC-MSCU cell is illustrated in Fig. 6.12. The design of these building blocks are discussed in the following paragraphs:

6-bit Latch

The architecture of a 6-bit latch is shown in Fig. 6.13. In this design six D-type flip flops selected from the standard cell library of TSMC 90nm technology are used. The delay element is used after the sixth latch is to ensure that a reconstructed clock pulse is available for the next 6-bit latch block.

Digital-to-Analog Converter

As discussed before, the initial conditions are sent to the chip as binary information. The magnitudes of initial conditions are represented by five bits while the sixth bit shows the sign of initial condition. On the other hand, to perform analog decoding, current mode initial conditions are required. Therefore, initial conditions should be converted to analog current signals. To do this, a digital-to-analog converter (DAC) is required.

There are several choices for DAC structure such as: current switched DACs, binary weighted DACs, segmented DACs, R2R ladder DACs and delta-sigma DACs.

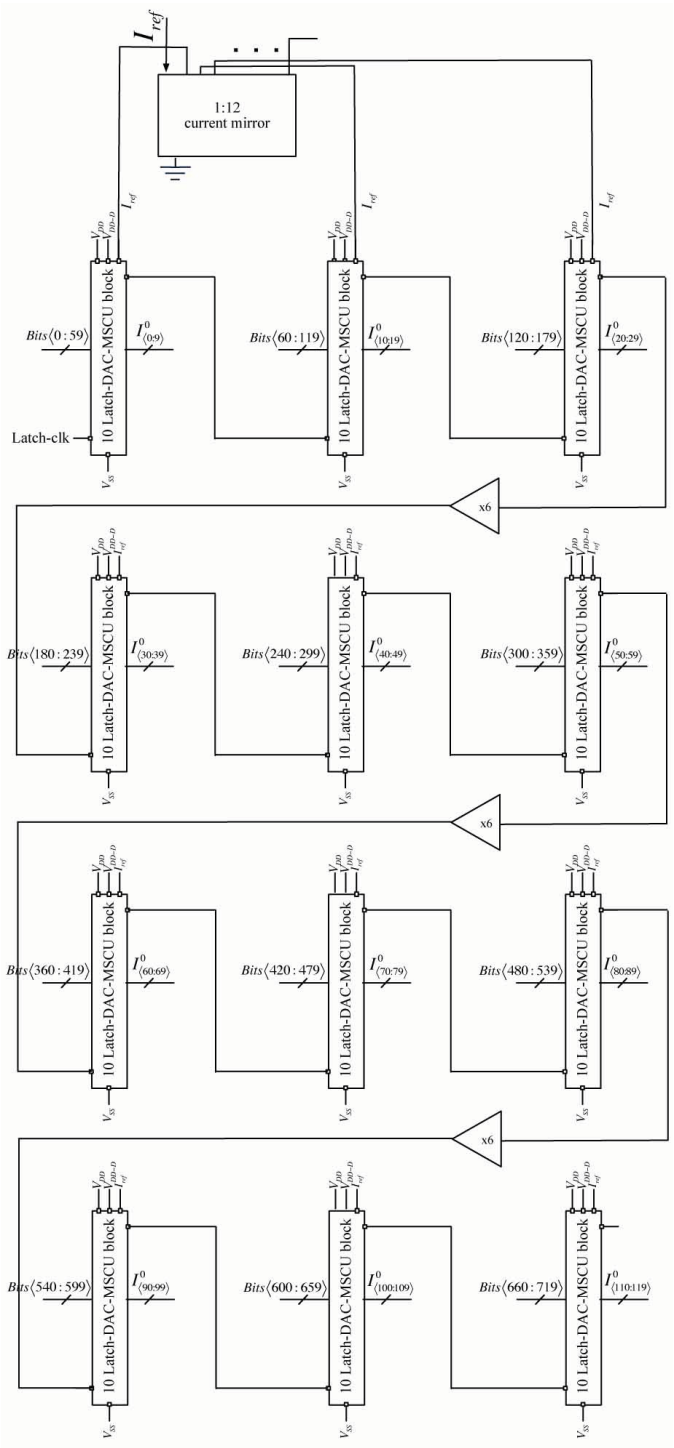


Figure 6.11: Architecture of Latch-DAC-MSCU

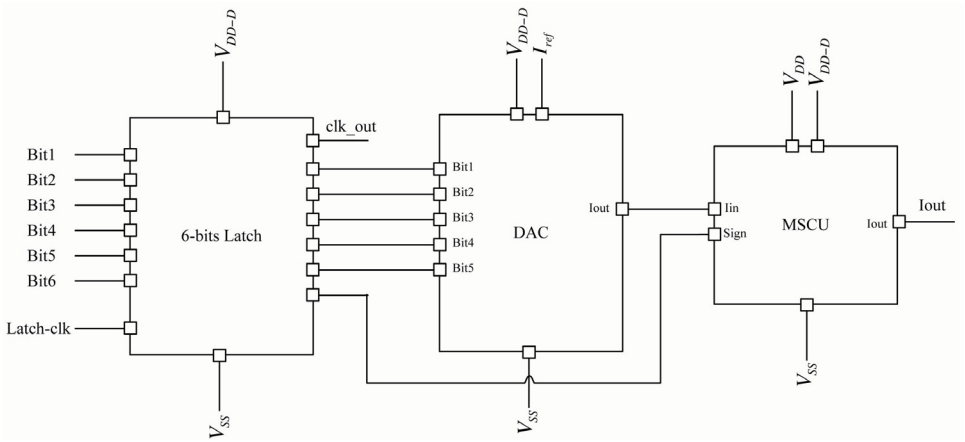


Figure 6.12: Architecture of a Latch-DAC-MSCU cell

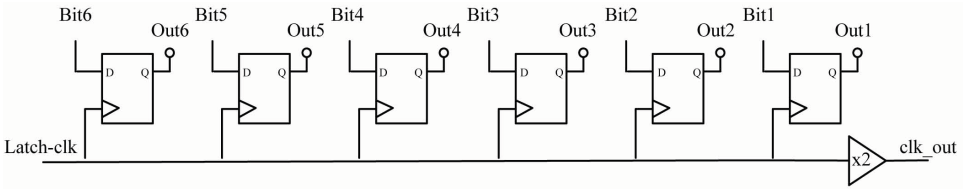


Figure 6.13: Design of a 6-bit latch

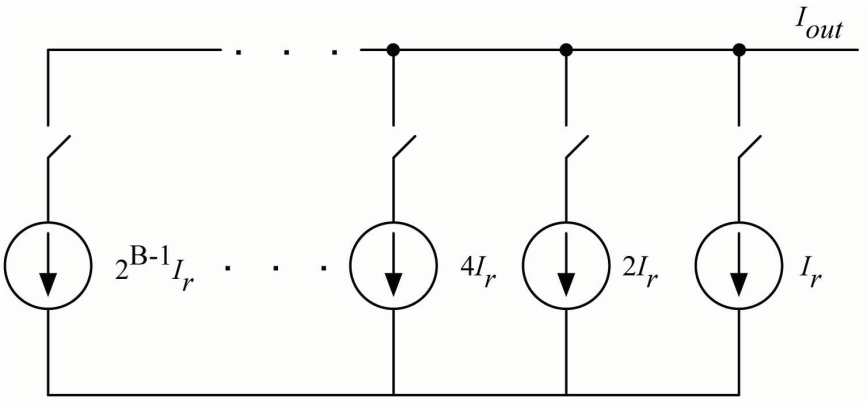


Figure 6.14: Binary weighted DACs structure

The objective is to find a relatively simple, small and accurate DAC. Based on the discussion in Section 6.1, since the DAC requires only 5-bit resolution the best choice is a binary weighted DAC (binary weighted current sources DAC). The structure of these DACs is shown in Fig. 6.14. The advantages of binary weighted DACs are their simple structure and low number of required switches. The disadvantage of this type of DAC is that their Differential Non-Linearity (DNL) is high. In other words, inaccuracy due to mismatch can be high, making them unsuitable for high resolution DACs.

Based on Fig. 6.14, 5 current sources are required for a 5-bit resolution DAC. Another parameter which needs to be designed is I_r. In Section 6.1 we have shown that the analog decoder can decode the received codeword correctly if the maximum LLR values are limited to 7. In this design the maximum I_{out} corresponding to DAC word "11111" is 1085nA. Therefore, each LLR maps to 155nA. It means that 155nA, 310nA, 465nA, ... represent LLR of 1, 2, 3, ..., respectively. On the other hand each bit corresponds to 35nA. Hence, I_r in Fig. 6.14 is 35nA. The schematic of the designed DAC is illustrated in Fig. 6.15.

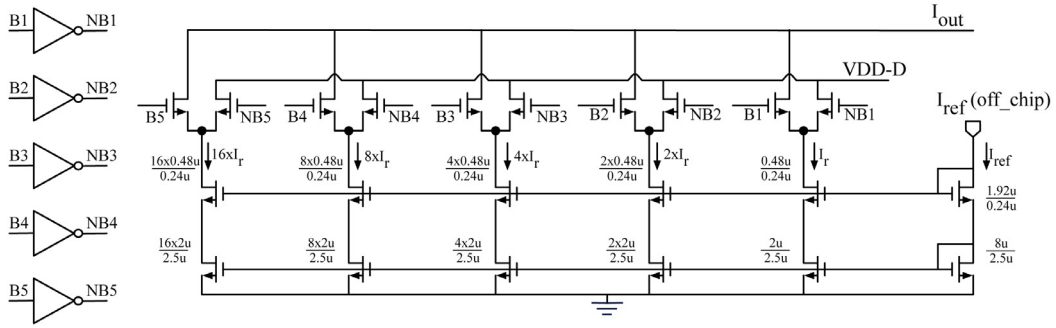


Figure 6.15: Binary weighted DACs structure

In Fig. 6.15, the transistors at the top of the schematic act as switches, while transistors used in the cascode structure at the bottom of the circuit act as current sources. At first, digital data B1, B2, B3, B4 and B5 are applied to NOT gates, chosen from the standard cell library, to obtain signals NB1, NB2, NB3, NB4 and NB5. These signals are connected to the gates of the switches. If the signal is high ($B_i = 1V$), the switch is on and V_{DS} of that particular switch is close to zero. Therefore, the current source is connected to I_{out}. On the other hand, $NB_i = 0V$ and the corresponding switch is off. In the same way if $B_j = 0$ then $NB_j = 1$, therefore the current is disconnected from I_{out} and it is connected to VDD. Transistor sizes are reported on the schematic while all switches have $\frac{W}{L} = \frac{0.5um}{100nm}$.

I_{ref} in Fig. 6.15 is provided by an off-chip source through the current mirrors. Due to offset current of parallel paths and limitation in off-chip current source, I_{ref} is chosen to be at least 4 times bigger than the current required for DAC operation. Therefore, I_{ref} is 140nA and it scales down to 35nA required for the DAC.

It should be noted that all the switches in the DAC schematic are low threshold voltage transistors while the rest are high threshold transistors. In TSMC 90nm technology transistors with three different threshold voltages are available. These

transistors are:

Transistor type	Threshold voltage
Standard threshold voltage (SVT)	313 mV
Low threshold voltage (LVT)	208 mV
High threshold voltage (HVT)	424 mV

Table 6.1: Transistors vs their threshold voltage

The designed DAC unit has been simulated using Spectre. Simulation results are summarized in the Table 6.2.

	Ideal Case	Simulation result
I_{ref}	140nA	141nA
I_r	35nA	35.74nA
$2 \times I_r$	70nA	70.17nA
$4 \times I_r$	140nA	140.1nA
$8 \times I_r$	280nA	280.6nA
$16 \times I_r$	560nA	561.2nA

Table 6.2: Spectre simulation results for DAC unit without mismatch while the output of the DAC is connected to a P-type current mirror

DAC word	Mismatch		Transient Response	
	$\frac{\Delta I}{I}$ for the worst case (over 100 iterations)	$\sigma_{norm} = \frac{\sigma}{AVG}$	$T_{d_{(L \to H)}}$	$T_{d_{(H \to L)}}$
0 0 0 0 1	10.69%	4.06%	71 ns	37 ns
1 0 0 0 0	8.67%	3.07%	18 ns	3 ns
1 1 0 0 0	9.13%	3.15%	14.6 ns	6.8 ns
0 0 0 1 1	8.02%	3.28%	25 ns	15 ns
0 0 1 1 0	8.25%	3.25%	16 ns	10 ns
0 1 1 1 1	8.86%	3.25%	15 ns	8.5 ns

Table 6.3: Mismatch simulation results for DAC unit

Mismatch analysis have been performed on the DAC unit. Simulation results for some sample DAC words are shown in Table 6.3. ΔI is the difference between the ideal case (no mismatch) and the case with mismatch. The current I represents the

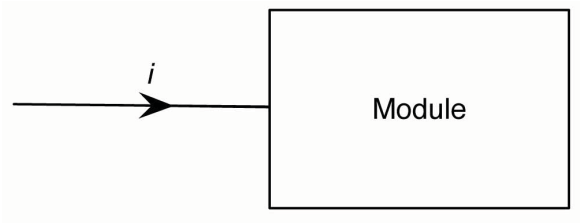


Figure 6.16: Current direction convention

ideal case (no mismatch). Also σ_{norm} is the standard deviation of mismatch over the average of mismatch. Moreover, Table 6.3 shows transient response of the DAC unit where $T_{d_{(L\rightarrow H)}}$ is the time delay when I_{out} changes its state from low to high and $T_{d_{(H\rightarrow L)}}$ is the case when I_{out} changes from high to low.

Magnitude and Sign Combiner Unit (MSCU) for DAC

Before discussing the MSCU, a positive and a negative current direction should be defined. Fig. 6.16 shows the direction for a positive current. Any current which flows in the opposite direction of i is negative.

One of the most commonly used blocks in the TS-LDPC analog decoder is the current mirror. Current mirrors conduct current in one of the two forms shown in Fig. 6.17. If we write a KCL on the super node identified with the dashed line in Fig. 6.17(a), we have,

$$-i_1 - i_2 = 0 \longrightarrow i_2 = -i_1. \tag{6.1}$$

In the same way, if we write a kcl on the super node in Fig. 6.17(b), we have,

$$i_1 + i_2 = 0 \longrightarrow i_2 = -i_1. \tag{6.2}$$

Therefore, it can be concluded that for each current mirror, input and output currents have the same magnitude but in opposite directions. The current mirror in Fig. 6.17(a) represents an N-type current mirror while Fig. 6.17(b) corresponds to a P-type current mirror. In this design, we use a cascode current mirrors as illustrated in Fig. 6.18. It can be seen that the module in Fig. 6.17(a) works like the N-type current mirror of Fig. 6.18(a). Also, the module shown in Fig. 6.17(b) corresponds to the P-type current mirror of Fig. 6.18(b). Based on the current direction convention agreed in Fig. 6.16, N-type mirrors are used for positive currents while their output has negative direction and P-type mirrors are utilized for negative currents and their outputs currents are positive. It should be noted that both i_1 and i_2 in Fig. 6.17 are positive quantities. It can be summarized that current mirrors, regardless of their type, have the the same magnitude but opposite direction.

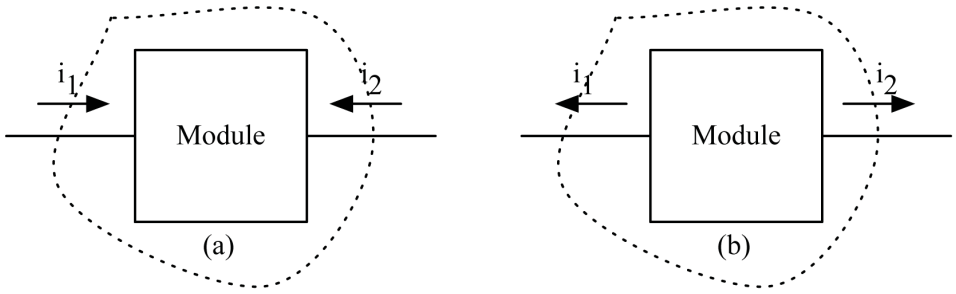


Figure 6.17: Current direction in current mirrors

As already discussed, the magnitude of the initial condition is converted to an analog current using the DAC unit. Fig. 6.9 shows that the output current of the DAC unit passes to the MSCU block along with the sign bit. The MSCU block is responsible for combining the magnitude and sign of the initial condition. In fact the MSCU block decides on the direction of current according to the sign bit received by the decoder. Based on the current direction convention defined above, the output

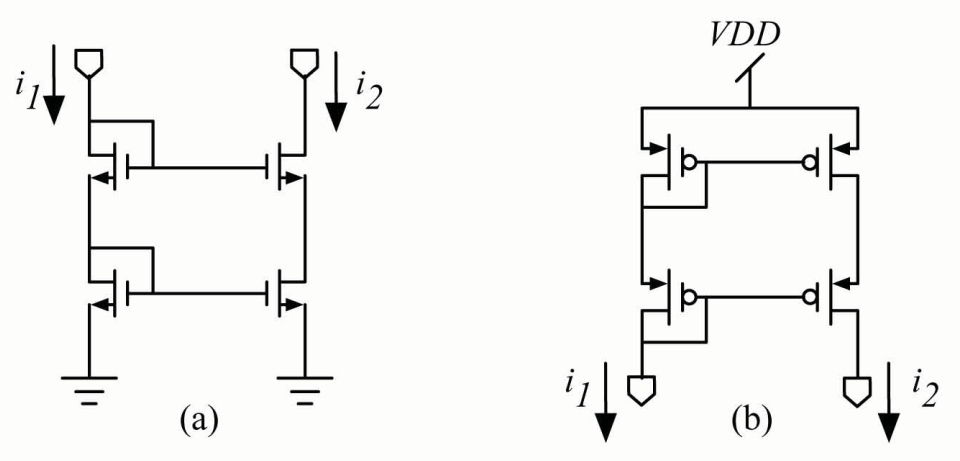


Figure 6.18: Cascode current mirrors (a) N-type (b) P-type

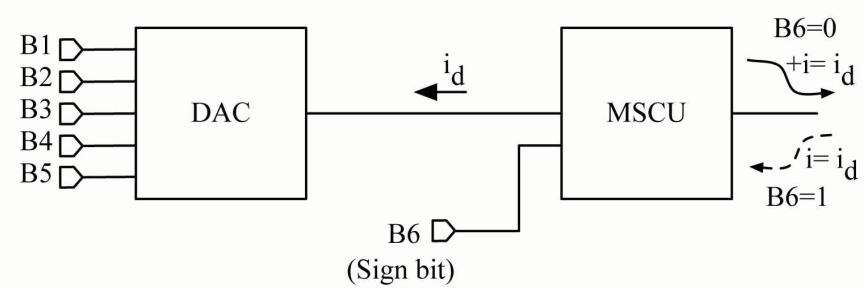


Figure 6.19: MSCU block's output based on the sign bit and DAC current

of the MSCU should be a positive current if the sign bit (the sixth bit of the initial condition word) is 0 and the current should be negative if the sign bit is 1. Fig. 6.19 shows these two possible scenarios. It can be observed that the output current of DAC is always negative, regardless of the sign of the DAC word.

Fig. 6.20 demonstrates the schematic of the MSCU block. This unit is designed specially to be used along with the DAC. The sign bit (B6) passes through a NOT gate to obtain NB6 signal. B6 and NB6 voltages control the gates of the complementary switch. If DAC word is positive then B6 is 0 and consequently NB6 is 1. Therefore, the upper complementary switch is on and current flows in the solid line direction

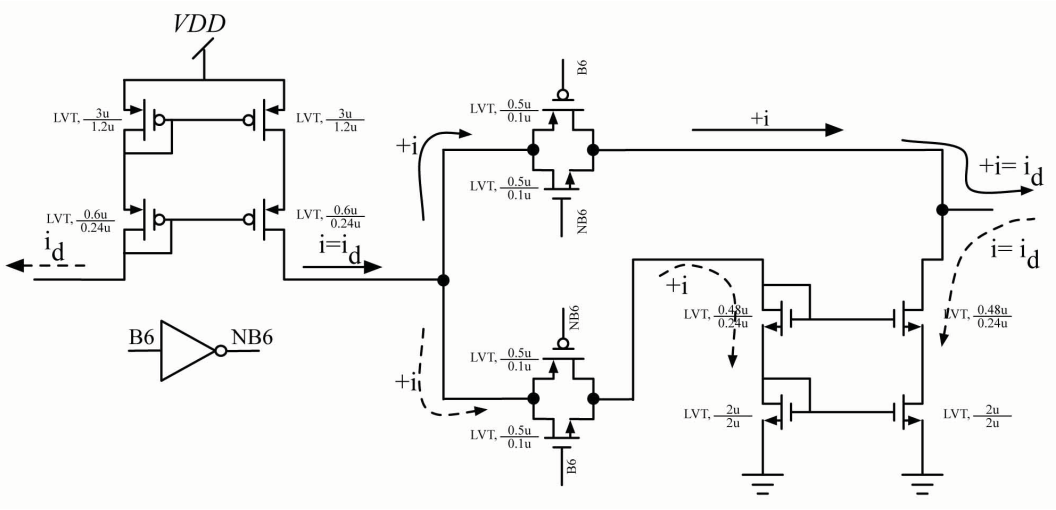


Figure 6.20: Magnitude and Sign Combiner Unit schematic

which results in positive current. If DAC word is negative, then lower complementary switch is on and current flows in the direction of dashed line, resulting a negative current at the output. The key point is that two complementary switches cannot be on simultaneously. It should be noted that the complementary switch is used to be able to conduct current entirely in one branch while current in the other branch is completely cut. That means current flows either in the solid line direction or in the dashed line direction.

6.3 Output Stage of Analog Decoder

The last block of the top level schematic presented in Fig. 6.1 is the Output-Stage block. This block is responsible for capturing the estimated coded bits $\widehat{c}$ and transmit these bits to an off-chip module. As it is shown in Fig. 6.21, the Output-Stage block consists of two 120-bit memory blocks as follows:

1. 120-bit parallel latch

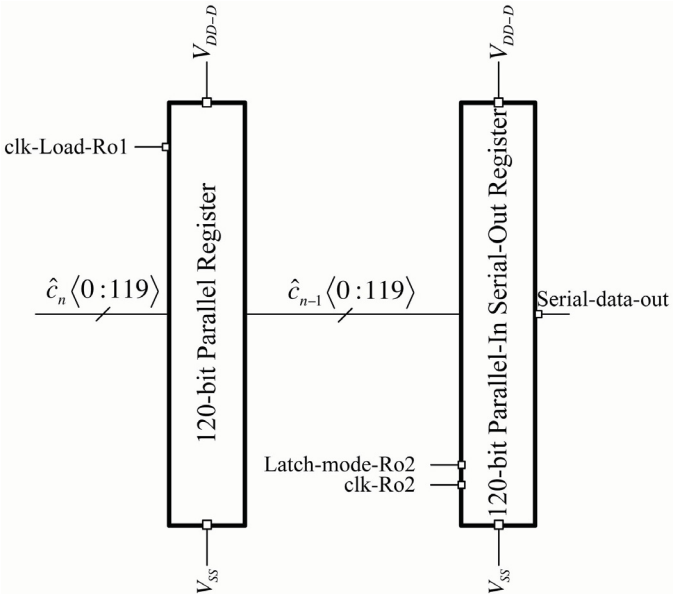


Figure 6.21: Output stage block

2. 120-bit serial/parallel latch

The 120-bit parallel latch consists of 120 latch blocks as shown in Fig. 6.22. To have perfect clocking, repeaters (NOT gates) are provided after every 10 latches. In the same way, the 120-bit serial/parallel latch block is designed. This latch is designed to have a capability of capturing the data and serially send the stored information to an off-chip module. Therefore, while the decoder is doing the decoding process, the data can be transferred to the off-chip module. If Latch-mod-Ro2=1V, with the positive edge of the clk-Ro2 the parallel data is captured in the latch block. If Latch-mod-Ro2=0V then with the positive edge of the clock, 0 enters to the 120th latch and the data stored in the first latch is sent to the off-chip module. Fig. 6.23 shows the scheme of the 120-bit serial/parallel latch. Due to the layout plan, after every 10 latches a repeater (NOT gates) are provided to assure a valid signal at the input of the next latch.

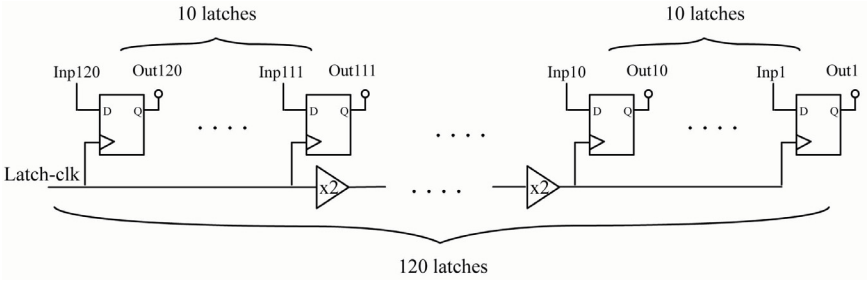


Figure 6.22: 120-bit parallel latch block

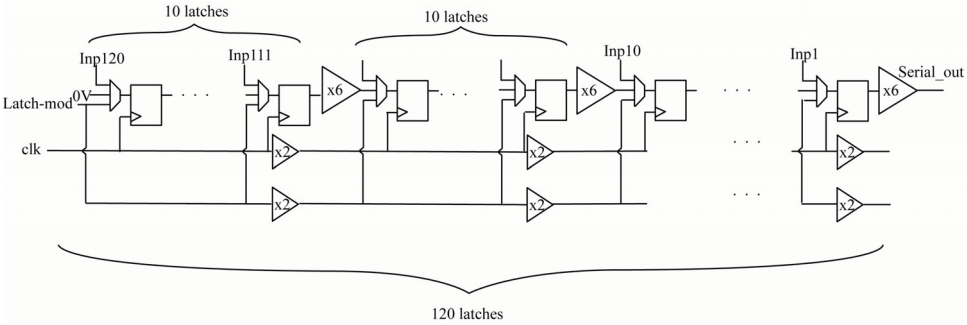


Figure 6.23: 120-bit serial/parallel latch block

6.4 Conclusion

In this chapter, the architecture and structure of the designed chip was proposed. It has been shown that the decoder chip consists of one analog decoder heart and two digital input and output stage blocks required to deliver the received signal to the analog decoder and transfer the estimated codedwords to the off-chip modules, respectively. Basics for adapting communication definitions such as LLR values to the physical values such current and voltage have been described. In this chapter, the structure of the input and output stages of the decoder chip was discussed. Circuit level designs of these blocks were illustrated in detail. The decoder analog heart will be discussed in Chapter 7.

Chapter 7

Design and Implementation of TS-LDPC Analog Decoder

In this chapter, the design and implementation of the Analog-Decoder block (Fig. 6.1) which is the analog heart of the decoder is discussed. The Analog-Decoder block is responsible for performing the decoding process based on the Min-Sum algorithm introduced in Chapter 4. This block receives analog initial conditions from the Input-Stage block or specifically MSCU units. After a certain time, known as decoding settling time (T_d), the hard-decided values (estimated codeword $\widehat{\mathbf{c}}$) are captured by a 120-cell latch at the Output-Stage block.

In this thesis the analog decoder for a (120,75) TS-LDPC code is designed and implemented. The Tanner Graph of such a code is illustrated in Fig. 6.4. This graph consists of two main building blocks: variable (VA) nodes and check (CHK) nodes. The number of required variable nodes is the same as the codeword length N and number of check nodes is the same as the number of parity check equations. Therefore, 120 variable nodes and $N - K = 45$ CHK nodes are required. The circuit

level architecture of VA nodes and CHK nodes is presented. In this chapter, VA node and CHK node circuits are simulated using Spectre. Based on the simulation results the designs are modified in order to achieve the required performance. Moreover, circuits are simulated in the presence of mismatch. It is observed that the variation of the output of each block is in the acceptable range defined in Chapter 5.

7.1 Architecture and Design of the Variable Node

Variable nodes are designed to perform equations 4.8 and 4.9. Both of these equations deal with the summation operation. Therefore variable nodes should perform addition task. If multiple wires carrying analog currents are connected together the resultant current follows Kirchhoff's current law. Therefore, variable nodes can be seen as a connection of wires carrying currents from check nodes to variable nodes. As discussed in Chapter 6, LLR messages M are represented by currents I as,

$$I = 155nA \times M. \tag{7.1}$$

A snapshot of a variable node 1 is illustrated in Fig. 7.1 . It is observed that both inputs and outputs of the analog decoder are connected to the variable nodes. Assume $I_{CH_1 \to V_1}$, $I_{CH_2 \to V_1}$ and $I_{CH_3 \to V_1}$ are currents from CHK 1, CHK 2 and CHK 3 to VA 1, respectively and I_1^0 is the initial condition from the Input-Stage block while I_{V_1} is the current that goes to the hard decision module. As discussed before, currents are acting as the LLR messages from check nodes to variable nodes and vice versa. Therefore, currents $I_{CH_1 \to V_1}$, $I_{CH_2 \to V_1}$, $I_{CH_3 \to V_1}$, I_1^0 and I_{V_1} represent $M_{CH_1 \to V_1}$, $M_{CH_2 \to V_1}$, $M_{CH_3 \to V_1}$, M_1^0 and M_{V_1}, respectively. In Fig. 7.1, using KCL

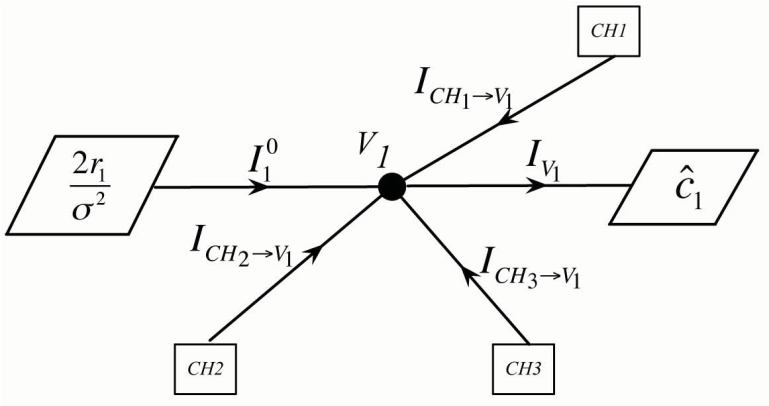


Figure 7.1: Snapshot of Variable node 1

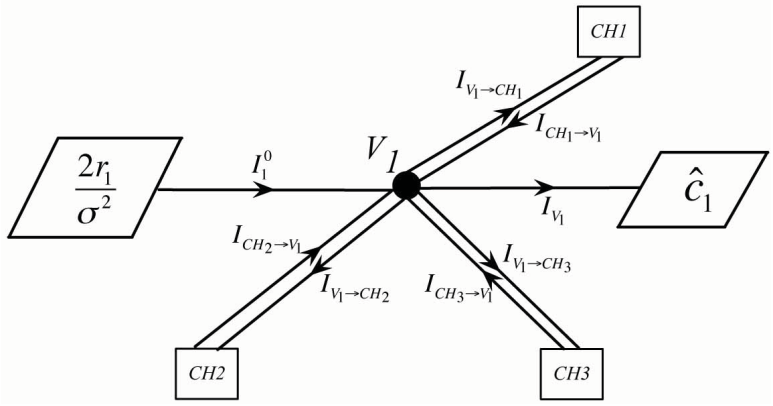


Figure 7.2: Implementation of bidirectional connections for variable node 1

$I_{V_1} = I_1^0 + I_{CH_1 \to V_1} + I_{CH_2 \to V_1} + I_{CH_3 \to V_1}$. This exactly implements equation 4.9 for variable node 1.

All the connections between variable nodes and check nodes are bi-directional connections. This means that currents can flow from a VA node to a CHK node and vice versa. Since, bi-directional connection is not physically feasible, we can assume that there is a one-way connection from CHK node to VA node and a one-way wire from VA to CHK node. This is shown in Fig. 7.2.

As mentioned in Chapter 4, during phase 2 of the Min-Sum algorithm messages

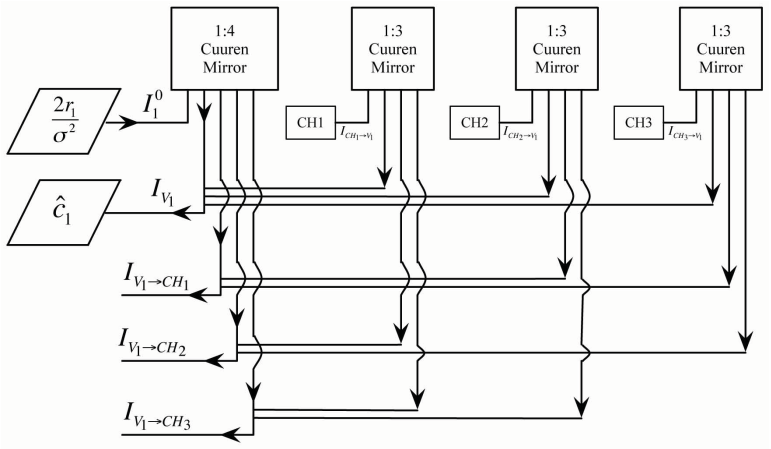


Figure 7.3: The circuit which performs equation 7.2

(currents) are sent from CHK nodes to VA nodes then VA nodes update their state and send back the new information to the CHK nodes based on equation 4.8. Let's expand equation 4.8 focusing on variable node 1. Considering equation 4.8 and Fig. 7.2, we have,

$$\begin{aligned} I_{V_1 \to CH_1} &= I_1^0 + I_{CH_2 \to V_1} + I_{CH_3 \to V_1} \\ I_{V_1 \to CH_2} &= I_1^0 + I_{CH_1 \to V_1} + I_{CH_3 \to V_1} \\ I_{V_1 \to CH_3} &= I_1^0 + I_{CH_1 \to V_1} + I_{CH_2 \to V_1}. \end{aligned} \tag{7.2}$$

These equations can be implemented using the circuit as shown in Fig. 7.3.

Now, let's expand equation 4.9 with respect to VA node1.

$$I_{V_1} = I_1^0 + I_{CH_1 \to V_1} + I_{CH_2 \to V_1} + I_{CH_3 \to V_1} \tag{7.3}$$

Comparing equations 7.2 and 7.3, we can derive new expressions for message updating in VA node 1 as follows,

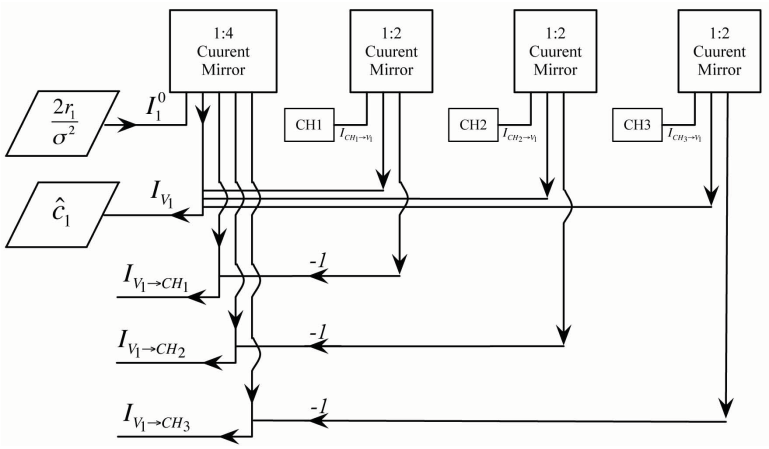


Figure 7.4: Architecture of variable node 1

$$
\begin{aligned}
I_{V_1 \to CH_1} &= I_{V_1} - I_{CH_1 \to V_1} \\
I_{V_1 \to CH_2} &= I_{V_1} - I_{CH_2 \to V_1} \\
I_{V_1 \to CH_3} &= I_{V_1} - I_{CH_3 \to V_1}.
\end{aligned} \tag{7.4}
$$

In these new expressions we reuse current I_{V_1} to update the variable node message. In this case, we can save some circuitry and wiring which results in saving die area. Equation 7.4 can be generalized for VA node message updating sequence as follows,

$$
I_{V_i \to CH_j} = I_{V_i} - I_{CH_j \to V_i} \tag{7.5}
$$

Implementation of VA node 1 using the new expression is shown in Fig. 7.4. In this figure, summation of signals and current carried by a wire with gain of -1, denote the subtraction.

As discussed before, LLR messages can be positive or negative. Consequently the currents representing LLR messages are either positive or negative. Therefore, the

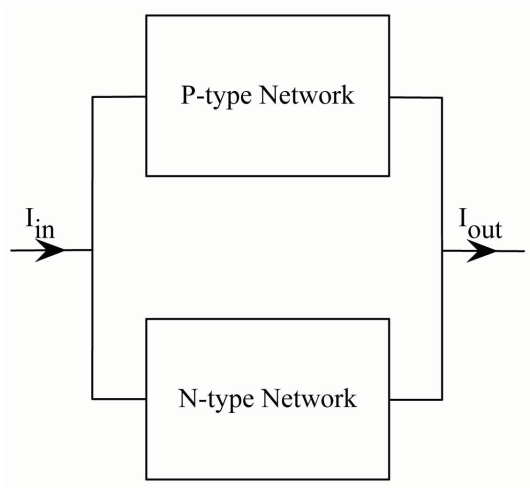


Figure 7.5: Architecture of a current buffer

current mirrors in Fig. 7.4 must handle both positive and negative currents. To do this, a combination of N-type and P-type current mirrors should be used where P-type mirror is connected to V_{DD} and N-type current mirror is at the bottom. This circuit is called current buffer. Fig. 7.5 shows a sample current buffer. It has been discussed that current mirrors replicate the input current at the output in the opposite direction ($I_{out} = -I_{in}$). In current buffers, the positive current ($I_{in} > 0$) is handled by the N-type mirror and the P-type mirror deals with the negative current ($I_{in} < 0$). N-type and P-type mirrors can be designed to be robust against mismatch, noise and non-linearity. In this thesis, cascode current mirrors in Fig. 6.18, are used in the current buffer structure. Fig. 7.6 illustrates circuitry of a current buffer. To implement the VA node structure 1:4 and 1:2 current buffers are required. The schematics of these current buffers are depicted in Fig. 7.7. It should be noted that all the transistors used in the current buffers are LVT transistors.

Based on the current buffers illustrated in Figs. 7.6 -7.7, we design the structure of a variable node. The structure of a variable node with degree of 3 is implemented

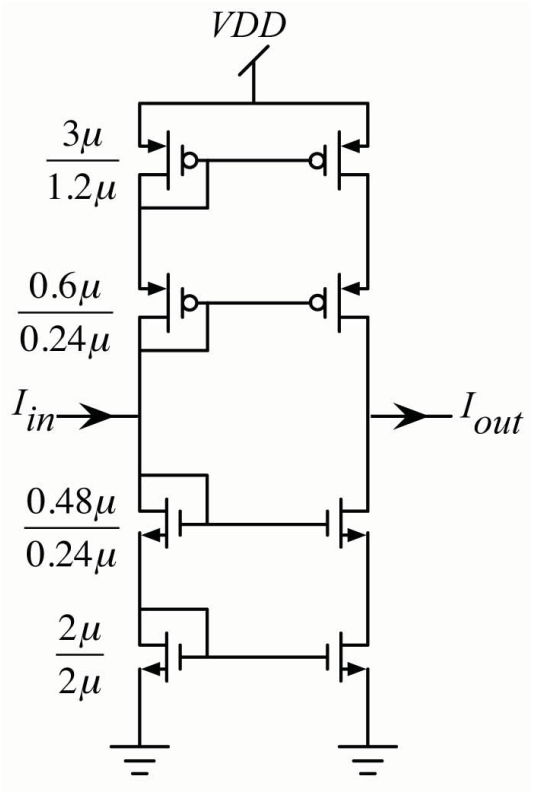


Figure 7.6: One-input one-output (1:1) current buffer

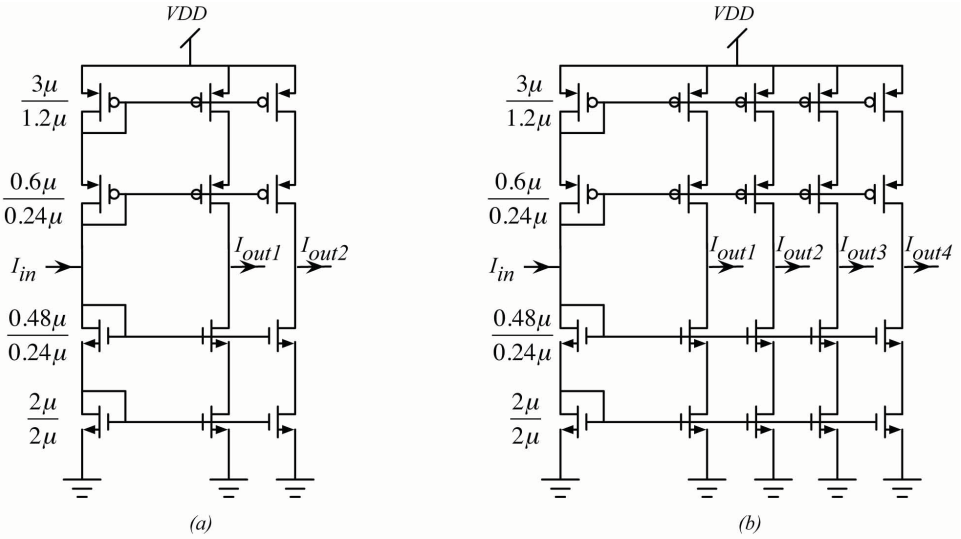


Figure 7.7: Circuitry level scheme of: (a) one-input two-output (1:2) current buffer (b) one-input four-output (1:4) current buffer

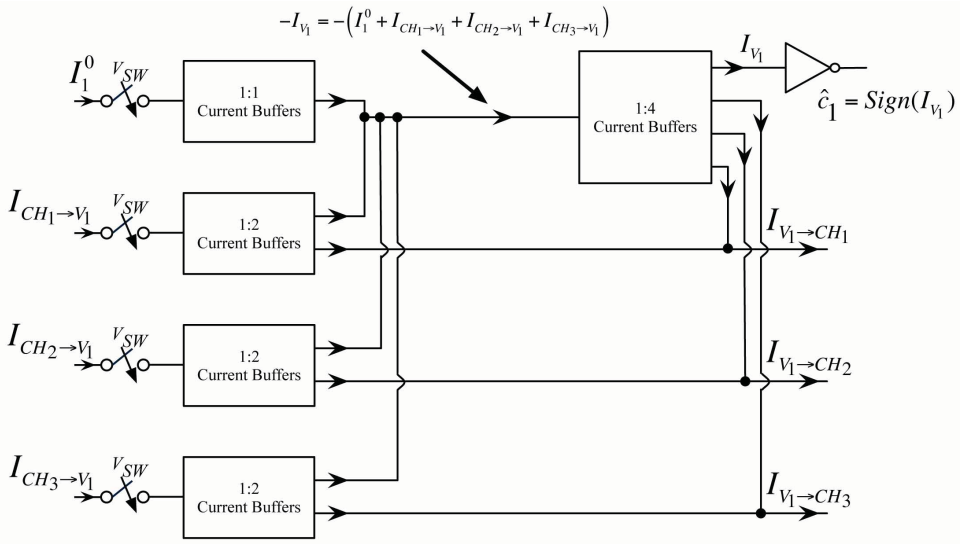


Figure 7.8: Circuit level structure of variable node 1

in Fig. 7.8. The currents at the end of the first stage of the current mirrors are replicas of the input current but in opposite direction. These currents are added up at node 1, according to the equation 4.9, the summed current is $-I_{V_i}$. When this current passes through the second current mirror the output is I_{V_i}. At node 2, I_{V_i} and the negative of the inputs are summed, this simulates the subtraction in equation 7.5.

We may use a hard decision to decide on the estimated coded bit based on equation 4.10. To make the decision, the current I_{V_i} is passed through a NOT gate. If the current is positive the parasitic capacitance at the input of the NOT gate is charged up, therefore the output is 0V. If the current is negative then the parasitic capacitance at the input of the gate is discharged and the output voltage or $\widehat{c_i}$ is 1V. This scenario is depicted in Fig. 7.9. The estimated codeword is captured after passing the settling time.

Variable nodes are simulated using Spectre. Two snapshots of the DC analysis

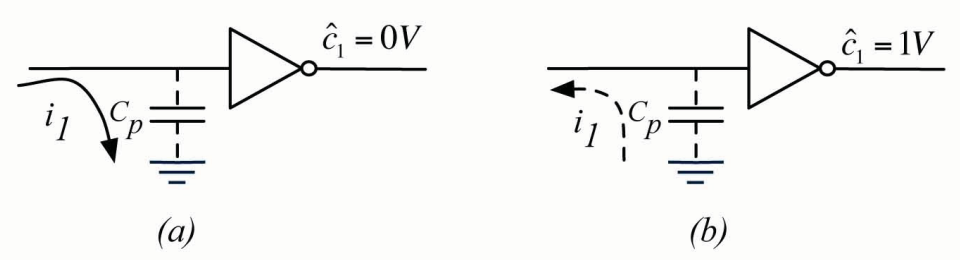


Figure 7.9: Decision making scenario

results are shown in Figs. 7.10 and 7.11. In Fig.7.10, initial condition $I_{V_1}^0$ is $+500nA$, $I_{CH_1 \to V_1}$ is $+100nA$, $I_{CH_3 \to V_1}$is $+300nA$ and $I_{CH_2 \to V_1}$ changes from $-1.5\mu A$ to $+1.5\mu A$. It can be observed that the output current from the VA node 1 to the CHK node 1 ($I_{CH_1 \to V_1}$) is the summation of all the signals except $I_{CH_1 \to V_1}$ and changes from $-700nA$ ($-1.5\mu A + 300nA + 500nA = -700nA$) to $2300nA$ ($+1.5\mu A + 300nA + 500nA = 2300nA$). The voltage $\widehat{c_i}$, the estimated coded bit, is high when the summation of all currents is negative and it changes its state to low when the summation of all currents is positive. In Fig. 7.11, $I_{CH_2 \to V_1}$ changes from $1.5\mu A$ to $-1.5\mu A$. In this case $I_{CH_1 \to V_1}$ changes from $2300nA$ ($1.5\mu A + 300nA + 500nA = 2300nA$) to $-700nA$ ($-1.5\mu A + 300nA + 500nA = -700nA$). When the summation of the currents are positive then $\widehat{c_i}$ is low and when the summation of the currents changes its direction $\widehat{c_i}$ changes its value to high. Moreover, transient analysis is performed on the VA node circuit. One snap shot of the transient analysis is shown in Fig. 7.12. It is observed that the rise time is $26ns$ and the fall time is $34ns$. At lower input currents the rise time is increased to $27.5ns$ and fall time is decreased to $30ns$. The rise/fall time is defined as the time that the signal reaches to 90% of its final value.

Mismatch simulations are done for the variable node in Fig. 7.8 with 100 samples and results are summarized in table 7.1. It is noticed that the maximum variation

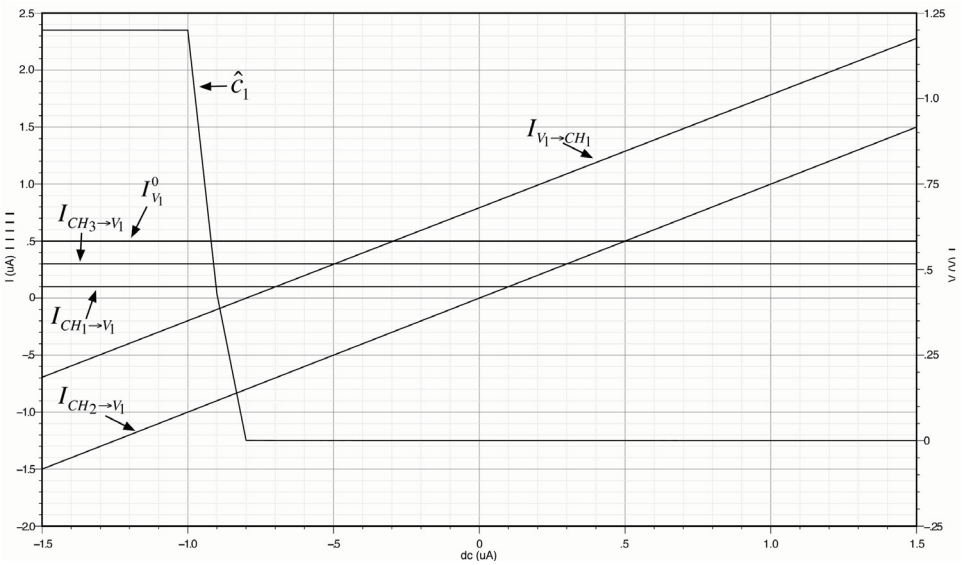


Figure 7.10: DC analysis on variable node 1-snapshot 1

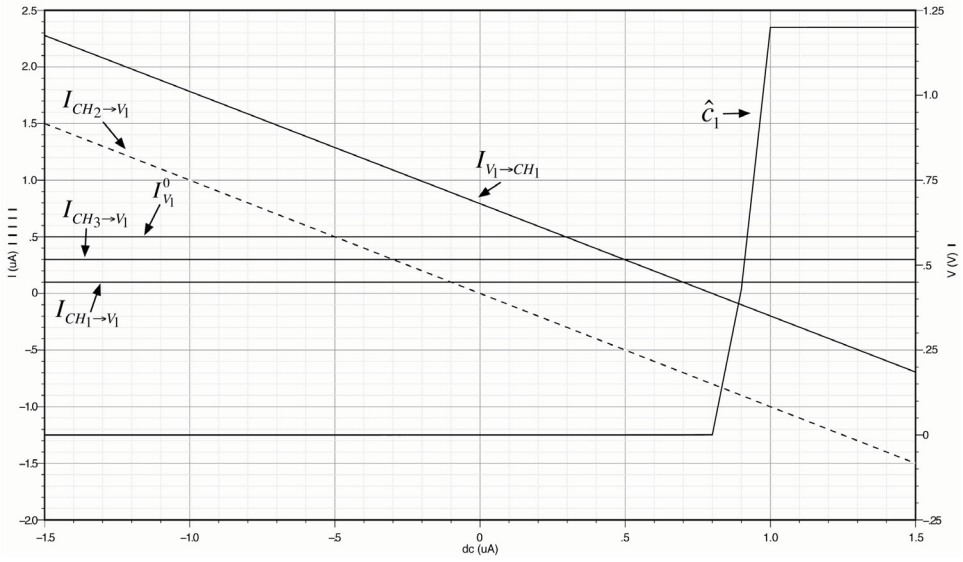


Figure 7.11: DC analysis on variable node 1-snapshot 2

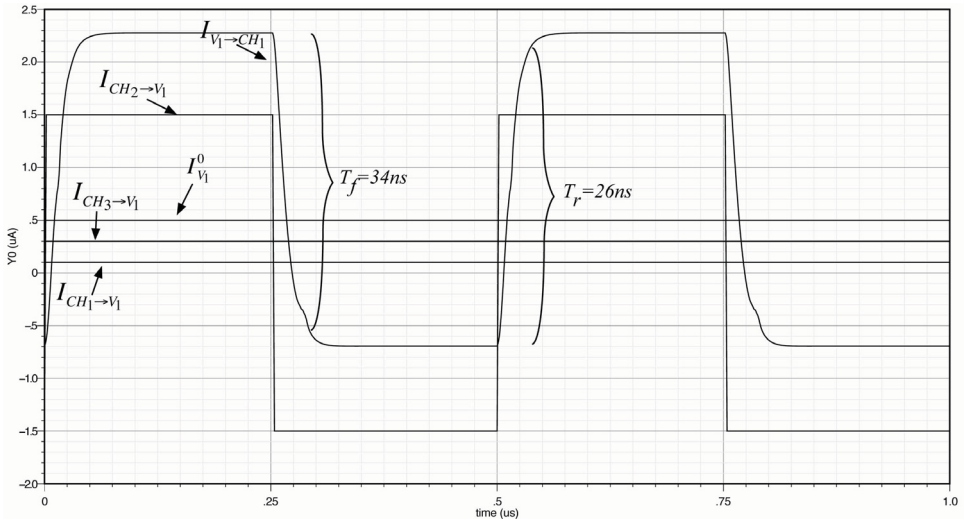


Figure 7.12: Transient analysis on variable node

due to mismatch satisfies the acceptable range defined in Chapter 5. From this, it is concluded that the effect of mismatch on VA nodes does not contribute to inaccuracy in the decoder context.

Case	Mismatch	
	$\frac{\Delta I}{I}$ for the worst case	$\sigma_{norm} = \frac{\sigma}{AVG}$
$-1.5\mu A \to +1.5\mu A$	19.5%	12.3%
$+1.5\mu A \to -1.5\mu A$	19.18%	13.88%

Table 7.1: Mismatch simulation results for Variable node unit

7.2 Architecture and Design of the Check Node

The check node's structure is the most complicated module in the design of the analog decoder performing Min-Sum algorithm. Check nodes update their messages based on equation 4.7. To compute the outgoing message of a check node j to variable node i, the following operations are performed:

1. Separate sign and magnitude of input signals.

2. Multiply the signs of all the received signals except the received signal from VA node i.

3. Find the minimum of the magnitudes of all the received signals except the received signal from VA node i.

4. Combine the result of the sign multiplier and the minimum of the magnitudes.

$$I_{CH_j \to V_i} = \underbrace{\underbrace{\left[\prod_{i' \in R_{j \setminus i}} sign(I_{V_{i'}} \to I_{CH_j})\right]}_{2} \cdot \underbrace{\min_{i' \in R_{j \setminus i}} \{|I_{V_{i'}} \to I_{CH_j}|\}}_{3}}_{4} \tag{7.6}$$

For the (120,75) TS-LDPC code, the degree of the check node is 8. It means that each check node is connected to 8 different variable nodes. CHK node j sends and receives messages from VA nodes through 8 bi-directional connections. Let's consider CHK node 1 in Fig. 6.4 as an example. This CHK node collects information from the VA nodes 1, 2, 3, 4, 5, 6, 7 and 120 then sends new information back to the variable nodes. Based on equation 7.6, the details of message updating are as follows,

$$\begin{aligned} I_{CH_1 \to V_1} = [&sign(I_{V_2} \to I_{CH_1}) \times sign(I_{V_3} \to I_{CH_1}) \times sign(I_{V_4} \to I_{CH_1}) \\ &\times sign(I_{V_5} \to I_{CH_1}) \times sign(I_{V_6} \to I_{CH_1}) \times sign(I_{V_7} \to I_{CH_1}) \\ &\times sign(I_{V_{120}} \to I_{CH_1})] \cdot min(|I_{V_2} \to I_{CH_1}|, |I_{V_3} \to I_{CH_1}|, \\ &|I_{V_4} \to I_{CH_1}|, |I_{V_5} \to I_{CH_1}|, |I_{V_6} \to I_{CH_1}|, |I_{V_7} \to I_{CH_1}|, \\ &|I_{V_{120}} \to I_{CH_1}|) \end{aligned} \tag{7.7}$$

$$
\begin{aligned}
I_{CH_1 \to V_2} = [&sign(I_{V_1} \to I_{CH_1}) \times sign(I_{V_3} \to I_{CH_1}) \times sign(I_{V_4} \to I_{CH_1}) \\
&\times sign(I_{V_5} \to I_{CH_1}) \times sign(I_{V_6} \to I_{CH_1}) \times sign(I_{V_7} \to I_{CH_1}) \\
&\times sign(I_{V_{120}} \to I_{CH_1})] \cdot min(|I_{V_2} \to I_{CH_1}|, |I_{V_2} \to I_{CH_1}|, \\
&|I_{V_4} \to I_{CH_1}|, |I_{V_5} \to I_{CH_1}|, |I_{V_6} \to I_{CH_1}|, |I_{V_7} \to I_{CH_1}|, \\
&|I_{V_{120}} \to I_{CH_1}|)
\end{aligned} \tag{7.8}
$$

$$
\begin{aligned}
I_{CH_1 \to V_3} = [&sign(I_{V_1} \to I_{CH_1}) \times sign(I_{V_2} \to I_{CH_1}) \times sign(I_{V_4} \to I_{CH_1}) \\
&\times sign(I_{V_5} \to I_{CH_1}) \times sign(I_{V_6} \to I_{CH_1}) \times sign(I_{V_7} \to I_{CH_1}) \\
&\times sign(I_{V_{120}} \to I_{CH_1})] \cdot min(|I_{V_1} \to I_{CH_1}|, |I_{V_2} \to I_{CH_1}|, \\
&|I_{V_4} \to I_{CH_1}|, |I_{V_5} \to I_{CH_1}|, |I_{V_6} \to I_{CH_1}|, |I_{V_7} \to I_{CH_1}|, \\
&|I_{V_{120}} \to I_{CH_1}|)
\end{aligned} \tag{7.9}
$$

$$
\begin{aligned}
I_{CH_1 \to V_4} = [&sign(I_{V_1} \to I_{CH_1}) \times sign(I_{V_2} \to I_{CH_1}) \times sign(I_{V_3} \to I_{CH_1}) \\
&\times sign(I_{V_5} \to I_{CH_1}) \times sign(I_{V_6} \to I_{CH_1}) \times sign(I_{V_7} \to I_{CH_1}) \\
&\times sign(I_{V_{120}} \to I_{CH_1})] \cdot min(|I_{V_1} \to I_{CH_1}|, |I_{V_2} \to I_{CH_1}|, \\
&|I_{V_3} \to I_{CH_1}|, |I_{V_5} \to I_{CH_1}|, |I_{V_6} \to I_{CH_1}|, |I_{V_7} \to I_{CH_1}|, \\
&|I_{V_{120}} \to I_{CH_1}|)
\end{aligned} \tag{7.10}
$$

$$
\begin{aligned}
I_{CH_1 \to V_5} = [&sign(I_{V_1} \to I_{CH_1}) \times sign(I_{V_2} \to I_{CH_1}) \times sign(I_{V_3} \to I_{CH_1}) \\
&\times sign(I_{V_4} \to I_{CH_1}) \times sign(I_{V_6} \to I_{CH_1}) \times sign(I_{V_7} \to I_{CH_1}) \\
&\times sign(I_{V_{120}} \to I_{CH_1})] \cdot min(|I_{V_1} \to I_{CH_1}|, |I_{V_2} \to I_{CH_1}|, \\
&|I_{V_3} \to I_{CH_1}|, |I_{V_4} \to I_{CH_1}|, |I_{V_6} \to I_{CH_1}|, |I_{V_7} \to I_{CH_1}|, \\
&|I_{V_{120}} \to I_{CH_1}|)
\end{aligned} \tag{7.11}
$$

$$
\begin{aligned}
I_{CH_1 \to V_5} = [&sign(I_{V_1} \to I_{CH_1}) \times sign(I_{V_2} \to I_{CH_1}) \times sign(I_{V_3} \to I_{CH_1}) \\
&\times sign(I_{V_4} \to I_{CH_1}) \times sign(I_{V_5} \to I_{CH_1}) \times sign(I_{V_7} \to I_{CH_1}) \\
&\times sign(I_{V_{120}} \to I_{CH_1})] \cdot min(|I_{V_1} \to I_{CH_1}|, |I_{V_2} \to I_{CH_1}|, \\
&|I_{V_3} \to I_{CH_1}|, |I_{V_4} \to I_{CH_1}|, |I_{V_5} \to I_{CH_1}|, |I_{V_7} \to I_{CH_1}|, \\
&|I_{V_{120}} \to I_{CH_1}|)
\end{aligned} \tag{7.12}
$$

$$
\begin{aligned}
I_{CH_1 \to V_5} = [&sign(I_{V_1} \to I_{CH_1}) \times sign(I_{V_2} \to I_{CH_1}) \times sign(I_{V_3} \to I_{CH_1}) \\
&\times sign(I_{V_4} \to I_{CH_1}) \times sign(I_{V_5} \to I_{CH_1}) \times sign(I_{V_6} \to I_{CH_1}) \\
&\times sign(I_{V_{120}} \to I_{CH_1})] \cdot min(|I_{V_1} \to I_{CH_1}|, |I_{V_2} \to I_{CH_1}|, \\
&|I_{V_3} \to I_{CH_1}|, |I_{V_4} \to I_{CH_1}|, |M_{V_5} \to M_{CH_1}|, |M_{V_6} \to M_{CH_1}|, \\
&|M_{V_{120}} \to M_{CH_1}|)
\end{aligned} \tag{7.13}
$$

$$
\begin{aligned}
I_{CH_1 \to V_5} = [&sign(I_{V_1} \to I_{CH_1}) \times sign(I_{V_2} \to I_{CH_1}) \times sign(I_{V_3} \to I_{CH_1}) \\
&\times sign(I_{V_4} \to I_{CH_1}) \times sign(I_{V_5} \to I_{CH_1}) \times sign(I_{V_6} \to I_{CH_1}) \\
&\times sign(I_{V_7} \to I_{CH_1})] \cdot min(|I_{V_1} \to I_{CH_1}|, |I_{V_2} \to I_{CH_1}|, \\
&|I_{V_3} \to I_{CH_1}|, |I_{V_4} \to I_{CH_1}|, |I_{V_5} \to I_{CH_1}|, |I_{V_6} \to I_{CH_1}|, \\
&|I_{V_7} \to I_{CH_1}|).
\end{aligned} \tag{7.14}
$$

We can break a signal into its magnitude and sign in a *Sign and Magnitude Extractor Unit* (MSEU). A *Sign and Magnitude Combiner Unit* (MSCU) collects the magnitude and based on the received sign, the output is produced. Finally a minimizer unit is responsible for finding the minimum value of its inputs and copying the minimum input at the output. The complementary sign ($\overline{Sign}$) of a current can be saved and treated as a $\overline{Sign}$ bit. If the current is positive the sign is 0V and $\overline{Sign}$ is 1V. In the same way, if the current is negative the sign is 1V and $\overline{Sign}$ is 0V. In this case, multiplying the signs is equivalent of XORing the sign bits or XNORing the $\overline{Sign}$ bits. The architecture of the circuit performing equation 7.7 is shown in Fig. 7.13.

The check node circuit should be able to perform required operations in equations 7.7 - 7.14. A complete CHK node can be implemented with the same concept as the circuit in Fig. 7.13. Considering equations 7.7 - 7.14, it can be observed that the sign and magnitude of each of the inputs of the CHK node is used 7 times to obtain 7 different outputs. To reuse a current mode signal, a current mirror should be employed. However, a voltage mode signal can be reused easily by making a direct connection to the desired voltage. Therefore, to reuse the magnitude of each of the

signals (currents) in 7 different places a 1:7 P-type current mirror is used. Since the sign of the input signals are represented in the form of voltage, the sign bit can be used in 7 different modules by simply crossing over connections.

Based on the above discussion and considering the circuit shown in Fig. 7.7 the architecture of a degree 8 CHK node is illustrated in Fig. 7.14. Three main blocks are used in this design:

1. Sign and Magnitude Extractor Unit (MSEU)

2. Minimizer

3. Sign and Magnitude Combiner Unit (MSCU)

The details of these blocks are presented in the following sections.

Switches at the input of variable nodes and check nodes in Figs. 7.14 and 7.8 are provided to control the decoding time T_d. By closing the switches operated by V_{SW}, the analog processor starts decoding. Currents representing LLRs start flowing between variable nodes and check nodes. At the end of each decoding cycle, following a decoding time, T_d, switches are opened to reset the analog decoder module. This is necessary to start an unbiased decoding process.

The design of the 1:7 P-type current mirror is shown in Fig. 7.15. The 7-input XNOR gates are designed using the XNOR gates available in the standard cell libraries of the TSMC90nm technology. The design of a 7-input XNOR gate is depicted in Fig. 7.16. NOT gates are directly chosen from the standard cell library.

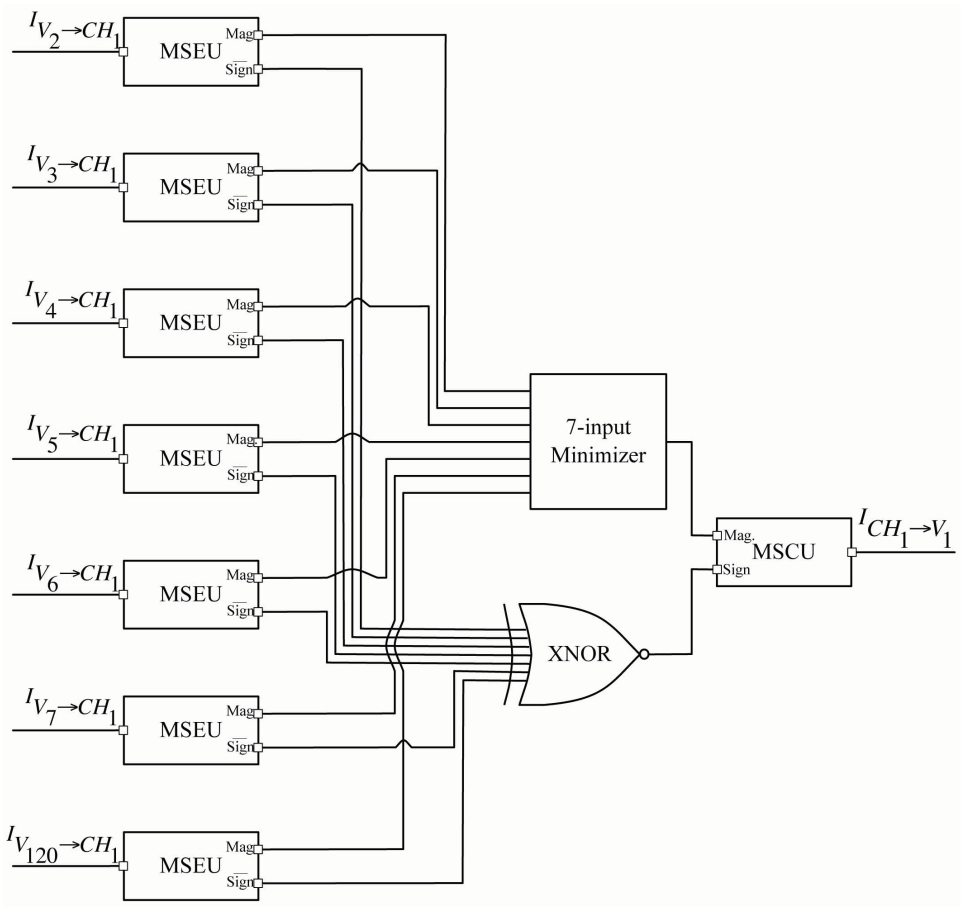


Figure 7.13: Architecture of the circuit performing equation 7.7

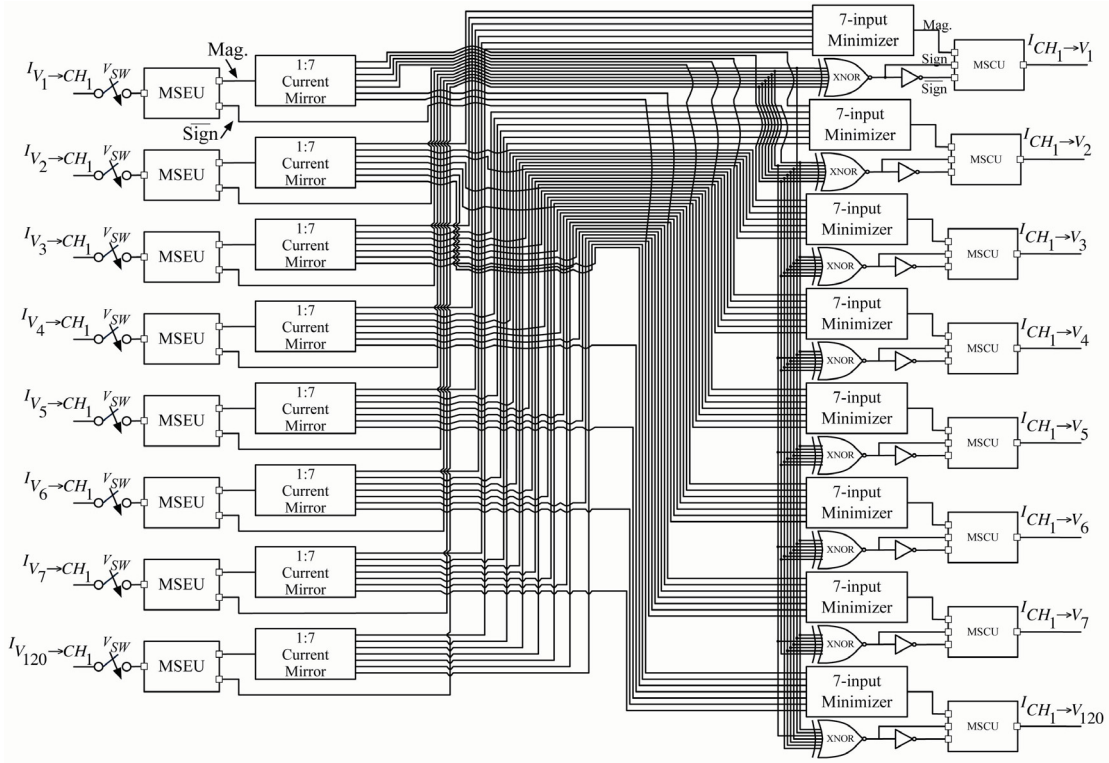


Figure 7.14: Architecture of the check node 1

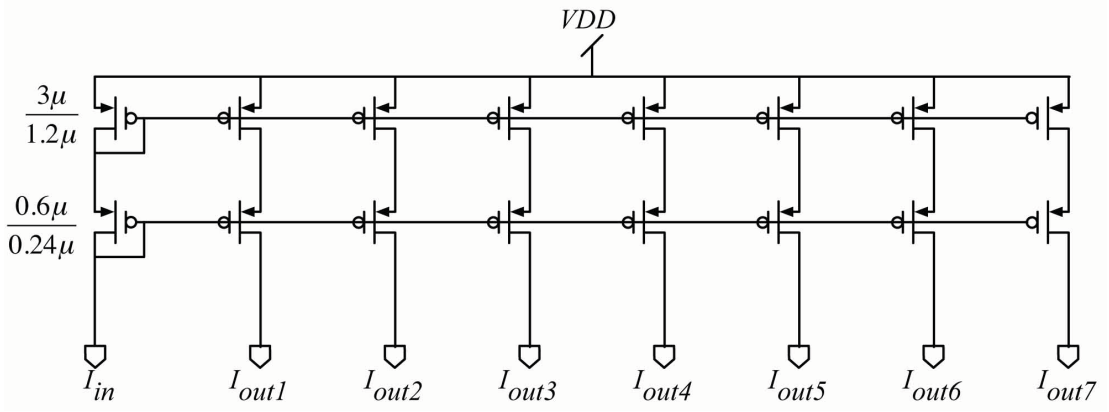


Figure 7.15: One-input seven-output (1:7) P-type current mirror

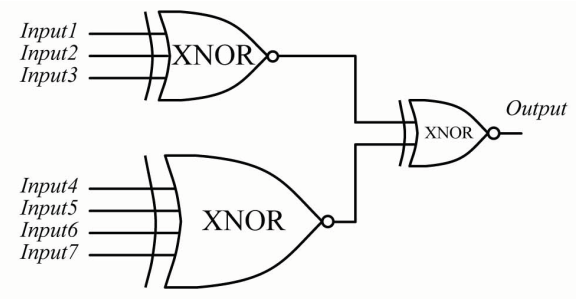


Figure 7.16: Design of a 7-input XNOR gate

7.2.1 Magnitude and Sign Extractor Unit (MSEU)

In equation 7.6, the sign and magnitude of the messages are treated separately. Therefore, for the first step magnitude and sign of the signal should be separated. The Magnitude and Sign Extractor Unit (MSEU) is designed to separate magnitude and sign of signals. The architecture of the MSE block is depicted in Fig. 7.17. This architecture is inspired by a circuit introduced in [28]. Fig. 7.14 shows that MSEU is followed by a P-type current mirror. This means that the MSEU sinks the current. In Fig. 7.17, regardless of the direction of the input current the output is negative or it sinks the current. Since the MSEU block should accommodate both positive and negative currents, the input stage should be a current buffer. Moreover, Fig. 7.17 illustrates that the output of the sign extractor unit is the complementary sign ($\overline{Sign}$). The sign extractor unit is comprised of two NOT gates as shown in Fig. 7.18. Two consecutive NOT gates are used to have a stable voltage at the output of the unit.

The circuit level design of the MSEU block is shown in Fig. 7.19. The input current i can be positive (solid line) or negative (dashed line). Positive current flows to the N-type cascode current mirror which results in activation of the two cascade

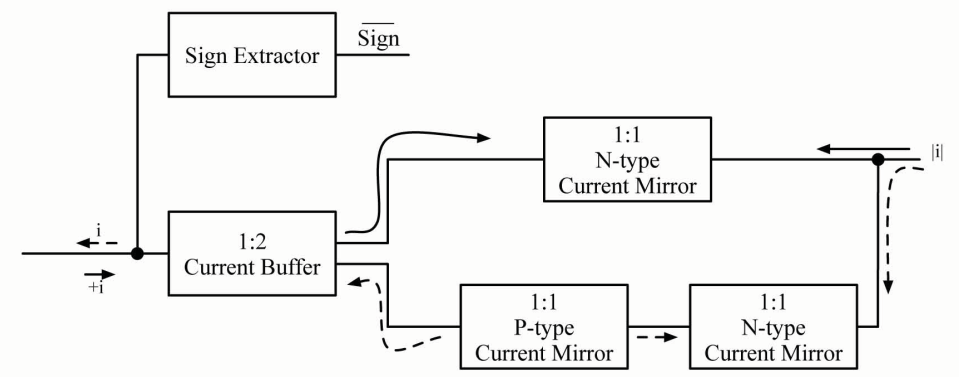


Figure 7.17: Architecture of the Sign and Magnitude Extractor Unit (MSEU)

P-type and N-type current mirrors at the bottom branch, respectively. Negative currents (dashed line) sinks the current from the P-type current mirror at the top and results in activation of the N-type mirror at the top branch. Fig. 7.19 shows that for both positive and negative input current the output current is negative with magnitude of $|i|$.

When the input current is positive, the current flows to the N-type current mirror and gives rise to the voltage at node A which makes $\overline{Sign}$ high voltage. In the same way, negative input current sinks current from the input stage P-type current mirror and the voltage at node A decreases which makes $\overline{Sign}$ low voltage. It is crucial to carefully chose the transistor sizes of the NOT gates in order to:

$$V_{th} = V_A|_{i=0} \tag{7.15}$$

where Vth is the switching threshold voltage of the NOT gates and $V_A|_{i=0}$ is the voltage of node A when the input current (i) is zero.

The simulation result showing MSEU performance is illustrated in Fig. 7.20. It can be seen that the output current is always negative. It means that the current

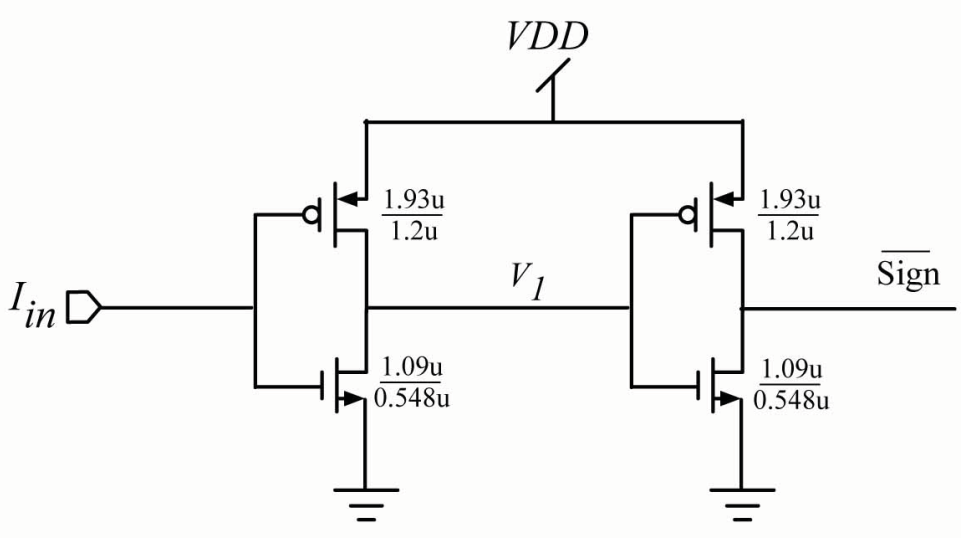


Figure 7.18: Sign extractor block

is always sunk from the current source. This is in agreement with the CHK node structure since the output of MSEU block is connected to a P-type current mirror as shown in Fig. 7.14.

7.2.2 Magnitude and Sign Combiner Unit (MSCU)

As it can be observed in Fig. 7.14, the outputs of the minimizer and the XNOR gate are applied to the MSCU. The MSCU block is responsible for combining the sign and magnitude of the messages obtained in a CHK node. This operation is the fourth step in determining equation 7.6. At this step, the output of the sign multiplication unit and magnitude minimizer module are combined. If the multiplication of the signs is +1 then the output is $min(inputs)$ and if the multiplication of the signs is -1 then the output is $-min(inputs)$. Therefore, it can be concluded that the output of the MSCU block is in the form of $(-1)^{Sign}.min(inputs)$. The circuit performing this operation is shown in Fig. 7.21. Since the output of the minimizer is always

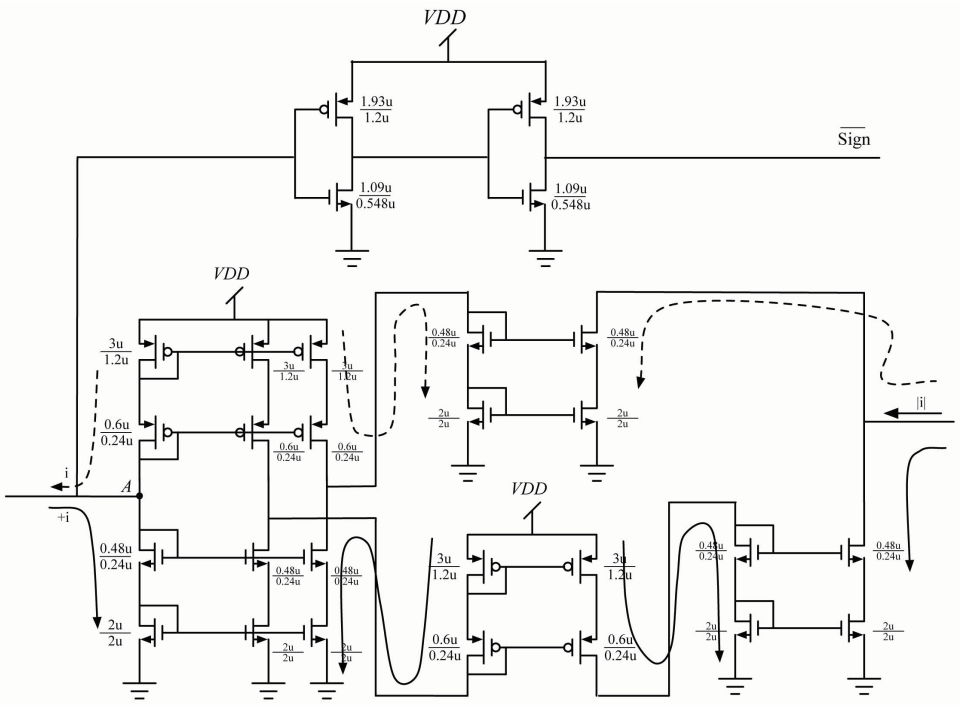


Figure 7.19: Sign and Magnitude Extractor Unit (MSEU)

positive, then MSCU circuit is designed based on the positive input current. To have a flawless switching performance, we use complementary MOS switches in the MSCU design. As it can be seen on the design, a complementary voltage set is applied to the inputs of the switch. In this circuit, the controlling voltages of the switch are $Sign$ and $\overline{Sign}$. It is obvious that at each time instance, if $Sign = 0V$ then $\overline{Sign} = 1V$ and vice versa.

If the result of the sign production is positive, $Sign = 0V$ and $\overline{Sign} = 1V$, the upper switch is ON and the current passes to the output in positive direction while the lower switch is OFF and no current flows in the lower branch. In the same way, if the sign production is negative, $Sign = 1V$ and $\overline{Sign} = 0V$, the upper switch is OFF and no current passes through the upper branch. Since the lower switch in

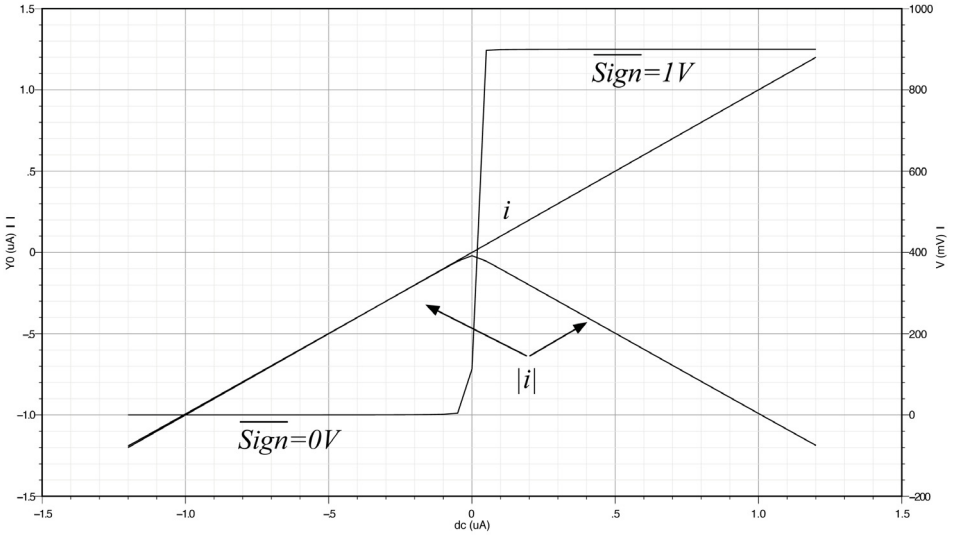


Figure 7.20: Simulation results for Magnitude and Sign Extractor Unit (MSEU)

ON, therefore all the current i goes to the N-type current mirror. Consequently the current at the output is in negative direction.

7.2.3 Minimizer

Step three in performing equation 7.6, is determining the minimum of the inputs' magnitudes. It should be noted that the minimizer circuit is the most complicated part of a check node. There are a lot of circuits available to carry out the minimizing operation.

Analog *Winner-Take-All* (WTA) circuits act as a maximizer and are used widely in neural networks. WTAs select the maximum from a set of inputs. The counterpart of WTA circuits is called *Loser-Take-All* (LTA). The LTA module chooses the minimum from a selection of inputs. In fact the functionality of an LTA circuit is the same as a minimizer. In the literature, minimizers are mostly designed based on WTA circuits. However there are some independent designs for LTA circuits. There are two types

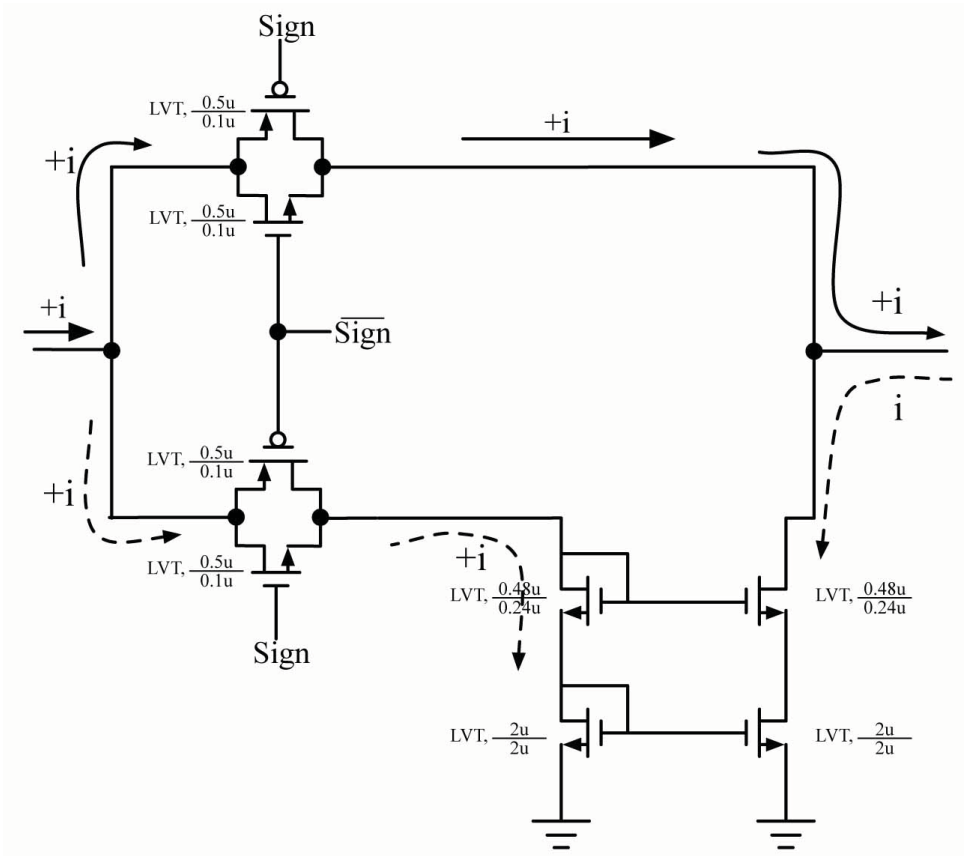


Figure 7.21: Schematic of the Magnitude and Sign Combiner Unit (MSCU)

of WTAs and LTAs: voltage mode and current mode.

In our design, the LLR values are mapped to the currents and the decoder circuit is designed based on the current mode modules. If a voltage mode LTA is used then intermediate current to voltage and voltage to current converters are required. These converters reduce the accuracy of the circuit. Hence, if a current mode LTA (minimizer) is used the converters are avoided. Moreover, current-mode circuits could have better bandwidth and dynamic range compared to voltage-mode circuits [87]. Therefore, current mode LTA circuits are preferred in this design.

Fig. 7.22 shows a general architecture of an LTA designed based on a WTA. In this

design inputs are subtracted from a fixed current (I) and the resultant current ($I-I_{in}$) is fed to the WTA. Therefore, the maximum output corresponds to the minimum I_{in}. We can subtract the output of the WTA from I to obtain I_{in}. In current mode analog decoders, the LLR values are increased at high SNR. This results in growth in current magnitude corresponding to the LLR. Therefore, at high SNR, I_{in} has larger value and the current $I - I_{in}$ has a smaller value which makes it sensitive to the mismatch. Consequently this current cannot be copied perfectly to the output and introduces an error. Hence, minimizers designed based on the current mode WTAs perform poorly at higher SNR results in degradation of the error performance of the decoder at high SNR. This problem affected the error performance of the analog decoder designed in [28].

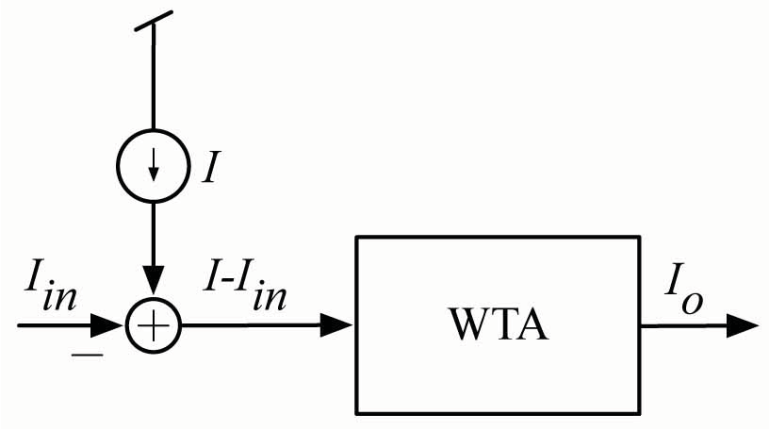


Figure 7.22: General architecture of an LTA designed based on a WTA

Based on the above discussion a direct design of the current mode LTAs is more suitable for our analog decoder circuit. LTA circuits proposed in [88–95] were investigated thoroughly based on accuracy, speed, stability, chip area and power consumption factors. We have designed the minimizer based on the one proposed in [89]. Modifications are done mainly based on accuracy, mismatch and stability criteria.

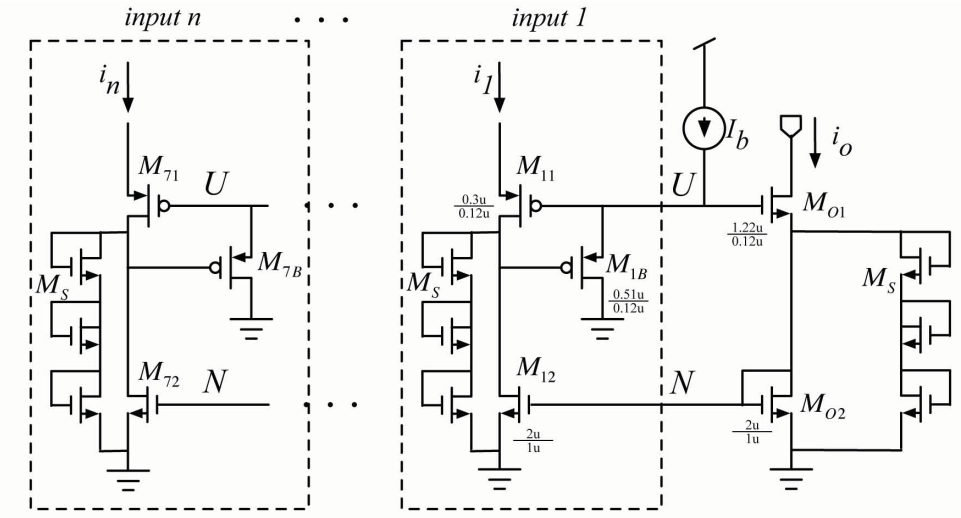


Figure 7.23: Loser-Take-All (LTA) circuit

Fig. 7.23 shows the LTA (minimizer) used in this design. Its accuracy is improved by adjusting bias current and transistor sizes. Diode-connected transistor stacks (M_S) are added to the circuit to reduce latency and increase its accuracy by providing additional paths for current.

The LTA shown in Fig. 7.23 has n input stages and one common stage consisting of transistors M_{o1}, M_{o1} and the current source I_b. Transistor M_{o1} acts as a voltage controlled current source while the gate of M_{o2} controls the source voltage of the M_{o1}. On the other hand the gate of M_{o2} is connected to M_{i2} of the ith stage. In each stage, M_{i2} converts input current to the voltage at its drain. If the current i_i increases, then the drain voltage of M_{i2} increases. This results in decreasing the V_{SG} of the M_{iB} transistor. Therefore M_{iB} enters cut-off region and no current goes through this transistor. This scenario is repeated for all M_B transistors except the M_B of the minimum input stage. Eventually the current I_b flows completely to the M_B transistor of the minimum stage. Therefore, the the voltage of node U and the

gate voltage of M_{o1} is controlled by the minimum cell. Since the drain and the source of the transistor M_{o1} are almost fixed, therefore increase in the gate voltage of the transistor results in moving the operation point toward the triode region. We know that in the triode region the transistor works as a linear voltage controlled current source, therefore the current of M_{o1} is controlled by the voltage of node U which varies based on the minimum input current.

Basically, the linearity between the input and the output is determined by M_B transistors. Hence, the $\frac{W}{L}$ ratio of M_B transistor should be chosen in order to have an output as close as possible to the input. On the other hand the step response of the circuit depends on the size of M_B transistors.

Consider a 2-input LTA. Assume that i_1 changes from $0A$ to $1.2\mu A$ while $i_2 = 500nA$. The simulation result is shown in Figs. 7.24 and 7.25. It can be observed that when I_1 increases, the gate voltage of M_{1B} increases as well. This decreases the current passing through M_{1B}. Consequently the share of M_{2B} of the current I_b increases. When I_1=I_2, gate voltages of both transistors M_{1B} and M_{2B} are equal. Further increase in I_1 results in controlling the voltage at node U (gate of transistor M_{o1}) by M_{2B}.

It has been already discussed that the MS algorithm is an approximation of the SP algorithm. In both algorithms variable nodes perform summation operation. Hence, the approximation is done on the check node function. The difference between the check node operation in SP algorithm and MS algorithm can be visualized. Figs. 7.26 and 7.27 show the behaviour of check node performing SP algorithm and minimizer approximation based on minimizer in Fig. 7.23. To have a better understanding, the error between the performance of digital MS algorithm check node and SP algorithm

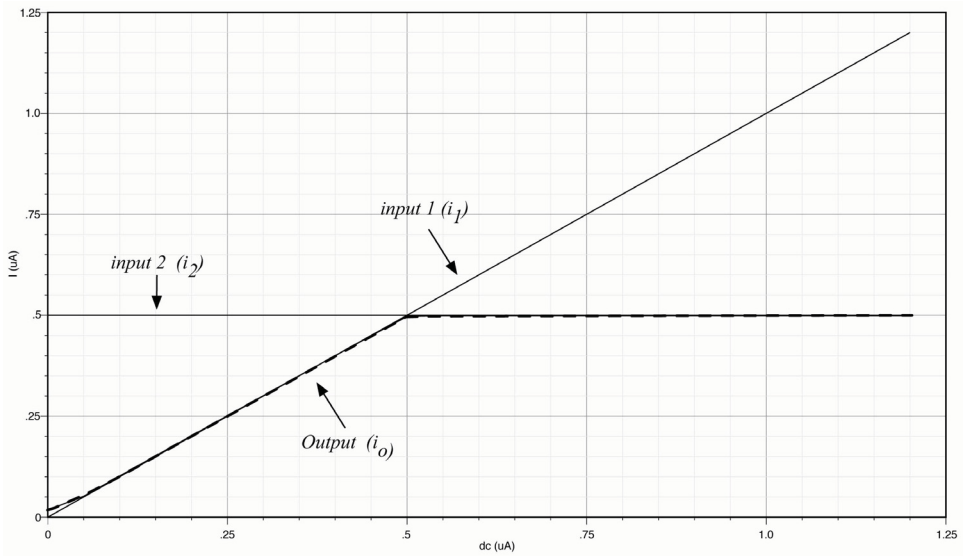


Figure 7.24: 2-input minimizer performance

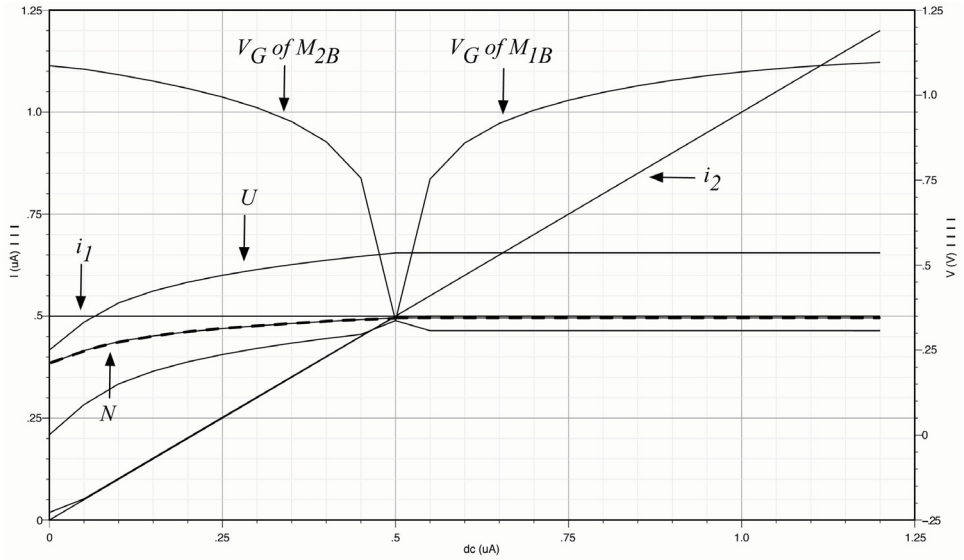


Figure 7.25: DC analysis for 2-input LTA

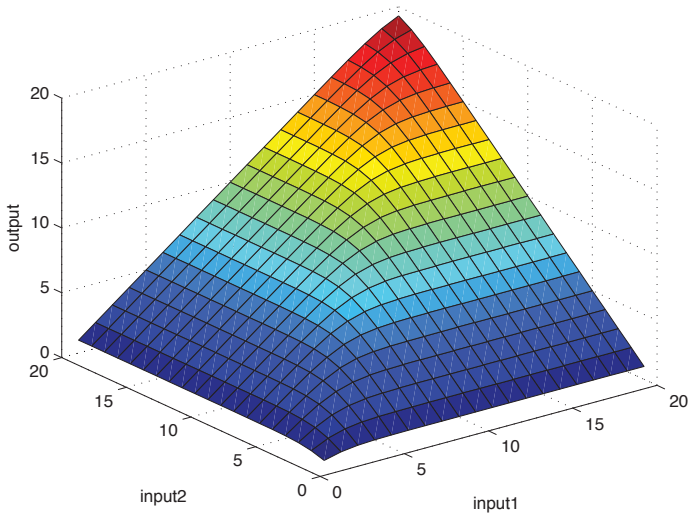


Figure 7.26: Behaviour of an ideal check node performing SP algorithm

check node is depicted in Fig. 7.28. Moreover, the error between the analog MS algorithm check node (designed in this thesis) and SP algorithm check node is illustrated in Fig. 7.29. It should be noted that the current range of $100nA - 1100nA$ is mapped to LLR $0 - 20$. In other words, each LLR is represented by $50nA$. It can be observed that the maximum error when a digital MS algorithm CHK node is used is around 0.7 and the error when the analog MS algorithm CHK node is used is also around 0.7. On the other hand, the error surface shows that at high LLRs corresponding to higher currents the error for all inputs are a little bit more. We believe that this will not cause any problem in the decoder performance since it has been shown in Chapter 5 that even if the LLR is saturated to 10, still the error performance is acceptable. If V_{DD} is increased this problem will be solved but error at low LLRs are higher. It is not desirable to add error at low LLR since the reliability of messages is much less than intermediate and high LLRs.

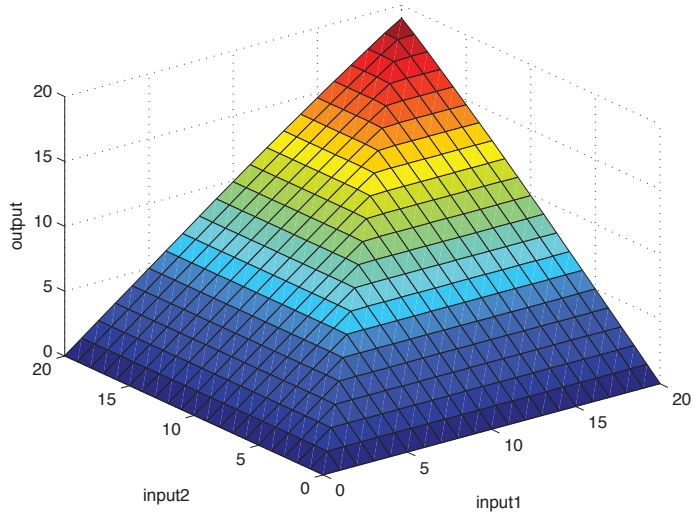


Figure 7.27: Behaviour of a check node performing MS algorithm using analog minimizer

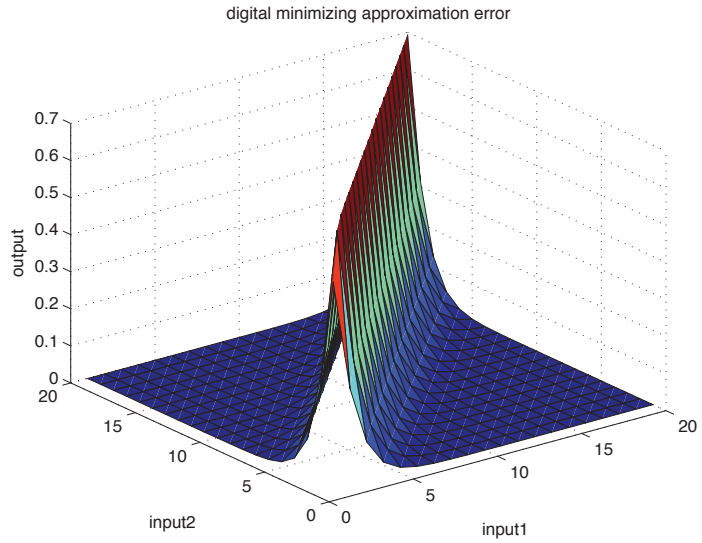


Figure 7.28: Error between an SP algorithm check node and digital MS algorithm check node

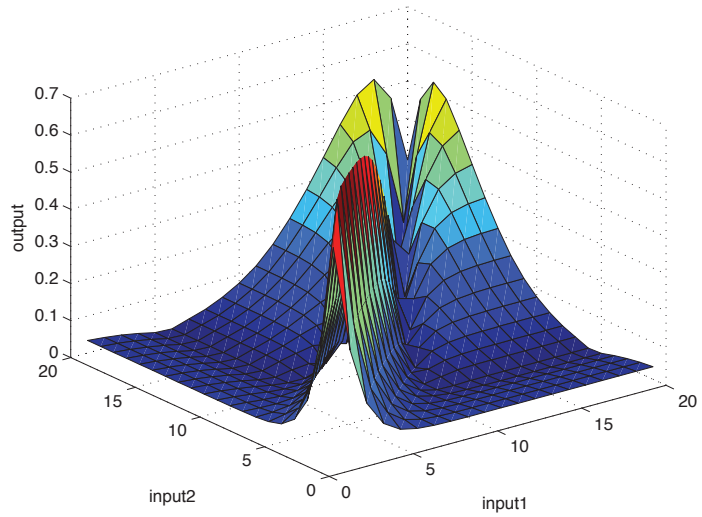


Figure 7.29: Error between an SP algorithm check node and analog MS algorithm check node

High degree minimizers

As shown in Fig. 7.14, eight 7-input minimizers are required to implement a check node. To implement 7-input minimizers different design strategies are considered. Five different circuits have been designed to perform minimizing operation. These designs are summarized as:

- Design 1: 7-input minimizer circuit designed directly based on the minimizer shown in Fig.7.23.
- Design 2: 7-input minimizer designed by cascading 2-input minimizers.
- Design 3: 7-input minimizer designed by cascading 4-input and 2-input minimizers.

Minimizer design	Transient Response				Frequency Response	
	$I_{min} = 500nA$		$I_{min} = 100nA$		f_{-3dB}	f_{-3dB}
Design number	T_r	T_f	T_r	T_f	@ $500nA$	@ $1uA$
Design 1	138ns	141ns	246ns	136ns	80MHz	105MHz
Design 2	72ns	73ns	147ns	89ns	28.8MHz	31.19MHz
Design 3	84ns	71ns	194ns	100ns	97MHz	128MHz
Design 4	62ns	63ns	114ns	100ns	87MHz	107MHz

Table 7.2: Comparison of transient responses and frequency responses of the designed minimizer circuits

Design number	Dissipated Current	Power Consumption (μW)	Estimated Gate Area (μm^2)
Design 1	$30.76\mu A$ - $94.09\mu A$	36.91-112.9	565.965
Design 2	$67.36\mu A$ - $109.82\mu A$	80.88-131.79	557.8
Design 3	$75.68\mu A$ - $120.48\mu A$	90.82-144.57	613.15
Design 4	$64.7\mu A$ - $114.46\mu A$	77.64-137.35	649.78

Table 7.3: Comparison of dissipated current, power consumption and estimated gate area, when inputs changes from $0-1.2\mu A$ in the designed minimizer circuits

- Design 4: 7-input minimizer designed by cascading 5-input and 3-input minimizers.

As mentioned before, current mirrors are required to reuse intermediate current mode signals. Comparison between the designed minimizers are performed based on some important criteria such as mismatch, $-3dB$ cut-off frequency, delay, accuracy, area and power consumption. Results of these comparisons are summarized in Tables 7.2-7.4 . Simulation results have shown that the accuracy of the direct design is the most promising one. In this work, direct design for 7-input minimizer is used based on accuracy comparison and simulation results are summarized in Tables 7.2-7.4.

The check node proposed in Fig. 7.14 has been simulated using different input scenarios. Some of the step response simulation results are summarized in table 7.5. Two snapshots of transient simulations are shown in Figs. 7.30 and 7.31. Fig. 7.32

Design number	Mismatch		Estimated Energy efficiency / CHK node
	$\frac{\Delta I}{I}$ for the worst case	$\sigma_{norm} = \frac{\sigma}{AVG}$	
Design 1	13.06%	5.4%	1.02 pJ/b
Design 2	24.47%	9.88%	1.54 pJ/b
Design 3	18.45%	6.85%	1.4 pJ/b
Design 4	18.9%	7.64%	1.13 pJ/b

Table 7.4: Comparison of mismatch, energy efficiency/each check node and accuracy in the designed minimizer circuits

shows the performance of the check node when one input changes from $-1.2\mu A$ to $1.2\mu A$ while the other input is $500nA$ and the rest of the inputs are at $1.3\mu A$. It can be observed that there is a distortion around $0A$. When $I_{min} = 0A$ the output is $-64nA$ which is close to 1 LLR. To resolve this issue a current source of $100nA$ is added at the input of LTA and subtracted at the output of LTA. Simulation result is shown in Fig. 7.33. When the number of inputs increases, the delay of the step response increases due to loading capacitors. Moreover the 1:7 current mirrors used in check node structure (Fig.7.14) increases the delay.

Mismatch analysis has been performed on the check node module. In this simulation 100 samples of the output current are obtained using Monte Carlo analysis while one of the inputs changing from $-1.2\mu A$ to $1.2\mu A$ and the rest of inputs are $1.3\mu A$. The results of mismatch analysis is presented in table 7.6. Moreover, Fig. 7.34 shows the output current with the ideal case (no mismatch), average of all 100-sample output currents including mismatch effect, the worst cases due to mismatch and standard deviation.

Transient Response							
$I_{min}: 0 \rightarrow 200nA$		$I_{min}: 200nA \rightarrow 400nA$		$I_{min}: -200nA \rightarrow 200nA$		$I: 200nA \rightarrow 700nA$, $I_{min} = 400nA$	
T_r	T_f	T_r	T_f	T_r	T_f	T_r	T_f
180ns	160ns	112 ns	129ns	143ns	110ns	88ns	100ns

Table 7.5: Step response simulation results for check node proposed in Fig. 7.14 while T_r and T_f are representing the rise time and fall time, respectively.

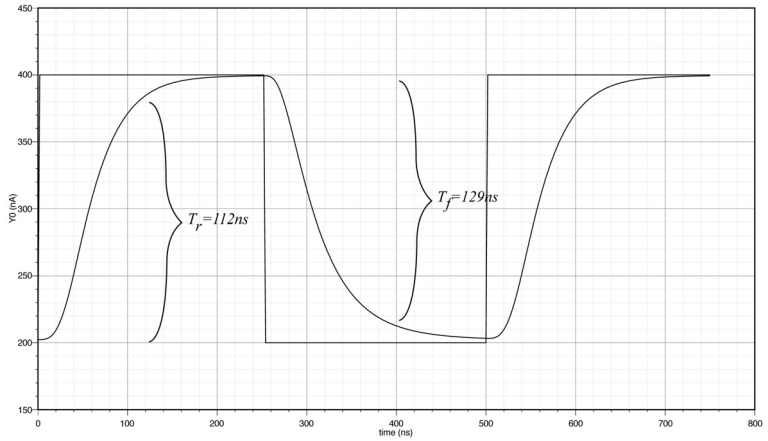


Figure 7.30: Step response of the check node when the minimum input changes from 200nA to 400nA.

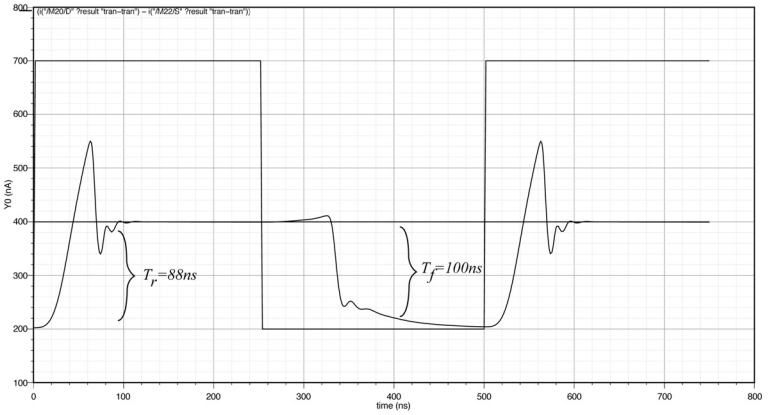


Figure 7.31: Step response of the check node including the input current mirrors and estimation of wiring cap when one of the inputs changes from 200nA to 700nA while the other input is 400nA.

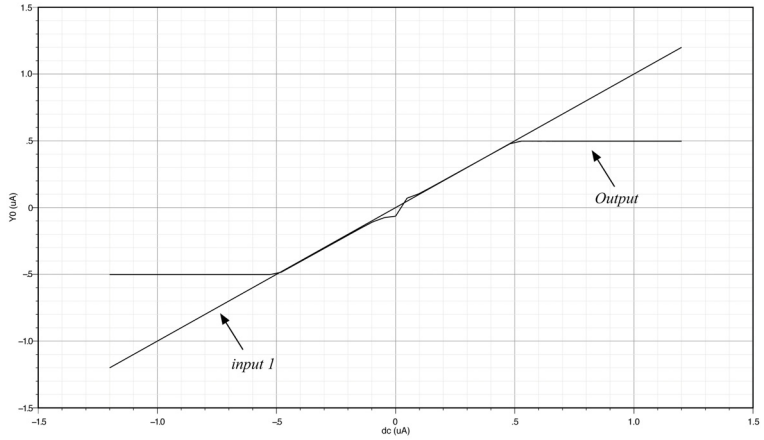


Figure 7.32: Performance of the check node when one input changes from $-1.2\mu A$ to $1.2\mu A$ while the other input is $500nA$

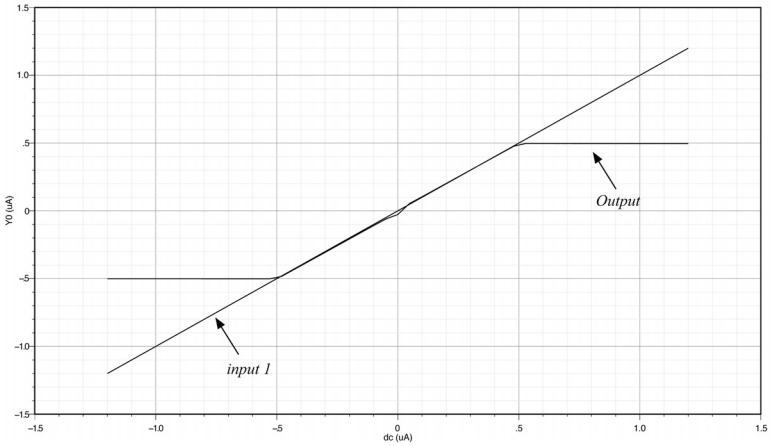


Figure 7.33: Performance of the modified check node when one input changes from $-1.2\mu A$ to $1.2\mu A$ while the other input is $500nA$

Mismatch	
$\frac{\Delta I}{I}$ for the worst case (over 100 iterations)	$\sigma_{norm} = \frac{\sigma}{AVG}$
26.5%	10.45%

Table 7.6: Mismatch simulation results for check node module.

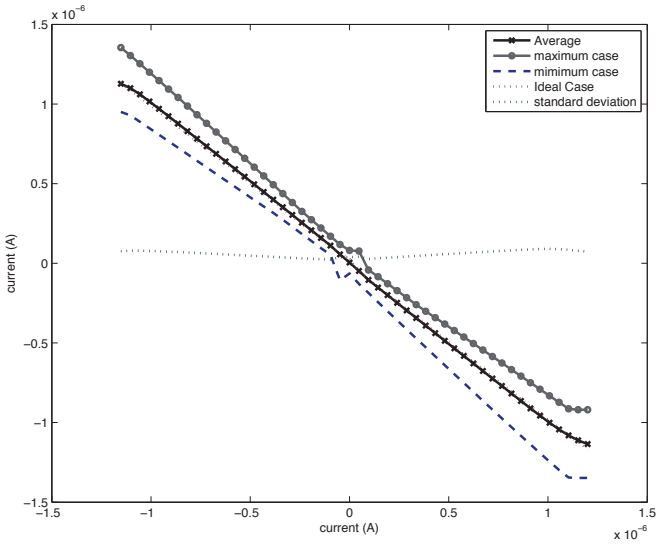


Figure 7.34: Mismatch analysis for the check node

Early Termination Signal

In some cases at higher SNR, decoding can be concluded before the maximum settling time is reached. A unique feature of this analog decoder is the generation of an Early Termination Signal raised if all parity-check equations are satisfied allowing a decoding decision to be made before T_d has elapsed. This signal can be used to shut-down the analog decoder circuit until new LLRs are arrived. This leads to a reduction in power consumption. Moreover, ETS gives us the opportunity to measure the decoding time.

If all parity check nodes are satisfied then the codeword is successfully decoded.

A parity check equation is satisfied when the XOR of all the digital received messages of that particular check node is zero. Therefore, the extracted signs of the analog received messages (currents) are applied to an 8-input XOR gate to test the parity check satisfaction. Eventually, the output of all the 45 XORs are passed to a 45-input OR gate. If the result of this OR gate is zero then all the parity check equations are satisfied and Early Termination Signal (ETS) goes high. This signal is passed to an off-chip module for further processing through the ETS pin. The 8-input XOR gate is designed in the same way as depicted in Fig. 7.16. The OR gate is implemented using fourteen 4-input, one 3-input and one 2-input OR gates.

7.3 Conclusion

In this chapter the analog processor performing decoding on a (120,75) TS-LDPC code is proposed and designed which is one of the largest analog decoders. This analog decoder is designed based on the Tanner graph of (120,75) TS-LDPC decoders. The Tanner graph is comprised of 120 variable nodes and 45 check nodes which are connected through the edges based on the parity check matrix of the code. The edges which connect VA nodes to CHK nodes are Bi-directional connections and are implemented using two separate wires.

The decoder starts decoding by receiving channel outputs and calculating LLR values. The proposed analog decoder is designed based on the current mode modules. Therefore, LLR values are mapped to analog currents through a mapping scheme. The design of VA nodes and CHK nodes are proposed and designed in this chapter. The VA node and CHK node modules are modified based on required accuracy, speed, power consumption and chip area. In order to control the decoding time switches are

provided at the inputs of VA nodes and CHK nodes. These switches are also used to reset the decoder's modules at the beginning of each decoding cycle. If the decoder reaches to a decision before the decoding time is passed, Early Termination Signal is activated.

Base on the design proposed in this chapter, the analog decoder chip has been implemented. The fabricated chip has been tested and measurement results are discussed in the next chapter.

Chapter 8

Measurement Results of TS-LDPC Analog Decoder Chip

In this chapter the fabricated analog decoder chip is presented. This analog decoder is implemented using TSMC 90nm technology provided by CMC Microsystems. We have tested the chip using a Teradyne Flex tester available at the CMC Mixed-Signal Laboratory.

In this chapter we briefly introduce the tester's equipment and capabilities. The designed test board is also presented. Measurement results are provided and comparison between simulation and measurement results are shown. Finally, this work is compared with existing analog decoders and some of their digital counterparts.

8.1 Teradyne Flex Tester

The Teradyne Flex tester gives us the opportunity to combine test resources for digital, analog and RF circuits. However, in this work we focus on digital and analog

resources as required in our test procedure. The tester utilizes different modules each responsible to provide one type of signalling. In this thesis, we use the following instruments of the Teradyne Flex tester.

DC30: This device provides voltage and current sources in the DC domain. This module is also capable of measuring the DC voltages. There are 20 voltage and current channels in total. Each channel provides upto 30V of voltage and 100mA of current. If more current is required the channels can be programmed to support 10V of voltage and upto 200mA of current.

HSD200: This module provides High Speed Digital signals. In total, there are 96 channels available in this instrument. The digital voltages can change from -1V to +6V. Four of these channels can be used for high voltage signals from 0V to 20V. The time resolution for each channel is 40ns. It should be mentioned that the HSD200 module is not a high frequency instrument and the minimum allowable period T is 100ns.

The above-mentioned instruments are controlled by IG-XL software which consists of Microsoft Excel and the VBT (Visual Basic for Test) module. The test procedure is defined in the VBT module while IG-XL works as the interactive environment.

Test flow is defined in IG-XL software based on the procedure determined in the VBT module. The pin type (pin map) is introduced in IG-XL software. Each pin is mapped to one of the tester channels through IG-XL software. DC values are defined in the VBT module, while voltage levels for digital signals are described in IG-XL software. Timing of digital signals are specified in IG-XL software, as well. For generation of each signal, timing should be defined in the IG-XL software. If the

value of a signal is set to 1 then a voltage is produced based on the level and the timing set. The following information is specified in the time set:

Period: The period of the signal should be defined even if the signal is not periodic.

Format: Signals are defined in one of the following forms:

- Non-Return (NR)
- Return-to-High (RH)
- Return-to-Low (RL)
- Retun-to-Complement

Data edge: Defines the positive edge of signal.

Return edge: Defines the negative edge of signal.

off: Defines the end time of one period.

Mode: If the signal is an output signal, then the mode is "edge" otherwise it is "off".

Open: Defines the edge of comparison for the output signals.

Example of a periodic, non-periodic and output signal (sampled at the defined edge) is illustrated in Fig. 8.1. If the value of a signal is set to be 1, then the output of that particular channel follows one of the waveforms depicted in Fig. 8.1 according to the signal definition, otherwise the output is 0V. All the values of the pins are stored in a file known as the pattern file. The tester generates signals based on the values defined in the pattern file the the waveform defined in the time set.

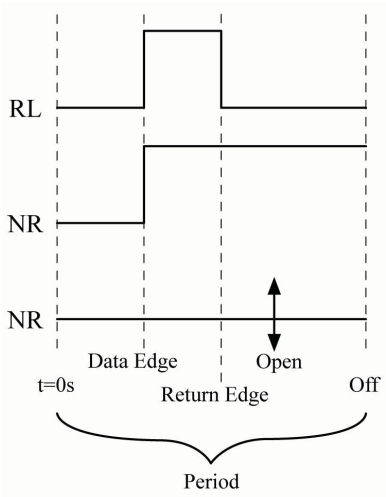


Figure 8.1: Exmaple describing timing specification of different signals in Teradyne Flex tester

8.2 Test Board

To ease the testing and assembly on the Printed Circuit Board (PCB), the fabricated decoder chip was packed using Quad Flat Package CQFP-44 pins. This allows us to easily test different copies of the chip. All required DC and AC signals are provided by the Teradyne Flex tester. Due to the tester limitation in sourcing nano-ampere range currents, two external current sources are used to provide I_{ref} and I_{adj}. To use different chip samples on a single PCB, a socket is used to host the chip. The schematic of the test bench is presented in Fig. 8.2. The PCB is mounted to the tester using two SQW-150 connectors. These are two 100 pins connectors that are used to transfer the signals from the tester to the chip. The coupling capacitors are placed between the voltage source and ground to keep the voltages more constant. Line terminator resistors are provided at the signal routes.

In this design, Kelvin connection is used to receive the desired DC voltage at each pin. The Kelvin connection methodology is different from the conventional routing

scheme but results in improved testing. Conventional and Kelvin connections are shown in Fig. 8.3. In the conventional connection, parasitic resistor introduced by the routing results in deviation from the desired voltage supply. In Fig. 8.3(b) Kelvin connection is presented. Two sensing connections are used to measure the voltage while they are in high impedance mode. Therefore, no current is following on these paths and voltage drop due to parasitic elements is zero. The voltage at V_{DD} pin of the DUT (Device Under the Test) is measured accurately by the tester. Based on the measured voltage, the tester adjusts its supply voltage to get the desired value. The crossing of the force and sense lines should be as close as possible to the DUT.

8.3 Timing Scheme of the Decoder Chip

As mentioned in Chapter 7, LLR information is required in order to start the decoding procedure. For each Signal-to-Noise Ratio, Analog LLR values are calculated using Matlab software and their magnitude is clipped to 7. Each LLR value represented by 5-bit quantization and the sign of the LLR values is stored in the 6th bit. Therefore, 720 bits are produced for each codeword utilizing Matlab software. Values used in the pattern file to specify the controlling signals such as clock pulses, loading signals and signals controlling the switches are generated by Matlab software, as well. All values are written to a text file which is converted to a pattern file by the Tester. The tester executes the pattern file values line by line. During execution of each line, based on the values of the parameters and the defined waveform, signals are generated.

In addition to LLR bits, the following six controlling signals are generated:

- clk-Serial-in

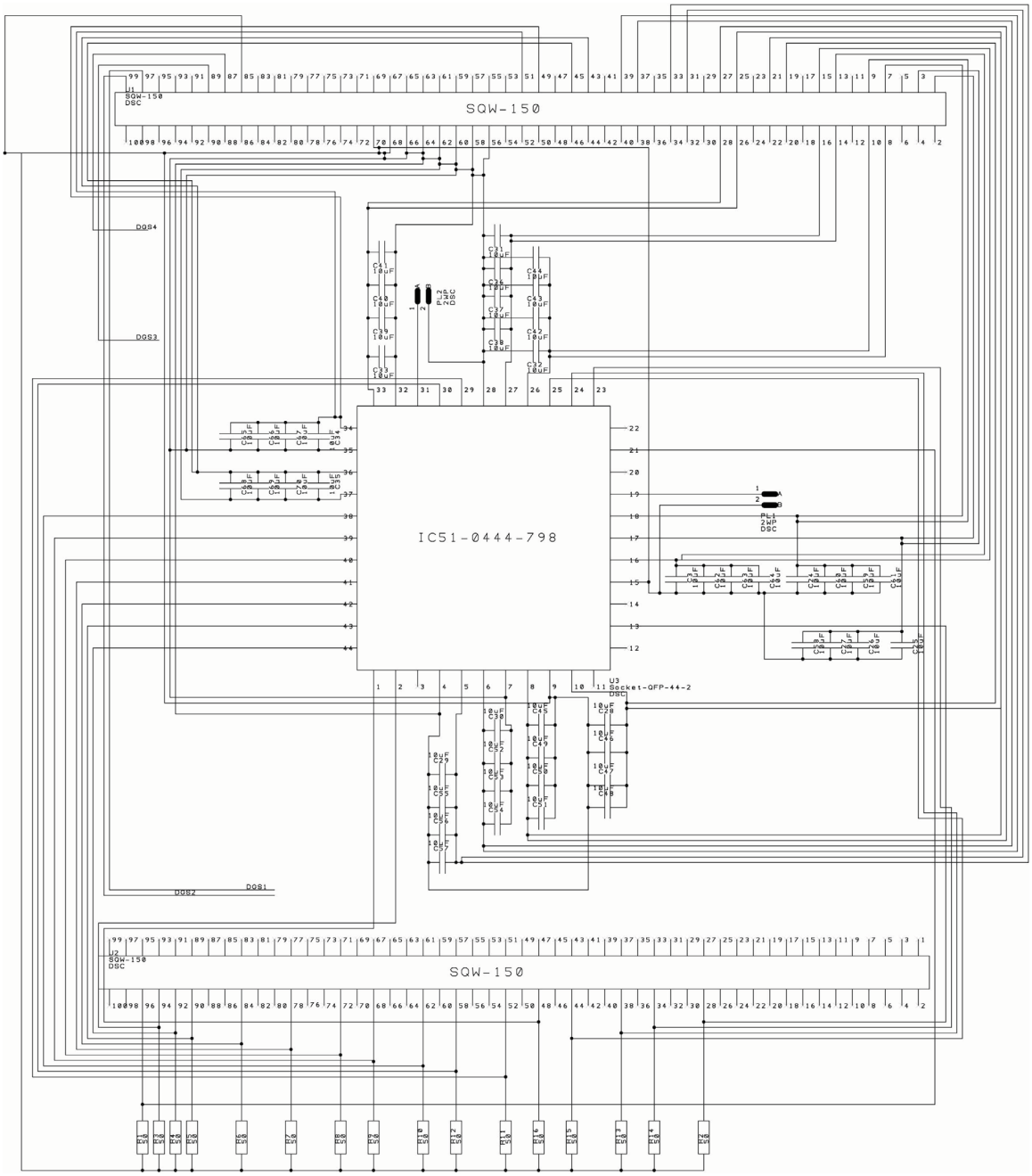


Figure 8.2: Test bench schematic

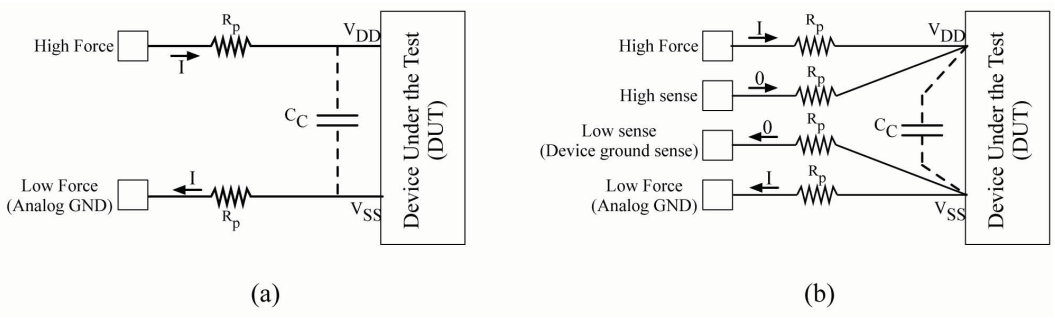


Figure 8.3: Test bench schematic

- clk-Latch-Data-in
- V_{SW}
- clk-load-Ro1
- Latch-mode-Ro2
- clk-Ro2

Three following timing scenarios can be performed on the decoder chip.

Scenario 1

In this scenario the LLR bits are transferred to the chip serially. Therefore, 720 clock pulses are required to save the LLR values to the input stage memory. Based on the specifications provided by socket manufacturer and CMC recommendation, $10MHz$ is the maximum frequency at which the socket works reliably. This means that the period of each clock pulse is $100ns$. Consequently, to store 720 bits in the memory block $72\mu s$ is required. In the same way, $120 \times 100ns = 12\mu s$ is needed to transfer the estimated codeword ($\widehat{\mathbf{c}}$) to the output module. Timing diagram for the scenario 1 is depicted in Fig. 8.4.

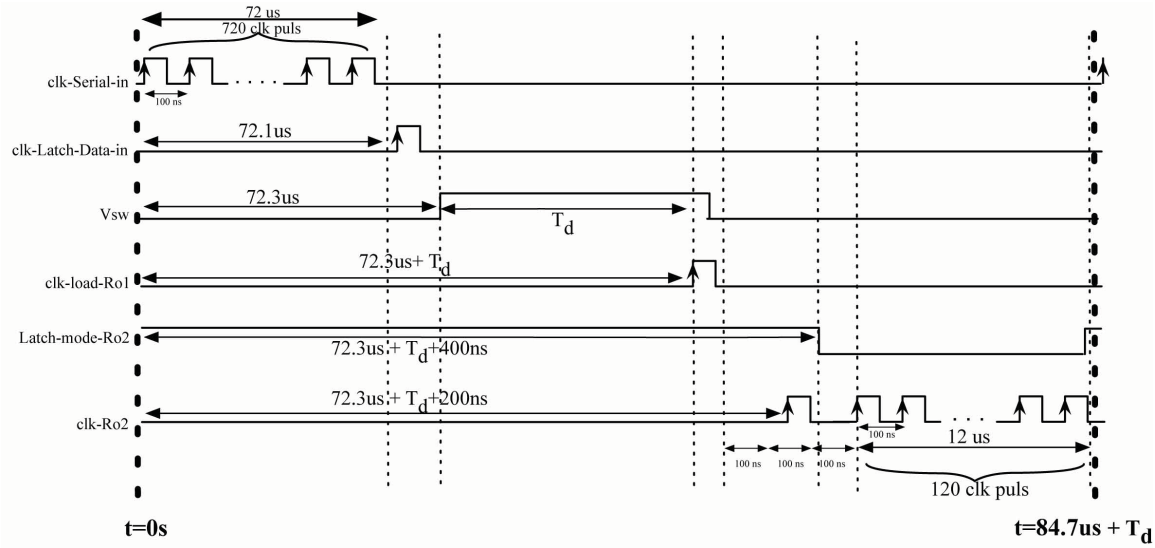


Figure 8.4: Timing diagram for scenario 1 that LLR bits are loaded into the chip serially.

Scenario 2

In this case, LLR bits are sent to the chip in a semi-serial mode. That is, 720 bits are split into six 120-bit blocks. Each 120-bit block is sent to one of the six Bit-stream pins as described in Section 6.1. This results in shorter loading time. The timing diagram describing this scenario is shown in Fig. 8.5. It can be seen that the decoding throughput increases significantly. In order to utilize the decoder chip in semi-serial mode, Data-in-Mode signal should be grounded.

Scenario 3

In this scenario, the decoder chip performs three different tasks simultaneously. At time interval n, the chip loads LLR bits $LLR-Bits_n < 0 : 719 >$ while decoding the codeword received at time interval $n-1$ and transmitting the estimated codeword which was received at time interval $n-2$ ($\widehat{\mathbf{c}}_{n-2}$) to the off-chip module. In this case,

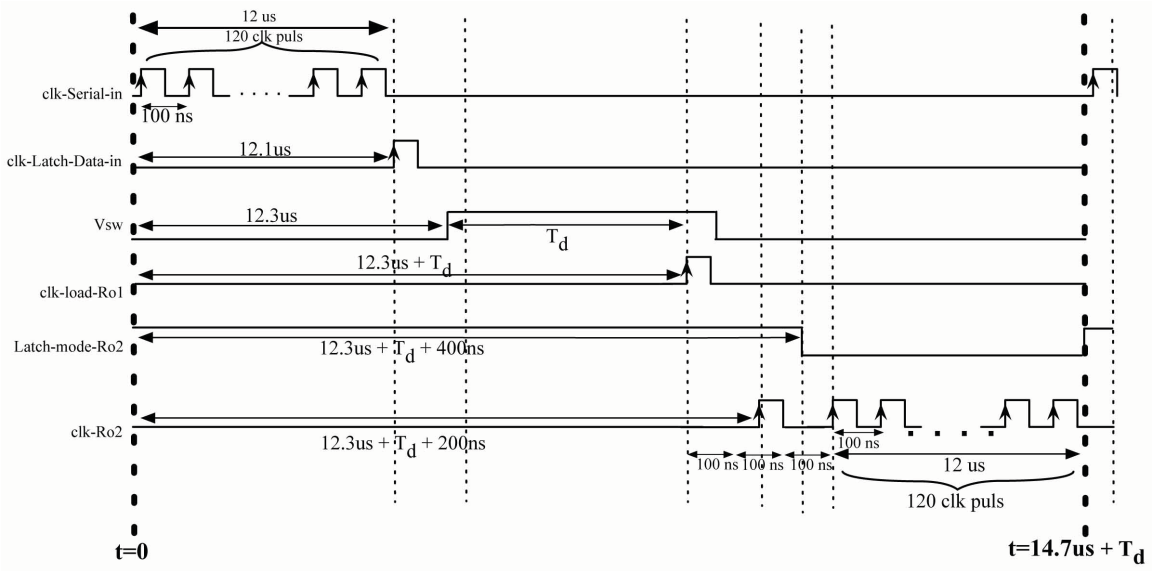


Figure 8.5: Timing diagram for the case that LLR bits are loaded into the chip semi-serially.

the decoder is set to work in semi-serial mode. Using this decoding scheme, the chip can be used as a roughly real-time decoder if the delay of two codewords is neglected. The timing diagram of the high speed decoding scenario (scenario 3) is presented in Fig. 8.6.

Some of the generated control signals are shown in Figs. 8.7-8.12. In these examples the clock pulse with period of $10\mu s$ is considered. Scenario 1 is assumed in presenting these signals.

8.4 The Fabricated (120,75) TS-LDPC Analog Decoder Chip

The described analog decoder scheme has been implemented and fabricated using TSMC 90nm technology. The die micrograph of the (120,75) TS-LDPC analog CMOS

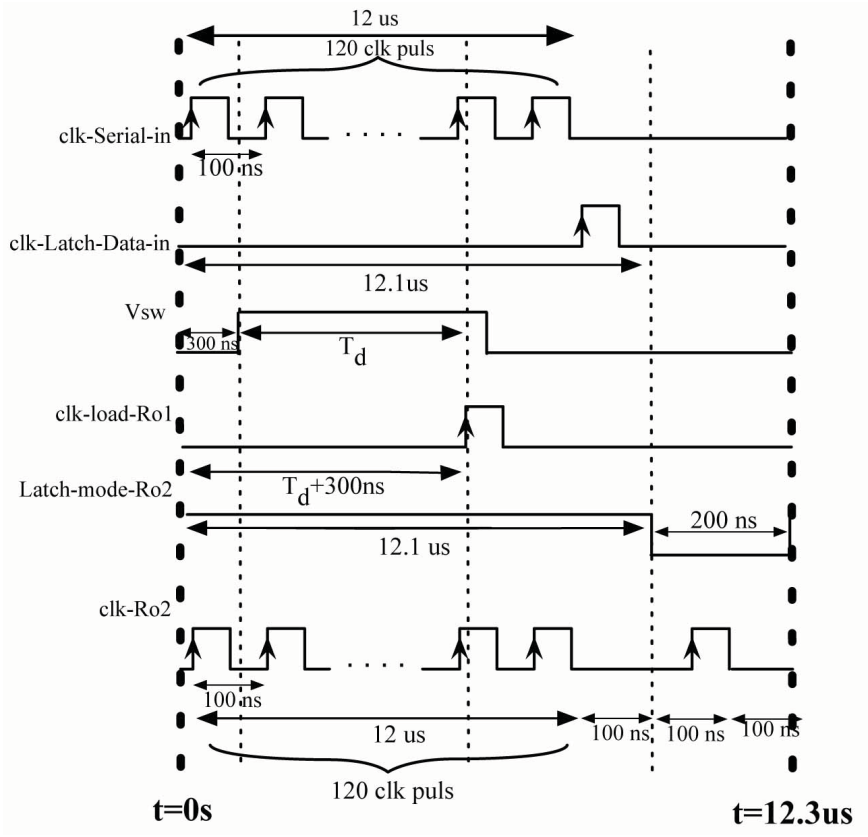


Figure 8.6: Timing diagram for the scenario 3

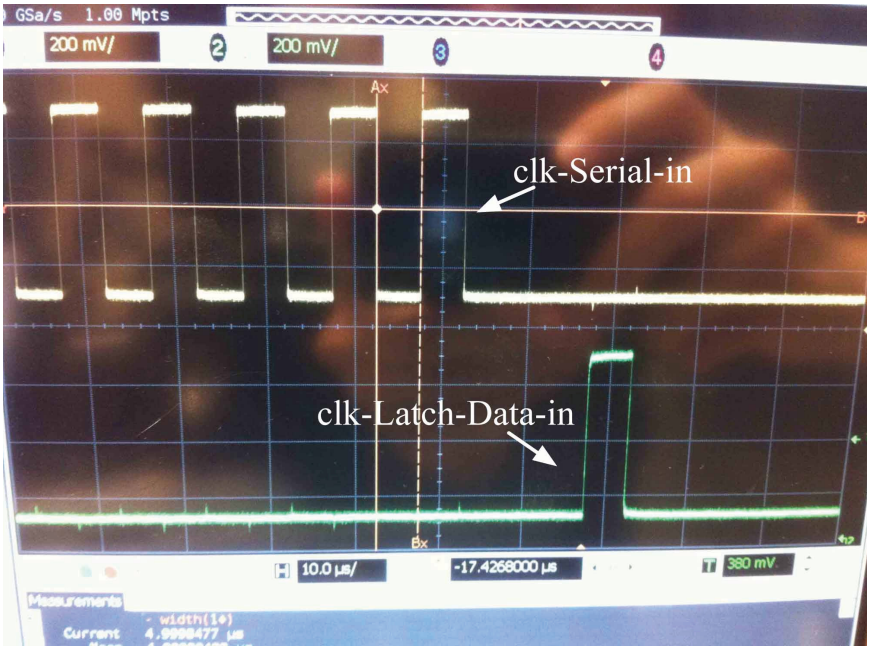


Figure 8.7: clk-Serial-in and clk-Latch-Data-in signals are presented.

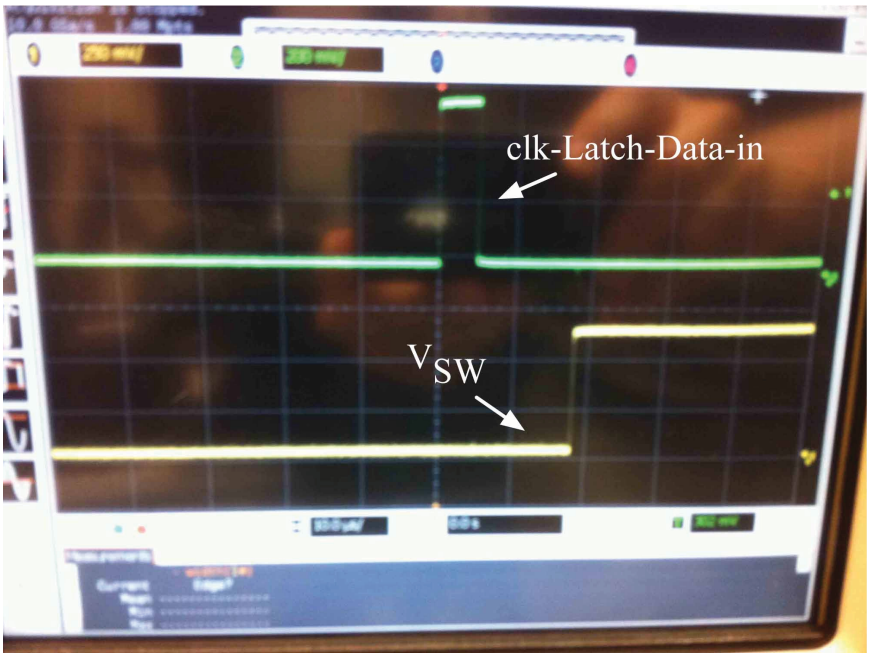


Figure 8.8: clk-Latch-Data-in and V_{SW} signals are presented.

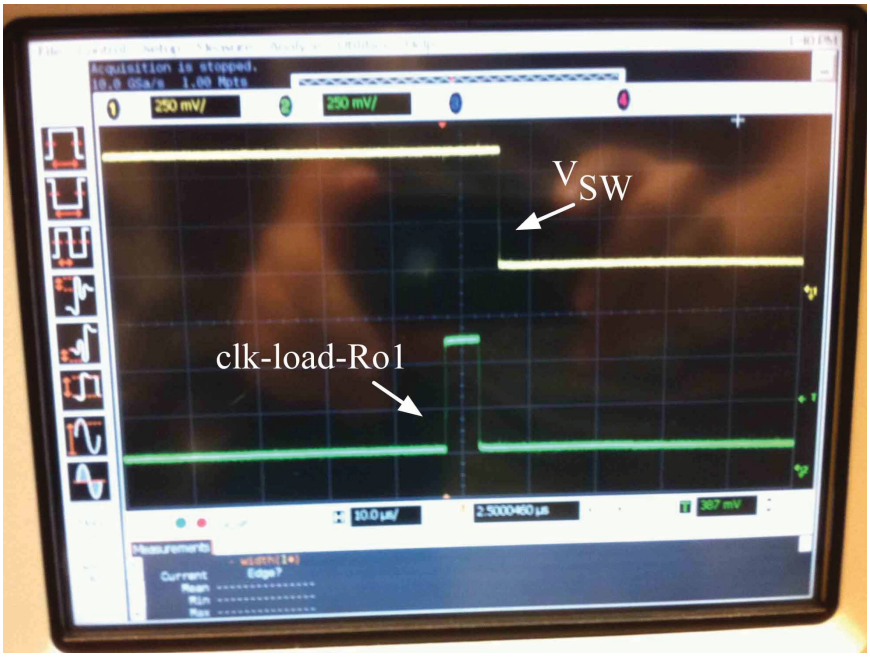


Figure 8.9: V_{SW} and clk-load-Ro1 signals are presented.

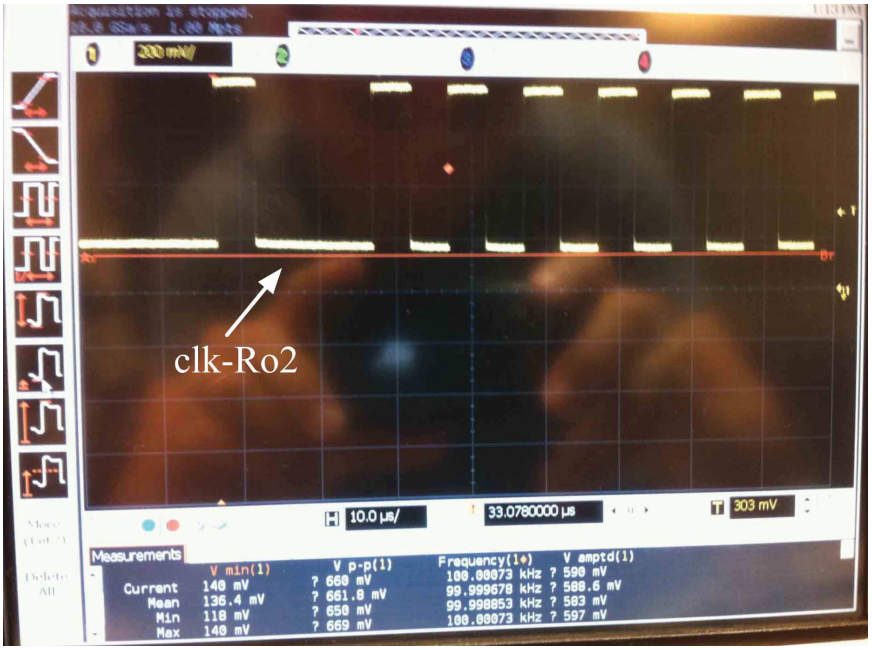


Figure 8.10: clk-Ro2 signal is presented.

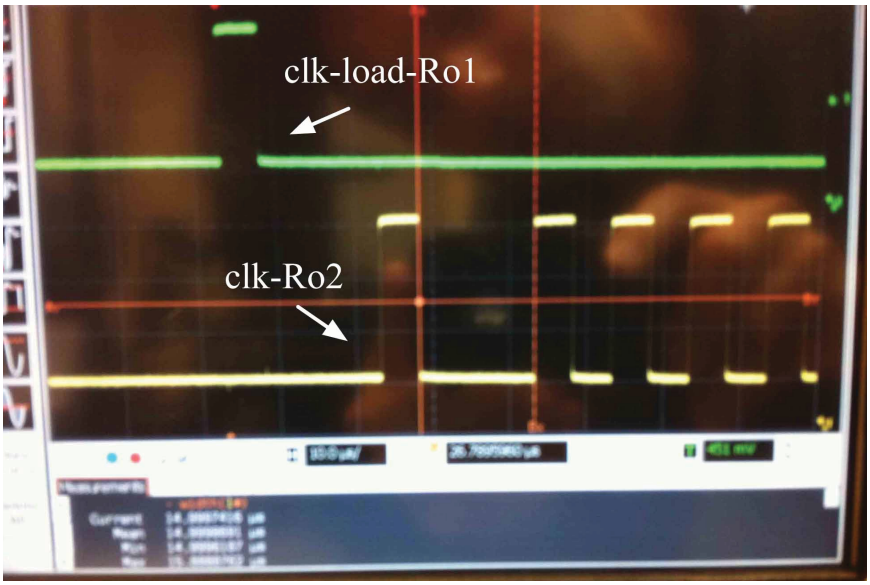


Figure 8.11: clk-load-Ro1 and clk-Ro2 signals are presented.

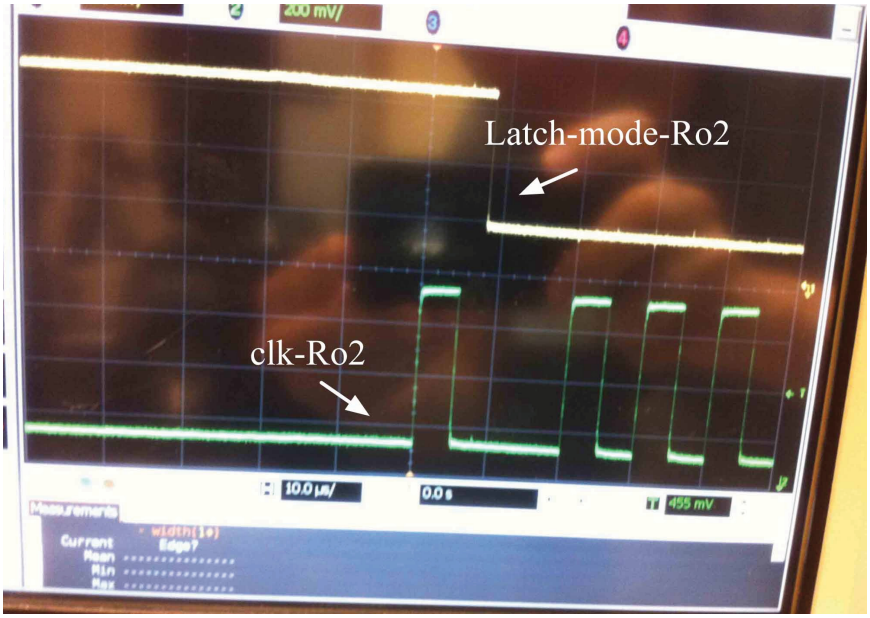


Figure 8.12: Latch-mode-Ro2 and clk-Ro2 signals are presented.

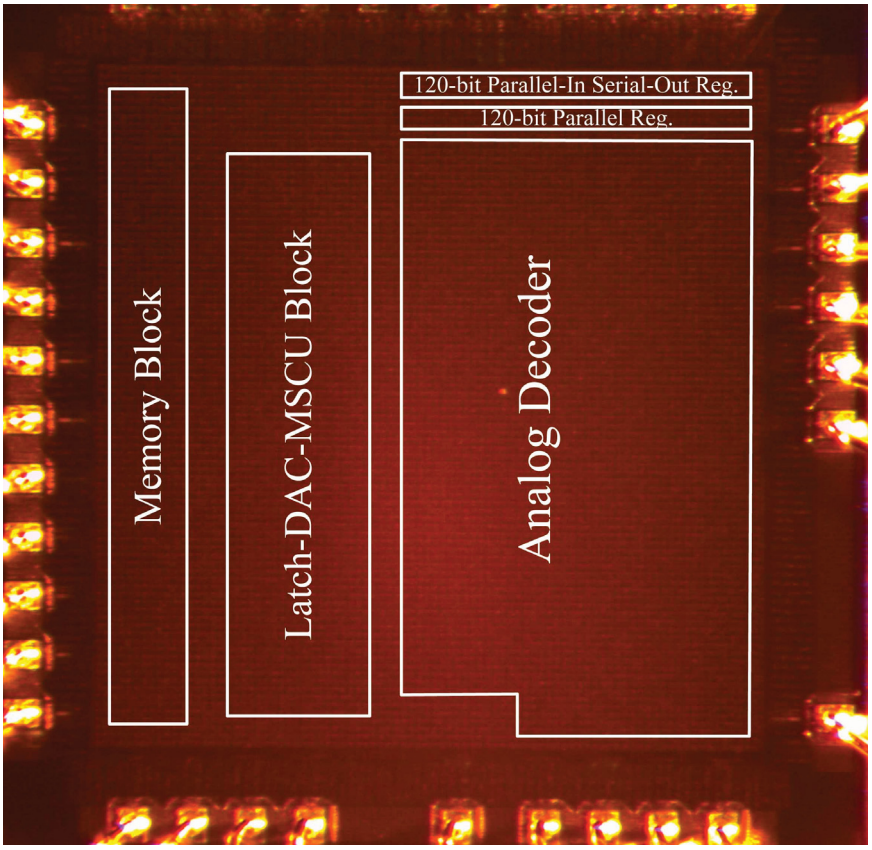


Figure 8.13: The die micrograph of the (120, 75) TS-LDPC analog CMOS decoder.

decoder is shown in Fig. 8.15. The constituting blocks of the decoder and their locations are illustrated on the die micrograph. The chip occupies $1.8 \times 1.8 = 3.24mm^2$ of die area while analog decoder core uses $1.379mm^2$. Ignoring regions void of circuitry, the total area of DACs, VA nodes and CHK nodes is only $0.75mm^2$. It is worth mentioning again that in this work, digital circuitry uses 1.2V while analog circuitry use 1.1V and 0.85V supplies. The reference current required for DACs is provided by an off-chip current source. Fig. 8.14 shows the test set-up while Fig. 8.13 depicts the decoder chip resting on the socket.

Figure 8.14: Teradyne Flex tester while the (120,75) TS-LDPC analog decoder IC is under test.

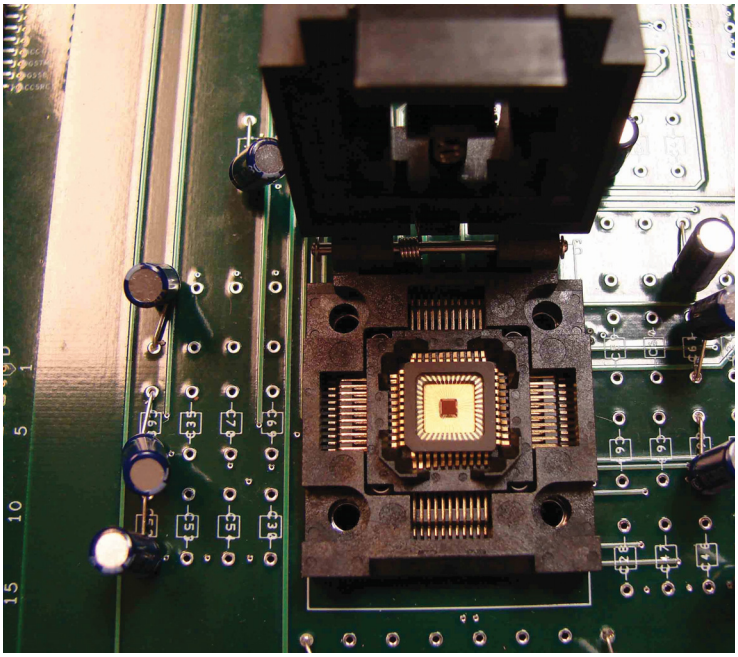


Figure 8.15: Snapshot of the (120,75) TS-LDPC analog decoder IC in the CQFP-44 pin package.

8.5 Chip Measurement Results and Discussion

The fabricated (120,75) TS-LDPC decoder is tested. The measurement results are presented in this section. In the testing process, using Matlab software, information bits are generated randomly and passed to the (120,75) TS-LDPC encoder. The codeword is modulated using BPSK modulation scheme where bits 0 and 1 are mapped to 1 and -1, respectively. The generated signals are sent through the AWGN channel. In the channel the random noise with a particular SNR is added to signals. At the receiver, the channel outputs are received and LLR values are produced using equation (3.10). The LLR values are quantized and fed into the chip through the tester.

Along with the LLR bits the generated signals in Figs. 8.7-8.12 are applied to the decoder chip. The chip is tested using two different procedures. In the first method, the outputs of the decoder (Serial-data-out and Early Termination Signal) are visualized using the tester associated software. In this method, 720-bit LLR block is sent to the chip. The tester converts the two binary outputs of the chip to decimal base format and plots the resultant value. In the decimal scheme, Serial-data-out is b_0 (LSB) and Early Termination Signal is b_1 (MSB). The output waveform is saved and the decimal values are converted to binary format. Then Serial-data-out is compared to the codeword and the estimated information bits are extracted and compared to the information bits. The required decoding time is observed by looking at the Early Termination Signal. This gives us a good estimate on the decoding time. Fig. 8.16 shows a snapshot of the tester output visualizing the chip's outputs. This method of chip testing is not practical for medium to high SNR at which the number of required bits (blocks of codewords) is high for evaluation of BER.

To measure the BER of the decoder chip at a particular SNR, a 720-bit block

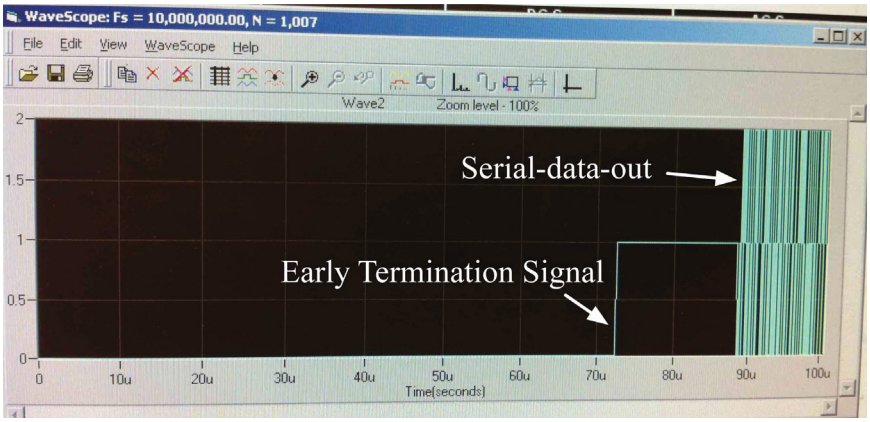


Figure 8.16: Snapshot of the decimal based decoder chip's output illustrating Early Termination Signal and Serial-data-out.

of LLR bits associated to each codeword is generated by Matlab as described in the above-mentioned paragraphs. These blocks along with the original codeword are fed to the tester. The tester compares the Serial-data-out ($\widehat{\mathbf{c}}$) with the original codeword ($\mathbf{c}$) and the number of errors are generated. This method gives us the opportunity to test the decoder chip with high number of codewords.

The values for the DAC reference current (I_{ref}) and the check nodes' correction current (I_{adj}) are defined in Chapters 6 and 7 based on the circuit simulations. In order to obtain the optimal I_{ref} and I_{adj} currents, the chip was tested using different current values. The measurement results are presented in Fig. 8.17. It can be observed that for $I_{adj} = 0A$, the BER performance at low SNR degrades while with $I_{adj} = 100nA$ the measured BER is comparable with simulation at low SNR. At high SNR, the error performance is slightly better than the case of $I_{adj} = 100nA$. The minimizer output is saturated at current of $1200nA$. By adding the correction current we boost the currents associated with very low LLRs. This increases the speed and accuracy at low LLR values while at high LLR values this results in faster saturation

of the minimizer's output. Hence, eliminating the correction current contributes to a better error rate performance at high SNR. Further increase in I_{adj} does not improve the BER performance while it adds to the chip overall power consumption.

Translating LLR bits to analog currents is one of the crucial steps of an analog decoder chip testing. Therefore, the optimal value for the DAC reference current should be obtained. Simulation results show that increasing the reference current I_{ref} from its nominal value ($140nA$) to $200nA$ does not further improve the error performance at low SNR. By decreasing the I_{ref}, the resolution of the currents representing LLR is decreased and the error performance degrades at low SNR. On the other hand, at high SNR the error performance improves. Decreasing the reference current results in smaller currents at high SNR. This delays the saturation of the minimizer's output. Consequently, a more accurate output is obtained at the check node which contributes to better error performance.

Based on the above discussion, there is a trade-off in choosing the I_{ref} and I_{adj}. Targeting low and medium SNR, $I_{ref} = 140nA$ and $I_{adj} = 100nA$ are good choices. If the decoder is working mainly at SNR of 5 and more, then $I_{ref} = 100nA$ and elimination of I_{adj} is suggested. In this work, we choose the nominal values of $140nA$ for I_{ref} and $100nA$ for I_{adj}.

The decoding time T_d is one of the most critical parameters in every decoder module. It is well known that, the longer decoding time, the better error performance. In this decoder chip, the decoding time is enforced by V_{SW} controlling signal. The decoding time has been changed from $T_d = 15\mu s$ to $T_d = 100ns$. The measurement results are presented in Fig. 8.18. It is observed that for $T_d = 15\mu s$ the measured BER competes with the software simulation result. At high SNR, the measured

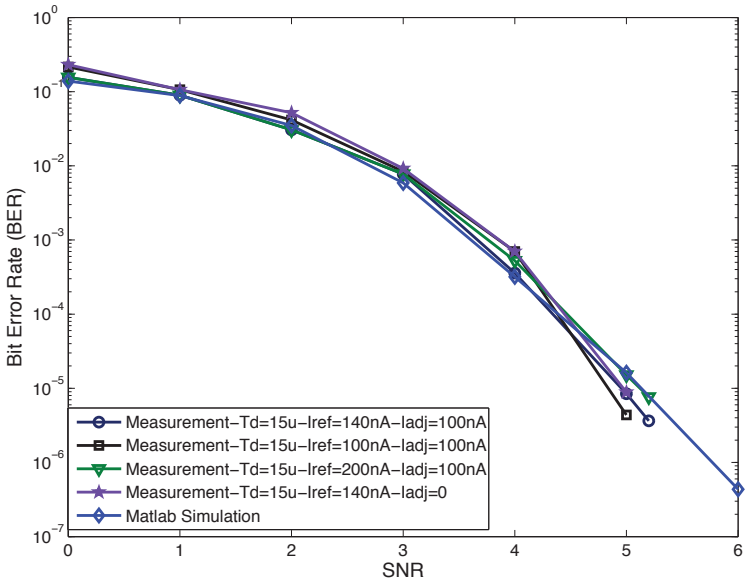


Figure 8.17: The chip measurement results with different I_{ref} and I_{adj} currents

BER slightly outperforms the simulation results. In Matlab software simulations no clipping is imposed on the LLR values while during the chip testing LLR values are clipped. The slight superiority of the measured BER compared to simulated BER is due to the effect of clipping. The clipping bounds the overconfident reliabilities at the output of the check nodes. This improves the error performance of MS algorithm [22]. When the decoding time reduces to $1\mu s$, the error performance is slightly degraded especially at low SNR. Fig. 8.18 shows the direct relationship of the error performance and the decoding time. It is illustrated that for $T_d = 500ns$, the measured BER differs by 0.1dB at BER of 10^{-5} from the software simulated decoder. This difference is 0.2dB when T_d is reduced to $100ns$. The measurement results depict that the fabricated chip works very close to software simulations.

As mentioned before, the Early Termination Signal is the sign of deciding on the

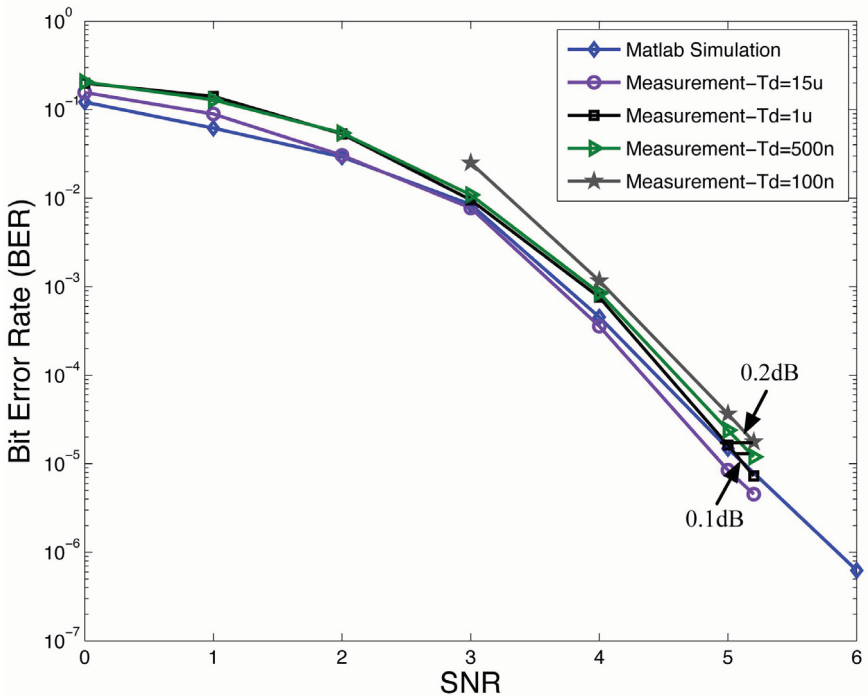


Figure 8.18: Comparison of the chip measurements with different decoding times and Matlab simulation results

codeword. Fig. 8.19 show the Early Termination Signal and V_{SW} at low SNR. The figure depicts that the decoder comes to the decision at $400ns$. The magnitude of the Early Termination Signal is small since the line was heavily loaded. The ETS was used to measure the required decoding time. The average required decoding time is illustrated in Fig. 8.20. At SNR of 0dB-2dB, the decoding time is not observed since the decoder cannot estimate the codeword truly.

The decoder chip proposed in this work utilizes three different voltage sources, V_{DD}, $V_{DD-MSEU}$ and V_{DD-D}. V_{DD} and $V_{DD-MSEU}$ provide necessary voltages to the analog circuitry while V_{DD-D} is the voltage used in the digital circuity. The average measured power consumption at the analog circuitry is $13mW$. The energy efficiency of analog decoders are defined as,

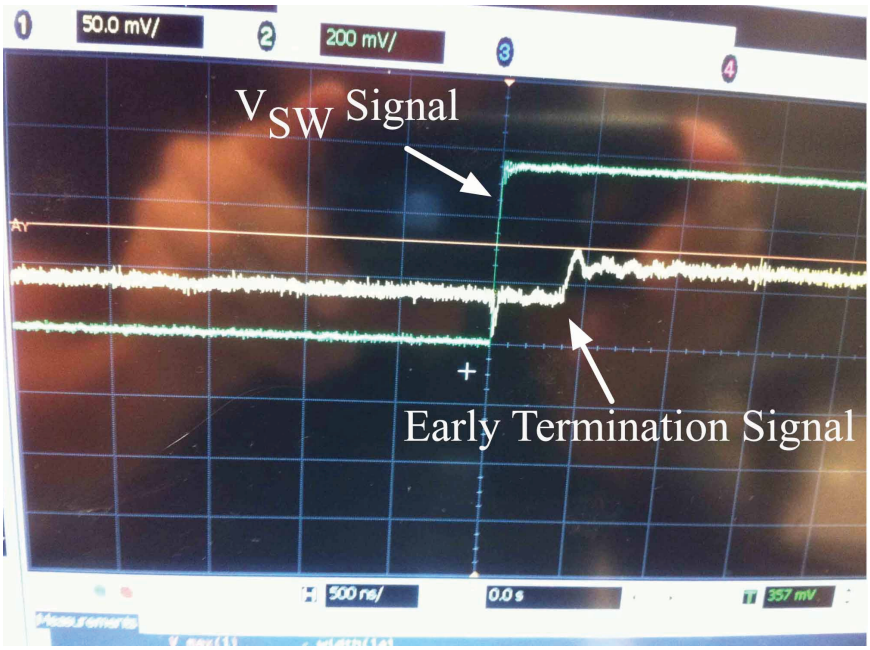


Figure 8.19: Snapshot showing V_{sw} and Early Termination Signal at low SNR. The Early Termination line is heavily loaded.

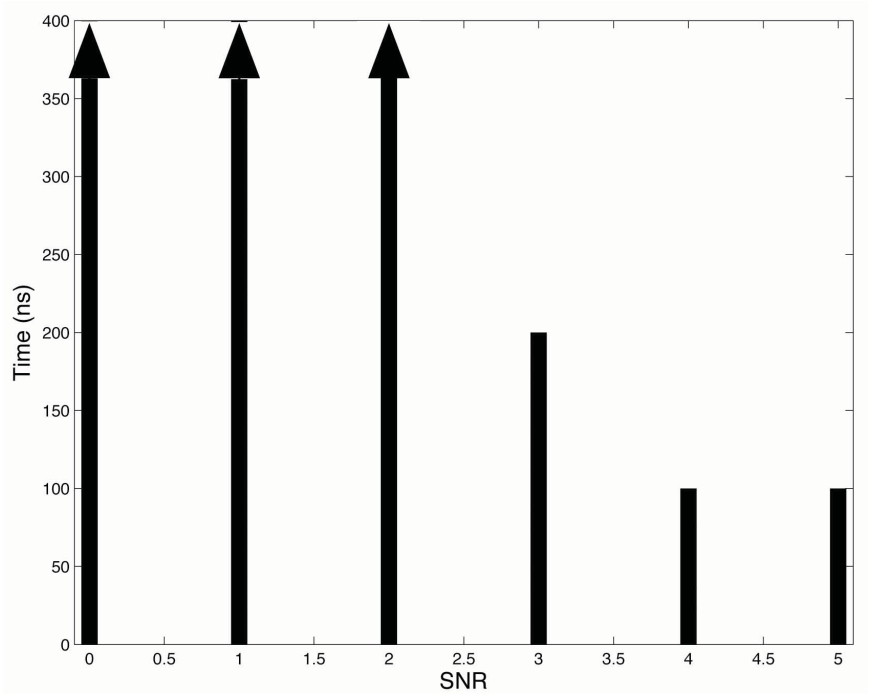


Figure 8.20: The histogram of the average required decoding time

$$\eta = \frac{P}{K \times T_d} \tag{8.1}$$

where P, K and T_d are power consumption, number of information bits and decoding time, respectively.

In this work, the chip is tested in scenario 1 where LLR bits are loaded before decoding takes place. The chip is tested with different decoding times T_d. It is shown that the higher T_d, the better error performance. Increasing T_d results in higher energy efficiency. Various throughputs and energy efficiencies of the chip have been tested.

Information throughput performance is based on the measured decoding time, assuming consecutive codewords could be decoded each T_d. For $T_d = 1\mu s$, the throughput of the chip is $75MHz$ and the energy efficiency is $0.173nJ/b$. The measurement results showed that the error performance of the chip is acceptable when the decoding time is $100ns$ which is the shortest measurable decoding time. This implies a maximum measurable information throughput of $750Mb/s$ and a minimum energy efficiency of $17pJ/b$.

8.6 Comparison Between the Analog Decoders

In this section, the proposed and fabricated analog decoder is compared with some analog and most recent digital decoders. Table 8.1 summarizes the comparison. Comparison between the proposed analog decoder and digital decoders show that the TS-LDPC analog decoder is the only analog decoder that competes with the digital decoders from error performance point of view. This work yields slightly better

efficiency than the digital decoders albeit operating on a shorter codeword. It has a small die area. However, the die area can be further reduced significantly with a careful interconnection scheme. The throughput of the proposed analog decoder is higher than the throughput of the decoder proposed in [96] at 5.5dB. Though, the lower throughput of the analog decoder compared to the rest of the digital decoders can be justified by the shorter length of the codeword.

Table 8.1 shows that the proposed analog decoder deals with one of the longest length code that has ever been implemented. The energy efficiency of the TS-LDPC decoder is superior compared to the rest of the analog decoders. The information throughput of this decoder is higher than the rest of the analog decoders by a great factor. Among the analog decoders, the TS-LDPC decoder has the best error performance with BER of almost identical to theory. Therefore, the proposed analog decoder is superior to all of the analog decoders in terms of error performance, throughput and energy efficiency.

Type	Author	Code	CMOS Technology	Core Area (mm^2)	Power (mW)	Throughput (Mb/s)	Energy Efficiency (nJ/b)	SNR loss at BER of $10^{-5}dB$
Digital	Darabiha et al. [96]	LDPC (660,480)	$0.13\mu m$ 1.2V, 0.6V	7.3	518 @ 4dB 398 @ 5.5dB	2440 480	0.156 @4dB 0.12 @5.5dB	–
	Zhang et al. [97]	RS-LDPC (2048, 1720)	65 nm 0.7V	6.67	144	6670	0.0215	0.2
	Chen et al. [98]	LDPC-CC $N = 491, R = \frac{1}{2}$	90 nm 1.2V	2.24	284	2370	0.024	≈ 0
Analog	Gaudet et al. [38]	Turbo code $N_B = 16, R = \frac{1}{3}$	$0.35\mu m$ 3.3V	1.32	185	13.3	13.9	1.5
	Winstead et al. [30]	(8,4) Hamming Trellis Graph Factor Graph	$0.18\mu m$ 1.8V	0.002 0.02	0.15 0.087	3.7 3.7	0.04 0.22	1 1
	Hemati et al. [28]	LDPC (32,8)	$0.18\mu m$ 1.8V	0.57	5	6	0.83	2
	Vogrig et al. [29]	Turbo $N_B = 40, R = \frac{1}{3}$	$0.35\mu m$ 3.3V	4.1	6.8 (core)	2	3.4	1
	Gu et al. [32]	LDPC (32,8)	$0.5\mu m$ 3.3V	5.4	1.25-7.75 (core)	12.8	0.098-0.6059	0.6
	This work	TS-LDPC (120,75)	90 nm 1.2V, 1.1V, 0.85V	1.38	13 (core)	750	0.017	0.2

Table 8.1: Compasion table of recent analog and digital decoders

8.7 Conclusion

In this chapter, the test procedure for the decoder chip proposed in Chapter 7 was presented. The decoder chip was tested using Teradyne Flex tester available at CMC Mixed-Signal at the Advanced Mixed Signal Systems Laboratory. During the testing, DC and digital high speed signals provided by the tester were utilized. The methods to generate the required signals were briefly discussed.

It was shown that the decoder can be used in three different decoding scenarios. In the first method, LLR bits are serially loaded to the chip memory and decoding is started. At the end of the decoding cycle the estimated codeword is captured at the output register and transferred to the off-chip module. In the second scenario, LLR bits are loaded in to the memory using semi-serial mode. This method increases the loading speed by factor of 6. In the third scenario at time instance n, LLR bits are transferred to the chip, while the chip is performing the decoding procedure on the code received at $n-1$ and sending the estimated codeword of $n-2$ to the off-chip module. In this work, the analog decoder was tested using the first decoding scenario.

The chip measurement results were illustrated in this chapter. It has been shown that the error performance of the decoder chip is almost identical to the simulation (theory) results. It was shown that the proposed analog decoder is one of the largest length code analog decoder which has been implemented to the date. Moreover, the comparison table depicts that the proposed analog decoder is superior than the rest of analog decoders in the sense of error performance, energy efficiency, throughput and chip area. Therefore, a new state-of-the-art is set for the analog decoding. This work was compared with the most recent digital decoders as well.

Chapter 9

Conclusion and Future Work

In this thesis, we considered a class of structured regular LDPC codes, called Turbo-structured LDPC (TS-LDPC) codes. This class of LDPC codes can be designed with any arbitrary desired column weight, row weight and girth. It has been discussed that the TS-LDPC codes can be designed to have a large girth which is a crucial parameter in designing LDPC codes. Large girths result in more efficient iterative decoding and guarantee large minimum Hamming distance (d_{min}) between the codewords which leads to lower error floor at high SNR. The Parity check matrix of TS-LDPC codes can be constructed from a much smaller shift matrix $\mathbf{S}$. Lu showed that TS-LDPC codes outperform the randomly generated LDPC codes at high SNR. Moreover, it has been presented that TS-LDPC codes have much lower error floor compared to randomly generated LDPC codes. Therefore, TS-LDPC codes are good candidates to be adopted in most communication applications.

The channel decoders are generally implemented using digital circuitry. Recently, implementation of analog decoders has been targeted as a research direction by many VLSI research groups. It has been argued that analog decoders are preferred in many

communication systems due to their higher speed, lower power dissipation and smaller chip area compared to their digital counterparts.

Iterative decoding algorithms such as Sum-Product and Min-Sum algorithms are implemented by analog circuitry. Since Min-Sum algorithms have low complexity in the implementation of analog decoders, Min-Sum and Min-Sum with correction factor algorithms have been reviewed and adapted with TS-LDPC codes. Three different iterative decoding schemes, Sum-Product, Min-Sum and Min-Sum with correction factor, have been used to evaluate the error performance of the TS-LDPC codes. Simulation results have shown that the error performance of the Min-Sum algorithm is comparable with the Sum-Product algorithm, for the same block length. Moreover, the superior error floor of TS-LDPC codes is preserved when decoded using Min-Sum algorithm. Therefore, Min-Sum algorithm was utilized in the TS-LDPC analog decoder in this thesis.

It was assumed that there is a one to one relationship between the required number of iterations and continuous settling time in analog decoders. We have shown that the required number of iterations for the decoder using MS algorithm is comparable with the SP-based decoder. However, at higher SNR the number of iterations for MS algorithms and SP algorithm were the same. To test the suitability of the MS-based TS-LDPC decoder, some analog impairments such as mismatch, offset and noise were added to the decoder. Simulation results showed that degradation of the error performance of TSLDPC codes due to the analog impairments is negligible. It was depicted that the TS-LDPC MS-based decoder is tolerable to 20% fluctuation in the output of the variable and check node blocks [23].

In this thesis, the architecture and structure of the decoder chip was proposed.

It was illustrated that the decoder chip consists of one analog decoder heart and two digital input and output stage blocks. The input stage unit is responsible for converting 720 LLR bits and convert the digital LLR values to analog current signals. The output stage block captures the estimated codeword and sends it to the off-chip module.

The analog decoder block is the analog heart of the decoder. It receives the current mode LLR values and starts the decoding procedure based on the Min-Sum algorithm. After the decoding time T_d passes, the estimated codeword is ready to be captured by the output stage block. If the decoding is concluded before T_d has been elapsed, the decoder activates an Early Termination Signal. This signal can be used to measure the actual decoding time. Moreover, the Early Termination Signal can be used to shut down the analog decoder module to reduce the average power consumption.

The fabricated MS-based TS-LDPC analog decoder was tested using Teradyne Flex Tester. The control signals were generated by the tester and LLR bits were produced by Matlab software and fed to the tester. Tester was sending all control signals and LLR bits to the chip under the test simultaneously.

The measurement results showed that the error performance of the decoder chip is almost identical to the simulation (theory) results. The decoder chip was compared to digital and analog decoders. It was argued that the proposed decoder competes with digital decoders from error performance point of view. On the other hand, the analog decoder yields slightly better efficiency than digital decoders albeit operating on a shorter codeword.

The proposed analog decoder was compared with previously implemented decoders. The proposed analog decoder is one of the longest length code analog decoder that has ever been implemented. It has been shown that the decoder proposed in this work is superior to all of the analog decoders in terms of error performance, throughput and energy efficiency. This sets a new state-of-the-art for analog decoding.

9.1 Future Work

This dissertation is concluded by proposing some possible future work. It should be noted that the suggested future work is not limited to the following ideas.

- Due to the tester limitations some features of the chip did not tested.
  - More measurements can be done on the decoder chip to test the decoding settling times of less than $100ns$ and the possibility of having higher throughputs.
  - The error floor of the decoder chip can be reached and presented.
  - The decoder chip can be decoded using the third decoding scenario (high speed decoding scenario) outlined in Section 8.3.
- This decoder can be redesigned to decode larger length codes.
- The throughput of analog decoders is increased by a factor of 57 in this work. However, to compete with digital decoders, the throughput needs to be further increased. It was discussed that the delay in the check node module contributes to the overall delay and consequently to the throughput of the chip. Therefore, to increase the decoder throughput, the delay in the check node module can be

decreased. This can be done by moving to a fabrication technology less than 90nm or redesigning the check node module.

- Routing scheme is another factor that contributes to the die area as well as the overall delay. A more careful interconnection scheme can be used in implementation of future decoder chips.

- One of the analog decoder features is reconfigurability. This can be another interesting specification of the future chip. A reconfigurable TS-LDPC analog decoder chip is capable of decoding different TS-LDPC codes with different **H** matrices. It is possible to connect the variable nodes to the check nodes through the switches. Based on the TS-LDPC parity check matrix, the switches can be turned on and off.

Bibliography

[1] International Telecommunication Union (ITU), "http://www.itu.int/itu-d/ict/statistics/at_glance/keytelecom.html," 2012.

[2] C. Shannon, "A Mathematical Theory of Communication," *The Bell System Technical Journal*, vol. 27, pp. 379 – 423 (Part one),623 – 656 (Part two), October 1948, reprinted in book form, University of Illinois Press, Urbana, 1949.

[3] R. G. Gallager, "Low-Density Parity-Check Codes," Ph.D. dissertation, Massachusetts Institute of Technology (MIT), 1963.

[4] D. J. C. Mackay and R. M. Neal, "Near Shannon Limit Performance of Low-Density Parity-Check Codes," *Electronics Letters*, vol. 32, no. 18, pp. 1645 – 1646, 1996.

[5] D. J. C. Mackay, "Good Error Correcting Codes Based on Very Sparse Matrices," *IEEE Transactions on Information Theory*, vol. 45, no. 2, pp. 399 – 431, 1999.

[6] *ETSI Standard TR 102 376 V1.1.1: Digital Video Broadcasting (DVB) User Guidelines for The Second Generation System for Broadcasting, Interactive Services, News Gathering and Other Broadband Satellite Applications (DVB-S2)*, ETSI Std. TR 102 376, February 2005.

[7] A. Morello and V. Mignone, "DVB-S2: The Second Generation Standard for Satellite Broad-Band Services," in *Proceedings of the IEEE*, vol. 94, no. 1, January 2006, pp. 210 – 227.

[8] *IEEE Standard for Information Technology-Telecommunications and Information Exchange between Systems-Local and Metropolitan Area Networks-Specific Requirements Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications*, IEEE Std. 802.3an, September 2006.

[9] *IEEE Standard for Local and Metropolitan Area Networks Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems Amendment 2: Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands and Corrigendum 1*, IEEE Std. 802.16e, February 2006.

[10] *IEEE Draft Standard for Information Technology-Telecommunications and Information Exchange between Systems-Local and Metropolitan Area Networks-Specific Requirements-Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Amendment : Enhancements for Higher Throughput*, IEEE Std. 802.11n/D2.00, February 2007.

[11] K. S. Andrews, D. Divsalar, S. Dolinar, J. Hamkins, C. R. Jones, and F. Pollara, "The Development of Turbo and LDPC Codes for Deep-Space Applications," in *Proceedings of the IEEE*, vol. 95, no. 11, 2007, pp. 2142 – 2156.

[12] A. Kavcic and A. Patapoutian, "The Read Channel," in *Proceedings of the IEEE*, vol. 96, no. 11, 2008, pp. 1761 – 1774.

[13] R. Tanner, "A Recursive Approach to Low-Complexity Codes," *IEEE Transactions on Information Theory*, vol. IT-27, no. 5, pp. 533 – 547, 1981.

[14] J. Lu and J.M.F. Moura, "TS-LDPC Codes: Turbo-Structured Codes With Large Girth," *IEEE Transaction of Information Theory*, vol. 53, no. 3, pp. 1080 – 1094, 2007.

[15] J.M.F. Moura, J. Lu, and H. Zhang, "Structured Low-Density Parity-Check Codes," *IEEE Signal Processing Magazine*, vol. 21, no. 1, pp. 42 – 55, 2004.

[16] K.S. Zigangirov and M. Lentmaier, "On the Asymptotic Iterative Decoding Performance of Low-Density Parity-Check Codes," in *Proceedings of International Symposium on Turbo codes and related topics*, 2000, pp. 39 – 42.

[17] G. Forney, "On Iterative Decoding and The Two-Way Algorithm," in *International Symposium on Turbo codes and related topics*, Brest, France, 1997, pp. 12 – 25.

[18] N. Wiberg, "Codes and Decoding on General Graphs," Ph.D. dissertation, Deptartment of Electrical Engineering, Linkping University, Linkping, Sweden, 1996.

[19] J. Chen and M. Fossorier, "Density Evolution for Two Improved BP-Based Decoding Algorithms of LDPC Codes," *IEEE Communications Letters*, vol. 6, no. 5, pp. 208 – 210, May 2002.

[20] A. Anastasopoulos, "A Comparison between The Sum-Product and The Min-Sum Iterative Detection Algorithms Based on Density Evolution," in *IEEE Global Telecommunications Conference (GLOBECOM)*, vol. 2, 2001, pp. 1021 – 1025.

[21] F. Zarkeshvari and A.H. Banihashemi, "On Implementation of Min-Sum Algorithm for Low-Density Parity-Check (LDPC) Codes," in *IEEE Global Telecommunications Conference (GLOBECOM)*, vol. 2, 2002, pp. 1349 – 1353.

[22] J. Zhao, F. Zarkeshvari, and A.H. Banihashemi, "On Implementation of Min-Sum Algorithm and its Modifications for Decoding Low-Density Parity-Check (LDPC) Codes," *IEEE Transactions on Communications*, vol. 53, no. 4, pp. 549 – 554, 2005.

[23] A.R. Rabbani Abolfazli, Y. R. Shayan, and G.E.R. Cowan, "TS-LDPC Analog Decoding Based on The Min-Sum Algorithm," in *26th Biennial Symposium on Communications (QBSC)*, 2012, pp. 162 – 167.

[24] A. Demosthenous, C. Verdier, and J. Taylor, "A New Architecture for Low Power Analogue Convolutional Decoders," in *Proceedings of 1997 IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 1, 1997, pp. 37 – 40.

[25] N. Correal, J. Heck, and M. Valenti, "An Analog Turbo Decoder for an (8,4) Product Code," in *The 2002 45th Midwest Symposium on Circuits and Systems (MWSCAS)*, vol. 3, 2002, pp. III – 632 – 635.

[26] C. Winstead, V.C. Gaudet, and C. Schlegel, "Analog Iterative Decoding for Error Control Codes," in *IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, vol. 3, 2003, pp. 1539 – 1542.

[27] N. Nguyen, C. Winstead, V.C. Gaudet, and C. Schlegel, "A 0.8V CMOS Analog Decoder for An (8,4,4) Extended Hamming Code," in *Proceedings of the 2004*

International Symposium on Circuits and Systems (ISCAS), vol. 1, 2004, pp. I – 1116 – 1119.

[28] S. Hemati, A.H. Banihashemi, and C. Plett, "A 0.18-um CMOS Analog Min-Sum Iterative Decoder for a (32,8) Low-Density Parity-Check (LDPC) Code," *IEEE Journal of Solid-State Circuits*, vol. 41, no. 11, pp. 2531 – 2540, 2006.

[29] D. Vogrig, A. Gerosa, A. Neviani, A. Amat, A. Gi, G. Montorsi, and S.Benedetto, "A 0.35-μm CMOS Analog Turbo Decoder for the 40-Bit Rate 1/3 UMTS Channel Code," *IEEE Journal of Solid-State Circuits*, vol. 40, no. 3, pp. 753 – 762, 2005.

[30] C. Winstead, N. Nguyen, V.C. Gaudet, and C. Schlegel, "Low-Voltage CMOS Circuits for Analog Iterative Decoders," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 53, no. 4, pp. 829 – 841, 2006.

[31] C. Kong and S. Chakrabartty, "Analog Iterative LDPC Decoder Based on Margin Propagation," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 54, no. 12, pp. 1140 – 1144, 2007.

[32] M. Gu and S. Chakrabartty, "A 100 pJ/bit, (32,8) CMOS Analog Low-Density Parity-Check Decoder Based on Margin Propagation," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 6, pp. 1433 – 1442, 2011.

[33] A. Acampora and R. Gilmore, "Analog Viterbi Decoding for High Speed Digital Satellite Channels," *IEEE Transactions on Communications*, 1978.

[34] H.L. Lou, "Implementing the Viterbi Algorithm," *IEEE Signal Processing Magazine*, vol. 12, no. 5, pp. 42 – 52, 1995.

[35] M.S. Shakiba, D.A. Johns, and K.W. Martin, "BiCMOS Circuits for Analog Viterbi Decoders," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 45, no. 12, pp. 1527 – 1537, 1998.

[36] H. Kim, H. Son, J. Lee, I. Kim, and H. Kim, "An Analog Viterbi Decoder for PRML using Analog Parallel Processing Circuits of the CNN," *2006 10th International Workshop on Cellular Neural Networks and Their Applications*, pp. 1 – 6, 2006.

[37] A.G. Amat, G. Montorsi, S. Benedetto, D. Vogrig, A. Neviani, and A. Gerosa, "An Analog Turbo Decoder for the UMTS Standard," in *Proceedings of International Symposium on Information Theory*, 2004.

[38] V. C. Gaudet and P. G. Gulak, "A 13.3-Mb/s 0.35μm CMOS Analog Turbo Decoder IC with a Configurable Interleaver," *IEEE Journal of Solid-State Circuits*, vol. 38, no. 11, pp. 2010 – 2015, 2003.

[39] C. Winstead, "Analog Iterative Error Control Decoders," Ph.D. dissertation, University of Alberta, Edmonton, AB, Canada, 2005.

[40] F. Lustenberger, M. Helfenstein, G. S. Moschytz, H. A. Loeliger, and F. Tarkoy, "All analog decoder for (18,9,5) tail-biting trellis code," in *Proceedings of the 25th European Solid-State Circuits Conference (ESSCIRC)*, 1999, pp. 362 – 365.

[41] S. Hemati and A.H. Banihashemi, "Full CMOS Min-Sum Analog Iterative Decoder," in *Proceedings of IEEE International Symposium on Information Theory*, 2003, p. 347.

[42] S. Y. Chung, G. D. Forney, Jr., T. J. Richardson, and R. Urbanke, "On the Design of Low-Density Parity-Check Codes within 0.0045 dB of the Shannon Limit," *IEEE Communications Letters*, vol. 5, no. 2, pp. 58 – 60, 2001.

[43] A.J. Felstrom and K.S. Zigangirov, "Time-Varying Periodic Convolutional Codes with Low-Density Parity-Check Matrix," *IEEE Transactions on Information Theory*, vol. 45, no. 6, pp. 2181 – 2191, 1999.

[44] M.C. Davey and D.J.C. Mackay, "Low-Density Parity Check Codes Over GF(q)," *IEEE Communications Letters*, vol. 2, no. 6, pp. 165 – 167, 1998.

[45] T.J. Richardson, M.A. Shokrollahi, and R.L. Urbanke, "Design of Capacity Approaching Irregular Low-Density Parity Check Codes," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 619 – 637, 2001.

[46] R. Zhang, *Linear Block Codes*. Department of Electrical and Computer Engineering, Drexel University, Technical Report ECE-S622/T602, Fall 2002.

[47] D. J. C. Mackay, "http://www.inference.phy.cam.ac.uk/mackay."

[48] J. Compello, D.S. Modha, and S. Rajagopalan, "Designing LDPC Codes using Bit-Filling," *IEEE International Conference on Communications (ICC)*, vol. 1, pp. 55 – 59, 2001.

[49] Y. Mao and A. H. Banihashemi, "A Heuristic Search for Good Low-Density Parity-Check Codes at Short Block Lengths," *IEEE International Conference on Communications (ICC)*, vol. 1, pp. 41 – 44, 2001.

[50] R. Townsend and E. Weldon, "Self-Orthogonal Quasi-Cyclic Codes," *IEEE Transactions on Information Theory*, vol. 13, no. 2, pp. 183 – 195, 1967.

[51] M. Karlin, "New Binary Coding Results by Circulants," *IEEE Transactions on Information Theory*, vol. 15, no. 1, pp. 81 – 92, 1969.

[52] Y. Kou, S. Lin, and M. P. C. Fossorier, "Low-Density Parity-Check Codes Construction Based on Finite Geometry," *IEEE Global Telecommunications Conference (GLOBECOM)*, vol. 2, pp. 825 – 829, 2000.

[53] ——, "Low-Density Parity-Check Codes construction based on Finite Geometries : a rediscovery and new results," *IEEE Transactions on Information Theory*, vol. 47, no. 7, pp. 2711 – 2736, 2001.

[54] J. H. Dinitz and D.R. Stinson, *A Brief Introduction to Design Theory.* Jhon Wiley & sons, 1992, ch. Contemporary Design Theory: A Collection of Surveys (J. H. Dinitz and D.R. Stinson, eds.), pp. 1 – 12.

[55] D. J. C. Mackay and M. C. Davey, *Evaluation of Gallager Codes for Short Block Length and High Rate Applications.* Springer-Verlag, New Yor, 2000, vol. 123, ch. Codes, Systems and Graphical Models; IMA Volumes in Mathematics and its Applications (B. Marcus and J. Rosenthal, eds.), pp. 113 – 130.

[56] S.J. Johnson and S.R. Weller, "Construction of Low-Density Parity-Check Codes from Kirkman Triple Systems," *IEEE Global Telecommunications Conference (GLOBECOM)*, vol. 2, pp. 970 – 974, 2001.

[57] ——, "Regular Low-Density Parity-Check Codes from Combinatorial Design," in *Proceedings of IEEE Information Theory Workshop*, 2001, pp. 90 – 92.

[58] B. Vasic, "Structured Iteratively Decodable Codes Based on Steiner Systems and Their Application in Magnetic Recording," *IEEE Global Telecommunications Conference (GLOBECOM)*, vol. 5, pp. 2954 – 2960, 2001.

[59] B. Vasic, E.M. Kurtas, and A.V. Kuznetsov, "LDPC Codes Based on Mutually Orthogonal Latin Rectangles and Their Application in Perpendicular Magnetic Recording," *IEEE Transactions on Magnetics*, vol. 38, no. 5, pp. 2346 – 2348, 2002.

[60] B. Vasic, "High-Rate Low-Density Parity-Check Codes Based on Anti-Pasch Affine Geometries," *IEEE International Conference on Communications (ICC)*, vol. 3, pp. 1332 – 1336, 2002.

[61] H. Song and B. V. K. Vijaya Kumar, "Low-Density Parity-Check Codes for Partial Response Channels," *IEEE Signal Processing Magazine*, vol. 21, no. 1, pp. 56 – 66, 2004.

[62] J. Lu and J.M.F. Moura, "Partition-and-Shift LDPC Codes," *IEEE Transactions on Magnetics*, vol. 41, no. 10, pp. 2977 – 2979, 2005.

[63] ——, "Partition-and-Shift LDPC Codes for High Density Magnetic Recording," *Digests of the IEEE International Magnetics Conference (INTERMAG Asia)*, pp. 991 – 992, 2005.

[64] ——, "Structured LDPC Codes for High-Density Recording: Large Girth and Low Error Floor," *IEEE Transactions on Magnetics*, vol. 42, no. 2, pp. 208 – 213, 2006.

[65] *IEEE 802.16e: Air Interface for Fixed and Mobile Broadband Wireless Access Systems*, IEEE Std. 802.16e, 2006.

[66] D.J.C. Mackay, S.T. Wilson, and M.C. Davey, "Comparison of Constructions of Irregular Gallager Codes," *IEEE Transactions on Communications*, vol. 47, no. 10, pp. 1449 – 1454, 1999.

[67] T. J. Richardson and R. L. Urbanke, "Efficient Encoding of Low-Density Parity-Check Codes," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 638 – 656, 2001.

[68] J. Lu and J.M.F. Moura, "Linear Time Encoding of LDPC Codes," *IEEE Transactions on Information Theory*, vol. 56, no. 1, pp. 233 – 249, 2010.

[69] D.A. Johns and K. Martin, *Analog Integrated Circuit Design*. John Wiley & Sons, 1997.

[70] B.A. Minch, "MOS Translinear Principle for All Inversion Levels," *IEEE Transactions on Circuits and Systems-II: Express Briefs*, vol. 55, no. 2, pp. 121 – 125, 2008.

[71] B. Gilbert, "A Precise Four-Quadrant Multiplier with Subnanosecond Response," *IEEE Journal of Solid-State Circuits*, vol. 3, no. 4, pp. 365 – 373, 1968.

[72] ——, "The Multi-Tanh Principle: A Tutorial Overview," *IEEE Journal of Solid-State Circuits*, vol. 33, no. 1, pp. 2 – 17, 1998.

[73] J. Lu, J.M.F. Moura, and U. Niesen, "A Class of Structured LDPC Codes with Large Girth," *2004 IEEE International Conference on Communications*, vol. 1, pp. 425 – 429, 2004.

[74] J. Lu and J.M.F. Moura, "Turbo Design for LDPC Codes with Large Girth," *4th IEEE Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pp. 90 – 94, 2003.

[75] H. Xiao-Yu, E. Eleftheriou, D.M. Arnold, and A. Dholakia, "Efficient Implementations of The Sum-Product Algorithm for Decoding LDPC Codes," *IEEE Global Telecommunications Conference (GLOBECOM)*, vol. 2, pp. 1036–1036E, 2001.

[76] R. Chen, H. Huang, and G. Xiao, "Relation between Parity-Check Matrixes and Cycles of Associated Tanner Graphs," *IEEE Communications Letters*, vol. 11, no. 8, pp. 674 – 676, 2007.

[77] V. Bhardwaj, N.P. Pathak, and A. Kumar, "Structured LDPC Codes with Linear Complexity Encoding," *WRI International Conference on Communications and Mobile Computing (CMC)*, vol. 1, pp. 200 – 203, 2009.

[78] S. Hemati and A. Banihashemi, "Dynamics and Performance Analysis of Analog Iterative Decoding for Low-Density Parity-Check (LDPC) Codes," *IEEE Transactions on Communications*, vol. 54, no. 1, pp. 61 – 70, 2006.

[79] M.J.M. Pelgrom, A.C.J. Duinmaijer, and A.P.G. Welbers, "Matching Properties of MOS Transistors," *IEEE Journal of Solid-State Circuits*, vol. 24, no. 5, pp. 1433 – 1439, 1989.

[80] P.G. Drennan and C.C McAndrew, "Understanding MOSFET Mismatch for Analog Design," in *Proceedings of the IEEE Custom Integrated Circuits Conference*, 2002, pp. 449 – 452.

[81] C. Winstead and C. Schlegel, "Density Evolution Analysis of Device Mismatch in Analog Decoders," in *Proceedings of International Symposium on Information Theory (ISIT)*, 2004.

[82] F. Lustenberger and H.A. Loeliger,, "On Mismatch Errors in Analog-VLSI Error Correcting Decoders," *The 2001 IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 4, pp. 198 – 201, 2001.

[83] P.R. Kinget, "Device Mismatch: An Analog Design Perspective," *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1245 – 1248, 2007.

[84] P. Kinget, "Device Mismatch and Tradeoffs in the Design of Analog Circuits," *IEEE Journal of Solid-State Circuits*, vol. 4, no. 6, pp. 1212 – 1224, 2005.

[85] B. Razavi, *Design of Analog CMOS Integrated Circuits*. McGraw-Hills, 2001.

[86] D.A. Johns and K. Martin, *Analog Integrated Circuit Design*, 1st ed. Wiley & Sons, 1997.

[87] C. Toumazou, F. I. Lidgey, and D. G. Haigh, *Analogue IC Design: The Current-Mode Approach*. Stevenage, UK: Peregrinus, 1990.

[88] N. Donckers, C. Dualibe, and M. Verleysen, "A Current-Mode CMOS Loser-Take-All with Minimum Function for Neural Computations," in *Proceedings of The 2000 IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 1, 2000, pp. 415 – 418.

[89] M. Rahman, K.L. Baishnab, and F.A. Talukdar, "A High Precision VLSI Loser-Take-All Circuit for Neural Networks and Fuzzy Systems," *IEEE International Conference on Computational Intelligence for Measurement Systems and Applications (CIMSA)*, pp. 28 – 31, 2009.

[90] A. Fish, V. Milrud, and O. Yadid-Pecht, "High Speed and High Resolution Current Loser-Take-All Circuit of O(N) Complexity," in *Proceedings of the 2004 11th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, 2004, pp. 234 – 237.

[91] T. Temel, "High-Performance Current-Mode Multi-Input Loser-Take-All Minimum Circuit," *Electronics Letters*, vol. 44, no. 12, pp. 718 – 719, 2008.

[92] M. Sasaki, T. Inoue, Y. Shirai, and F. Ueno, "Fuzzy Multiple-Input Maximum and Minimum Circuits in Current-Mode and Their Analyses Using Bounded-Difference Equations," *IEEE Transactions on Computers*, vol. 39, no. 6, pp. 768 – 774, 1990.

[93] M. Asloni, A. Khoei, and K. Hadidi, "Design of Analog Current-Mode Loser-Take-All Circuit," *IEICE Transaction on Electronics*, vol. E89-C, no. 6, pp. 819 – 822, 2006.

[94] C.Y. Huang, C.J. Wang, and B.D. Liu, "Modular Current-Mode Multiple Input Minimum Circuit for Fuzzy Logic Controllers," *Electronics Letters*, vol. 32, no. 12, pp. 1067 – 1069, 1996.

[95] C. Dualibe, P. Jespers, and M. Verleysen, "Embedded Fuzzy Control for Automatic Channel Equalization after Digital Transmissions," *The IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 3, pp. 173 – 176, 2001.

[96] A. Darabiha, A. C. Carusone, and F. R. Kschischang, "Power Reduction Techniques for LDPC decoders," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 8, pp. 1835 – 1845, 2008.

[97] Z. Zhang, V. Anantharam, M.J. Wainwright, and B. Nikolic, "An Efficient 10GBASE-T Ethernet LDPC Decoder Design with Low Error Floors," *IEEE Journal of Solid-State Circuits*, vol. 45, no. 4, pp. 843 – 855, 2010.

[98] C. Chen, Y. Lin, H. Chang, and C. Lee, "A 2.37-Gb/s 284.8 mW Rate-Compatible (491,3,6) LDPC-CC Decoder," *IEEE Journal of Solid-State Circuits*, vol. 47, no. 4, pp. 817 – 831, 2012.