Turbo Codes
A VLSI Implementation

By
Moeed Israr

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

Department of Electronics
Carleton University
Ottawa, Ontario, Canada

September 2006
© 2006 Moeed Israr

Library and
Archives Canada

Bibliothèque et
Archives Canada

Published Heritage
Branch

Direction du
Patrimoine de l'édition

395 Wellington Street
Ottawa ON K1A 0N4
Canada

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

ISBN: 978-0-494-23340-5

Our file Notre référence

ISBN: 978-0-494-23340-5

NOTICE:

The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protègent cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

Abstract

Turbo Codes – A VLSI Design is a Masters Thesis that involves research, architecture development, implementation and verification of a turbo encoder. The design achieves a maximum clock frequency of 333 MHz and features an on-chip memory and its controller. The VLSI implementation is carried out using generally accepted methodology, consisting of behavioral modeling and synthesis using Cadence and Synopsis. Test vectors generated by models developed in Matlab were used to confirm expected functionality. Individual behavioral blocks were verified prior to amalgamating them to form a turbo encoder. Synthesis and timing simulations were executed to determine power and area requirements along with the critical path. Results show successful simulations of the turbo encoder with a clock rate of 333 MHz.

The information used in this thesis comes in part from the research program of Dr. Tad Kwasniewski and his associates in the VLSI in Communications group. The research results appearing in this thesis represent an integral part of the ongoing research program. All research results in this thesis including tables, graphs and figures but excluding the narrative portions of the thesis are effectively incorporated into the research program and can be used by Dr. Kwasniewski and his associates for educational and research purposes, including publication in open literature with appropriate credits. The matters of intellectual property may be pursued cooperatively with Carleton University and Dr. Kwasniewski and when appropriate.

Acknowledgements

I would like to thank my supervisor, Dr. Tad Kwasniewski, for providing an interesting topic, direction and valuable advice.

Also thanks to Mr. Toqeer Israr, Mr. Mohammad Usama, Mr. Henry Wong and Mr. Gord Allan for their time and effort. Their advice and encouragement considerably enhanced the quality of this thesis. During the course of the program, I have come to appreciate the Electronics Research Group & Support Staff at Department of Electronics; Hardware, IS and IT support who have also contributed in numerous ways.

I would also like to thank my Brothers, Sisters and Wife who have provided me with much needed support. Finally, I would like to thank my Parents, for achieving this deed would not have been possible with out their guidance and support.

Table of Contents

1	INTRODUCTION	1
1.1	MOTIVATION.....	1
1.2	OBJECTIVES.....	4
1.3	CONTRIBUTIONS.....	5
1.4	REFERENCE TO PUBLICATION.....	7
1.5	THESIS OUTLINE.....	8
2	BACKGROUND	9
2.1	VLSI IMPLEMENTATION	10
2.1.1	<i>Previously Established Results</i>	<i>10</i>
2.1.2	<i>Area Optimization Techniques</i>	<i>12</i>
2.1.3	<i>Power Consumption.....</i>	<i>13</i>
2.1.4	<i>Clock Signal and System Timing.....</i>	<i>15</i>
2.2	TURBO ENCODERS	17
2.2.1	<i>Recursive Systematic Convolutional Encoder</i>	<i>19</i>
2.2.2	<i>Interleaving Strategies</i>	<i>22</i>
2.2.3	<i>Punctured Codes</i>	<i>24</i>
2.2.4	<i>Termination Methods</i>	<i>28</i>
2.3	TURBO DECODER	29
2.3.1	<i>Iterative Decoder Structure</i>	<i>31</i>
2.3.4	<i>MAP Algorithm Modifications</i>	<i>39</i>
3	ARCHITECTURE & DEVELOPMENT	41
3.1	DATA GENERATION MODULATION SCHEME.....	44
3.1.1	<i>Data Generation.....</i>	<i>44</i>
3.1.2	<i>Binary Modulation</i>	<i>45</i>
3.2	CHANNEL NOISE AND COMPARATOR	46
3.2.1	<i>Channel Noise</i>	<i>46</i>
3.2.2	<i>Comparator.....</i>	<i>46</i>
3.3	TURBO ENCODER.....	47
3.3.1	<i>Recursive Systematic Convolutional Encoder with Termination</i>	<i>54</i>
3.3.2	<i>Pseudo Random Index Interleaver.....</i>	<i>64</i>
3.3.3	<i>Memory and Memory Controller.....</i>	<i>68</i>
3.4	TURBO DECODER	77
3.4.1	<i>Log MAP Decoding</i>	<i>78</i>
3.4.2	<i>System performance vs. number of decoder Iterations</i>	<i>84</i>
3.4.3	<i>De-Interleaver</i>	<i>85</i>
4	HARDWARE VERIFICATION AND OPTIMIZATION	86
4.1	VERIFICATION.....	88
4.2	REDUCED AREA AND POWER FIGURES	89
4.3	TIMING ANALYSIS	91
4.4	SAVINGS DUE TO INTERLEAVER DESIGN	92
4.5	SAVINGS DUE TO DUAL-PORT ON-CHIP MEMORY DESIGN	94
4.6	SINGLE-PORT ON-CHIP MEMORY DESIGN	95
5	CONCLUSION AND FUTURE WORK	97

REFERENCES.....	101
BIBLIOGRAPHY.....	105
APPENDIX A – DESIGN PROCESS	108
APPENDIX B – THEORETICAL CALCULATIONS	109
APPENDIX C – TRANSFER FUNCTIONS	110
APPENDIX D – COMPARISON OF PERFORMANCE	117

List of Figures

FIG. 1. A SCC ENCODER WITH OUTER AND INNER CODING BLOCKS	17
FIG. 2. A PCC ENCODER WITH TWO ENCODERS AND AN INTERLEAVER	18
FIG. 3. TURBO ENCODER FUNCTIONAL DIAGRAM.....	19
FIG. 4. AN EXAMPLE RSC ENCODER WITH FEEDBACK, $G_1(D)$ BEING $[1+D+D^2]$ AND FEEDFORWARD, $G_2(D)$ BEING $[1+D^2]$	20
FIG. 5. 1/2 CODE RATE WITH ONE SYSTEMATIC SEQUENCE AND ONE PARITY SEQUENCE.	24
FIG. 6. TRELLIS DIAGRAM FOR THE (2,1) NONSYSTEMATIC ENCODER [1].	25
FIG. 7. 1/3 CODE RATE WITH NO PUNCTURING USES IDENTICAL TRELLIS STRUCTURES IN RSC1 AND RSC2 FOR ENCODING PARITY1 AND PARITY2.	26
FIG. 8. TRELLIS DIAGRAM FOR BOTH RSC1 AND RSC2.....	26
FIG. 9. PUNCTURING WITH 2/3 CODE.....	27
FIG. 10. TRELLIS DIAGRAM OF A 2/3-RATE PUNCTURED CODE PRODUCED BY PERIODICALLY DELETING SYMBOLS (MARKED WITH 'X') FROM A RATE 1/2 CODE [16].....	28
FIG. 11. A SAMPLE TERMINATION SCHEME	29
FIG. 12. BLOCK DIAGRAM OF A CONSTITUENT DECODER	30
FIG. 13. TURBO DECODER FUNCTIONAL DIAGRAM.....	31
FIG. 14. THE FIGURE ABOVE SHOWS A BLOCK DIAGRAM OF A LOG MAP DECODER	34
FIG. 15. BLOCK DIAGRAM OF A MAP DECODER	34
FIG. 16. DIGITAL COMMUNICATION CHANNEL.....	41
FIG. 17. DESIGN PROCESS AND TOOLS USED DURING THE COURSE OF THE STUDY	43
FIG. 18. SIGNAL POINTS FOR BINARY ANTIPODAL SIGNALS	45
FIG. 19. ENCODER FUNCTIONAL DIAGRAM WITH 1/3 RATE CODING	47
FIG. 20. MATLAB SIMULATION BLOCKS FOR THE TRANSMIT STAGE	49
FIG. 21. HARDWARE BLOCK DIAGRAM.....	50
FIG. 22. TURBO ENCODER MODULES	51
FIG. 23. TURBO ENCODER BLOCK DIAGRAM	51
FIG. 24. TOP ENCODER INTERNAL HARDWARE.....	52
FIG. 25. BEHAVIOR WAVEFORMS FOR THE TURBO ENCODER SHOWS THE OUTPUT STREAMS MATCH THE TEST VECTORS.....	52
FIG. 26. BEHAVIOR WAVEFORMS – TOP LEVEL TURBO ENCODER FOR THE LAST SECTION OF THE FRAME.....	53
FIG. 27. TRANSFER FUNCTION VS. BER	55
FIG. 28. STRUCTURE OF RECURSIVE SYSTEMATIC ENCODER.....	56
FIG. 29. STRUCTURE OF RECURSIVE SYSTEMATIC ENCODER WITH TERMINATION BITS	56
FIG. 30. TRELLIS DIAGRAM REPRESENTS THE NEXT STATE CONDITIONS FOR ALL POSSIBLE INPUTS WHEN STARTING INITIALLY FROM $S = 0$	57

FIG. 31. TRELLIS DIAGRAM SHOWING THE OUTPUTS ASSOCIATED WITH EACH STATE TRANSITION.	58
FIG. 32. TERMINATION TRELLIS	61
FIG. 33. INTERNAL HARDWARE OF RSC USED TO COLLECT TERMINATION BITS	62
FIG. 34. INTERNAL HARDWARE OF RSC1 USED TO GENERATE PARITY1	62
FIG. 35. INTERNAL HARDWARE OF RSC2 USED TO GENERATE PARITY2	63
FIG. 36. BEHAVIOR WAVEFORMS OF RSC1	63
FIG. 37. BEHAVIOR WAVEFORMS OF RSC2 USING INTERLEAVED DATA.	63
FIG. 38. INTERLEAVING DEPTH VS. BER	65
FIG. 39. 10 BIT, ALL STATE PSEUDO RANDOM INTERLEAVER	67
FIG. 40. WAVEFORM OF THE BEHAVIOR MODEL	67
FIG. 41. INTERNAL HARDWARE OF DATA INTERLEAVER.....	67
FIG. 42. MEMORY CONTROLLER – WRITE TO MEMORY FUNCTIONAL DIAGRAM.....	69
FIG. 43. MEMORY CONTROLLER – READ FROM MEMORY FUNCTIONAL DIAGRAM FOR RSC1	70
FIG. 44. MEMORY CONTROLLER – READ FROM MEMORY FUNCTIONAL DIAGRAM FOR RSC2	71
FIG. 45. DUAL-PORT SDRAM CHIP ARCHITECTURE	73
FIG. 46. WAVEFORM OF THE BEHAVIOR MODEL ACCESSING MEMORIES.....	73
FIG. 47. INTERNAL HARDWARE OF THE MEMORY CONTROLLER.....	74
FIG. 48. INTERNAL HARDWARE OF MEMORY CONTROLLER’S DE-SERIALIZING SUB-BLOCK.	75
FIG. 49. INTERNAL HARDWARE OF MEMORY CONTROLLER’S COUNTER.	75
FIG. 50. INTERNAL HARDWARE OF MEMORY CONTROLLER’S COUNTER1.	76
FIG. 51. TURBO DECODER FUNCTIONAL DIAGRAM.....	78
FIG. 52. LOG MAP DECODER BLOCK DIAGRAM WHERE IS THE SYMBOL FOR MEMORY.	79
FIG. 53. LOG MAP DECODER TRELLIS SHOWING THE ALPHA, BETA AND GAMMA CALCULATIONS REQUIRED FOR EACH POSSIBLE STATE OF THE BIT IN THE SEQUENCE.	79
FIG. 54. GAMMA CALCULATION BLOCK DIAGRAM.....	80
FIG. 55. FOR EACH TIME STEP, GAMMA CALCULATIONS ARE REQUIRED FOR ALL TRELLIS NODES WHERE MC IS $K - 1$ (WHERE ‘K’ IS THE CONSTRAINT LENGTH OF THE TRANSMITTER).	80
FIG. 56. FOR EACH TIME STEP ALPHA CALCULATIONS ARE REQUIRED FOR ALL TRELLIS NODES.....	81
FIG. 57. ALPHA CALCULATION FLOW DIAGRAM.....	81
FIG. 58. SAMPLE STATE TRANSITIONS.....	82
FIG. 59. FOR EACH TIME STEP, BETA CALCULATIONS ARE REQUIRED FOR ALL TRELLIS NODES.....	82
FIG. 60. BETA CALCULATION FLOW DIAGRAM.....	83
FIG. 61. ITERATIONS VS. BER 1024 BIT FRAME, $G_1(D) = [1\ 1\ 1\ 1]$, $G_2(D) = [1\ 0\ 1\ 1]$, SINGLE TRELLIS TERMINATION, 1/3 CODE RATE AND RANDOM INTERLEAVER.....	84
FIG. 62. DESIGN PROCESS AND TOOLS USED	87
FIG. 63. AREA REQUIREMENTS FOR EACH TURBO ENCODER BLOCK	89
FIG. 64. POWER CONSUMPTION FOR EACH TURBO ENCODER BLOCK.....	90

FIG. 63. AREA REQUIREMENTS FOR EACH TURBO ENCODER BLOCK	89
FIG. 64. POWER CONSUMPTION FOR EACH TURBO ENCODER BLOCK.....	90
FIG. 65. ARCHITECTURE WITH DUAL-PORT ON-CHIP MEMORY	94
FIG. 66. ARCHITECTURE WITH SINGLE-PORT ON-CHIP MEMORY	96
FIG. 67. PERFORMANCE GRAPH OF 1/3 RATE FROM 'HANDBOOK OF RF AND WIRELESS TECHNOLOGIES' [23].....	117
FIG. 68. INTERLEAVING DEPTH VS. BER	118

List of Tables

TABLE 1. PREVIOUSLY ESTABLISHED RESULTS.....	11
TABLE 2. RESULTS ESTABLISHED IN 2005.....	12
TABLE 3. SUMMARY OF SYMBOLS USED IN TURBO ENCODER	19
TABLE 4. SUMMARY OF SYMBOLS USED IN DECODER ITERATIONS.....	32
TABLE 5. SUMMARY OF SYMBOLS USED IN LOG- MAP DECODING	35
TABLE 6. SYSTEM PARAMETERS OF THE TURBO ENCODER	48
TABLE 7. SUMMARY OF TURBO ENCODER DESIGN.....	53
TABLE 8. SIMULATION CONDITION FOR TRANSFER FUNCTION ANALYSIS.....	54
TABLE 9. NEXT STATE CALCULATIONS FOR THE TRELLIS OF FIG. 31 USING TRANSFER FUNCTION 1: $G = [1 \ 1 \ 1; 1 \ 0 \ 1]$	59
TABLE 10. TERMINATION -1	60
TABLE 11. LAST STATE AND LAST OUTPUT TABLE -1	61
TABLE 12. SUMMARY OF RSC2 ENCODER DESIGN.....	64
TABLE 13. PARAMETERS FOR THE INTERLEAVING DEPTH SETUP	65
TABLE 14. SUMMARY OF INTERLEAVER DESIGN	68
TABLE 15. SUMMARY OF MEMORY CONTROLLER DESIGN	76
TABLE 16. INITIAL AND OPTIMIZED POWER AND AREA FIGURES	89
TABLE 17. SUMMARY OF ENCODER DESIGNS.....	92
TABLE 18. SUMMARY OF THESIS FINDINGS (ALSO SEE TABLE 19)	97
TABLE 19. PREVIOUSLY ESTABLISHED RESULTS AND FINDINGS OF THE THESIS.....	98
TABLE 20. COMPARISON OF MAXIMUM CLOCK FREQUENCY OF PREVIOUSLY PUBLISHED RESULTS AND THESIS FINDINGS.....	99
TABLE 21: S0 FORMULA VARIABLES.....	111
TABLE 22: OUTPUT2 FORMULA VARIABLES.....	111
TABLE 23 TRANSFER FUNCTION 1: $G = [1 \ 1 \ 1 \ 1; 1 \ 0 \ 1 \ 1]$	112
TABLE 24. TERMINATION -1	112
TABLE 25. LAST STATE AND LAST OUTPUT TABLE -1	113
TABLE 26. TRANSFER FUNCTION 2: $G = [1 \ 1 \ 1 \ 1; 1 \ 1 \ 0 \ 1]$	114
TABLE 27. TERMINATION - 2 : $G = [1 \ 1 \ 1 \ 1; 1 \ 1 \ 0 \ 1]$	114
TABLE 28. LAST STATE / LAST OUTPUT - 2: $G = [1 \ 1 \ 1 \ 1; 1 \ 1 \ 0 \ 1]$	114
TABLE 29. TRANSFER FUNCTION 3: $G = [1 \ 0 \ 1 \ 1; 1 \ 1 \ 1 \ 1]$	115
TABLE 30. TERMINATION - 3 $G = [1 \ 0 \ 1 \ 1; 1 \ 1 \ 1 \ 1]$	115
TABLE 31. LAST STATE / LAST OUTPUT - 3 $G = [1 \ 0 \ 1 \ 1; 1 \ 1 \ 1 \ 1]$	115
TABLE 32. TRANSFER FUNCTION 4: $G = [1 \ 0 \ 0 \ 1; 1 \ 0 \ 1 \ 1]$	116
TABLE 33. TERMINATION - 4: $G = [1 \ 0 \ 0 \ 1; 1 \ 0 \ 1 \ 1]$	116

TABLE 34. LAST STATE / LAST OUTPUT - 4: $G = [1\ 0\ 0\ 1; 1\ 0\ 1\ 1]$	116
TABLE 35 COMPARISON OF PERFORMANCE FIGURES FROM [23] AND THESIS. THE % DIFFERENCE IS CALCUTED BY $[\text{SNR}_{\text{THESIS}} - \text{SNR}_{\text{HANDBOOK}}] / \text{SNR}_{\text{THESIS}}$	118
TABLE 36 THE TABLE CAPTURES THE DIFFERENCES IN PARAMETERS AND HOW THEY IMPACT THE PERFORMANCES.	119

Table of Abbreviations

3GPP	Third Generation Partnership Project
ARQ	Automatic Request for Repeat
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
PAM	Pulse Amplitude Modulation
CD1	Constituent Decoder 1
CD2	Constituent Decoder 2
CDMA	Code Division Multiple Access
DUT	Device Under Test
ECC	Error Correcting/Control Code
FCC	Federal Communications Commission
LLR	Log Likelihood Ratio
LogMAP	Logarithmic Maximum a posteriori
MAP	Maximum <i>a posteriori</i>
NMOS	n-type MOSFET
MAX LogMAP	method to calculate Maximum Logarithmic Maximum a posteriori
PCC	Parallel Concatenated Convolution
PMOS	p-type MOSFET
Rate	Original data / Encoded bits
RSC	Recursive Systematic convolutional
RTL	Register Transfer Level
SCC	Serial Concatenated Convolution code
SISO	Soft Input Soft Output
SNR	Signal to Noise Ratio
SOVA	Soft Output Viterbi Algorithm
UEP	Unequal Error Protection Codes

Table of Symbols

Variable	Comment
α	result of forward recursion. Alpha contain $n * 2^m$ elements in MAP algorithm whereas in Log-MAP, it contain 'n' elements.
$\alpha_{i-1}(\hat{s})$	$(i-1)^{th}$ result of forward recursion.
β	result of backward recursion. Beta contain ' $n * 2^m$ ' elements in MAP algorithm whereas in Log-MAP, it contain 'n' elements.
$\beta_i(s)$	i^{th} result of backward recursion.
γ	is the state transistions branch metric
$\gamma(\hat{s}, s)$	branch transition probabilities
$(\hat{s}, s) \Rightarrow c_i = +1$	when the input bit c_i is +1, these are the set of transitions that can occur from the previous state, ' s ', to the present state, s
$(\hat{s}, s) \Rightarrow c_i = -1$	when the input bit c_i is -1, these are the set of transitions that can occur from the previous state, ' s ', to the present state, s
σ^2	noise variance
a	fading amplitude
c_i	input bit at the turbo decoder
D	state of a generator polynomial
E_b	energy per bit
$g_1(D)$	feedback polynomial
$g_2(D)$	output polynomial
i	is the position of the bit being processed in the sequence
$I_L^1(u)$	interleaved extrinsic sequence for CD1
$I_L^2(u)$	interleaved extrinsic sequence for CD2
I_x	interleaved input
k	number of input bits per symbol period
K	constraint length of the transmitter
$L(c_i y)$	probability of receiving the bit ' c_i ' given y , the received symbol.
$L(c_i)$	i^{th} bit of the received sequence
$L(u_{a_i})$	is the i^{th} apriori value

Variable	Comment
$L^1(c)$	received sequence for constituent decoder 1 (CD1)
$L^1(ua)$	apriori sequence for CD1
$L^1(ue)$	extrinsic sequence for CD1
$L^2(c)$	received sequence for constituent decoder 2
$L^2(ua)$	apriori sequence for CD2
$L^2(ue)$	extrinsic sequence for CD2
m	memory length at the transmitter (K-1)
n	number of bits in each packet
$P(c_i)$	a-priori likelihood of c_i
R	rate of the convolutional code
$S_i = s$	present state in the trellis
x	input vector at the turbo encoder
x_i	symbol of the i^{th} transition.
x_{il}	are bits in the received sequence x_i
x_{il} and y_{il}	are bits in the received sequence y_i and x_i for parity 1
x_{is}	decoded bit, c_i
y_{il}	are bits in the received sequence y_i
y_{is}	is the received bit of the transmitted systematic bit
$y_{j<i}$	$i - 1$ received symbols,
$y^{(3)}$	parity 2 sequence bit
$y^{(2)}$	parity 1 sequence bit
$S_{i-1} = \acute{s}$	previous state in the trellis
$y^{(1)}$	systematic bit
$y_{j>k}$	received sequence up to current transition
$P(S_{i-1} = \acute{s} \wedge y_{j<i})$	Bayes's rule that $P(\acute{s} y_{j<i}) * P(y_{j<i})$

Glossary

The terms a-priori, a-posteriori, and extrinsic information are fundamental concepts of the iterative decoding of turbo codes used throughout this thesis and are defined as follows:

a-priori

a-priori information is also known as intrinsic information. It is a likelihood value that complements information about the same bit from other sources (such as received sequence or the code constraints). In turbo decoding, a-priori information and received sequence are collected before decoding can begin [1].

a-posteriori

The a-posteriori LLR information of a input bit, c_i , is the output information a decoder generates using parity, systematic and a-priori sequences [1].

extrinsic

The extrinsic information refers to a figure calculated by a decoder. It is an output calculated using the received sequence and a priori information for each bit c_i [1]. The extrinsic values do not require the use of systematic bits. *“Typically, the [constituent] decoder ... processes the received bits and a-priori information surrounding the systematic bit, and uses this information and the code constraints to provide information about the value of $[c_i]$ ”* [1].

1 INTRODUCTION

1.1 *Motivation*

Error control codes (ECC) are an integral part of digital communication systems. It involves encoding original data and/or introducing parity bits to achieve improved bit error rate (BER). It is widely used in but not limited to 3G wireless, other mobile, satellite and radio communications [2]. Systems that are sensitive to disruptions or delay such as medical telemetry, video conferencing require ECC such that automated repeat requests (ARQ) are not issued to re-transmit data. Such applications also include real-time measurements and control applications.

Voice and data transmissions vary significantly in bandwidth requirements, acceptable latency, and bit error rate. All transmission modes are susceptible to systematic and random noise, which impacts wireless communications more than wired communications. To compensate for noise, techniques such as increasing the power of the signal transmitter, ARQ or ECC are used. Increasing the signaling power may improve the signal-to-noise (SNR) ratio; however, it poses its own challenges. The power levels must comply with the Federal Communications Commission's (FCC) regulations about transmission power levels. Moreover, increasing the signaling power decreases the battery life; this is a concern to users with hand held devices. Finally, increasing power levels may have minimal impact on the signal quality, if the source of

the noise is close to the transmitter. ARQ is a viable option when latency and disruptions can be tolerated. Having to verify and request to re-send corrupted packets results in increased bandwidth requirements and delays which may be unacceptable for video conferencing or real-time applications. ECC may not be suitable for all applications but with turbo encoding and decoding, it comes close to the theoretical limits of data transmission in presence of noise [2].

ECC integrates extra information, known as parity bits, into the signal (digital data). Depending on the encoding used, the receiver employs a decoder that utilizes parity bits to detect and correct errors that may have occurred during transmission. Global System for Mobile Communications, Third Generation and Next Generation mobile wireless communication system can consist of two distinct channel coding methods: convolutional and turbo coding. Convolutional coding is naturally suited for applications with little tolerance to latency, such as voice communications and control channels. For applications that require high data rates in terms of block size and number of blocks, turbo codes provide significantly better bit error rate performance. It is an ECC scheme that approaches the theoretical limits predicted by Claude Shannon [3]. The scheme and a practical decoding algorithm were later introduced by a group of French researchers [1].

The turbo codes are parallel concatenated codes (PCC) that require the input stream to be encoded by two constituent encoders with a data interleaver between them [5]. Based on the SNR of the channel, the turbo encoder can be designed to operate at 1/2 or 1/3 code rate.

At the receiver, the turbo decoder employs two trellis-based soft-input-soft-output (SISO) decoders to process the received sample sequence. The received symbols, digitized analog inputs (soft inputs), are processed to calculate reliability figures by the decoder outputs. Depending on the bit-error-ratio (BER) or latency requirements, the received signals and the reliability figures are processed iteratively. At the final iteration, the soft-decisions and the soft-input received symbols are used to calculate the transmitted bit. Trellis-based decoders which generate soft outputs, such as MAP (see section 3.4) or soft output Viterbi algorithm are appropriate for turbo decoding. For the purposes of this study, MAP has been selected as the constituent decoder.

Another key component of turbo encoding/decoding is the interleaver. The interleavers re-arrange the data sequences of both the received symbols and reliability figures to match the expected scrambling function.

Digital ASIC implementation of turbo encoding, and architecture development of the encoder and decoder were led by system level simulations in Matlab. The simulations helped determine the impact of decoder iterations, encoder transfer function and block size on latency, throughput and BER. Hardware design of a turbo encoder with dual-port on-chip memory targeting 0.18 μm CMOS technology was implemented. It reduces the memory requirements in half. Architectural, behavioral and processing tools are made to reduce power and area requirements. Measured power, area and timing figures are presented in section 3.3.

1.2 Objectives

The objectives of this thesis are as follows:

1. To implement a turbo encoder hardware structure targeting 0.18 μm CMOS technology, along with on-chip memory and its controller. Architectural and block level considerations are made to achieve reduced power and area, while achieving increased maximum clock frequency, minimal latency and maximum throughput.
2. To develop a turbo encoding/decoding architecture using Matlab. System level simulations are able to verify the effectiveness and efficiency of the turbo encoding and decoding process. The system is a digital communication channel consisting of random data generation, turbo encoding, 2-PAM modulation/demodulation, additive white gaussian noise (AWGN) channel, turbo decoding and BER calculations.

1.3 Contributions

The contributions of this research are as follows:

1. The following turbo encoder ASIC results based upon 1.8V V_{DD} have been achieved:
 - i. area of 0.986 mm²
 - ii. power of 46.1 mW
 - iii. maximum clock frequency of 333 MHz.
2. Initial results of the turbo encoder ASIC were published in IEEE's Workshop on System on Chip 2005. The ASIC required a core area of 1.2 mm², had a maximum clock rate of 250 MHz and consumed 56 mW.
3. An ASIC performing turbo encoding has been developed. It features
 - i. 0.18 μ m CMOS on-chip memory which operates at a maximum frequency of 333 MHz.
 - ii. an architecture that employs dual-port memory design, reducing the storage (memory) requirements by one-half.
 - iii. the design of pseudo random index interleaver (PR-II), eliminating the memory requirement for storing index addresses.
4. Prior to VLSI implementation; a Matlab-based simulator was developed to verify and achieve performance targets of turbo codes as part of a digital communication channel. The simulator provides insight into the influence of various design parameters on BER, throughput, latency and computational requirements. For instance, the following turbo encoding aspects have been studied and analyzed:

- i. Various encoder transfer functions were simulated for BER performances.
- ii. Effects of varying decoder iterations on BER have also been studied and are presented in section 3.4.2.

1.4 Reference to Publication

A summary of the research, analysis, methodology and results presented in this thesis have been published at ‘The 5th IEEE International Workshop on System on Chip for Real-Time Applications’, 2005. The publication, ‘Turbo Codes – A Digital IC DESIGN’ has been quoted and paraphrased numerous times in the thesis.

1.5 Thesis Outline

The topics addressed by this thesis are as follows:

- Chapter 1 is the introduction, which explains the motivation and scope of the research.
- Chapter 2 provides an overview of the prior research in the area and includes the discussion of VLSI implementation issues, turbo encoder and turbo decoder.
- Chapter 3 presents turbo encoder VLSI Implementation and describes the development of the architecture of the turbo encoder and turbo decoder. The architecture has been verified by simulations. A digital communication channel, consisting of turbo coding/decoding, 2-PAM modulation/demodulation and AWGN Channel, has been simulated. Overview, algorithms, design parameters and simulation results have been presented. The system performance of the architecture has been compared with the most recent literature on the subject.
- Chapter 4 details the VLSI architectural issues in a top-down fashion. It includes implementation methodology, synthesis area breakdown, speed constraints, net profile and power estimates.
- Chapter 5 draws conclusions of the research and provides directions for future studies on this topic.

2 BACKGROUND

Turbo codes, introduced in 1993, constitute a breakthrough in the coding community. While achieving excellent bit-error-rate performance, turbo codes suffer from a low decoding throughput performance [7]. Until recently, latency, implementation of complex equations and extensive memory usage resulted in the ASIC's power and area figures beyond the feasibility of handheld devices [8]. With reduction in the algorithm complexity and increased demand for high bandwidth requirements, 3G Wireless has adopted turbo codes as an integral part of their standard. The subsequent sections of the chapter introduce:

- VLSI design and its implementation issues
- turbo encoding
- turbo decoding

2.1 VLSI Implementation

The complexity and the size of the turbo encoder lends itself to a development of an ASIC, although development using an FPGA is certainly achievable. The VLSI design process for an ASIC requires thorough research to develop architecture, clear specifications and logic design. Synthesis of logic into circuits, verification and layout complete the implementation phase. During the development of a turbo encoder, multiple levels of abstraction, such as simulation code, state graphs, register transfer level (RTL), transistors and memories, were managed. In complex designs, it is common to deal with conflicting constraints such as low power versus high speed, or low cost versus high performance. Once expected functionality was achieved, the design was optimized for maximum operational frequency, while improving area and power requirements. The following subsections describe previously published results along with various power consumption components. Methods for power and area reduction, clock signal stability and system timing are also introduced.

2.1.1 Previously Established Results

Published results show a turbo decoding VLSI, using 0.18 μm CMOS technology with 1.8 V DC rail, operating at 110.8 MHz [6]. The publication, authored by ten researchers, presents a turbo encoder/decoder design which uses 85K logic gates, requires 239 Kb of RAM and has an overall core area of approximately 9mm^2 (excluding power rings, I/O drivers and pads).

Another publication, [25], developed both an encoder and a decoder. Based on the turbo encoder presented in the publication, the technique stores the interleaved data packet, and original data packet, RSC1-encoded-data packet to synchronize the data streams. The parameters for the turbo encoder VLSI are presented in the table below.

	Estimated previously published results with its parameters [25]
Component encoders	RSC encoders
encoding Rate	1/3 rate with no puncturing
RSC transfer function	$[17_H; 23_H]$
Interleaver	Block Interleaver
Frame size L	1048 bits
Termination method	No termination
Area ^a	$\sim 1.81 \text{ mm}^2$ ^c
Maximum Frequency ^b	130 MHz ^d
Power	85.33 mW

Table 1. Previously established results.

- a. The area results have been scaled from 0.25 μm CMOS technology to 0.18 μm CMOS technology.
- b. The frequency results have been scaled by using $f_{t-0.25 \mu\text{m CMOS}} = 30 \text{ GHz}$ and $f_{t-0.18 \mu\text{m CMOS}} = 70 \text{ GHz}$ [26]
- c. The figure includes both the encoder and the decoder. Individual encoder figures are not provided.
- d. The maximum frequency includes only the encoder.

During the final compilation of the thesis, a third publication was published [28]. It uses SiGe BiCMOS and 0.13 CMOS technology to achieve a throughput of 10 Gbps. The parameters for the turbo encoder/decoder VLSI are presented in the table below.

	Estimated previously published results with its parameters [REF]
Component encoders	BCH
encoding Rate	1/3 rate with no puncturing
Tansfer function	BCN1 (144,128) x BCN2 (256, 239)
Interleaver	Block Interleaver
Frame size L	NA
Termination method	NA
Area ^a	40 x 40 mm ²
Throughput	10 Gbps
Power	10 W

Table 2. Results established in 2005.

Considering the study uses BCH encoding scheme and block interleavers, the performance of the system and the hardware implementation is vastly different and does not relate to a direct comparison with this study. Having said that, the hardware performance has to be noted and studied for future reference.

2.1.2 Area Optimization Techniques

Area reduction strategy varies according to the complexity of the design, the technology involved and the application. An acceptably optimized design meets timing requirements and occupies the smallest area. Generally accepted optimization approach targets

architecture level, implementation at the RTL code level, and successive passes through logic synthesis using software tools [10].

The approaches to architectural and RTL code level optimization are similar. In both instances, area optimization is achieved through considering acceptable options. Analysis and creative thinking are required to develop further enhancements that improve area figures. In fact, the VLSI architecture presented in the thesis, results in significantly smaller area requirements.

Logic synthesis, a central design step for most ASIC design projects, creates the net-list. The net-list is then mapped into circuits using standard cell libraries. In digital integrated circuits, for a given technology, reducing the number of gates reduces the area requirements. Optimal gate-level net-list is created by carrying out successive passes using an automated synthesis tool.

The optimization carried out at the architecture level, RTL code level, and logic synthesis reduces the gate count, thus reducing the design size.

2.1.3 Power Consumption

The power consumed by electronic circuits is a key concern in hand held devices. Large batteries and heat dissipation are directly linked to power consumed by the circuits. With ever-increasing operational frequencies and complexity, designers seek to minimize the

power consumption. In 0.18 μm CMOS digital circuits, dynamic power, short-circuit power, and leakage power constitute the bulk of the figure [9].

Switching Power Dissipation

Upon changing states from logic '0' to logic '1' or vice versa, a transistor dissipates power. The dissipation is due to the energy required from the power supply for a transistor output to make a full transition from one power rail to another. The power used for the transition is frequency- dependent. Moreover, power consumption of a block is largely dependent on its utilization and operational frequency.

Leakage Power Dissipation

Due to nonzero reverse leakage and sub-threshold currents, CMOS circuits when connected to a power supply, tend to leak current even when the transistors are hold their states [9]. The leakage current is negligible for a transistor in 0.18 μm CMOS technology. However, for an IC with hundreds and thousands of transistors and gate size of 90 nm or smaller, the cumulative leakage current can significantly impact the total power figures. The processing parameters for a transistor determine the magnitude of the leakage current.

Power Reduction Techniques

In VLSI, the following techniques decrease the amount of required power:

1. Reduction of switching activity (algorithmic simplifications)
2. Parallel processing (duplication of hardware) to reduce the overall clock frequency for blocks requiring high speeds.
3. Pipelining (slow down max clock frequency) by breaking the algorithms down into smaller sections to be executed sequentially.

2.1.4 Clock Signal and System Timing

Clock signals are vital reference signals in a digital system and they must be reliable. A perfect clock signal would generate reflections; probably some increased power loss but would have specific duty cycles, and be free of skew. Variations in the length of clock net paths and load conditions influence the clock arrival times at the destination [9]. The maximum disparity of arrival times determines the clock skew.

Typically a chip receives one or more clock signals, and generates subsequent nets to distribute the reference signals to the clocked logic taking into account load conditions [9]. Load conditions are deterministic and are balanced by ensuring that nets created to distribute the clock have an identical number of fan-out gates. The clock signals must be buffered in multiple stages to handle the high fan-out loads. In addition to buffers, automated layout tools also help generate clock paths' geometry that reduces the clock skew.

In a conventional digital design, the following issues are considered: [9]

- 50% duty cycle of a clock signal helps
 - when the signal is being passed through a series of invertors
 - with timing when both rising and falling edges are active edges.
- Excessively small rise and fall times can result in reflections in the external interconnects.
- Having large fan-out increases load capacities, which slows the rise and fall times.

- Matching of characteristic impedance of the clock distribution line and receivers.

However, this is not usually considered for chips smaller than 10x10 mm.

2.2 Turbo Encoders

Concatenated codes apply two levels of coding, an inner code and an outer code to obtain a better BER with simpler encoding and decoding than that needed for a single code. Two such types of codes are serially concatenated code (SCC) and parallel convolutional codes (PCC).

Fig. 1. A SCC encoder with outer and inner coding blocks

SCC employs two levels of coding, an inner-code and an outer-code. The SCC's inner-encoder processing begins after the outer-code has encoded the input data sequence; the decoding process of the outer-code begins after decoding the received data sequence by the inner code. This approach has been used in space communications, with convolutional codes as the inner code and low redundancy Reed-Solomon codes as the outer code.

Fig. 2. A PCC encoder with two encoders and an interleaver

PCC has similar constituent blocks as SCC, but unlike SCC, its inner and the outer codes operate simultaneously on the data. It reduces latency at the encoding and the decoding stages. Similar reductions in latency are achieved at the decoding. An interleaver is incorporated between the codes to de-correlate the received symbols affected by error bursts.

A turbo encoder achieves low complexity due to concatenation of codes and low latency by parallel processing. It consists of two encoders with a random interleaver between them (RI) [11]. To elaborate further, the turbo encoder accepts 'x' as its inputs and using Recursive Systematic Codes (RSC1) generates outputs, $y^{(1)}$ and $y^{(2)}$. In addition to $y^{(1)}$ and $y^{(2)}$, the turbo encoder generates I_x , an interleaved input sequence which is then sent to RSC2 in order to generate the parity sequence $y^{(3)}$. Finally, the information sequence $y^{(1)}$ and the parity check sequences of the two encoders, $y^{(2)}$ and $y^{(3)}$, are multiplexed to generate a 1/3 rate turbo code sequence. Fig. 3 below shows a block diagram of a typical turbo encoder.

Fig. 3. Turbo encoder functional diagram

x'	interleaved input
$y^{(1)}$	systematic bit
$y^{(2)}$	parity 1 sequence bit
$y^{(3)}$	parity 2 sequence bit
$g_1(D)$	feedback polynomial
$g_2(D)$	output polynomial

Table 3. Summary of symbols used in Turbo Encoder

RSC encoders, interleaver, puncturing and termination form an integral part of turbo encoding. These are described in later subsections.

2.2.1 Recursive Systematic Convolutional Encoder

RSC encoder can be thought of as a convolutional encoder that retains the original data sequence (systematic bits stream), and it has been modified (to send feedback) to improve

encoding strength. As shown in the figure above, the RSC encoder accepts the input data bit sequence of length N , denoted by 'x' and produces two outputs, $y^{(1)}$ and $y^{(2)}$. The first output, a.k.a. the systematic output, denoted by $y^{(1)}$ is identical to the input sequence; whereas the second output is the parity check sequence, denoted by $y^{(2)}$. Both $y^{(1)}$ and $y^{(2)}$ are transmitted as turbo encoder outputs. A traditional RSC encoder features a

- systematic bit stream
- convolutional encoder and
- feedback (recursive) polynomial.

A 1/2-rate RSC code is a convolutional code represented by two polynomials, $g_1(D)$ and $g_2(D)$. $g_1(D)$ is the feedback polynomial whereas $g_2(D)$ is the feed-forward polynomial utilized in convolutional encoding.

Fig. 4. An example RSC encoder with feedback, $g_1(D)$ being $[1+D+D^2]$ and feedforward, $g_2(D)$ being $[1+D^2]$

Convolutional encoder

Convolutional encoder is an integral part of the RSC encoder which utilizes a linear feed-forward shift register to generate a 1/2 rate convolutional code. The input data stream is a binary sequence, given by

$x = (\dots, x_{l-1}, x_l, x_{l+1}, \dots, x_{l+1}, \dots)$, where l is a time instant.

The output is two sequences $y^{(1)}$ and $y^{(2)}$

$$y^{(1)} = (\dots y_{l-1}^{(1)}, y_l^{(1)}, y_{l+1}^{(1)}, \dots y_{l+1}^{(1)}, \dots)$$

$$y^{(2)} = (\dots y_{l-1}^{(2)}, y_l^{(2)}, y_{l+1}^{(2)}, \dots, y_{l+1}^{(2)}, \dots), \quad \text{where } y_l^{(j)}, j=1,2 \text{ are binary symbols.}$$

At time l the input to the encoder is x_l and the output is a code block y_l .

As mentioned in 2.2.1, y_l consists of, $y_l^{(1)}$, systematic output sequence and $y_l^{(2)}$, encoded output sequence. These elements are calculated as follows:

$$y_l^{(1)} = x_l, \text{ whereas}$$

$$y_f = x_l + x_{l-1} + x_{l-2}$$

$$y_l^{(2)} = y_f + x_{l-2}$$

Note that '+' denotes modulo-2 (XOR) addition and the encoder must store the past two information bits.

Systematic Bit Stream [12]

Having systematic bits as part of the output bit stream gives an opportunity to send the original data along with the parity sequences. In a systematic convolutional code the first output sequences are an exact replica of the input information sequences.

Recursive bit stream (Feedback) [13]

The feedback polynomial is essential in improving the encoding strength of the RSC encoder. $g_1(D)$ of the transfer function is the feedback polynomial. The feedback in the encoder enhances the encoding strength, bringing the BER performance to par with nonsystematic encoders.

2.2.2 Interleaving Strategies

The interleaver can affect the bit error rate (BER) performance of a particular turbo-code by an order of magnitude or more [6]. Interleaving is a standard signal processing technique used in a variety of communications systems. The process manipulates the temporal order of the received sequence without impacting the original symbols values [12]. The classical use of interleaving is to rearrange the locations of errors introduced in transmission, allowing for the use of random error correcting codes at the receiver. Such a situation occurs due to error bursts in a channel (e.g., wireless communications channels) and concatenated coding.

The two common interleavers are block, and convolutional [14], [15]. In a classical block interleaver, the input data is written in rows of a set of memory elements set up as a matrix, and then read as columns. There is a tradeoff for block-interleavers because they have fixed interleaving depths. With a fixed interleaving depth, the interleaver adequately spreads out the adjacent bits if each burst of errors does not exceed the interleaver depth, but it would be over-designed if the error-bursts are on average much shallower than the interleaver depth [24]. Typically, a matrix 8 x 8, block interleaver has

a fixed interleaving depth of 8. If the error-burst occurs at the 9th element, then two errors will be adjacent. Similarly, if the error occurs at the 2nd elements then these two errors will only be segregated by 8 bits. If it was known that only these errors were present, the distance between them would be much greater. Looking at another matrix interleavers configuration of 2x32 would space these two errors 32 elements apart. Interleavers with such low interleaving depth are not useful for long burst of errors. In practice, the number of error bit sequences are unknown and can vary in both length and occurrences.

A pseudo random interleaver is a type of convolutional interleaver. It scrambles the input data by a permutation of N elements with no repetitions. The PRI produces a continuous block code and changes the order of the input sequences. If the input sequences to the two component decoders are de-correlated, there is a high probability that some of the errors in the first decoder should become correctable in the second decoder [12]. The interleaver parameters are [12]:

- *“Delay of an interleaver, which is the total delay of the data after interleaving and de-interleaving.”*
- *Memory of an interleaver, which is the minimum number of memory elements required to implement a causal version of a given interleaver.”*

Dispersion of an interleaver denotes the ‘randomness’ of the interleaver. Due to the random nature of error bursts and recent research demonstrating better performance of Pseudo Random interleaver (PRI) compared to Block interleaver; PRI was researched and simulated.

2.2.3 Punctured Codes

Punctured codes allow code rates to be increased. When experiencing a wide range of SNR it may be required to use a different code rate without changing its encoding and trellis structure. An example of such a scheme is unequal error protection codes (UEP) where an encoder and decoder with variable code rates are used. Puncturing typically results in lower performance and are suited for situations in which SNR is high. Puncturing is achieved by not transmitting certain code symbols of the parity sequences. The following sub-sections describe 1/2, 1/3, and 2/3 code rate designs.

1/2 Rate Code

A 1/2 rate code refers to an encoder design where the output symbol contains twice as many bits as the original sequence. The figure below depicts a 1/2 code rate.

Fig. 5. 1/2 Code Rate with one systematic sequence and one parity sequence.

An encoder with $g_1(D) = [1+D+D^2]$ and $g_2(D) = [1+D^2]$ has four possible states. The following trellis diagram depicts this in the vertical axis. Each state is uniquely identified using two bits. Each node receives two inputs and transmits two outputs to the next node.

Fig. 6. Trellis diagram for the (2,1) nonsystematic encoder [1].

1/3 Rate Code

A 1/3 rate code produces a 3 bit output symbol for every input bit provided to the decoder. In case of turbo encoding, the symbol consists of a systematic bit and two parity bits. An identical trellis is used to create parity1 and parity2 bit sequences. However, input to create parity2 bit sequence is the interleaved version of the input used to create parity1 bit sequence. See Figure 7 below.

Fig. 7. 1/3 Code Rate with no puncturing uses identical trellis structures in RSC1 and RSC2 for encoding parity1 and parity2.

The figure below is the trellis used for both RSC1 and RSC2.

Fig. 8. Trellis diagram for both RSC1 and RSC2

2/3 Rate Code

A 2/3 rate code is established when not every alternating parity bit is included in the output symbol. The systematic bits become part of all the output symbols whereas all even bits for parity1 and odd bits for parity2 are discarded. When SNR ratio is high, and the system performance can be lowered without reaching the minimum performance requirements, 2/3 rate code are particularly useful in increasing the bandwidth.

Fig. 9. Puncturing with 2/3 Code

In terms of trellis diagram, if one of the four encoder output symbols is removed (see Fig. 10), the encoder produces three coded symbols for every two information bits. That is, every leading symbol of the rate 1/2 code is punctured from every other branch. The new code is time varying and has a code rate of 2/3. The trellis diagram of the punctured (3,2) code is shown in Fig. 10. Again the vertical axis shows the possible states in which X indicates the deleted symbols in the alternate transitions.

Fig. 10. Trellis diagram of a 2/3-rate punctured code produced by periodically deleting symbols (marked with 'X') from a rate 1/2 code [16].

2.2.4 Termination Methods

Termination scheme is added to the component encoders of a Turbo Encoder. Trellis termination drives the encoder to a known state, which enhances the encoding strength and reduces complexity of decoding stage [17]. In an encoder without a feedback, sending zero into the shift register would rapidly clear it. However, the codes employed in this study are recursive and it is not possible to terminate the trellis by padding zero tail bits. The tail bits are dependent on the state of the component encoder due to the last bit in the received data sequence. A control switch in both of the RSC encoders can toggle between the data sequence and the flush bits to incorporate the tail bits. The flush bits are inversely encoded such that they produce zero at the input of the shift register. The zero at the input of the shift register drives the states to zero after a finite number of

clock cycles. With a pseudo-random interleaver, it is highly unlikely that this method will terminate both encoders simultaneously. Generally, only the first component encoder is designed to reach a known state. The figure below shows a trellis and its paths to termination.

Fig. 11. A sample termination scheme

2.3 Turbo Decoder

A turbo decoder consists of constituent decoders, interleavers, and de-interleavers. It also has a strategy to strike a balance between error correction and complexity.

A conventional turbo decoder consists of two constituent decoders. Each constituent decoder accepts the received signal, and a-priori values, $L(u_{a_i})$, and generates soft decision, $L(c_i)$, and extrinsic values, $L(u_{e_i})$, (see Fig. 12) for further computations. Depending on the SNR of the channel, the constituent decoders make multiple decoding passes over the received samples. The extrinsics and the soft-decisions generated by both

constituent decoders in each pass eventually form the bases for the hard-decisions outputs.

Fig. 12. Block diagram of a constituent decoder

Interleaving strategy is an integral part of turbo codes;, the turbo decoder therefore provides the interleaving and de-interleaving features. This allows the two constituent decoders to operate on each other's data.

The following subsections describe the turbo decoder components in further detail and Fig. 13 presents a functional diagram.

c_i	symbol of the i^{th} transition.
$L^1(c)$	received sequence for constituent decoder 1 (CD1)
$L_n^1(c)$	sequence for constituent decoder 1 (CD1) from n^{th} iteration
$L(ua)$	apriori sequence
$L(ue)$	extrinsic sequence
$L^1(ua)$	apriori sequence for CD1
$L^1(ue)$	extrinsic sequence for CD1
$L^1(u)$	initial interleaved extrinsic sequence for CD1
$L^2(c)$	received sequence for constituent decoder 2 (CD2)
$L_n^2(c)$	sequence for constituent decoder 2 (CD2) from n^{th} iteration
$L^2(ua)$	apriori sequence for CD2
$L^2(ue)$	extrinsic sequence for CD2
$L^2(u)$	initial interleaved extrinsic sequence for CD2

Table 4. Summary of symbols used in decoder iterations

First Iteration

In the first decoding pass, both constituent decoders process their associated received sequences simultaneously (see Fig. 13). $L^1(c)$ is the received sequence for constituent decoder 1 (CD1) whereas $L^2(c)$ is the received sequence for constituent decoder 2 (CD2). The apriori bit sequences, $L^1(ua)$ for CD1 and $L^1(ua)$ for CD2 are both set to a neutral value, 'zero', which does not influence the outputs. Using the apriori sequence, each of the decoder processes the received bits, producing extrinsic sequences of $L^1(ue)$, and $L^2(ue)$.

Subsequent Iterations

Prior to commencing subsequent iterations, $L^1(ue)$ is interleaved and is represented by $L^2(ua)$. Due to interleaving at the encoding stage, the soft outputs must be interleaved and de-interleaved before CD1 and CD2 can commence operations. Care has to be taken to provide appropriately sequenced CD1 outputs to CD2 as inputs and vice versa [1]. Similarly, $L^2(ue)$ is de-interleaved and becomes $L^1(ua)$.

CD1 then interprets the channel outputs, but now employs $L^1(ua)$, to attain a more precise set of $L^1(ue)$. Similarly, the CD2 utilizes $L^2(ua)$ to generate $L^2(ue)$. With each iteration, the BER of the decoded sequence provides an equivalent or improved estimate of the original sequence. However, it results in diminishing returns because as the iteration increases, the rate of improvement in BER decreases [1]. Hence, due to latency, typically five and up to eight iterations are used.

Turbo Decoding Basics

This section describes extrinsic and intrinsic information and how it is used in iterative decoding of turbo codes. The constituent decoders implement a decoding algorithm that supports soft-outputs. In this study, the MAP algorithm has been implemented in the decoder. MAP algorithm introduced by Bahl et al. in 1974 calculates “*a-posteriori probabilities of the states and the transition of a Markov source observed in memoryless noise*” [1]. Bahl demonstrated that MAP algorithm can be successfully used to decode convolutional codes. MAP decoding process consists of two gamma calculations: forward recursion and backward recursion. In Fig. 14, gamma calculations and the

recursions are carried out for each trellis transitions before soft decisions and extrinsic values can be calculated. The entire packet is received and then it creates a trellis, shown earlier in Fig. 6, with 2^m state-nodes for each of the bits in the packet. During forward recursion, the branch values, called gamma and alpha are calculated and stored; during backward recursion, gamma and beta are calculated and stored. Once the trellis paths have been traversed, $L(c_i)$ and $L(ue_i)$ are determined [1].

Fig. 14. The figure above shows a block diagram of a Log MAP decoder

When used to decode convolutional codes, MAP algorithm is optimal when it comes to recovering the original bit sequence [1]. The original MAP algorithm traversed all the trellis paths and hence was deemed too computationally intensive for most applications.

A block diagram for a log - MAP decoder is shown below:

Fig. 15. Block diagram of a MAP decoder

The following sections describe the workings of a MAP Decoder, starting with a table that captures the variables used.

$\gamma(\hat{s}, s)$	branch transition probabilities
α	result of forward recursion. Alpha contain ' $n * 2^m$ ' elements in MAP algorithm whereas in Log-MAP, it contain ' n ' elements.
β	result of backward recursion. Beta contain ' $n * 2^m$ ' elements in MAP algorithm whereas in Log-MAP, it contain ' n ' elements.
$\hat{C}(s, s) \Rightarrow c_i = +1$	when the input bit c_i is +1, these are the set of transitions that can occur from the previous state, ' s ', to the present state, s
$\hat{C}(s, s) \Rightarrow c_i = -1$	when the input bit c_i is -1, these are the set of transitions that can occur from the previous state, ' s ', to the present state, s
σ^2	noise variance
$\gamma(\hat{s}, s)$	branch transition probabilities
$\beta_i(s)$	i^{th} result of backward recursion.
$\alpha_{i-1}(\hat{s})$	$(i-1)^{\text{th}}$ result of forward recursion.
a	fading amplitude
c_i	input bit
$\mathbf{x}$	interleaved input
k	number of input bits per symbol period
K	natural constraint length of the transmitter
$L(c_i \mathbf{y})$	log ratio of receiving the bit ' c_i ' given $\mathbf{y}$, the received symbol.
$L(c_i)$	i^{th} bit of the received sequence
$P(a \wedge b)$	Bayes's rule stating $P(a b) * P(b)$
$\mathbf{y}_{j>k}$	received sequence up to current transition
$\mathbf{y}$	the received symbol.

Table 5. Summary of symbols used in log- MAP decoding

In a MAP algorithm, once the sequence has been received, it calculates the probability, $L(c_i|y)$ of each symbol being a +1 or -1. The $L(c_i|y)$ is given by [1]

$$L(c_i | \underline{y}) = \ln \left(\frac{P(c_i = +1 | \underline{y})}{P(c_i = -1 | \underline{y})} \right) \quad (1)$$

which can be described as

$$= \ln \left(\frac{\sum_{(s,s) \Rightarrow c_i = +1} \alpha_{i-1}(s) \gamma_i(s, s) \beta_i(s)}{\sum_{(s,s) \Rightarrow c_i = -1} \alpha_{i-1}(s) \gamma_i(s, s) \beta_i(s)} \right) \quad (2)$$

where (See Fig. 53)

$$\begin{aligned} \alpha_{i-1}(s) &= P(S_{i-1} = s \wedge y_{j < i}) \\ &= \sum_{all\ s} \alpha_{i-1}(s) \gamma(s, s) \end{aligned} \quad (3)$$

at time $i-1$, the trellis' probability of being in state s is represented by $\alpha_{i-1}(s)$.

Furthermore, the value of received channel sequence to this point met its expectation of being $y_{j < i}$ [1].

$$\begin{aligned} \beta_i(s) &= P(y_{j > i} | S_i = s) \\ &= \sum_{all\ s} \beta_{i-1}(s) \gamma(s, s) \end{aligned} \quad (4)$$

At time i , the trellis's probability of being in state s is represented by $\beta_i(s)$. Furthermore, the value of future received channel sequence from this point met its expectation of being $y_{j > i}$ [1].

$$\begin{aligned} \gamma_i(s, s) &= P(\{y_i | S_i = s\} | S_{i-1} = s) \\ &= P(y_i | \underline{x}_i) P(c_i) \end{aligned}$$

At time $i-1$, the trellis's probability that was in state s is represented by $\gamma_i(s,s)$.

Furthermore, the value of received channel sequence at this point is y_i [1].

$$\begin{aligned} P(\underline{y}_i | \underline{x}_i) &= \prod_{l=1}^n P(y_{il} | x_{il}) \\ &= \prod_{l=1}^n \frac{1}{\sqrt{2\pi\sigma}} e^{\left(-\frac{E_b R}{2\sigma^2} (y_{il} - ax_{il})^2\right)} \end{aligned} \quad (5)$$

where, [1]

The MAP algorithm determines values for alpha and beta for all the states and transitions of 'i' from 0 to N-1. Gamma, $\gamma_i(s,s)$ for all likely transitions from state $S_{i-1} = s$ to $S_i = s$, again for $i=0,1\dots N-1$ are also calculated. Once alpha, beta and gamma calculations are complete, MAP decoder computes conditional LLRs $L(c_i|y)$.

As mentioned earlier, log - MAP decoder produces the following output [1]:

$$L(c_i | y) = \ln \left(\frac{\sum_{(s,s) \Rightarrow u_i = +1} \alpha_{i-1}(s) \gamma_i(s,s) \beta_i(s)}{\sum_{(s,s) \Rightarrow u_i = -1} \alpha_{i-1}(s) \gamma_i(s,s) \beta_i(s)} \right) \quad (6)$$

and it can be written as

$$= L(c_i) + L_r y_{is} + L_e(c_i) \quad (7)$$

where,

$$L_e(c_i) = \ln \left(\frac{\sum_{(s,s) \Rightarrow u_i = +1} \alpha_{i-1}(s) X_i(s,s) \beta_i(s)}{\sum_{(s,s) \Rightarrow u_i = -1} \alpha_{i-1}(s) X_i(s,s) \beta_i(s)} \right) \quad (8)$$

$L(u_{ai})$ is the *a-priori* LLR

$$L_r = \frac{4a}{2\sigma^2}, \text{ represents the channel reliability value} \quad (9)$$

y_{is} is the transmitted systematic bit at the receiver

$x_{is} = c_i$, and

$$c_i(\hat{s}, s) = e^{\left(\frac{L_c}{2} \sum_{l=2}^n y_{il} x_{yl} \right)} \quad (10)$$

As seen in (6), LLR, $L(c_i|y)$, comprises of three terms: $L(c_i)$, $L_r y_{is}$, and $L_e(c_i)$. “*The a-priori LLR term $L(c_i)$ comes from $P(c_i)$ in the expression for the branch transition probability $\gamma(\hat{s}, s)$* ” [1]. As mentioned earlier, when there is no independent a-priori knowledge, LLR $L(c_i)$ is zero, hence having a-priori probability $P(c_i) = 0.5$ [1]. Each of the constituent decoder produces LLR $L(c_i)$ which then is re-sequenced and provided to the other decoder.

“*The second term $[L_r y_{is}]$ in (6) is the soft output of the channel for the systematic bit c_i , which was transmitted across the channel and received as y_{is} .*” [1]. The channel reliability figure $[L_r]$ is relative to SNR and impacts a-posteriori LLR $[L(c_i | y)]$ accordingly. When the SNR is high, L_r has a significant influence on LLR whereas at low SNR it has relatively less influence [1].

Finally, the extrinsic LLR for the bit c_i , $L_e(c_i)$, is calculated using the a-priori information sequence and the received channel information sequence but does not use y_{is} or $L(c_i)$ of c_i .

2.3.4 MAP Algorithm Modifications

Due to calculations required for each bit in the sequence, MAP algorithm is far more complex than Viterbi algorithm and hence is not suitable for most applications [1]. With the recent advent of turbo codes, high bandwidth requirements, and simplification in the MAP algorithm, it has taken center stage in ECC.

Baier and Erfanian et al. presented the Max-Log-MAP algorithm, which simplifies the algorithm by carrying out the calculations and keeping results in log domain [18], [19].

Using log-domain and an approximation, $\ln\left(\sum_i e^{x_i}\right) \approx \max_i(x_i)$, selects the maximum value during each calculation of alpha, beta and gamma, drastically reduces the required number of calculations. $\alpha_{k-1}(s')$ is calculated during forward recursion, $\beta_k(s)$ is calculated during backward recursion, along with branch transition probabilities $\gamma(s',s)$. Once the approximate component parameters have been calculated, LLR can be determined. Considering the maximum values are being used, the complexity is only three times higher than Viterbi Algorithm but it is sub-optimal [1]. In fact its performance is 0.35dB lower than MAP algorithm [1].

Robertson et al. studied the max-log MAP algorithm and developed an improvement which enhances its error correction strength to match the original MAP algorithm's performance [18]. A technique based on Jacobian algorithm, $\ln(e^{x_1} + e^{x_2})$, is employed to determine a correction factor. Approximately, eight correction factors, $f_c(\delta)$, are

calculated and stored for repeated use [1]. Hence, with little added complexity, log MAP performance matches that of MAP algorithm.

3 ARCHITECHTURE & DEVELOPMENT

To evaluate turbo codes design and performance, a simulation environment of a digital communication channel, see Fig. 16, consisting of Pseudo Random Bit Generator, 2-PAM Modulation and De-modulation, Channel Noise, turbo encoder, turbo decoder and Comparator – was first created in Matlab. The performance of the system in terms of BER was compared with published research and summarized in Appendix D. Matlab's built-in functions, existing simulation code [20], and the blocks developed are simulated. During the development of the turbo communication channels, a legitimate technique of code substitution was applied to verify correct functionality of the architecture and development [20]. The architecture and the design parameters were initially based on the 3G Wireless Standard [27].

Fig. 16. Digital Communication Channel

The turbo code hardware development process (Fig. 17) includes architecture development, hardware development, synthesis, timing analysis and optimization. The architecture was confirmed using Matlab and then the functionality and performance benchmarked against the published research [23]. Turbo encoder hardware behavior models were created using Mentor Graphics (Modelsim) and Cadence (X-L Verilog). The synthesis was carried out by Synopsis (Design Analyzer) using 'slow-slow' (worst case) timing models in the 0.18 μm CMOS VST libraries. Furthermore, pre-layout gate level simulations were carried out to reaffirm the functionality. The timing of the critical path was determined and then modifications were made to optimize it. Lastly, estimated prediction for power and area estimated from the optimizer were compared with the power and area of the synthesized logic.

Fig. 17. Design process and tools used during the course of the study

For verification of error correction, a sample sequence was created using the 'rand' function in Matlab. Outputs for the turbo encoder were hand-calculated. During the design process, the simulation environment was verified against the theoretical calculations; the behavior models were verified against test vectors generated by the simulation environment. The output sequences of both systems were compared to verify operation.

The following sections present the performance of a digital communication channel using turbo codes, and development, synthesis and analysis of the turbo encoder and its sub-blocks.

3.1 Data Generation Modulation Scheme

An input sequence with values between 0 and 1 is generated using a 'rand' function in Matlab. The mean, μ is 0.5 and the use of the 2-PAM modulation scheme makes standard deviation irrelevant. The numbers from 0 to 0.5 were translated to 1, whereas numbers between 0.5 and 1 are translated to -1. The figures 1 and -1 represent the bits of the signal space used to map information.

3.1.1 Data Generation

A pseudo random sequence was generated to represent traffic through the system. A true random value is generated by a physical process like a fair dice throw, whereas pseudo-random numbers are generated by software functions and are deterministic. For a given initial value, identical sequences of numbers are generated.

When designing and implementing complex algorithms, it is essential to verify results and having pseudo-random input data helps identify errors and debug it. A point of comparison would not be possible with 'random numbers'. For the system level

simulations, a particular 'seed' was not specified in order to generate random number sequences.

3.1.2 Binary Modulation

In 2-PAM signaling, the two signal waveforms are $s_1(t) = g(t)$ and $s_2(t) = -g(t)$, where $g(t)$ is a pulse that is nonzero in the interval $0 \leq t \leq T_b$ and zero otherwise. Let the energy in the pulse $g(t)$ be given by E_b .

The 2-PAM modulation is one-dimensional and hence its geometric representation is a one dimensional vector $s_1 = \sqrt{E_b}$ and $s_2 = -\sqrt{E_b}$. Fig. 18 illustrates the two signal points:

Fig. 18. Signal points for binary antipodal signals

Assuming that the two signals are equally likely and that signal $s_1(t)$ was transmitted, then the received signal from the matched filter or correlation demodulator is

$$r = s_1 + n = \sqrt{E_b} + n$$

Here n represents the Additive Gaussian noise component, which has zero mean and a variance of $\sigma_n^2 = \frac{1}{2}N_0$. In this case, the decision rule compares r with the threshold

zero. If $r > 0$, the decision is made in favor of $s_1(t)$, and if $r < 0$, the decision is made that $s_2(t)$ was transmitted. Clearly, the two conditional probability density functions of r are:

$$p(r | s_1) = \frac{1}{\sqrt{\pi N_0}} e^{-(r - \sqrt{E_b})^2 / N_0}, \text{ and}$$

$$p(r | s_2) = \frac{1}{\sqrt{\pi N_0}} e^{-(r + \sqrt{E_b})^2 / N_0}$$

3.2 Channel Noise and Comparator

3.2.1 Channel Noise

Additive White Gaussian Noise (AWGN) with statistically independent samples (corrupting data samples) has been modeled and implemented as Channel Noise. AWGN is implemented by linear addition of wideband or white noise with a constant spectral density and Gaussian distribution to the transmitted data. The model does not account for the phenomena of fading, frequency selectivity, interference, nonlinearity or dispersion. However, it produces a simple model that helps determine the behavior of a system. The relative power of noise in an AWGN channel is typically described by quantities such as SNR and E_b/N_0 .

3.2.2 Comparator

Random data is generated by the random data generator grouped, encoded and transmitted as fixed length packets called frames. The decoded frame at the receiver is

compared to the original frame for any differences. If difference(s) between the original and the decoded frames are detected it is noted as a frame error. However, the system performance is reported as a bit error rate (BER), once a predetermined number of frame errors has been observed. Output bits that do not agree with the inputs are reported as errors.

3.3 Turbo encoder

As described in the earlier sections, turbo codes are based on parallel-concatenated convolutional codes (PCC codes). The turbo encoder architecture employed for the study is shown in Fig. 19. The design consists of 1024-bit interleaver, 2 3-state RSC encoders and has an option of establishing 1/2 or 1/3 code rate output sequence.

Fig. 19. Encoder functional diagram with 1/3 Rate coding

Matlab Simulations

The turbo encoding scheme models were created using Matlab. Table 6 summarizes the simulation parameters, whereas Fig. 20 shows the block diagrams involved. Individual blocks were created and verified against the expected results. Once all the functions had been developed, they were fully integrated and verified again for correct operability. The system level simulations created random data, turbo-encoded it, and then compared it against the expected resultant. The simulation environment allows various design parameters to be modified methodically for observation of their impact. The parameters are as follows:

SNR	$[0 - 0.8]^a$
Modulation	2 - PAM
Component encoders	RSC encoders
encoding Rate	1/3 rate with no puncturing
RSC transfer function	$[F_H, B_H]^b$
decoder algorithm	Log – MAP
Interleaver	10 bit Pseudo Random Interleaver
Frame size L	1024 bits, $[\text{range } 64 - 4096]^c$
decoder iterations	5, $[\text{range } 1 - 8]^d$
Termination method	Single trellis termination ^e
Stopping Criteria	100 Frame Errors

Table 6. System parameters of the turbo encoder

- a. Signal to Noise ratio is typically swept between 0 & 0.8 dB for system performance graphs.
- b. The RSC transfer function was selected to be $[1\ 1\ 1\ 1; 1\ 0\ 1\ 1]$.
- c. The frame size was varied from 64 to 4096 bits to evaluate impact on the system performance, due to varying interleaving depth.
- d. The decoder iterations were varied from 1 to 8.

- e. Data stream to the first encoder was terminated, whereas the second data stream was unterminated (see section 3.3.1).

Simulation and implementation for each of the blocks are described in the subsequent sections.

Fig. 20. Matlab simulation blocks for the transmit stage

Hardware Development

The turbo encoder developed in this study consists of the input stream and interleaved input stream being encoded RSC1 and RSC2. Typically RSC1 encoder encodes as it receives data; however, in order to establish 1/3 Code Rate, the original data and RSC1 encoded data are delayed. The interleaver provides RSC2 with the data sequence after an entire packet has been received. To avoid storing original data and the encoded data, in this architecture, the original data is stored and then provided to RSC1 and RSC2 encoders simultaneously. Instead of storing both the original data and the RSC1 sequence, storing only the original data reduces the memory requirements in half. Even after the reduced requirements, (see Fig. 63) memory represents 80% of the total design, and sharing storage reduces the encoder area requirements by 45%.

The design specifications and elements of system level design facilitated the hardware development. Fig. 20 and Fig. 21 show that the turbo encoder hardware design follows the architecture and simulation environment. In addition to the design blocks shown in Fig. 20, on chip memory and memory controller were designed and incorporated on the chip, to satisfy the turbo-encoder storage requirements.

Fig. 21. Hardware Block Diagram

The following are the top-level views of the turbo encoder, the synthesized block diagram and the waveform resulting from the behavior modeling.

Fig. 22. Turbo encoder modules

Fig. 23. Turbo encoder block diagram

Fig. 24. Top encoder internal hardware

Fig. 25. Behavior waveforms for the turbo encoder shows the output streams match the test vectors.

Fig. 26. Behavior waveforms – top level turbo encoder for the last section of the frame

Analysis and Optimization

Fig. 25 shows behavior waveforms for clock, encoded data, data bits from memory being retrieved by the interleaver and the interleaved encoded data. With the exception of the clock signal, theoretical bit stream has also been displayed in the waveform diagram. Notice that a comparator waveform automatically compares the theoretical and the observed theoretical waveforms to detect differences.

The following table summarizes the timing, power and area figures for turbo encoder, memory and the logic.

	Turbo encoder blocks	Memory	Overall Results
Timing at 333 MHz (or 3.00 ns) slack (ns)	0.60	0.49	0.49
Area Total cell area (μm^2)	11688	82315	94003
Total Dynamic Power (mW)	7	39	46

Table 7. Summary of turbo encoder design

3.3.1 Recursive Systematic Convolutional Encoder with Termination

The RSC encoder transfer function significantly impacts the coding performance. Simulations were carried on selected transfer functions to establish a function that yields minimum BER. The following table summarizes the simulation conditions.

SNR	0.8
Modulation	2 - PAM
Component encoders	RSC encoders
encoding Rate	1/3 rate with no puncturing
RSC transfer function	range of functions. See graph below.
decoder algorithm	Log – MAP
Interleaver	a random interleaver which was initially generated using Matlab 'rand' function.
Frame size L	1024 bits
decoder iterations	5
Termination method	Single trellis termination
Stopping Criteria	100 Frame Errors

Table 8. Simulation condition for transfer function analysis

The bar graph in Fig. 27 shows $g = [1111; 1011]$ having the lowest BER among the other transfer functions simulated. Considering its performance for the simulation parameters mentioned above, $g = [1111; 1011]$ was selected for implementation.

Fig. 27. Transfer function vs. BER

1/2-rate RSC encoders used in the design have a memory of 3. The generator polynomials, feedback and feedword are $g_1(D) = 1111$ and $g_2(D) = 1011$.

Bits sequence generated by RSC1 has been terminated, whereas RSC2 generates an open-ended sequence. Shift registers are used to create the memory of the encoders. The original states of the 8-state RSC encoders are '0' at the start of encoding. The transfer function consists of $g_1(D)$ and $g_2(D)$, which are a feedback and feed forward polynomials respectively. Fig. 28 illustrates the shift registers and XORs at the appropriate taps.

Fig. 28. Structure of recursive systematic encoder

Termination

To improve the BER performance, trellis termination pads tail bits to the end of the input data stream. These are needed to bring the RSC encoder to a null state. The tail bits trigger a logic path based on outputs of the RSC encoder state graph that forces the shift registers to a Null-state. Additional circuitry is required in RSC encoder1 as a selectable option to introduce appropriate termination bits at the end of each packet

Fig. 29. Structure of recursive systematic encoder with termination bits

The following section contains

- trellis for the encoder
- transfer function table
- termination table
- last State/last Output table

Trellis Diagram

The following trellis diagram represents the next state conditions for $G(D) = [1111; 1011]$

Fig. 30. Trellis diagram represents the next state conditions for all possible inputs when starting initially from $s = 0$.

Fig. 31. Trellis diagram showing the outputs associated with each state transition.

Transfer Function Table:

Transfer Function tables have been generated using formulae in Microsoft Excel. The generator matrix has been strategically placed at a 'relative' location to the input, state, state+ (next-state) and outputs. Please see Appendix C for three more transfer functions and details on methodology for developing the tables.

A	B	C	D	E	F	G	H	I	J	K	L	M
233	g1	1		1	1	1						
234	g2		1	0	1	1						
235	feedback											
236	xor											
	(x,s1,s2,s3)										x	(s0,s2,s3)
237	x	s0	s1	s2	s3	state	s1+	s2+	s3+	state+	output1	output2
238	0	0	0	0	0	0	0	0	0	0	0	0
239	0	1	0	0	1	1	1	0	0	4	0	0
240	0	1	0	1	0	2	1	0	1	5	0	0
241	0	0	0	1	1	3	0	0	1	1	0	0
242	0	1	1	0	0	4	1	1	0	6	0	1
243	0	0	1	0	1	5	0	1	0	2	0	1
245	0	0	1	1	0	6	0	1	1	3	0	1
244	0	1	1	1	1	7	1	1	1	7	0	1
245	1	1	0	0	0	0	1	0	0	4	1	1
246	1	0	0	0	1	1	0	0	0	0	1	1
247	1	0	0	1	0	2	0	0	1	1	1	1
248	1	1	0	1	1	3	1	0	1	5	1	1
249	1	0	1	0	0	4	0	1	0	2	1	0
250	1	1	1	0	1	5	1	1	0	6	1	0
251	1	1	1	1	0	6	1	1	1	7	1	0
252	1	0	1	1	1	7	0	1	1	3	1	0

Table 9. Next state calculations for the trellis of Fig. 31 using transfer function 1: $g = [1 \ 1 \ 1 \ 1; 1 \ 0 \ 1 \ 1]$

Termination Tables

The termination reflects the forcing of the RSC encoder states to '0'. Considering that the feedback and encoding polynomials are fixed, the input, 'x' is the only variable that can influence the Encoder states. The termination sequences have been established to

deliver a calculated 'input' value to 'x'. At request, 'input' is provided to the encoder such that 's0' takes on a '0' value. Since the constraint length is '4', a maximum of three such 'x' values will be required to force the 'state' values to '000'.

The values of states and next-states with their inputs and output have been established in the 'Transfer Function Tables'. Transitions that result in 's0' being null are selected by sorting the Transfer Function Table with respect to 's0' and then 'state'. The resultant top eight rows are the required fields for termination.

state	x	s0	state+	output1	output2
0	0	0	0	0	0
1	1	0	0	1	1
2	1	0	1	1	1
3	0	0	1	0	0
4	1	0	2	1	0
5	0	0	2	0	1
6	0	0	3	0	1
7	1	0	3	1	0

Table 10. Termination -1

Fig. 32. Termination trellis

'Last Output' and 'Last State' Tables

Once the Transfer Function Table has been established, it is sorted with respect to 'x' and within the same sorting function with respect to 'state+'. The resulting table outputs the desired previous state and output results.

x	state+	s0	state	output1	output2
0	0	0	0	0	0
0	1	0	3	0	0
0	2	0	5	0	1
0	3	0	6	0	1
0	4	1	1	0	0
0	5	1	2	0	0
0	6	1	4	0	1
0	7	1	7	0	1
1	0	0	1	1	1
1	1	0	2	1	1
1	2	0	4	1	0
1	3	0	7	1	0
1	4	1	0	1	1
1	5	1	3	1	1
1	6	1	5	1	0
1	7	1	6	1	0

Table 11. Last State and Last output Table -1

Fig. 33, Fig. 34, Fig. 36 and Fig. 37 illustrate the schematic views of the RSC encoders and the waveform resulting from the behavior of modeling. A set of control signals is generated at the instant a packet is stored in memory. The bits are stored in the memory and then read back to both encoders via separate ports. The storing and retrieval results in delays of the bit sequence. Hence, the control signals were delayed to synchronize the control signals and the data stream.

Fig. 33. Internal hardware of RSC used to collect termination bits

Fig. 34. Internal hardware of RSC1 used to generate parity1

Fig. 35. Internal hardware of RSC2 used to generate parity2

Fig. 36. Behavior waveforms of RSC1

Fig. 37. Behavior waveforms of RSC2 using interleaved data.

Analysis and Improvements

The following table summarizes the timing, power and area figures for the RSC encoders.

It can be seen that RSC1 has greater area and power figures than RSC2. This is due to the fact RSC1 contains a termination logic to force the values of the states to zero.

	RSC to collect termination bits	RSC1	RSC2
Timing at 333 MHz (or 3.00 ns) slack (MET) (ns)	0.40	0.53	0.53
Area Total cell area (μm^2)	1174	1561	1561
Total Dynamic Power (μW)	606	810	810
Cell Leakage Power (nW)	78	72	72

Table 12. Summary of RSC2 encoder Design

3.3.2 Pseudo Random Index Interleaver

Interleavers provide immunity to error bursts and are essential in digital communication systems. The selection of a particular interleaver impacts the over BER for the system by an order of magnitude or more [21]. As discussed in the earlier sections, PRI performs considerably better than block interleavers. Even for PRI, the interleaving depth strongly influences the encoding strength. Table 13 summarizes the simulation parameters, whereas Fig. 39 shows the impact on PRI performance as the encoding interleaving depth varies with the size of the data sequence. According to the architecture, turbo code blocks must be capable of handling 1024 bit packet. Therefore, a 1024 bit interleaver was implemented in the Turbo Encoding VLSI.

Fig. 38. Interleaving depth vs. BER

SNR	[0 – 0.8]	No. of decoder iterations	5
Modulation	2 - PAM	Termination method	Single trellis
Component encoders	two RSC encoders	Stopping criteria	100 Frame Errors
encoding Rate	1/3 rate	interleaver	pseudo random
RSC transfer function	[1111: 1011]	Frame size	[64 - 4096] bits
decoder algorithm	Log – MAP		

Table 13. Parameters for the interleaving depth setup

The interleaver provides temporal diversity by scrambling the original data sequence. It handles 1024 bits (including the 4 flush bits) by employing shift registers and dual port on-chip memory.

The 10-bit Linear Shift Back Register, seen in Fig. 39, has been designed with a tap configuration that yields all 1024 bit sequences. The designed tap configuration also disallows the states to transition into a non-recoverable 'stuck at 0' state. This has been accomplished in two steps. Firstly, a tap sequence is used that yields a 2^{n-1} pseudo random bit sequence. Secondly, the additional circuitry detects and forces the shift register to leave the stuck-in state.

The extra circuitry added in the PR-II is also beneficial in terms of address allocations. It allows all 2^n addresses. As seen in Fig. 39, C is produced by noring all the taps except B. C is then Xored with B and D to produce A. The outer-loop (B ORed C) when ORed, inverts the resultant calculated by D. If the bits in the minor loop back are '0's then the most significant tap, B must be '1'. When this is true, the MSB is reverted to logic 0. As a result, the feedback and the register states transition to 0s. The inversion in the next cycle sets the feedback back to '1' and the transition continues. Hence, the improved design of the shift register allows all 2^n addresses to be interleaved.

Fig. 39. 10 bit, all state pseudo random interleaver

The following is the waveform of the behavior model:

Fig. 40. Waveform of the behavior model

Fig. 41. Internal hardware of data interleaver

Analysis and Improvements

The following table summarizes the timing, power and area figures for the data interleaver.

Parameter	Data interleaver
Timing at 333 MHz (or 3.00 ns) slack (MET) (ns)	1.42
Area Total cell area (μm^2)	1171
Total Dynamic Power (μW)	407
Cell Leakage Power (nW)	48

Table 14. Summary of interleaver design

3.3.3 *Memory and Memory Controller*

On-chip dual-port memory stores the initial data packet. Once a packet has been completely stored, two different memory locations are simultaneously read and transmitted using two independent ports. RSC1 receives the original data stream, whereas RSC2 receives bits in accordance with the interleaved sequence retrieved by the pseudo random index interleaver (PR-II). To establish 1/3 rate, the original packet, RSC1 output and the RSC2 output are synchronized by introducing buffers. The architecture developed uses a dual-port memory design which does not require the storage of RSC1 encoded bits.

Write Access

The memory controller has been designed to operate two ports simultaneously. The SDRAM does not have the intelligence to avoid collision during two simultaneous write accesses by two ports; therefore, it has been handled at the turbo encoder hardware design (see Fig. 21). As the figure shows, initially, the turbo encoder stores only 1 set of data sequence in its received form.

Having two ports of the memory, provides the flexibility to read the original data sequence from one port and simultaneously read the interleaved sequence from the second port. This eliminates the need to store an interleaved data sequence or the RSC encoder output and also requires only one port being enabled to write the bits to the memory at any given time.

Fig. 42 shows that a single port is used to transfer the 8-bit register values to the memory. The incoming data to the memory controller is serial bits that are converted to 8-bit register values. At every 8th clock cycle, the registered values are written to the memory. Moreover, the address is generated incrementally to store the sequence in its original order. The write cycle starts when the write signal is asserted and stops at the end of the packet. There is a 'one packet pause' to allow the packet to be processed.

Fig. 42. Memory controller – write to memory functional diagram

Read Access for RSC1

As soon as the memory write access ends for a given packet, the original sequence and its interleaved sequence are read concurrently through ports 1 and 2. The input data sequence to be provided to RSC1 is read every eight clock cycles. When storing, the input sequence maintained its temporal order hence, the 8-bit register values read from port 1 can be serialized before being transmitted to RSC1. In view of the fact that the packet was stored in the order in which it arrived, the memory cells are read successively at every 8th clock cycle. Each memory cell consists of 8 bits, and at every clock cycle, each bit is transmitted to RSC1 encoder. Fig. 43 below describes the memory read access for RSC1 encoder:

Fig. 43. Memory controller – read from memory functional diagram for RSC1

Read Access for RSC2

Read Access for RSC2 locates individual bit location within a register of a memory cell. The memory cell locations and the bit locations are identified by the Pseudo Random Index Generator module. The memory control block receives address locations of 10 bits. At every clock cycle, the first seven bits are transmitted to the SDRAM to identify and retrieve the 8 bit register. The three least significant bits are then used to identify the precise bit within a cell. The remaining 7 bits of the retrieved register are discarded. It can be seen in Fig. 44 that the memory access and the bit extraction occurs at every clock cycle.

Fig. 44. Memory controller – read from memory functional diagram for RSC2

Dual Port SDRAM Memory Architecture

Using dual-port on-chip SDRAM for turbo encoding leads to two immediate benefits, which in turn means faster max operational frequency and savings in terms of power and

area. Firstly, on-chip memory eliminates the need to employ external RAM devices; and the need for interconnects, saving area on the printed circuit board, while improving the timing margins. Secondly, dual-port SDRAM requires the data sequence to be stored only once. Because memory consumes 87% of power and area, dual-port design reduces the memory requirements by half.

SDRAM is a superior type of DRAM. While DRAM has an asynchronous interface, meaning that it responds instantly to its control signal, SDRAM has a synchronous interface. It waits for a reference signal's pulse before responding to its control inputs. In SDRAM, a clocked, internal finite state machine pipelines the instructions. Pipelining allows memory to process the next command before it has completed the current one.

For instance, in a write command can be executed a burst because the memory is no longer requires additional clock cycles to write to the memory. Similarly, in a pipelined read, latency is introduced but burst reads can be executed.

The dual-port SDRAM block diagram, shown in Fig. 45, has two ports and two control units in SDRAM. It nearly doubles the access rate of the memory. However, within the SDRAM, there is no provision for collision avoidance; thus, the architecture of turbo encoder ensures only one port carries out memory write access at a given location.

Fig. 45. Dual-Port SDRAM Chip Architecture

The following is the waveform of the behavior model.

Fig. 46. Waveform of the behavior model accessing memories

Fig. 47. Internal hardware of the memory controller

Fig. 48. Internal hardware of memory controller's de-serializing sub-block.

Fig. 49. Internal hardware of memory controller's Counter.

Fig. 50. Internal hardware of memory controller's Counter1.

Analysis and Improvements

The following table summarizes the timing, power and area figures for the memory controller and the memory:

	Memory Controller	Memory
Timing at 333 MHz (or 3.00 ns) slack (MET) (ns)	0.27	0.42
Area Total cell area (μm^2)	11737	82316
Total Dynamic Power (mW)	6.85	39.32

Table 15. Summary of Memory Controller design

3.4 *Turbo Decoder*

At the receiver, the turbo decoder uses trellis-based SISO decoders to interpret the received data sequence (see Fig. 13). The received signal and the a-priori information are used to determine the branch metrics of the trellis. The results are then used to determine the reliability of the decoder outputs. Depending on the signal quality and the BER required, the turbo decoder makes multiple decoding passes over the received samples. The final decision on the received sequence is derived from the received signal and the soft decisions calculated by the decoder. Interleaver and de-interleaver strategies are also developed in the decoder design, since they make interoperability between the decoder possible. The top-level turbo decoder simulation block follows the architecture developed in Fig. 13. The following sections describe the implementation of the constituent blocks.

Fig. 51. Turbo decoder functional diagram

3.4.1 Log MAP Decoding

The Log MAP algorithm has been simulated as part of the turbo decoder. The architecture (seen in section 2.2) that combines gamma, beta, and then LLR, is shown in the block diagram and the trellis below.

Fig. 52. Log MAP decoder Block diagram where $\square$ is the symbol for memory.

Fig. 53. Log MAP decoder trellis showing the alpha, beta and gamma calculations required for each possible state of the bit in the sequence.

γ - Gamma

As shown in Fig. 14, calculation of the transition probability values $\gamma_i(\hat{s}, s)$ is the first step when the received sequence and a-priori information becomes available. Fig. 54

illustrates the block diagram of the gamma function, whereas Fig. 55 shows gamma calculation for each node of the trellis.

Fig. 54. Gamma calculation block diagram

Fig. 55. For each time step, gamma calculations are required for all trellis nodes where M_c is $K - 1$ (where 'K' is the constraint length of the transmitter).

α - Alpha

As seen in section 3.4.1, $\alpha_i(s)$ values can be determined once $\gamma_i(s,s)$ has been calculated.

Fig. 56 shows the implementation of alpha for each node of the decoding trellis, whereas

Fig. 57 is the flow chart of calculation process.

Fig. 56. For each time step alpha calculations are required for all trellis nodes

Fig. 57. Alpha calculation flow diagram

- $\alpha[j,i]$ matrix of i columns, j rows
- L number of message bits in the frame
- M_c number of states in RSC
- $s(j)$ is the j th state of the trellis
- $S(i)$ represents all the 2^{M_c} states $s(j)$ at ' i ' in the trellis
- A Represents the states that can transition into s_i from states s_{i-1} . As shown in Fig. 58, state S_7 at i , $A = \{S_6, S_7\}$ at $i-1$.

Fig. 58. Sample state transitions

β - Beta

As seen above, $\beta_k(s)$ values can be determined after $\gamma_i(\cdot, s)$ has been established. Fig. 59 shows the implementation of β for each node of the decoding trellis, whereas Fig. 60 is the flow chart of calculation process.

Fig. 59. For each time step, beta calculations are required for all trellis nodes

Fig. 60. Beta calculation flow diagram

$\beta[j,i]$ matrix of i columns, j rows

B Represents the states that can transition into s_i from states s_{i+1} . As shown in

Fig. 58, state S_7 at i , $A = \{S_3, S_7\}$ at $i+1$.

The turbo decoder has only been implemented in Matlab and not in Hardware. It has been created in Matlab for system level analysis and performance evaluation.

3.4.2 System performance vs. number of decoder iterations

Typically, increasing the number of decoder iterations improves the BER of the received sequence. The simulation graph below obtains a measure of Bit Error Rate (BER) improvement as the number of decoder iterations increase. As expected (see Fig. 61), the BER improves with the increasing iteration number; however, the improvement rate of the BER decreases rapidly after the 5th iteration.

Fig. 61. Iterations vs. BER 1024 bit frame, $g1(D) = [1 \ 1 \ 1 \ 1]$, $g2(D) = [1 \ 0 \ 1 \ 1]$, single trellis termination, 1/3 code rate and random interleaver.

3.4.3 De-Interleaver

A de-interleaver essentially follows the same principle as the interleaver (see section 3.3.2). It restores the original temporal sequence that the interleaver scrambles in order to reduce vulnerability from error bursts. The de-interleaver receives the interleaver-scrambled packet or values generated by the decoder, and places them back into the original sequence. The Matlab function created for the de-interleaver could have been hard-coded, considering that the original scrambling sequence is known. However, a 10-bit pseudo random sequence (similar to the one used in scrambling) was created. The index of the sequence is identical to the original interleaver, and it uses the index values to retrieve the original sequence. This methodology eliminates the need to store interleaving indexes, thus potentially reducing the memory requirements for the turbo decoder.

4 HARDWARE VERIFICATION AND OPTIMIZATION

This research has emphasized developing architecture with smaller area and power requirements but greater timing margins. A Digital ASIC development and process guideline, described by Canadian Micro-Electronics Corporation (CMC), has been followed. Fig. 62 describes the digital flow and the ASIC development tools used. The analysis and figures for the synthesized hardware is based on 0.18 μm CMOS and VST libraries with worst-case parameters. The system level analysis and verification was done in Matlab, whereas Verilog Cadence XL and ModelSim were used for behavior modeling. Synopsys – Design Analyzer, along with VST libraries, were used for synthesis and extraction of area, power and timing figures.

The subsections of this chapter present

- verification techniques used
- power and area reduction
- timing analysis
- savings due to interleaver design
- savings due to the dual-port on-chip memory design
- analysis of single-port on-chip design.

Fig. 62. Design process and tools used

4.1 Verification

The design's functionality was verified by the creation of test benches. Matlab simulation obtained output vectors for the overall system were used in the test benches and have been referred to as theoretical results. A device under test was instantiated, and inputs used to generate Matlab result vectors were provided to the hardware. The resulting waveforms from the hardware were compared to the theoretical waveform using an automatic comparator waveform and were found to be identical. See Fig. 25, Fig. 36, Fig. 37, Fig. 40 and Fig. 46.

4.2 Reduced Area and Power Figures

The optimized core area and power figures of the synthesized turbo encoder and memory are 0.986 mm² and 46.01mW respectively (Table 16). The optimization was carried out by specifying the constraints in the tool. As predicted and shown in Fig. 63, the memory module represents the largest block in terms of area and power consumption.

	Area (mm ²)	Power (mW)
Published at IWSOC 2005 [4]	1.22	56.56
Optimized results reported in the Thesis	0.986	46.01

Table 16. Initial and optimized power and area figures

Fig. 63. Area requirements for each turbo encoder block

Fig. 64. Power consumption for each turbo encoder block

The design of the interleaver, Single-Port On-Chip Memory and Dual-Port On-Chip Memory were studied to reduce area and power requirements (see sections 4.5 and 4.6). Traditional interleavers are based on block interleaving, as explained in sections 2.1.2 and 3.3.2, but random interleavers have been shown to perform better by orders of magnitude [18] (see section 4.4).

4.3 Timing Analysis

The timing numbers presented in earlier sections are representative of 'slow slow,' worst case operating times. Dual-port memory operates at 333 MHz, whereas the turbo encoder block (other than memory) can operate 333 MHz without optimization.

	Turbo Encoder	Memory	Total
Slack Timing at 333 MHz (ns)	0.27	0.49	0.27

To improve speed, both the memory design and the turbo encoder design blocks were optimized using the synthesis tool. With stricter constraints, the turbo encoder blocks (excluding memory) met timing at 450 MHz.

The memory was recompiled and redesigned to increase the operational frequency. Limitations due to memory compiler, Virage's design and 0.18 μm CMOS technology bound the operational frequency of Dual-Port memory to 333 MHz. Single Port memory can operate at 450 MHz, but it alters the architecture and, as section 4.6 explained, this architecture increases area requirements by 60%, power consumption by 40% and, yet, operates at 225 MHz.

	Memory	Encoder Design	Overall Design
Reported [6]	--	110.8 MHz for encoder and decoder	110.8 MHz for encoder and decoder
Dual-Port On-Chip Memory Design	333 MHz	450 MHz	333 MHz
Single-Port On-Chip Memory Design	450 MHz but 2 cycles required for write access, hence effective frequency of 225 MHz	450 MHz	225 MHz

Table 17. Summary of encoder designs

4.4 Savings due to Interleaver Design

Among random interleavers, optimized random interleaver (ORI) and pseudo random interleavers were evaluated. The ORI simulations were carried out to determine the index which provides the maximum distance between adjacent bit locations. It was observed and discussed in section 3.3.2 that BER performance of the system improves with increasing distance between adjacent bits. The sequence with maximum distance between adjacent bits is referred to as optimized random sequence. In the experiments carried out in this study, compared to pseudo random sequence, optimized random sequences performed marginally better, but a need arises to store individual bit locations for all of the bits. 10K memory would be required to locate the optimized sequence and it would increase memory requirements by a factor of 8.

Pseudo random sequence, in contrast, is generated using shift registers and xor gates at various taps. The seed is fixed and the pseudo random sequence identical for successive packets; therefore, the bit locations are not required to be stored. Unlike optimized interleaver, where 10K memory or ROM, is required for every 1K packet, the PRI design has been reduced to only 10 registers and tens of gates. Considering there are at least eight other parameters which can influence the performance of a turbo codes, using ORI was considered and then eliminated from the design.

4.5 Savings due to Dual-port on-chip memory design

An architecture with dual-port on-chip memory is described in section 3.3.3. The design requires the input data sequence to be stored in memory as it arrives. The bits are stored in an 8-bit register and then written to the memory every 8th clock cycle. The location of a particular bit in the memory corresponds to the order in which it arrived. Once the entire packet has been stored, the same bits are retrieved in two sequences, the first corresponding to the original sequence, and the second corresponding to the interleaved sequence. This concurrent accessing is achieved through two independent ports accessing the memory cells. (See Fig. 65.)

As shown in Fig. 63 and Fig. 64, compared to other turbo encoder blocks, the memory block requires the largest amount of area (75% of the entire turbo encoder) and consumes the most power.

Fig. 65. Architecture with dual-port on-chip memory

4.6 *Single-port on-chip memory design*

Single-port on-chip memory design, as shown in Fig. 66, was studied as a possible architecture for turbo encoder. The single main advantage of the single port memory architecture is the faster operational frequency. However, both the original and the interleaved original sequences are required to be stored. This doubles the memory requirements. Memory now occupies 89% of the turbo encoder area and consumes 60% more power.

Even a 60% increase in turbo encoder area requirements and power consumption fails to deliver higher speeds than architecture with dual-port on-chip memory design. The memory's write access cycle has a limitation of writing once every two clock cycles. In the dual-port memory case, the write cycle was carried out every 8th clock cycle, whereas, in a single port memory architecture, a write cycle is required for every bit and at every clock cycle to exploit fully the maximum clock frequency. Writing every 2nd clock cycle effectively lowers the design speed to 225 MHz. Therefore, it is this study's conclusion that the architecture with Dual-port Memory Design requires less area, consumes less power and operates at a higher frequency.

Fig. 66. Architecture with single-port on-chip memory

5 CONCLUSION AND FUTURE WORK

This thesis developed the coding hardware of a turbo encoder system, reducing power consumption and area, while increasing maximum operational clock frequency. During the course of study, a turbo encoding/decoding architecture was developed using Matlab, then the encoder was implemented in hardware and finally optimized for minimal power, area and maximum operating frequency. Initially, system level simulations were carried out to verify the effectiveness and efficiency of the turbo encoding and decoding process. The simulations clarified the algorithms involved and helped determine the impact of decoder iterations, encoder transfer function and block size on latency, throughput and BER. After exploring the algorithms, the turbo encoder with on-chip memory and controller was implemented using Verilog. The design features a pseudo random index interleaver that does not require the index addresses to be stored. An architecture has been employed that does not require storage of RSC1 outputs which further reduced the memory requirements by one half. This architecture has been realized using a dual-port memory design. The overall design has been synthesized and laid out using 0.18 μm CMOS technology, where methodology includes behavioral modeling, logic synthesis and timing analysis. The developed turbo encoder, as summarized in Table 18, has a clock rate of 333 MHz, and requires core area of 0.99 mm^2 and 46 mW of power.

	Area (mm^2)	Power (mW)	Speed (MHz)
Published at IWSOC 2005 [4]	1.2	56.6	250
Optimized results reported in the Thesis	0.99	46.0	333

Table 18. Summary of thesis findings (also see Table 19)

The turbo encoder presented in the publication [25], the technique stores the interleaved data packet, and original data packet, RSC1-encoded-data packet to synchronize the data streams. The architecture proposed in the thesis requires the packet to be stored once. It is then read sequentially and in the interleaved manner to supply the appropriate bits to the RSC encoders. Considering memory block requires the most amount of power and area, this study has reduced the memory requirements to 1/3.

	Previously published results with its parameters[25]	Thesis findings and its parameters.
Component encoders	RSC encoders	RSC encoders
encoding Rate	1/3 rate with no puncturing	1/3 rate with no puncturing
RSC transfer function	$[17_H; 23_H]$	$[F_H; B_H]$
Interleaver	Block Interleaver	10 bit PRI
Frame size L	2048 bits	1024 bits
Termination method	No termination	Single trellis termination
Area	$\sim 5.76 \text{ mm}^2$	1 mm^2
Maximum Frequency	130 MHz	333 MHz
Power	not available	49 mW.

Table 19. Previously established results and findings of the thesis.

	Memory (MHz)	Design (MHz)	Overall design (MHz)
Previously published [6]	--	110.8	110.8
Previously published [25]	--	130	130
Results published at IWSOC 2005 [4].	250	250	250
Optimized findings reported in the Thesis	333	450	

Table 20. Comparison of maximum clock frequency of previously published results and thesis findings

After the initial development of this turbo encoder, the results were published in IWSOC 2005 [4]. That turbo encoder required a core area of 1.3 mm^2 and 56 mW of power. However, since then, optimization has improved these figures to the clock rate being 333 MHz, requiring a core area of 0.99 mm^2 and 46.0 mW of power. Memory accesses for storage and retrieval take place at a maximum frequency of 333 MHz, whereas all other turbo encoding blocks can operate up to 450 MHz. Incorporating faster memory or eliminating the need for memory will allow the chip to operate faster. Using registers may alleviate this bottleneck but greatly increase area requirement. Memory consumes 75% of the total power and core area. In order to improve power and area figures, it is essential to reduce the storage and retrieval requirements. Smaller packet size or usage of frames will reduce the required area proportionally, but this will also degrade the decoding performance.

To advance the topic, architecture improvements from the encoder can be employed at the decoder stage. On-chip memory will eliminate the need to use external memory devices. Furthermore, analysis and comparison between using various types of memory

and on-chip memory will help create faster turbo encoding/decoding systems. Although reduction of switching activity through architecture improvements, pipelining, RTL and synthesis optimization have been achieved, techniques such as clock gating, power gating, parallel processing, and multiple- V_{dd} have not been harnessed. If applicable, these techniques could further improve the power and speed figures.

REFERENCES

- [1] J. P. Woodard and L. Hanzo, "Comparative Study of Turbo decoding Techniques: An Overview", IEEE Transactions on Vehicular Technology, vol. 49, No. 6, Nov. 2000, pp. 2208 – 2233.
- [2] http://www.techonline.com/community/ed_resource/feature_article/20016?print, Forward Error Correction.
- [3] L. Fanucci, C. Sicilia, and D. Sicilia, Electronics, "VLSI design of a high speed turbo decoder for 3rd generation satellite communication", 9th International Conference on Electronics Circuits and System. (ICESC) Dubrovnik, Croatia, vol. 2, Sept. 15-18, 2002., pp. 509 – 512.
- [4] M. Israr, Dr. T. Kwasniewski, "Turbo codes – A Digital IC Design", 5th IEEE International Workshop on System on Chip for Real-Time Applications, July. 20-24, 2005, pp. 341 – 346.
- [5] G. Maera, M. Mazza, G. Piccinini, F. Viglione, and M. Zamboni, "Architectural Strategies for Low-Power VLSI Turbo decoders", IEEE Transactions on Very Large Scale Integration (VLSI) Systems. Vol. 10, issue 3, June 2002, pp. 279 – 285.
- [6] M. A. Bickerstaff, D. Garret, T. Prokop, C. Thomas, B. Widdup, Gongyu Zhou, Linda M. Davis, Graeme Woodward, Chris Nicol, and Ran-Hong Yan, "A Unified Turbo/Viterbi Channel decoder for 3GPP Mobile Wireless in 0.18 – μm CMOS," IEEE Journal of Solid-State Circuits, Vol. 37, No. 11, November 2002, pp. 1555- 1564.
- [7] Z. Wang, Z. Chi and K. K. Parhi, "Area-Efficient High Speed decoding Schemes for Turbo/MAP decoders", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Volume 10, Issue 6, Dec. 2002, pp. 902 – 912.

- [8] F. Viglione, G. Masera, G. Piccinini, R. Roch, M. Zamboni, "A 50 Mbit/s iterative Turbo-decoder", Design, Automation and Test in Europe Conference and Exhibition 2000. March 27-30, 2000, pp. 176 – 180.
- [9] <http://lsiwww.epfl.ch/LSI2001/teaching/webcourse/ch07/ch07.html#7.2>, Low power VLSI circuits and systems.
- [10] <http://www.cse.sc.edu/~jimdavis/Research/Papers-PDF/ICCAD02-Davis-Akella-3.pdf> , On the Tradeoff between Productivity and Area Optimization in VLSI Design.
- [11] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error correcting coding and decoding: Turbo-codes," Proc. ICC'93, Geneva, Switzerland, May 1993, pp. 1064-1070.
- [12] C. Heegard, S. B. Wicker, "Turbo coding", Kluwer Academic Publishers, Norwell, Massachusetts 02061 USA, 1999, pp 47.
- [13] A. J. Viterbi and J.K. Omura, "Principles of Digital Communications and coding", McGraw Hill, 1979.
- [14] G. C. Clark. Jr. and J.B. Cain. "Error-Correction coding for Digital Communications". Plenum Press, 1981.
- [15] S. B. Wicker. "Error Control Systems for Digital Communications and Storage". Prentice Hall, Englewood Cliffs, NJ 07632, 1995.
- [16] K. C. Chang, "Digital Systems Design with VHDL and Synthesis", An Integrated Approach, IEEE Computer Society, 1999.

- [17] S. Crozier, "Turbo-Code Design Issues: Trellis Termination Methods, Interleaving Strategies, and Implementation Complexity", Invited presentation for Panel Session on "Application of Turbo codes" at International Conference on Communications (ICC'99), Vancouver, British Columbia, Canada, June 6-10, 1999.
- [18] A. J. Viterbi, "An intuitive justification and a simplified implementation of the MAP decoder for convolutional codes," IEEE J. Select. Areas Commun., Feb. 1997, pp. 260–264.
- [19] J. A. Erfanian, S. Pasupathy, and G. Gulak, "Reduced complexity symbol detectors with parallel structures for ISI channels," IEEE Trans. Commun., vol. 42, 1994, pp. 1661–1671.
- [20] <http://www.ee.vt.edu/~yufei/turbo.html>, MATLAB code for experiment on turbo codes.
- [21] A. Ambroze, G. Wade and M. Tomlinson, "Practical aspects of iterative decoding", IEE Proceedings – Communications, vol. 147, issue 2, April 2000, pp. 69 – 74.
- [22] P. H. Y Wu, S. M. Pisuk, "Implementation of a Low Complexity, Low Power, Integer-Based turbo decoder" Global Telecommunications Conference, 2001, (GLOBECOM '01). San Antonio, Texas, USA, vol. 2, 25-29 Nov. 2001, pp. 946 – 951.
- [23] M. C. Valenti, J. Sun, "Handbook of RF and Wireless Technologies", Page 375.
- [24] <http://www.crc.ca/en/html/fec/home/interleaver/interleaver>, Interleaver Technology.

[25] M. Jezequel, P. Penard, "Turbo4: a high bit-rate chip for turbo code encoding and decoding". IEE Colloquium, Turbo Codes in Digital Broadcasting - Could It Double Capacity? London, Nov. 1999, pp. 174. 22.

[26] <http://rfcmos.icu.ac.kr/pdf/domestic/conferences/12.pdf>, 0.25 μm CMOS transistor and inductor test patter layout for RFIC applications.

[27] 3GPP Organizational Partners, "3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Multiplexing and channel coding", 3GPP, TS 25.222 V6.2.0, Dec 2004.

[28] K. Ouchi, K.Kubo, T.Muzuochi, Y.Miyata, H. Yoshida, H.Tagami, K. Shimiz, T. Kobayashi, K. Shimoura, K. Onohara and K. Motoshima, "A Fully Integrated Block Turbo Code FEC for 10 Gb/s Optical Communication Systems". Optical Fiber Communication Conference, 2006 and the 2006 National Fiber Optic Engineers Conference, 5-10 March 2006, pp. 3 – 5.

BIBLIOGRAPHY

- [a] Z. Wang, H. Suzuki, and K. K. Parhi, "VLSI Implementation issues of turbo decoder Design for Wireless Applications", IEEE Workshop on Signal Processing Systems, (SiPS 99). Taipei, Taiwan, 20-22 Oct. 1999, pp. 503 – 512.

- [b] J. Hsu and C. Wang, "On Finite-Precision Implementation of a decoder for turbo codes", IEEE International Symposium on Circuits and Systems, ISCAS '99. Orlando, Florida, vol. 4, 30 May - 2 June 1999, pp. 423 – 426.

- [c] S. Crozier, "turbo-Code Design Issues: Trellis Termination Methods, Interleaving Strategies, and Implementation Complexity", Invited presentation for Panel Session on "Application of turbo codes" at International Conference on Communications (ICC'99), Vancouver, British Columbia, Canada, 6-10 June, 1999.

- [d] http://en.wikipedia.org/wiki/Turbo_code, Turbo Codes.

- [e] D. Divsalar and F. Pollara, "turbo codes for PCS application," in Proc. ICC'95, Seattle, WA, June 1995, pp.54-59.

- [f] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain," in Proc. Int. Conf. Communications, June 1995, pp. 1009–1013.

- [g] G. Battail, "Ponderation des symboles decodes par l'algorithme de Viterbi," Ann. Telecommun., vol. 42, no. 1–2, Jan. 1987, pp. 31–38.

- [h] J. Hagenauer and P. Hoeher, "A Viterbi algorithm with soft-decision outputs and its applications," IEEE Globecom, 1989, pp. 1680–1686.
- [i] G. D. Forney, "The Viterbi algorithm," Proc. IEEE, vol. 61, Mar. 1973, pp. 268–278.
- [j] M. Breiling and L. Hanzo, The super-trellis structure of turbo codes, in IEEE Trans. Inform. Theory, Sept. 2000.
- [k] "Optimum noniterative turbo-decoding," in Proc. PIMRC'97, Helsinki, Finland, Sept. 1997, pp. 714–718.
- [l] C. Berrou, "Some clinical aspects of turbo codes," in Proc. Int. Symp. turbo codes and Related Topics, Brest, France, Sept. 1997, pp. 26–31.
- [m] H. Nickl, J. Hagenauer, and F. Burkett, "Approaching Shannon's capacity limit by 0.27 dB using simple Hamming codes," IEEE Commun. Lett., vol. 1, Sept. 1997, pp. 130–132,
- [n] W. Koch and A. Baier, "Optimum and sub-optimum detection of coded data disturbed by time-varying inter-symbol interference," IEEE Globecom, Dec. 1990, pp. 1679–1684.
- [o] Peter H. Y Wu, S. M. Pisuk, "Implementation of a Low Complexity, Low Power, Integer-Based turbo decoder" Global Telecommunications Conference, 2001, (GLOBECOM '01). San Antonio, Texas, USA, vol. 2, 25-29 Nov. 2001, pp. 946 – 951.
- [p] B. Sklar, "A primer on turbo code concepts," IEEE Commun. Mag., Dec. 1997, pp. 94–102.
- [q] G.D. Forney, Jr., Concatenated codes. Cambridge, MA: MIT Press, 1966.

- [r] R. Prasad and H. Kobayashi, "Multi-Methodology design: An Experimental Comparison", in Proceedings IVC-96, 5th International Verilog HDL Conference, Santa Clara, CA, Open Verilog International, 1996, pp. 45-49.
- [s] W. F. Lee, "VHDL coding and Logic Synthesis with Synopsys", Academic Press, Inc., 2000.
- [t] P. Robertson, "Illuminating the structure of parallel concatenated recursive systematic (TURBO) codes," in Proc GlobeCOM'94, San Francisco, CA, No. 1994, pp.1298-1303.
- [u] A. Viterbi, "Approaching the Shannon limit: Theorist's dream and practitioner's challenge," in Proc. Int. Conf. Millimeter Wave and Far Infrared Science and Technology, 1996, pp. 1-11.
- [v] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," IEEE Trans. Inform. Theory, Mar. 1974, pp. 284-287.

APPENDIX A – DESIGN PROCESS

The following is a strategy in designing a turbo encoder and decoder:

- 1 Extensive research on the topic
- 2 System level implementation in Matlab
- 3 Developing functional specifications
- 4 Generating behavioral for the modules
- 5 Incorporating required 0.18u memory into the design
- 6 Functional Simulation
- 7 Simulating & synthesizing of individual module, then combining them
- 8 Zero-delay simulation (except for clock)
- 9 Composite Synthesis to ensure correct functionality
- 10 Test insertion (scan insertion)
- 11 Gate level test bench was created and simulated
- 12 Clock constraints
- 13 Timing Verification
- 14 Hand tooling to enhance critical path

APPENDIX B – THEORETICAL CALCULATIONS

MAP Calculations

Visit <http://dualist.stanford.edu/~ee487/PAPERS/turbo2.pdf>. [1] Pages 2 to 5 describe MAP algorithm and its derivations. The paper describes, forward recursion calculation of $\alpha_k(s)$ values, backward recursion calculation of $\beta_k(s)$ values and calculation of the $\gamma_k(s)$ values in further detail.

Iterative turbo decoding

Visit <http://dualist.stanford.edu/~ee487/PAPERS/turbo2.pdf>. [1] Pages 5 to 7 explain turbo decoding structure and have its derivations.

log-map algorithm

Visit <http://dualist.stanford.edu/~ee487/PAPERS/turbo2.pdf>. [1] Pages 7 to 9 describe derivations and correction factor leading to the Log-MAP algorithm.

APPENDIX C – TRANSFER FUNCTIONS

In order to evaluate coding strength of the transfer functions, poly2trellis, a Matlab function, has been employed and the following tables have been created:

- Transfer Function Tables
- Termination Tables
- Last Output / Last State Tables

A total of 9 sets of tables were created for the transfer functions, however, only 3 sets are presented below.

Poly2trellis

Poly2trellis is part of the Matlab Communication tool set.

Its inputs are:

- ConstraintLength,
- Feedforward polynomial and
- Feedback polynomial

It produces the following outputs [§]

- numInputSymbols
 - returns a value representing the number of symbols provided to the RSC encoder.
- NumOutputSymbols
 - returns a value representing the number of output symbols from the RSC encoder.
- numStates
 - returns the number of total states in the RSC encoder.
- nextStates
 - returns all of the 'Next state' combinations based on the current state and '0' or '1' transitions.
- outputs
 - returns all of the outputs based on the current state and '0' or '1' inputs.

Poly2trellis in conjunction with Formulae in Microsoft Excel have been used to develop the above mentioned tables.

[§] http://www.mathworks.com/access/helpdesk_r13/help/toolbox/comm/poly2trellis.html

Methodology for Developing Transfer Function Tables:

The Transfer Function tables have been generated using formulae in Microsoft Excel. The generator matrix has been strategically placed at a 'relative' location to the input, state, state+ (next-state) and outputs.

- 'x' is the input bit to the RSC Encoder
- 's0' is determined by a formula that has been entered to calculate the result of the feedback. The resultant can be seen at 's0' column. A sample formula developed is:
 $s0 = \text{ROUND}(\text{MOD}(((B238 * \$B\$233) + (D238 * \$D\$233) + (E238 * \$E\$233) + (F238 * \$F\$233)), 2), 0)$.
where:

Table 21: s0 formula variables

Feedback Polynomial Assignment	States Assignment
$B233 = g_{11}$	$B238 = x$
$D233 = g_{12}$	$D238 = s1$
$E233 = g_{13}$	$E238 = s2$
$F233 = g_{14}$	$F238 = s3$

- 'output2' is similarly calculated to determine the outputs of the feed-forward. A sample formula developed is: $\text{output2} = \text{ROUND}(\text{MOD}(((C238 * \$C\$234) + (D238 * \$D\$234) + (E238 * \$E\$234) + (F238 * \$F\$234)), 2), 0)$.

Table 22: output2 formula variables

Encoding Polynomial Assignments	States Assignments
$C234 = g_{11}$	$C238 = s0$
$D234 = g_{12}$	$D238 = s1$
$E234 = g_{13}$	$E238 = s2$
$F234 = g_{14}$	$F238 = s3$

- 'Output1' is identical to 'x'; $\text{output1} = x$.
- 'next-states', s1+, s2+ and s3+ are next states based on s0, s1 and s2:
 $s1+ = s0$.
 $s2+ = s1$.
 $s3+ = s2$.

Table 23 Transfer Function 1: $g = [1\ 1\ 1\ 1; 1\ 0\ 1\ 1]$

233	A	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>	<i>F</i>	<i>G</i>	<i>H</i>	<i>I</i>	<i>J</i>	<i>K</i>	<i>L</i>	<i>M</i>
		1		1	1	1							
234	g1												
235	g2		1	0	1	1							
236		feedback											
		xor											
		(x,s1,s2,s3)											
237		x	s0	s1	s2	s3	state	s1+	s2+	s3+	state+	output1	output2
238		0	0	0	0	0	0	0	0	0	0	0	0
239		0	1	0	0	1	1	1	0	0	4	0	0
240		0	1	0	1	0	2	1	0	1	5	0	0
241		0	0	0	1	1	3	0	0	1	1	0	0
242		0	1	1	0	0	4	1	1	0	6	0	1
243		0	0	1	0	1	5	0	1	0	2	0	1
245		0	0	1	1	0	6	0	1	1	3	0	1
244		0	1	1	1	1	7	1	1	1	7	0	1
245		1	1	0	0	0	0	1	0	0	4	1	1
246		1	0	0	0	1	1	0	0	0	0	1	1
247		1	0	0	1	0	2	0	0	1	1	1	1
248		1	1	0	1	1	3	1	0	1	5	1	1
249		1	0	1	0	0	4	0	1	0	2	1	0
250		1	1	1	0	1	5	1	1	0	6	1	0
251		1	1	1	1	0	6	1	1	1	7	1	0
252		1	0	1	1	1	7	0	1	1	3	1	0

Methodology for Developing Termination Tables

The termination reflects the forcing of the RSC encoder states to '0'. Considering, the feedback and encoding polynomials are fixed, the input, 'x', is the only variable that can influence the Encoder states. The termination sequences have been established to deliver a calculated 'input' value to 'x'. At request, 'input' is provided to the Encoder such that 's0' takes on a '0' value. Since the constraint length is '4', a maximum of three such 'x' values will be required to force the 'state' values to '000'.

The values of states and next-states with their inputs and output have been established in the 'Transfer Function Tables'. Transitions that result in 's0' being null are selected by sorting the Transfer Function Table with respect to 's0' and then 'state'. The resultant top eight rows are the required fields for termination.

Table 24. Termination -1

state	x	s0	state+	output1	output2
0	0	0	0	0	0
1	1	0	0	1	1
2	1	0	1	1	1
3	0	0	1	0	0
4	1	0	2	1	0
5	0	0	2	0	1
6	0	0	3	0	1
7	1	0	3	1	0

Methodology for 'Last Output' and 'Last State' Tables

Once the Transfer Function Table has been established, it is sorted with respect to 'x' and within the same sorting function with respect to 'state+'. The resulting table outputs the desired previous state and output results.

- 'Last Output' = output1 and output2 with values '0' changed to represent modulated symbol of '-1'.
- 'Last State' = state

The state in question is 'state+'.

Note: due to indexing limitations in Matlab, the state values were incremented by '1'.

Table 25. Last State and Last output Table -1

x	state+	s0	state	output1	output2
0	0	0	0	0	0
0	1	0	3	0	0
0	2	0	5	0	1
0	3	0	6	0	1
0	4	1	1	0	0
0	5	1	2	0	0
0	6	1	4	0	1
0	7	1	7	0	1
1	0	0	1	1	1
1	1	0	2	1	1
1	2	0	4	1	0
1	3	0	7	1	0
1	4	1	0	1	1
1	5	1	3	1	1
1	6	1	5	1	0
1	7	1	6	1	0

g1
g2

Table 26. Transfer Function 2: $g = [1 \ 1 \ 1 \ 1; 1 \ 1 \ 0 \ 1]$

1	1	1	1
1	1	0	1

xor (x,s1,s2,s3)					xor (s0,s1,s3)						
x	s0	s1	s2	s3	state	s1+	s2+	s3+	state+	output1	output2
0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	1	1	1	0	0	4	0	0
0	1	0	1	0	2	1	0	1	5	0	1
0	0	0	1	1	3	0	0	1	1	0	1
0	1	1	0	0	4	1	1	0	6	0	0
0	0	1	0	1	5	0	1	0	2	0	0
0	0	1	1	0	6	0	1	1	3	0	1
0	1	1	1	1	7	1	1	1	7	0	1
1	1	0	0	0	0	1	0	0	4	1	1
1	0	0	0	1	1	0	0	0	0	1	1
1	0	0	1	0	2	0	0	1	1	1	0
1	1	0	1	1	3	1	0	1	5	1	0
1	0	1	0	0	4	0	1	0	2	1	1
1	1	1	0	1	5	1	1	0	6	1	1
1	1	1	1	0	6	1	1	1	7	1	0
1	0	1	1	1	7	0	1	1	3	1	0

Table 27. Termination - 2 : $g = [1 \ 1 \ 1 \ 1; 1 \ 1 \ 0 \ 1]$

state	x	s0	state+	output1	output2
0	0	0	0	0	0
1	1	0	0	1	1
2	1	0	1	1	0
3	0	0	1	0	1
4	1	0	2	1	1
5	0	0	2	0	0
6	0	0	3	0	1
7	1	0	3	1	0

Table 28. Last State / Last Output - 2: $g = [1 \ 1 \ 1 \ 1; 1 \ 1 \ 0 \ 1]$

x	state+	s0	state	output1	output2
0	0	0	0	0	0
0	1	0	3	0	1
0	2	0	5	0	0
0	3	0	6	0	1
0	4	1	1	0	0
0	5	1	2	0	1
0	6	1	4	0	0
0	7	1	7	0	1
1	0	0	1	1	1
1	1	0	2	1	0
1	2	0	4	1	1
1	3	0	7	1	0
1	4	1	0	1	1
1	5	1	3	1	0
1	6	1	5	1	1
1	7	1	6	1	0

g1
g2

Table 29. Transfer Function 3: $g = [1\ 0\ 1\ 1; 1\ 1\ 1\ 1]$

1	0	1	1
1	1	1	1

xor (x,s2,s3)						xor (s0,s1,s2,s3)					
x	s0	s1	s2	s3	state	s1+	s2+	s3+	state+	output1	output2
0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	1	1	1	0	0	4	0	0
0	1	0	1	0	2	1	0	1	5	0	0
0	0	0	1	1	3	0	0	1	1	0	0
0	0	1	0	0	4	0	1	0	2	0	1
0	1	1	0	1	5	1	1	0	6	0	1
0	1	1	1	0	6	1	1	1	7	0	1
0	0	1	1	1	7	0	1	1	3	0	1
1	1	0	0	0	0	1	0	0	4	1	1
1	0	0	0	1	1	0	0	0	0	1	1
1	0	0	1	0	2	0	0	1	1	1	1
1	1	0	1	1	3	1	0	1	5	1	1
1	1	1	0	0	4	1	1	0	6	1	0
1	0	1	0	1	5	0	1	0	2	1	0
1	0	1	1	0	6	0	1	1	3	1	0
1	1	1	1	1	7	1	1	1	7	1	0

Table 30. Termination - 3 $g = [1\ 0\ 1\ 1; 1\ 1\ 1\ 1]$

state	x	s0	state+	output1	output2
0	0	0	0	0	0
1	1	0	0	1	1
2	1	0	1	1	1
3	0	0	1	0	0
4	0	0	2	0	1
5	1	0	2	1	0
6	1	0	3	1	0
7	0	0	3	0	1

Table 31. Last State / Last Output - 3 $g = [1\ 0\ 1\ 1; 1\ 1\ 1\ 1]$

x	state+	state	output1	output2
0	0	0	0	0
0	1	3	0	0
0	2	4	0	1
0	3	7	0	1
0	4	1	0	0
0	5	2	0	0
0	6	5	0	1
0	7	6	0	1
1	0	1	1	1
1	1	2	1	1
1	2	5	1	0
1	3	6	1	0
1	4	0	1	1
1	5	3	1	1
1	6	4	1	0
1	7	7	1	0

g1
g2

Table 32. Transfer Function 4: $g = [1\ 0\ 0\ 1; 1\ 0\ 1\ 1]$

1	0	0	1
1	0	1	1

x	xor (x,s3)				state	s1+	s2+	s3+	state+	x output1	xor (s0,s2,s3) output2
	s0	s1	s2	s3							
0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	1	1	1	0	0	4	0	0
0	0	0	1	0	2	0	0	1	1	0	1
0	1	0	1	1	3	1	0	1	5	0	1
0	0	1	0	0	4	0	1	0	2	0	0
0	1	1	0	1	5	1	1	0	6	0	0
0	0	1	1	0	6	0	1	1	3	0	1
0	1	1	1	1	7	1	1	1	7	0	1
1	1	0	0	0	0	1	0	0	4	1	1
1	0	0	0	1	1	0	0	0	0	1	1
1	1	0	1	0	2	1	0	1	5	1	0
1	0	0	1	1	3	0	0	1	1	1	0
1	1	1	0	0	4	1	1	0	6	1	1
1	0	1	0	1	5	0	1	0	2	1	1
1	1	1	1	0	6	1	1	1	7	1	0
1	0	1	1	1	7	0	1	1	3	1	0

Table 33. Termination - 4: $g = [1\ 0\ 0\ 1; 1\ 0\ 1\ 1]$

state	x	s0	state+	output1	output2
0	0	0	0	0	0
1	1	0	0	1	1
2	0	0	1	0	1
3	1	0	1	1	0
4	0	0	2	0	0
5	1	0	2	1	1
6	0	0	3	0	1
7	1	0	3	1	0

Table 34. Last State / Last Output - 4: $g = [1\ 0\ 0\ 1; 1\ 0\ 1\ 1]$

x	state+	state	output1	output2
0	0	0	0	0
0	1	2	0	1
0	2	4	0	0
0	3	6	0	1
0	4	1	0	0
0	5	3	0	1
0	6	5	0	0
0	7	7	0	1
1	0	1	1	1
1	1	3	1	0
1	2	5	1	1
1	3	7	1	0
1	4	0	1	1
1	5	2	1	0
1	6	4	1	1
1	7	6	1	0

APPENDIX D – COMPARISON OF PERFORMANCE

The following are graphs and turbo code performances taken from 'Handbook of RF and Wireless Technologies', pg 394 by C. Valenti. The figure below demonstrates performances of CDMA2000 for four code rates. The graph of 1/3 code rate is compared against the findings of the Thesis. The parameters of the graphs below are similar with the parameters set in the Thesis, however, it should be noted that SNR, modulation scheme, component encoder, encoding rate, encoding transfer function, interleaver type, frame size, termination or stopping criteria of the simulation can influence the performance significantly.

Parameter	Thesis	Handbook of RF and Wireless Technologies
SNR	AWGN	AWGN
Modulation	2 – PAM	BPSK
Component encoders	RSC encoders	RSC encoders
encoding Rate	1/3 rate with no puncturing	1/3 rate with no puncturing
RSC transfer function	1111;1011	1011;1011
decoder algorithm	Log – MAP	Log – MAP
Interleaver	random interleavers using Matlab 'rand' function	not stated
Frame size L	1024 bits & 2048 bits	1530
decoder iterations	5	8
Termination method	Single trellis termination	not stated
Stopping Criteria	100 Frame Errors	not stated

Fig. 67. Performance graph of 1/3 rate from 'Handbook of RF and Wireless Technologies' [23]

Fig. 68. Interleaving depth vs. BER

An comparison of the thesis finding with the previously published results. Note: the performance of the published result can not be taken as stated to measure the accuracy of the Turbo Code developed in this thesis. The published are to be taken as a ball-park estimate.

Target BER	SNR - Handbook	SNR - Thesis	% difference
10^{-1}	--	--	--
10^{-2}	0.28	0.25	1.2
10^{-3}	0.47	0.46	2.2
10^{-4}	0.66	0.7	-5.7*
10^{-5}	--	--	--

Table 35 Comparison of performance figures from [23] and Thesis. The % difference is calculated by $[\text{SNR}_{\text{THESIS}} - \text{SNR}_{\text{HANDBOOK}}] / \text{SNR}_{\text{THESIS}}$.

- The increased performance of the published results in [23] at lower BER can be attributed to the higher number of decoder iterations, termination scheme used and the transfer function workings. Note that at lower BER, the results are within close to 2% of the two studies.

Parameter	Thesis	Handbook of RF and Wireless Technologies	Paramet setting would result in better performance for
RSC transfer function	1111;1011	1011;1011	undetermined
frame size L	2048	1530	Thesis
decoder iterations	5	8	[23]
termination method	Single trellis termination	not stated	undertermined

Table 36 The table captures the differences in parameters and how they impact the performances.