Reconfigurable Cores for Wireless Appliances: Turbo Codes

VOLUME 1 (OF 2)

Edward Brown

A themed portfolio submitted to
The Universities of

Edinburgh,
Glasgow,
Heriot-Watt and
Strathclyde

For the Degree of
Doctor of Engineering in System Level Integration

Abstract

This portfolio thesis documents the work carried out by the author whilst enrolled in the Engineering Doctorate (EngD) programme. The author completed the majority of the work at the sponsoring company, Xilinx, in their Digital Signal Processing division.

The thesis introduces the subject of Turbo codes, highlighting the motivation behind their inclusion in international standards. Particular attention is given to the cdma2000 and UMTS third generation mobile telephony standards. Both the technical and commercial advantages/disadvantages of implementing Turbo codes in a Field Programmable Gate Array (FPGA) based system are discussed. The subject of third generation mobile technology is also discussed, this includes an introduction to spread spectrum and rake receivers.

The commercial relevance of all projects conducted is discussed. These projects allowed the sponsoring company to highlight the advantages of using FPGAs in third generation mobile base stations.

A novel system for testing forward error correction (FEC) codes is presented. Results obtained are shown and discussed. A novel parameterisable Turbo decoder will also be highlighted. The decoder in question allows the user to specify certain criteria that can be used to control the memory used by the decoder and its latency. A novel hardware architecture for Turbo decoders is proposed, as is a unique channel variance value that optimises a cdma2000 Turbo decoder. Other subjects covered are Duo-Binary Turbo codes, Turbo decoder hardware architectures and how to calculate the input values to Turbo decoders.

Declaration of Originality

The material in this thesis is entirely the results of my own independent research under the supervision of Dr J. M. Irvine and Mr B. Wilkie and is not the outcome of any collaborative work. All published or unpublished material used in this thesis has been given full acknowledgement.

Edward Brown

28 September 2005

Acknowledgements

On a professional level I would like to thank both of my supervisors, Bill Wilkie and Dr. James Irvine, for all the support and advice they have provided throughout my EngD. I am extremely grateful for the time they spent advising me on all matters, from reviewing my conference submissions to answering technical queries. Special thanks also go to Dr. Colin Carruthers and Dr. David Lawrie. Colin, along with Bill, awarded me the opportunity to work with Xilinx, while David proved an invaluable resource throughout the latter stages of my EngD project. I also would not have been able to enrol on the EngD without the help of my former supervisor, Patrick Lysaght. I would therefore, also like to thank Patrick for giving me the opportunity to do the EngD.

I would also like to thank all of Xilinx Edinburgh, especially the DSP team, for making me feel so welcome. Also to Ann and Rhona, thanks for driving and/or entertaining me on all those long journeys along the M8.

I would like to express my gratitude to the ISLI and all its staff, particularly Alexandra Buchanan for all the help and guidance she has provided. Other staff I would like to thank are Sian Williams for all her help in the library, and also for helping with EngD matters in the latter stages of my degree, including arranging my viva, and Wendy Glendinning for all her help during the MSc portion of the EngD.

On a more personal basis I'd like to thank my Mum and Dad for all the support they have given me throughout not only my EngD, but my undergraduate degree as well. I would like to thank my sister and nephew for always supporting me throughout both of my degrees. I'd like to thank all my friends, particularly Linda and Farah for supporting me throughout my EngD, Linda and Louise, my flatmates, for putting up with me throughout my write up and my golfing/drinking buddies, Richie, Steve, Nige, Jamie, Andy and Brian for providing some much needed distraction over the last 5 years.

Glossary

3G	Third generation mobile, an advancement from 2G standards, such as GSM and IS-95, that allows higher data rates. The European standard is the UMTS standard, the United States of America standard is the cdma2000 standard.
Alpha Probability	One of the three probabilities required to calculate final Turbo decoder output. The alpha probability is calculated by performing a forward trace of the trellis.
ASIC	Application Specific Integrated Circuit. An integrated circuit that is not reconfigurable. It is used for one particular purpose.
AWGN channel	Additive white Gaussian noise channel. A channel that is often used in communication system models. The name comes from the fact that the noise is linear and has an average value of 1 over its entire distribution.
BER	Bit error rate. A measure for how many bits in a particular sequence are in error. Usually, BER is given in powers of ten, e.g. 10^{-6}, meaning 1 bit in every 1 million bits are in error.
BERT	BER Tester. A platform that can be used to determine the BER of a particular system.
Beta probability	Required to calculate the final Turbo decoder output. Calculated by performing a backward trace of the trellis.
Block code	A type of forward error correction (FEC) code that relies only on the current input for its output.
cdma2000	The 3G standard to be implemented in the United States of America.
Coding gain	The difference in dB between two BER curves, when compared at the same BER value.
Convolutional code	A type of FEC code whose current output is dependent upon the previous m inputs.
DSP	Digital Signal Processor. A specialised microprocessor that is used in digital signal processing (DSP). It contains components, such as Multiply-Accumulators (MACs), that are often used in DSP.
DVB	Digital Video Broadcasting. A standard that uses Turbo coding on its reverse satellite channel.

$\frac{E_b}{N_0}$	A measure of signal to noise ratio, where E_b is the energy per bit and N_0 is the noise spectral density.
Fading channel	A channel model that is more like a real mobile channel than the AWGN channel introduced earlier. The most common fading channels are the Rayleigh channel and the Rician channel. The Rayleigh channel is a model where the source and destination have no direct line of sight. In the Rician channel, the source and destination has a direct line of sight or has a dominant component among the signals received by the destination.
Fast termination	A method of reducing the average number of iterations performed by the Turbo decoder.
FEC	Forward Error Correction. A method of detecting and correcting errors in a communications system.
FER	Frame Error Rate. A measure of error in an FEC system. As data in 3G systems is transmitted in frames, this provides a quantifiable measure of how many frames would be dropped for any given system.
FPGA	Field Programmable Gate Array. A reconfigurable integrated circuit containing DSP specific components, such as multipliers.
Gamma probability	The gamma probability is required to calculate the final output from the Turbo decoder. It is also known as the branch probability.
Interleaver	An interleaver is used to de-correlate data entering the second Turbo component encoder from the first Turbo component encoder, improving the BER performance of the Turbo code in the process.
LLR	Log likelihood ratio. Data input to a Turbo decoder is in the format of a LLR. An LLR gives the probability of a particular symbol representing either a 1 or a 0. The larger the magnitude of the LLR, the higher the probability that it is correct.
Log-MAP	Applying the MAP algorithm in the logarithm domain allows it to be implemented in hardware more easily, operations such as exponentials disappear and multiplications turn into additions.
MAP	Maximum a posteriori. The MAP algorithm is used in Turbo codes to calculate the probability that a particular bit at a particular point in time is either a 1 or a 0.

Max-log-MAP	A simpler variation of the log-MAP algorithm. The log-MAP algorithm requires a look up table to produce its output. The max-log-MAP algorithm simplifies the log-MAP by removing the look up table.
Max Scale	The Max Scale is a variation of the max-log-MAP algorithm. It applies a scaling factor to the Turbo decoder output, improving the BER performance of the Turbo decoder.
Puncturing	A method of increasing the information rate of a FEC code. The FEC code is punctured by removing certain symbols at the Turbo encoder output. The punctured symbols are replaced by an all zero codeword at the Turbo decoder input, thereby decreasing the BER performance.
RSC	Recursive systematic convolutional. Turbo component encoders are RSC encoders, an RSC encoder is one that contains feedback and whose input is a part of the encoder output symbol.
SISO	Soft-in, soft-out. Turbo component decoders are SISO decoders, by both inputting and outputting soft bits each component decoder can share information on what they think the Turbo decoder output should be. The main types of SISO decoder used implement either the MAP algorithm or the SOVA.
SOVA	Soft output Viterbi algorithm. A variation of the Viterbi algorithm that outputs soft bits, allowing it to be used to implement a Turbo component decoder.
Spread spectrum	Almost all 3G standards are based on spread spectrum systems. Spread spectrum is a term that describes a system that allows users to share both time and frequency.
Turbo code	A turbo code is a type of FEC, traditional Turbo codes use RSC encoders as a component encoder. Turbo product codes use block encoders as component encoders.
Turbo encoder	The Turbo encoder is used at the transmission side of the communications system. It contains 2 encoders separated by an interleaver. Each encoder is known as a component encoder.
Turbo decoder	The Turbo decoder is used at the receiver side of the communications system. It contains 2 SISO decoders separated by

	an interleaver and de-interleaver. Each SISO decoder is known as a component decoder.
UMTS	The name of the 3G standard implemented in Europe.
VHDL	Very high speed integrated circuit description language. A programming language used to model hardware systems. Widely used when designing FPGA circuits.
Sliding window	A technique used to reduce the overall latency of a Turbo decoder, allows the Turbo decoder to begin decoding before a complete set of data has been received.

Symbols Used

$\alpha_t(s)$	Forward state metric/probability
$\beta_{t-1}(s')$	Backward state metric/probability
$\gamma_k(s', s)$	Branch metric/probability
τ	Measure of time
i	Number of decoder iterations
L	Measure of memory
$L(u_k)$	Log-likelihood ratio
$L^e(u_k)$	Extrinsic data
$L^e_{IN}(u_k)$	Component decoder extrinsic input
$L^e_{OUT}(u_k)$	Component decoder extrinsic output
m	Number of component encoders/decoders
N	Block size
q	Symbol width
R	Code rate
S	Number of states in trellis
u_t	Turbo encoder input
$\hat{u}_t$	Turbo decoder output
w	Sliding window width
X_t^S	RSC 1 systematic output
$X_t^{'S}$	RSC 2 systematic output
X_t^P	RSC 1 parity output
$X_t^{'P}$	RSC 2 parity output
Y_t^S	SISO 1systematic input
$Y_t^{'S}$	SISO 2 systematic input
Y_t^P	SISO 1 parity input
$Y_{t\lambda}^{'P}$	SISO 2 parity input
$\mathbf{y}$	Turbo decoder input

Contents

List of Figures

List of Tables

1. Executive Summary

The initial specification of the Engineering Doctorate (EngD) industrial project stated that by the time of its completion the EngD Research Engineer (RE) would have had investigated new opportunities within wireless appliances to exploit the reconfigurability of Field Programmable Gate Arrays (FPGAs) as a key, product-differentiating feature. This was achieved through numerous projects, all of which were based around Turbo codes.

1.1. Projects

The first stage of the industrial project was to decide on a task that was mutually beneficial to the sponsoring company, Xilinx, and the RE. Xilinx had acquired a Turbo encoder and Turbo decoder from Frontier Designs [1] that conformed to the UMTS specification [2]. To create the Turbo encoder and decoder, Frontier used their A|RT Designer tool [3, 4] to convert C++ code to a synthesizable VHDL netlist. As Xilinx intellectual property (IP) is written in structural VHDL it can take a relatively long time to simulate. To overcome this problem Xilinx usually supplied a behavioural model of the design to customers for simulation purposes, this had not been supplied by Frontier. It was decided that this would be a perfect learning opportunity for the RE to familiarise himself with Turbo codes, while producing something that benefited the sponsoring company and their customers. This stage of the project included both a literature search and practical work. The work carried out in this portion of the industrial project is summarised in Appendices A-C {vol. 2/pp. 1-25}. The time scale to complete this portion of the project can be obtained from Figure 1. The reports produced by the RE during this project were of great commercial relevance to Xilinx, particularly 'Investigation of Turbo Decoder Hardware Architectures' , Appendix D {vol.2/pp. 26-40}. At this time Xilinx were designing a Turbo codec that complied with the cdma2000 standard [5]. 'Investigation of Turbo Decoder Hardware Architectures' allowed Xilinx to determine what hardware architecture should be used for their cdma2000 Turbo decoder.

Figure 1 highlights the commercial relevance and novelty of all projects undertaken by the RE and shows milestones passed by the RE as the industrial project progressed.

The second project tackled was to use a bit error rate test (BERT) platform [6] created by Nallatech to test the Frontier Turbo codec. The forward error correction (FEC) designs

were instantiated in the BERT using a VHDL wrapper. The FEC designs could be either written in VHDL or be an EDIF netlist. The FEC design under test (DUT) could then be stimulated by random data supplied by a Matlab script. Throughout the life of this project there were conflicts between the EDIF netlist supplied by Frontier and the Nallatech BERT, as a result the Frontier Turbo codec could not be tested. The development of the Nallatech BERT was passed to a staff Engineer at Xilinx, while the RE began to develop a recursive systematic convolutional (RSC) encoder that conformed to the cdma2000 Turbo code standard. In between the two projects the RE was able to present a poster [7] highlighting the advantages of using both Turbo codes and FPGAs.

While the cdma2000 encoder was being developed Xilinx became interested in the subject of Duo-Binary Turbo codes, which were to be used in the Digital Video Broadcasting (DVB) standard [8]. The RE was asked to research the subject of Duo-Binary Turbo codes and compile a report on the subject, Appendix E {vol. 2/pp. 41-50}, paying particular attention to the differences between Duo-Binary and standard Turbo codes. This allowed Xilinx to determine the effort required to convert their cdma2000 codec to a DVB codec.

Once the cdma2000 Turbo codec was complete it was tested using the Nallatech BERT platform, this helped Xilinx verify that their codec design was working correctly while creating marketing data for their design. As with most Xilinx cores the Turbo decoder was parameterisable. The user could specify parameters such as sliding window size, decoder input width and which algorithm to use for the decoder. The parameterisable aspect of the design allowed the RE to test the design using various configurations. The results of these tests were presented by the RE in two separate publications [9, 10]. These papers helped Xilinx publicise the reconfigurable nature of their design while allowing the RE to publish novel research.

One subject that was of interest to the RE and Xilinx was the subject of how to manipulate the input data to a Turbo decoder so that the best possible BER performance was achieved. The result of this investigation was the discovery of a novel variance value that optimised the performance of Xilinx's Turbo decoder; this is documented in Appendix F {vol. 2/pp. 51-65}. When this literature search was taking place it became apparent that the Nallatech BERT platform was not performing optimally: often the BERT would crash when a test was running. It was therefore necessary that the problems with the Nallatech BERT be

solved or a new system be designed. Around the same time a new version of the Xilinx System Generator tool [11] had recently been released. It was decided that the RE should implement a BERT in System Generator. Implementing a BERT in System Generator allowed the RE to create something that was completely novel and allowed Xilinx to investigate how the System Generator tool performs when it is used in system development. All results presented in this thesis were obtained using the System Generator BERT. A System Generator design can be altered repeatedly and implemented on any compatible Xilinx FPGA platform [12, 13, 14] further highlighting the reconfigurable nature of the System Generator tool and Xilinx designs.

1.2. Commercial Relevance

The subject of commercial relevance is dealt with more comprehensively in Chapter 5. A summary of the commercial relevance of all projects is summarised below in reference to Figure 1.

- **Behavioural models and literature search:** The report "Investigation of Turbo Decoder Hardware Architectures", Appendix D {vol. 2/pp. 26-40}.
- **Nallatech BERT:** The BERT was used by the RE to produce BER plots for Xilinx's cdma2000 Turbo codec, a selection of the results produced were presented by the RE [9].
- **cdma2000 Turbo encoder:** The RE designed a (RSC) encoder that conformed to the cdma2000 standard.
- **Investigation of calculating Turbo decoder inputs:** The outcome of this investigation was a report entitled "Calculating Inputs to Turbo Decoders", Appendix F {vol. 2/pp. 51-65}. The report suggested a novel value that could be used for channel variance, the value proposed optimised the Turbo decoder core created by Xilinx.
- **SysGen BERT:** The System Generator BERT allowed Xilinx to research and market their Turbo code cores and their 3G solutions in general.

1.3. Novelty

A summary of the novel aspects of the EngD project are shown below in reference to Figure 1.

- **Behavioural models and Literature search:** Presentation of original hardware architectures.

- **Nallatech BERT:** This project highlighted a novel parameterisable Turbo decoder [9].
- **cdma2000 Turbo encoder:** The RSC encoder was designed entirely using Xilinx structural libraries and was targeted to the Xilinx Virtex-II FPGA.
- **Investigation of calculating Turbo decoder inputs:** Presentation of a novel channel variance value that optimised the performance of a Turbo decoder.
- **SysGen BERT:** The System Generator BERT is novel due to the fact that it is designed completely in System Generator. Designing in System Generator allowed features to be added rapidly and for the testing system to be automated [15]. As it is designed in System Generator it allows non-FPGA designers to use and upgrade the BERT platform in an FPGA environment without using standard FPGA design techniques such as HDLs.

1.4. Milestones

The milestones referenced in Figure 1 are detailed below.

- 1: June 2001, 120 taught technical credits achieved.
- 2: February 2002, Turbo encoder behavioural model complete. Report on encoder behavioural model complete, Introduction to Turbo codes report complete.
- 3: June 2002, Turbo decoder behavioural model partially complete. Report on decoder behavioural model complete, Investigation of Turbo Decoder Hardware Architectures report complete.
- 4: December 2002, poster presentation, SET for Europe.
- 5: May 2003, Duo-Binary Turbo Codes report complete.
- 6: September 2003, paper published at IEE Colloquium on DSP Enabled Radio.
- 7: November 2003, Calculating Input Values for Turbo Decoders report complete.
- 8: January 2004, 60 taught business credits achieved.
- 9: May 2004, paper published at World Wireless Congress.
- 10: June 2004, System Generator BERT completed.

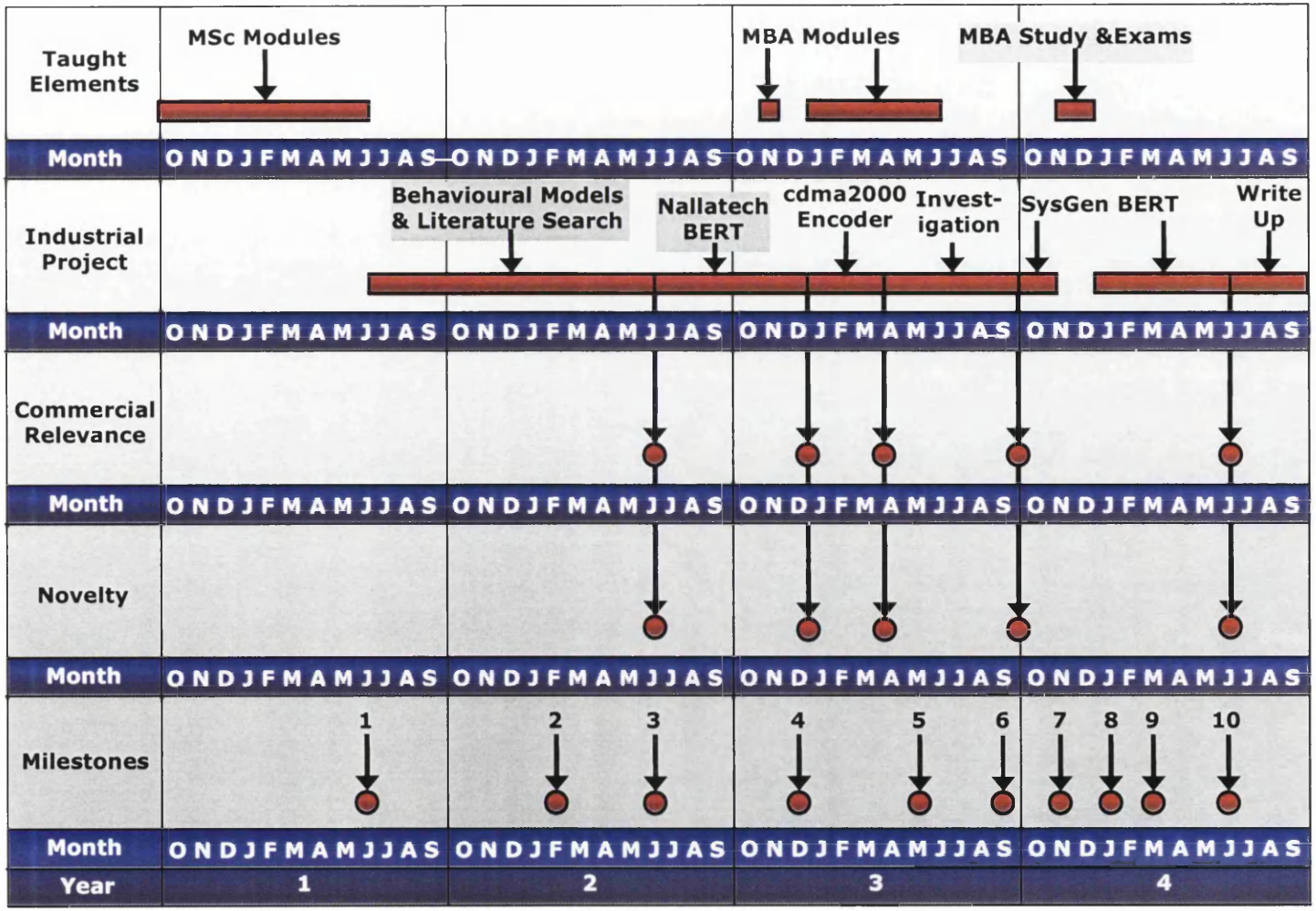

Figure 1: Work carried out by the EngD over a four year period

2. Portfolio Organisation

The Portfolio Thesis is presented in two volumes. Volume 1 presents the main motivations and outcomes of the work. It also shows the commercial relevance and novelty of the work. Volume 2 presents numerous appendices containing papers and reports compiled by the RE as the EngD progressed. All reports presented are on the subject of Turbo codes but cover numerous areas within the Turbo code field.

The remainder of Volume 1 is organised as follows: Chapter 3 discusses the taught technical and business credits that were chosen by the RE, showing how the classes chosen aided the RE in both his research and in gaining an understanding on how the projects undertaken impacted on Xilinx commercially. Chapter 4 highlights all publications made by the RE. The commercial relevance of projects undertaken by the RE is examined in Chapter 5. It is shown how the numerous projects conducted by the RE contributed to the commercial success of Xilinx's 3G solutions and how FPGAs can be used to reduce the cost of building a mobile base station.

Chapter 6 presents the technical background to the RE's industrial project. Firstly the subject of channel coding is introduced. Particular attention is given to the subject of convolutional codes and the differences between a standard convolutional encoder and a recursive systematic convolutional (RSC) encoder, used in Turbo codes, is given. An introduction to Turbo codes is then presented. Each component, the Turbo encoder, interleaver and Turbo decoder, in the Turbo codec are described separately. The Section on Turbo encoders includes a discussion on the standards looked at by the RE during the EngD, the UMTS [2], cdma2000 [5] and DVB [8]. The advantages and disadvantages of implementing a Turbo codec in a Field Programmable Gate Array (FPGA) based system are also discussed. Chapter 7 will highlight some of the results obtained by the RE and discuss the significance of these results. The results included tests that change inputs such as block size and code rate. Novel results such as a comparison between UMTS and cdma2000 Turbo codecs will also be analysed. Finally, Chapter 8 will summarise the thesis and propose future work that should be carried out.

Figure 2 shows how each of the chapters in Volume 1 relate to each other. All chapters in some way or another have an impact on the commercial relevance of the project. As stated

previously, the business modules have a direct link to the commercial relevance of the projects undertaken. Any results generated were used to promote Xilinx 3G solutions to potential customers, similarly any publications highlighting these results promoted Xilinx's 3G solutions to international audiences.

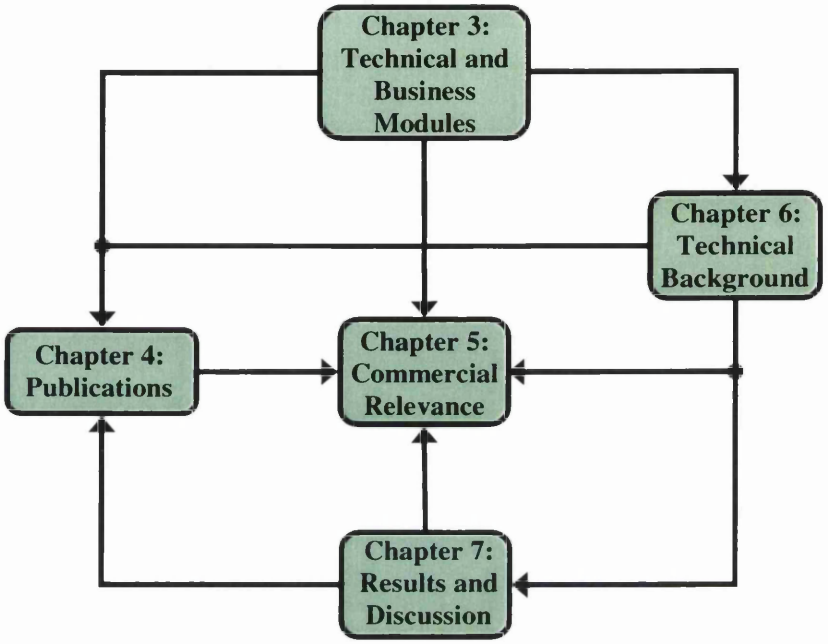

Figure 2: Organisation of Volume1

3. Taught Modules

Taught technical credits were obtained in the RE's first year at the Institute for System Level Integration (ISLI). The 60 business credits were obtained from the University of Strathclyde Graduate Business School.

3.1. Technical Credits

In total 120 credits were gained at the ISLI, the subjects taken by the RE to gain these credits are shown in Table 1. The classes at the ISLI provided an insight into many aspects of system on a chip (SoC) design. Classes that benefited the RE's industrial project were: Intellectual Property Block Authoring (IPBA); Intellectual Property Block Integration (IPBI); Communication Algorithms; and Mobile Communications. Both IPBA and IPBI contributed to the RE's knowledge of digital design with a particular emphasis on design reuse through parameterisation. Communication Algorithms and Mobile Communications helped the RE understand some of the basic concepts of mobile telephony and error control coding.

Table 1: Technical credits breakdown

Class	*Credits*
SoC Overview	2
System Partitioning	12
IPBA	12
IPBI	12
VLSI Design	12
Software Engineering	15
Microcontrollers and Microprocessors	12
Towards Deep Submicron	7
Communications Algorithms	12
Multimedia and Video	7
Mobile Communications	7
Broadband and Digital Networks	10
Total	**120**

3.2. Business Modules

Completing the business modules allowed the RE to be more aware of the commercial relevance of the work carried out during the industrial project. Sixty credits were gained by the RE from the Master of Business Administration (MBA) modules available at the University of Strathclyde Graduate Business School. The MBA modules chosen by the RE and their credit weighting are shown in Table 2.

Table 2: Business credits breakdown

Class	*Credits*
Marketing Management	12
Data Management	6
Information Systems	6
Managing People in Organisations	12
Finance and Financial Management	12
Financial and Management Accounting	12
Total	**60**

From a purely research point of view Information Systems was very beneficial for the RE. It helped the RE gain an understanding of how to manage and extract necessary information where an abundance of information is at the researcher's disposal. Both Finance and Financial Management and Financial and Management Accounting allowed the RE to recognise how important financial constraints and project deadlines were in allowing projects to be financially beneficial to the company. Marketing Management was of particular significance to the RE. It introduced the RE to new concepts that could be used to highlight work the RE had completed and allowed the RE to look at products from a customer's viewpoint. Gaining this ability meant that the RE could improve systems such as the System Generator BERT by adding features which improved its usability.

4. Publications

The RE made numerous presentations at international conferences. The publications made by the RE are highlighted below.

- **Reconfigurable Cores for Wireless Appliances** [7]: An introduction to both FPGAs and Turbo codes. The RE also highlighted the advantages of implementing Turbo codes in FPGAs.
- **A Memory-efficient Parameterisable FPGA Implementation of the cdma2000 Turbo Codec** [9]: Presented a novel parameterisable Turbo decoder. The RE used the Nallatech BERT to produce results that showed how changing the parameters of the Turbo codec core affected performance.
- **A Memory-efficient Parameterizable FPGA Implementation of the cdma2000 Turbo Codec** [10]: The paper presented here showed how the BERT could be used to calculate novel results such as a channel variance value that optimised the Turbo codec core.
- **Rapid Prototyping of a Test Harness for Forward Error Correcting Codes** [16]: The paper concentrated on the implementation of the System Generator BERT designed by the RE. It highlighted a number of inputs that made the BERT user programmable. The publication was included in the conference session 'Novel Applications of Reconfigurability'.
- **Rapid Prototyping of a Test Harness for Forward Error Correcting Codes** [15]: In this paper the RE presented a number of results that were produced using the System Generator BERT. The results included a comparison between the cdma2000 and UMTS Turbo codecs.

5. Commercial Relevance

By the time 3G mobile networks were launched in the UK [17] mobile telephony, due to its rapid acceptance by the consumer, was a relatively mature product. Consumers therefore had become accustomed to a certain level of quality for a certain price on mobile networks, for example seamless handover between base stations while on a call and near 100% network coverage. Mobile telephones were expected to be lightweight and compact, have a battery that could last for 2 to 3 days and offer basic functions such as games and a calendar. Both network operators and phone manufacturers faced problems when 3G telephony was initially launched, phones and network tariffs were expensive, battery life was short and phones were relatively large and heavy. One of the major technological challenges for 3G network operators was implementing an algorithm that could seamlessly handover calls between GSM and 3G networks. This is partly due to the fact that the UMTS standard implemented in Europe has no direct backward compatibility with GSM. Benson and Thomas [18], Jugl and Pampel [19] and Lugara et al [20] discuss the limitations of certain handover algorithms and present simulation results of these algorithms.

The majority of consumers did not see the advantages of 3G over more mature technologies such as GPRS. This led to a very slow take up in 3G phones. Three, the first company to launch 3G in the UK had only 210,000 customers by the end of their first year in March 2004, they had expected one million [21]. However, the market for 3G telephones is expanding, all but one of the companies able to provide 3G services was expected to launch 3G networks by December 2004 [22]. Three have attempted to attract customers by offering tariffs that are competitive with the 2G network operators' free minutes and free short message service (SMS) bundles. This has been backed with free trials of Three's 3G services such as football highlights and video conferencing. This seems to have been successful with Three reaching a total of 1·2 million customers in the UK by August 2004 [23].

5.1. FPGAs

The size and power consumption of FPGAs mean it is difficult for them to be used in mobile phones in the imminent future. However, given their reconfigurability they are ideal for use in mobile base stations or for hardware prototyping. The reconfigurable nature

of FPGAs is particularly important given the changing nature of 3G mobile standards. Both UMTS and cdma2000 standards changed a number of times between their inception and the final standard being set. This offers manufacturers of FPGAs a great advantage over the manufacturers of Application Specific Integrated Circuits (ASICs). An FPGA manufacturer can offer their 3G solutions to any customers and as the standards evolve so can the solutions offered by the FPGA manufacturers. However, when the final standard is released the ASIC manufacturer can offer a solution that is smaller, faster and consumes less power. Time to market is a very important part of any product development, the reconfigurability of FPGAs means that their solutions can always beat ASICs to the target market. As FPGAs are "off the shelf" parts they are also cheaper than ASICs for low volume production.

Initially FPGAs were used as 'glue logic' in mobile base stations, so called because it connected two or more complex systems. However, as the processing power of FPGAs has increased they have been used more extensively in base stations. Many base stations use a combination of ASICs, Digital Signal Processors (DSPs) and FPGAs to implement their desired system. ASICs offer the designer high speed, low power and small area, but are very costly to design and offer little freedom in terms of reconfiguration. DSPs and FPGAs both offer reconfigurability, while DSPs may be smaller and cheaper per unit than FPGAs they cannot offer the same processing power. This has led to FPGAs replacing both ASICs and DSPs in many base stations. FPGAs are ideal for use in components such as rake receivers and Turbo decoders as they allow multiple channels to be processed in parallel. This leads to FPGAs being less expensive per channel than DSPs [24]. Reconfigurability is not the unique selling point of FPGAs, as DSPs can also be reconfigured. The unique selling point of FPGAs is offering a chip that is reconfigurable and can offer the user a processing speed that can come relatively close to that of an ASIC.

As FPGA technology evolves they will eventually become small, low power devices relative to the ASICs of today. Obviously ASIC technology will also evolve and become smaller, faster and consume less power than future FPGAs. However, they cannot offer the reconfigurability and time to market advantages that future FPGAs will offer. Just as at the moment a base station manufacturer can reconfigure an FPGA remotely by transmitting a bit stream, in the future network operators will be able to reconfigure FPGAs on mobile telephones remotely. This is a key aim of software defined radio (SDR) [25]. The

reconfigurable nature of FPGAs may well allow them to eat into the ASICs share of the handset and base station market when SDR becomes common across all networks. The subject of SDR is discussed in more detail in Chapter 6.4.

5.2. Contribution

All work carried out by the RE has been of significant commercial relevance to Xilinx. The most significant piece of work carried out was undoubtedly the System Generator BERT. This was used both as a research tool and as a marketing tool. After creating their Turbo codec's the main problem facing Xilinx was functional verification and generating results to present to customers for marketing purposes. Initially the codec was functionally verified by comparing the output of the VHDL simulations with a C++ model of the codec. Any marketing or research data was also generated with the C++ model. The problem with this is that it can take weeks or even months to generate accurate plots. Using real hardware to test the system was the only real viable solution. The System Generator BERT was implemented in both a Nallatech XtremeDSP Kit [12] and an Annapolis WildcardII [13] PCMCIA card. Both of these contained Xilinx Virtex II 3000 chips [26]. Using the System Generator BERT meant that requests for plots could be taken from customers, generated and passed to the customer rapidly. In some cases the System Generator BERT could be given to the customer so that they could test the Turbo codec core for themselves. Another important aspect of the System Generator BERT is that it can be used in demonstrations and in conference presentations, an accurate BER plot could be generated in a matter of seconds. As the system was automated using various scripts it also meant that the user could set up a number of simulations and then leave the BERT running continuously. The BERT was designed to be extremely user friendly: to start a test the user needed only to be familiar with very basic Matlab skills, such as how to run a script.

The behavioural models designed by the RE were to be used by Xilinx customers who purchased the Xilinx/Frontier Turbo codec and therefore had great commercial relevance to Xilinx. The main purpose of the behavioural model was to allow the user to simulate the Turbo codec. As well as producing the same results as the structural design it also had to offer a significant simulation speed-up over the structural design and be easy to use. The behavioural model could also be used to verify the structural VHDL design.

The RE also designed a recursive systematic convolutional (RSC) encoder that conformed to the cdma2000 standard for Xilinx. Again, it could be used by customers in their cdma2000 codec.

The numerous reports created by the RE also had an impact on the commercial output from Xilinx. Two particular reports of significance were "Investigation of Turbo Decoder Hardware Architectures", Appendix D {vol. 2/pp. 26-40}, and "Calculating Input Values for Turbo Decoders"; Appendix F {vol. 2/pp. 51-65}. The latter proposed a novel value that optimised the performance of the Xilinx Turbo decoder. The former was used to determine which architecture should be used to implement the cdma2000 Turbo decoder.

6. Technical Background

In 1948 Claude Shannon published his channel coding theorem [27], this theorem showed that data could be transmitted in a noisy channel with little or no error, provided that the data rate does not exceed the channel capacity. The publication of this paper led to a number of different theorems, and therefore channel coding techniques, on how to achieve this goal. Common between all techniques is the inclusion of a channel encoder and channel decoder to the communications system. A simple communications system is shown in Figure 3.

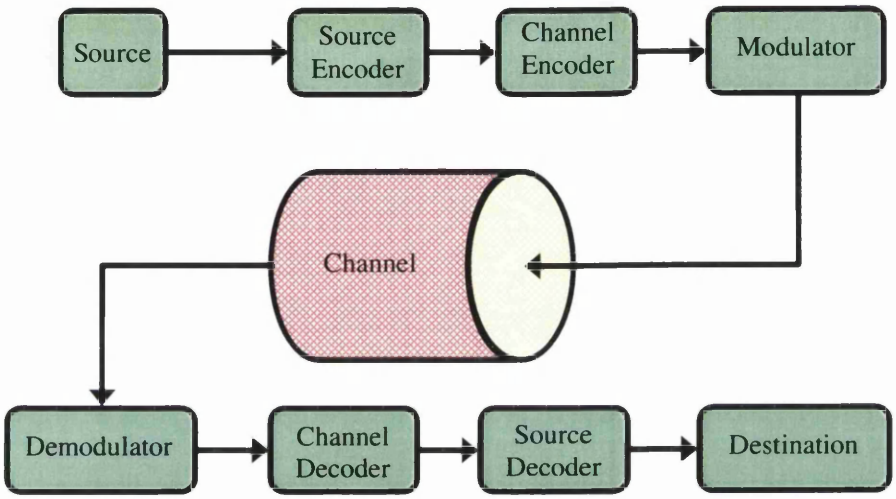

Figure 3: Overview of a simple communications system

The encoder and decoder implemented can perform simple error detection, where the system simply realises that one or more bits received is in error and take action such as requesting that the data in error be re-transmitted, or a more complex system called forward error correction (FEC). FEC systems both detect and correct errors, this enhancement is paid for by the complexity of the FEC decoder. Two coding techniques are used to implement FEC systems, block codes and convolutional codes. The major distinction between the two is that the output from a block encoder is completely reliant on its current input alone. The output from a convolutional code is reliant on the current input plus the previous m inputs, where m is the number of memory elements in the convolutional encoder. The number of memory elements in a convolutional encoder is known as the constraint length. Block encoders accept k-bits at its input and produce n parity bits from this input. In total, $n+k$ bits are transmitted. Of the bits transmitted, n is redundant as they contain no real information. Redundancy can be measured in terms of code rate, R. For a

block code redundancy can be measured using (1). Code rate is inversely proportional to the number of parity bits generated. Hence, as redundancy increases the code rate decreases and less useful information is transmitted. The advantage of increasing redundancy is that the error correcting power of the code is increased.

$$R = \frac{k}{n+k} \tag{1}$$

The code rate of a convolutional encoder can be calculated using (2), where k is the number of bits into the encoder and n is the number of bits out.

$$R = \frac{k}{n} \tag{2}$$

As Turbo codes are a convolutional FEC technique this portfolio thesis shall look at this type of code in more depth. The reader is referred to more comprehensive publications on the subject of block codes [28, 29] for more information on this subject.

6.1. Convolutional Codes

An example of a convolutional encoder is shown in Figure 4. The component encoders in Turbo encoders are recursive systematic convolutional (RSC) encoders. The encoder shown in Figure 4 is both non-recursive and non-systematic. The differences between the two types of convolutional encoders will be highlighted in this section. In Figure 4 the input, u_t, is passed through memory elements and modulo-2 adders to generate parity streams, X_t^{P0} and X_t^{P1}. Using (2) it can be shown that the code rate for this encoder is $\frac{1}{2}$.

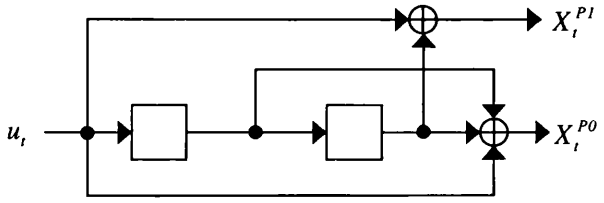

Figure 4: Convolutional encoder

A convolutional encoder can be represented using a polynomial. The polynomial that represents X_t^{P0} is shown in (3); the polynomial that represents X_t^{P1} is shown in (4)

$$g_1(D) = 1 + D + D^2 \tag{3}$$

$$g_2(D) = 1 + D^2 \tag{4}$$

Any non-systematic, non-recursive convolutional encoder can be converted to a RSC convolutional encoder using the rule shown in (5).

$$G(D) = 1, \frac{g_2(D)}{g_1(D)} \tag{5}$$

Figure 5 shows the equivalent RSC encoder created using (5) from (3) and (4).

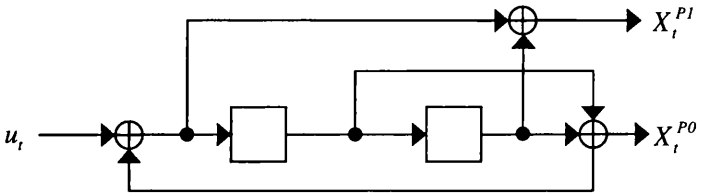

Figure 5: RSC encoder

For Turbo codes the main advantage of using RSC instead of non-RSC encoders is that when RSC encoders are concatenated in parallel they produce relatively more high weight code words, leading to a better BER performance [30]. Berrou et al [31] and Benedetto et al [32] both show that the BER performance of an RSC encoder is better than the BER performance of the corresponding non-RSC encoder.

To decode a code word it is advantageous for the decoder to know the starting state of the encoder that produced the code word. This is done by returning the encoder to the all zero state after a packet has been encoded. This is relatively easy for the non-RSC encoder as all that is required is to input a sequence of m zeros, where m is the number of memory elements. For the RSC encoder a process known as trellis termination [33] must be used to return the encoder to the all zero state. Figure 6 shows the process of trellis termination. When a block has been encoded the data input, u_t, to the first XOR gate is replaced by the feedback input to the first XOR gate, thereby always producing an input of zero. The outputs of the encoder when in the trellis termination state are known as tail bits.

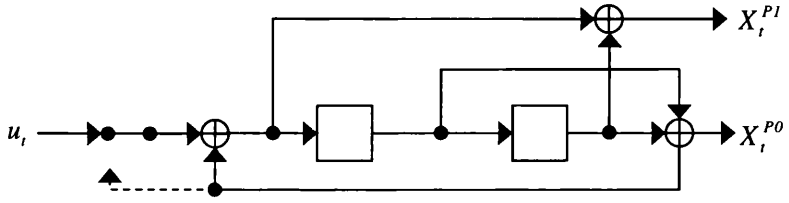

Figure 6: RSC encoder with trellis termination

6.2. Turbo Codes

Introduced in 1993 Turbo codes [31] are a relatively recent innovation in channel coding research. They have already been included in numerous international standards, including deep space telemetry [34], digital video broadcasting (DVB) [8], UMTS [2] and cdma2000 [5]. Their popularity has grown due to their BER performance at low signal to noise ratios, the initial paper [31] came within 0·7dB of the Shannon limit. The unprecedented performance of Turbo codes can be attributed to three main factors: the use of parallel concatenated convolutional encoders; the use of an interleaver between each component encoder and the use of iterative decoding. A top-level view of a Turbo encoder is shown in Figure 7. The systematic and parity outputs can be punctured before transmission. Puncturing is a process where certain encoder outputs are deleted so that the code rate is increased, hence increasing the information rate. When an encoder output is punctured it is replaced with an all zero codeword at the decoder input. Using puncturing will usually result in a degraded BER performance, as will be shown in Chapter 7. The systematic output from RSC2, $X_t^{'S}$, is always punctured, in this case the BER performance is not compromised as $X_t^{'S}$ can be reconstructed using an interleaver and the systematic data from RSC1, X_t^S. In Figure 7, X_t^{P0} represents parity data output by RSC1 and X_t^{P1} represents data output by RSC2.

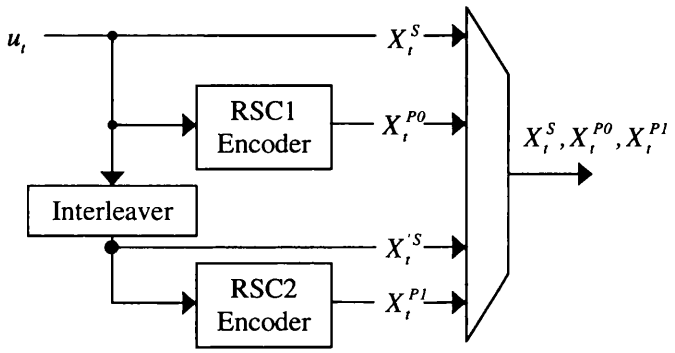

Figure 7: Top-level view of a Turbo encoder

A Turbo encoder is created from three components, two recursive systematic convolutional encoders (RSCs), discussed in Section 6.2.1, and an interleaver, discussed in Section 6.2.2. Each RSC is known as a component encoder.

A Turbo decoder is created using four components, two soft-in soft-out decoders, an interleaver and a de-interleaver. The top-level of a Turbo decoder is shown in Figure 8.

For i iterations each SISO shares its output data with the preceeding decoder. Data output from the SISO decoder, $L^e_{ab}(u_t)$, is the SISO estimate of the original data input to the corresponding Turbo encoder and is known as extrinsic data. As (7) will show, the systematic input to the decoder, Y_t^S, is contained in the natural output of the SISO decoder, $L(u_k)$. Therefore, in a real hardware implementation it is unlikely that only $L^e_{ab}(u_t)$ will be output by the SISO decoder, the actual value output will be $L^e_{ab}(u_t)+Y_t^S$. However, the SISO in Figure 8 shows only $L^e_{ab}(u_t)$ being output for clarity. In Figure 8, a λ subscript indicates that the associated variable is in interleaved form, relative to the original input data sequence. Y_t^{P0} represents the noisy version of the data output by RSC1 in Figure 7, $Y_{t\lambda}^{P1'}$ represents the noisy version of the data output by RSC2, After i iterations, the components shown as dashed lines are initialized. The hard decision (HD) block is a basic thresholder that compares the summed input to 0. If the input is greater than 0 the decoder estimate, $\hat{u}_k$, is a binary 1. If the summed input is less than 0 the decoder is estimate is a binary 0. If the summed input is exactly 0 there is the same probability that the output should be a 0 or a 1. In this case the system designer must decide what the output should be. This could involve an additional algorithm, i.e. alternating the output for a 0 input between 0 and 1 or simply making the output always be 1 or 0 when a 0 input is received.

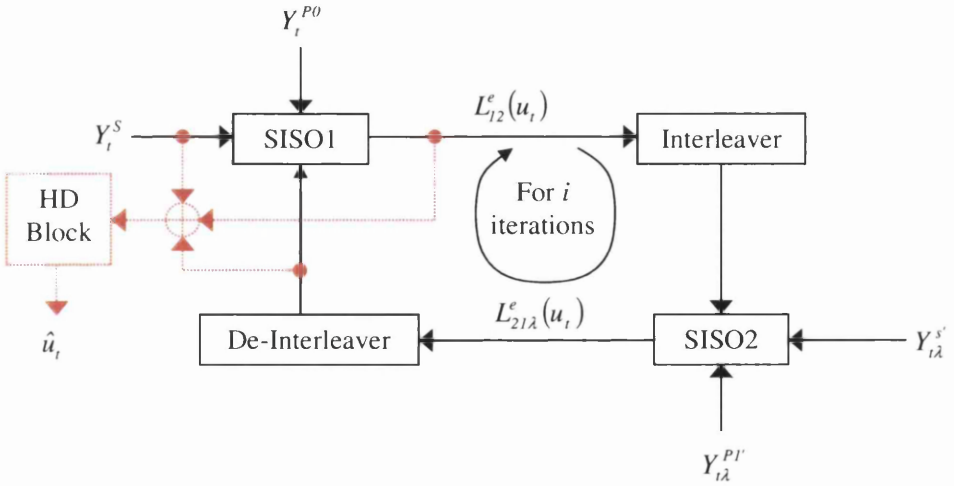

Figure 8: Top-level view of a Turbo decoder

Sharing of information between each SISO decoder is a major contribution to the performance of Turbo codes. The Turbo decoder and its components are discussed more comprehensively in Section 6.2.3.

6.2.1. Turbo Encoder

The input to Turbo encoder comes from a data set of size N, known as the block size. Traditionally, the input to the encoder is 1 bit wide. However, systems such as the DVB Turbo encoder accept multiple bit inputs. As Figure 7 shows the systematic output from RSC 2 is not transmitted. This is achieved by puncturing the Turbo encoder output. Puncturing the output of a Turbo encoder involves deleting certain bits according to a puncturing pattern.

The component RSC encoder for the UMTS standard is shown in Figure 9.

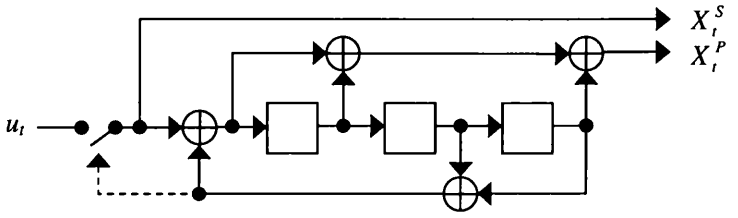

Figure 9: UMTS component encoder

As each UMTS RSC encoder outputs only one parity stream the UMTS Turbo encoder has a standard code rate of $\frac{1}{3}$.

A cdma2000 component RSC encoder is shown in Figure 10.

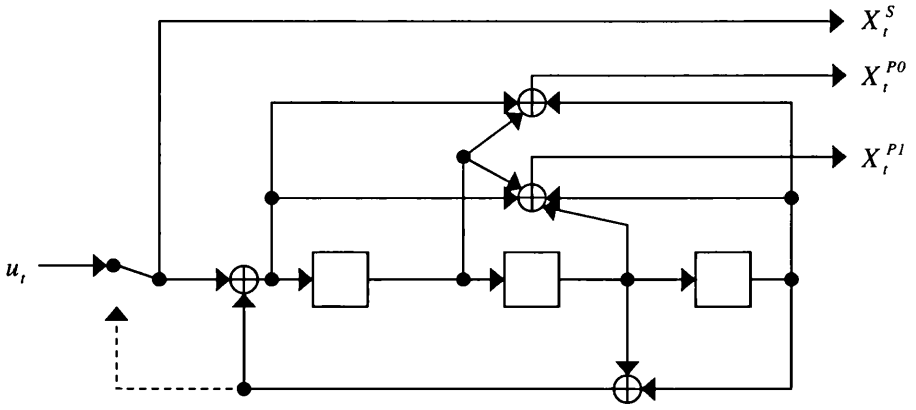

Figure 10: cdma2000 component encoder

The cdma2000 RSC encoder has two parity stream outputs and can therefore have a code rate of $\frac{1}{2}$, $\frac{1}{3}$, $\frac{1}{4}$ or $\frac{1}{5}$. The puncturing pattern of the cdma2000 standard is shown in Table 3.

Table 3: cdma2000 puncturing patterns for data

	Code Rate			
Output	$\frac{1}{2}$	$\frac{1}{3}$	$\frac{1}{4}$	$\frac{1}{5}$
X_t^S	11	11	11	11
X_t^{P0}	10	11	11	11
X_t^{P1}	00	00	10	11
$X_t'^S$	00	00	00	00
$X_t'^{P0}$	01	11	01	11
$X_t'^{P1}$	00	00	11	11

In the puncturing table a 0 represents a bit that is deleted and a non-zero number shows how many times the symbol in question is transmitted. For example, if data is being transmitted at a code rate of $\frac{1}{2}$ the output from the encoder will be X_t^S, X_t^{P0}, X_t^S, $X_t'^{P0}$. When the trellis termination is initialised the puncturing pattern is altered so that $X_t'^S$ is transmitted, certain bits are repeated so that the code rate during trellis termination matches the code rate requested by the user. The puncturing pattern for trellis termination is shown in Table 4.

Table 4: cdma2000 puncturing patterns for trellis termination

	Code Rate			
Output	$\frac{1}{2}$	$\frac{1}{3}$	$\frac{1}{4}$	$\frac{1}{5}$
X_t^S	111 000	222 000	222 000	333 000
X_t^{P0}	111 000	111 000	111 000	111 000
X_t^{P1}	000 000	000 000	111 000	111 000
$X_t'^S$	000 111	000 222	000 222	000 333
$X_t'^{P0}$	000 111	000 111	000 111	000 111
$X_t'^{P1}$	000 000	000 000	000 111	000 111

The DVB component RSC encoder is fundamentally different from the UMTS and cdma2000 RSC component encoders as it accepts two binary inputs for each time period, *t*,

highlighted in Figure 11. Appendix E {vol. 2/pp. 41-50}, gives an in-depth overview on the subject of duo-binary Turbo codes.

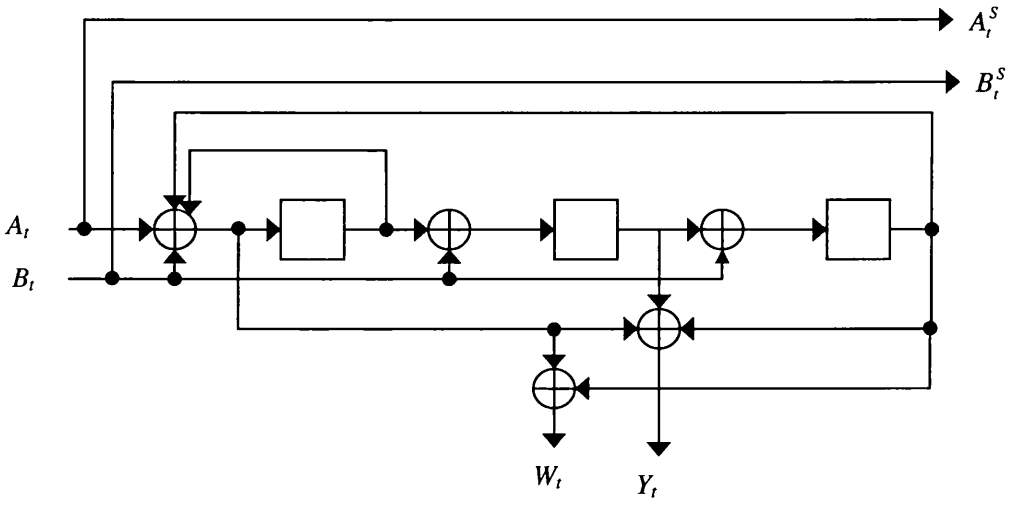

Figure 11: DVB component encoder

The DVB standard Turbo encoder can output data in any one of seven code rates. These code rates are shown in Table 5.

Table 5: DVB standard puncturing patterns

	Code Rate						
Output	$\frac{1}{3}$	$\frac{2}{5}$	$\frac{1}{2}$	$\frac{2}{3}$	$\frac{3}{4}$	$\frac{4}{5}$	$\frac{6}{7}$
Y_t	1	11	1	10	100	1000	100000
W_t	1	10	0	00	000	0000	000000

6.2.2. Interleaver

The interleaver improves the performance of the Turbo codec as it reduces the correlation between the data entering RSC1 and the data entering RSC2. By interleaving the data entering RSC2 it also protects the encoded data from burst errors. The input data to an interleaver is assigned an address. This address is then changed using a specific algorithm. A block diagram of the address generator for the cdma2000 interleaver is shown in Figure 12.

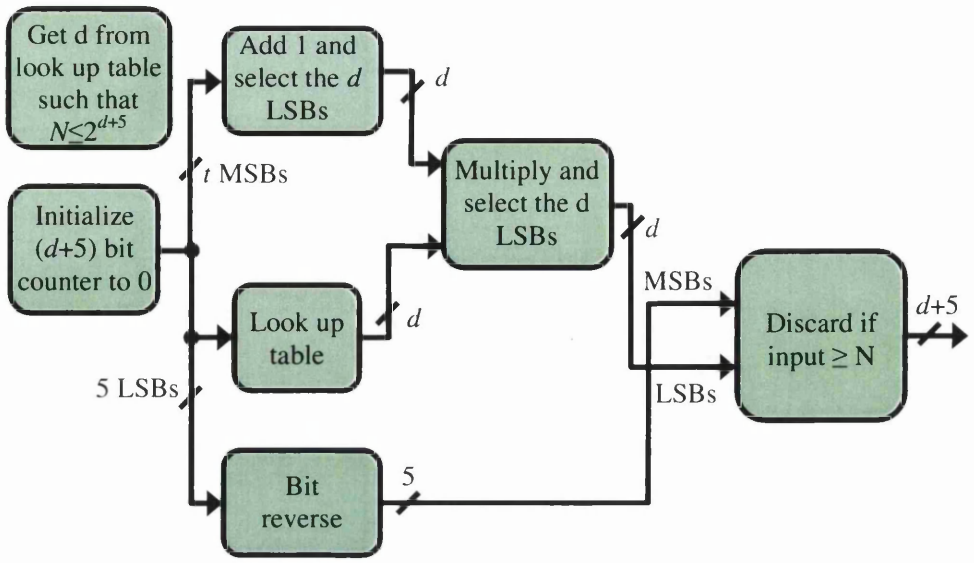

Figure 12: cdma2000 standard interleaver

If the address generated is greater than the block size of the current input data set then it is discarded. The components in the cdma2000 interleaver are easily implementable in hardware as they contain common mathematical operations such as counters, multipliers and bit reversals. The UMTS standard interleaver [2] is more mathematically complex and could be considered more difficult to implement in hardware.

6.2.3. Turbo Decoder

The key component of the Turbo decoder is the soft-in soft-out (SISO) decoder. SISO decoders, as their name suggests, accept soft value inputs and give soft value outputs. Soft values are j bits wide; the j bits represent a fractional number, an example of a decoder that accepts soft inputs is the Viterbi decoder [35]. One disadvantage of the traditional Viterbi decoder is that it only outputs single bits. By using soft inputs and soft outputs the BER performance of the decoder is improved. The main algorithms used to implement SISO decoders are the soft output Viterbi algorithm (SOVA) [36] and the Maximum a Posteriori (MAP) [37]. Using the MAP algorithm results in a better BER performance compared with the SOVA algorithm [38]. The MAP algorithm produces the most probable information bit per time instance for a given data set, the Viterbi algorithm produces a maximum likelihood sequence for a given data set. By performing a forward and backward traversal of the

trellis, the MAP algorithm produces two probabilities for the same time instance. This improves the BER performance of the Turbo decoder.

The Turbo decoders tested during the RE's industrial project were designed using variations of the MAP algorithm. Therefore, only this algorithm and its variants are described in this portfolio thesis. The variants described are the log-MAP algorithm, the Max-log-MAP algorithm and the Max Scale algorithm. All of these algorithms offer varying levels of performance and use varying levels of resources on the FPGA when implemented. Usually, as resources increase the BER performance of the implemented algorithm improves.

The soft inputs to the MAP algorithm are input in the form of a log-likelihood ratio, $L(u_t)$. A log-likelihood ratio shows the likelihood that a soft value represents a binary 1 or 0. Equation 6 shows how a log-likelihood ratio is calculated, where y_t represents the data received from the communications system demodulator.

$$L(u_t) = ln\left[\frac{P(u_t = 1 \mid y_t)}{P(u_t = 0 \mid y_t)}\right] \qquad (6)$$

If the result of (6) is positive then it is most likely that the result should be a binary 1, if the result is negative it is most likely that the result should be a binary 0. The probability that the result should be 0 or 1 increases as the magnitude of the result increases. If the result of (6) is exactly 0 then it is equally likely that the result is 0 or 1.

Ryan [39] shows that the output from a MAP decoder consists of three elements, shown in (7).

$$L(u_t) = L^e_{IN}(u_t) + L^e_{OUT}(u_t) + L_C Y^S_t \qquad (7)$$

The variable L_C is dependent on the variance of the channel, $L^e_{IN}(u_t)$ represents extrinsic data input to the SISO decoder, e.g. $L^e_{12}(u_t)$ into SISO2 in Figure 8. $L^e_{OUT}(u_t)$ represents the extrinsic output from the SISO Decoder, e.g. $L^e_{21\lambda}(u_t)$ out of SISO2 in Figure 8.

The three main variables used to calculate the outputs from the MAP decoder are $\alpha_t(s)$, the alpha probability of state s at time t, $\beta_t(s)$, the beta probability of state s at time t, and $\gamma_t(s', s)$, the branch probability. The alpha probability is produced by performing a forward trace of the trellis, the beta probability is produced by a backward trace of the trellis.

The branch probability is the probability that the trellis moves from state *s'* at time *t-1* to state *s* at time *t*. Ryan [39] shows that the branch metric can be calculated using (8).

$$\gamma_t(s',s) = exp\left[\tfrac{1}{2} X_t^S \left(L_{IN}^e(u_t) + L_C Y_t^S\right) + L_C\left(X_t^{P0} Y_t^{P0} + X_t^{P1} Y_t^{P1}\right)\right] \qquad (8)$$

For a standard Turbo decoder each state in the trellis has two branches entering it. This means that each state is associated with two alpha probabilities and two beta probabilities, all four probabilities which have a branch probability associated with them. Equation (9) shows how $\alpha_t(s)$ is calculated in a forward trace, this is shown graphically in Figure 13. The probability that the trellis is in state *s* at time *t-1* is equal to the probability that the previous state was *s'* and the trellis travelled along branch $\gamma_t(s', s)$ to get to state *s*. As Figure 13 shows each branch is associated with either a 0 or a 1. If a branch is associated with a 0 then the state it comes from will have a negative α probability, similarly a branch associated with a 1 will have come from a state with a positive alpha probability. Consequently (9) will yield either a positive or negative α depending on the magnitude of the two previous states.

$$\alpha_t(s) = \sum_{s'} \alpha_{t-1}(s')\gamma_t(s',s) \qquad t=1, \ldots, N \qquad (9)$$

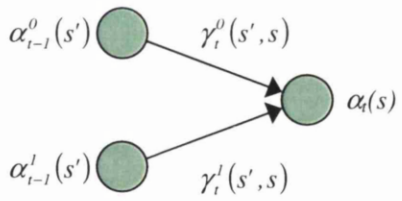

Figure 13: Calculating alpha probability

The beta probability is calculated in a similar fashion to the α probability, except it is calculated when a backward trace is performed. Figure 14 shows graphically how the beta probability is calculated. The actual equation performed is shown in (10). The state, *s*, at time *t-1* has two probabilities associated with it. The probability that the state at time *t* was $\beta_t^1(s')$ and the trellis travelled along branch $\gamma_t^1(s',s)$ and the probability that the state at time *t* was $\beta_t^0(s')$ and the trellis travelled along branch $\gamma_t^0(s',s)$. Whichever of the two probabilities is most likely will have the largest magnitude and therefore the output of (10) will be positive or negative depending on the result of (10).

$$\beta_{t-1}(s) = \sum_{s} \beta_t(s') \times \gamma_t(s',s) \qquad t = N, \ldots, 2 \qquad (10)$$

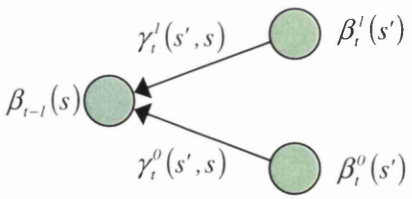

Figure 14: Calculating beta probability

As Figure 8 shows the output of the decoder after *i* iterations is given by (11).

$$L(u_t) = L_{12}^e(u_t) + L_{21}^e(u_t) + L_c y_t^S(u_t) \tag{11}$$

In terms of $\alpha_k(s)$, $\beta_k(s)$ and $\gamma_k(s', s)$ the output from the decoder is calculated as shown in (12).

$$L(u_t) = \frac{\sum_{\substack{(s',s)\\ u_t=+1}} \left(\alpha_{t-1}(s') \times \gamma_t^{1XX}(s',s) \times \beta_t(s)\right)}{\sum_{\substack{(s',s)\\ u_t=-1}} \left(\alpha_{t-1}(s') \times \gamma_t^{0XX}(s',s) \times \beta_t(s)\right)} \tag{12}$$

The MAP algorithm gives exceptional BER results at relatively low $\frac{E_b}{N_0}$, however, the MAP algorithm is extremely difficult to implement in hardware as it contains mathematical operations such as logarithms and divisions which are costly in terms of hardware resources. A modification of the MAP algorithm is the log-MAP algorithm [40]. As the name suggests the log-MAP algorithm is an implementation of the MAP algorithm in the logarithm domain. Taking the logarithm of (8), (9) and (10) yields (13), (14) and (15).

$$A_t(s) = ln(\alpha_t(s)) \tag{13}$$

$$B_t(s) = ln(\beta_t(s)) \tag{14}$$

$$G_t(s',s) = ln(\gamma_t(s',s))$$

$$G_t(s',s) = \left(\tfrac{1}{2} X_t^S \left(L_{IN}^e(u_t) + L_c y_t^S\right)\right) + \left(\tfrac{1}{2} L_c \left(y_t^{P0} X_t^{P0} + y_t^{P1} X_t^{P1}\right)\right) \tag{15}$$

The Jacobian logarithm, (16), can be used to obtain a formula for both (13) and (14) that can be easily implemented in hardware, shown in (17) and (18).

$$MAX*\left(A_t^0, A_t^1\right) = ln\left(e^{A_t^0} + e^{A_t^1}\right)$$

$$MAX*\left(A_t^0, A_t^1\right) = MAX\left(A_t^0, A_t^1\right) + ln\left(1 + e^{-|A_t^0 - A_t^1|}\right) \tag{16}$$

$$A_t(s) = \underset{s'}{MAX*}\left(A_{t-1}(s') + G_t(s',s)\right) \qquad t = 1, \ldots, N\text{-}1 \tag{17}$$

$$B_{t-1}(s') = \underset{s}{MAX}\,^{*}\left(B_t(s) + G_t(s',s)\right) \qquad t = N, \ldots, 2 \qquad (18)$$

Similarly the output from the decoder can also be calculated using the Jacobian logarithm, shown in (19).

$$L(u_t) = \underset{\substack{(s',s)\\ u_t=+1}}{MAX}\,^{*}\left(A_{t-1}(s') + G_t(s',s) + B_t(s)\right) - \underset{\substack{(s',s)\\ u_t=-1}}{MAX}\,^{*}\left(A_{t-1}(s') + G_t(s',s) + B_t(s)\right) \qquad (19)$$

The Max* computation involves adding both the alpha or beta metrics to the branch metric they are associated with, then comparing both of the summed metrics. The largest is selected and the offset is added from a look up table. This output is then passed to the next state in the trellis diagram. The add, compare, select, offset component is at the heart of all of the MAX* computations, it is shown graphically in Figure 15 [41]. One ACSO unit is needed for every state in the trellis diagram.

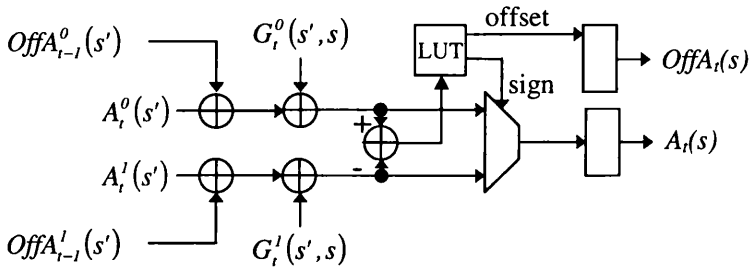

Figure 15: Add, compare, select, offset unit

The log-MAP algorithm can be further simplified by excluding the offset, $ln\left(1 + e^{-|A^0_t - A^1_t|}\right)$, in (16), resulting in an implementation called the Max-log-MAP algorithm. Chapter 7 will show that removing the offset results in a relatively poor BER performance. A compromise between the log-MAP and Max-log-MAP algorithm is the Max Scale algorithm. The Max Scale algorithm has a similar complexity to the Max-log-MAP algorithm but achieves a coding gain of between 0·2dB and 0·4dB compared with the Max-log-MAP algorithm [42]. The coding gain is achieved by multiplying the SISO decoder output by a scaling factor, *sf*, shown in (20).

$$L^e_{OUT}(u_t) = \left[L(u_t) - \left(L^e_{IN}(u_t) + L_c y^S_t\right)\right] \times sf \qquad (20)$$

6.3. Field Programmable Gate Arrays

Field Programmable Gate Arrays (FPGAs) are off the shelf hardware parts that offer a compromise between the speed of ASICs and the reconfigurability of DSPs. There are three main basic elements in FPGAs, these are: configurable logic blocks (CLBs); input output blocks (IOBs) and routing. The basic structure of an FPGA is shown in Figure 16.

In the Xilinx Virtex-II FPGA each CLB contains 4 slices. Each slice contains two 4-input look-up tables (LUTs), two registers and some combinatorial logic, used to implement fast carry chains, and two multiplexers. The resources used by any given design are measured in slices.

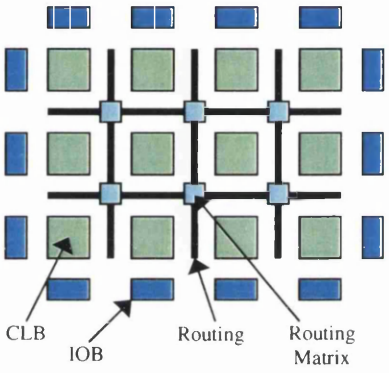

Figure 16: FPGA basic structure

The routing matrix is programmable, allowing the user to change routing for each new design downloaded to the FPGA. As has been stated previously, FPGAs offer a speed advantage over DSPs and a reconfigurability advantage over ASICs. However, both DSPs and ASICs are smaller and consume less power than FPGAs. To compete with DSPs FPGAs now contain elements such as memory and embedded multipliers. The Xilinx Virtex-II FPGA [26] has up to 168 18bit x 18bit embedded multipliers (EMults), along with 168 18Kbit block RAMs (BRAMs). The Xilinx Virtex-II Pro [43] extended the programmability of Xilinx FPGAs by including up to 2 embedded PowerPC processors. Adding these extra components allows FPGAs to outperform DSPs in applications that require intensive computing.

It is well known that the development time of FPGA design is significantly shorter than ASICs. However, DSPs have a shorter design development time than FPGAs. This is

mainly due to the proliferation of tools available to DSP designers and the fact that DSPs can be programmed using high level languages. To combat this Xilinx introduced their System Generator tool. System Generator is a Matlab Simulink add-on that allows FPGAs to be designed using a simple schematic capture environment using a flow that DSP designers are familiar with. The System Generator design flow is shown in Figure 17 [44]. System Generator includes various library components that can be used for DSP design. These vary in complexity from simple logic gates to a Viterbi decoder. Library components are designed in low level VHDL so are highly optimised for implementation in Xilinx FPGAs.

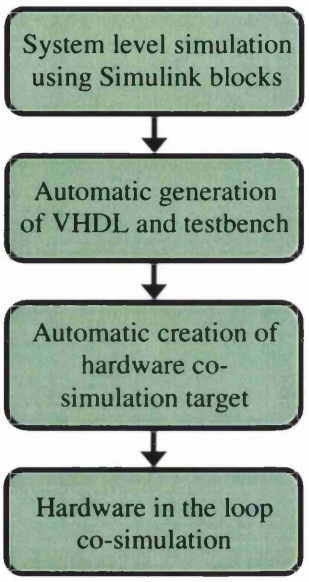

Figure 17: System Generator DSP design flow

The System Generator design flow can also be presented in a fashion that is familiar to hardware design engineers, as shown in Figure 18 [15].

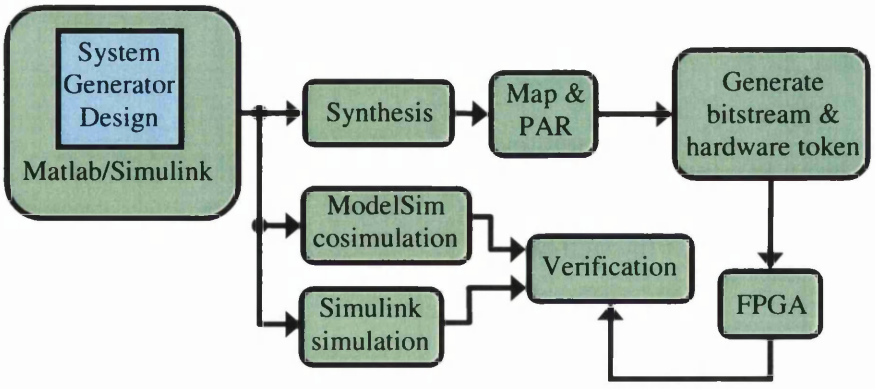

Figure 18: System Generator hardware design flow

This shows the flexibility of System Generator. While System Generator may not allow hardware engineers to fine tune designs in the same way a hardware description language (HDL) does, it does allow hardware and systems engineers to work together on the same design using a common platform, improving productivity.

6.4. Third Generation Mobile Technology

The main 3G standards are based around spread spectrum technology [45, 46, 47]. Spread spectrum uses spreading sequences such as Walsh codes and Gold codes to distinguish between different users. This is required as all users within a cell share both time and frequency. Figure 19 shows an example of the spreading process. Each user in a cell is assigned a specific chip sequence. The base station controlling a cell stores all chip sequences assigned to each user and uses a correlator to determine which user is transmitting to the base station. The chip rate for the cdma2000 standard is 3·6864 Mchips/sec[1], UMTS has a chip rate of 3·84 Mchips/sec. One other difference between the two standards is that the cdma2000 standard is synchronised with a GPS clock while the UMTS standard is completely asynchronous. This makes the UMTS receiver more complex, however the cdma2000 receiver is dependent on government satellites for the required GPS information. The cdma2000 standard is to be implemented in the United States, the UMTS standard will be used in Europe. One advantage of deploying a spread spectrum system in the US is that previous 2G mobile standards used spread spectrum systems. Whereas in Europe, the previous 2G system uses the GSM standard. Meaning, handover in the US cdma2000 system is simpler than in the European UMTS system. This is because previous US standards also used spread spectrum systems such as cdmaOne, whereas previous European systems have been based on the non spread spectrum GSM standard.

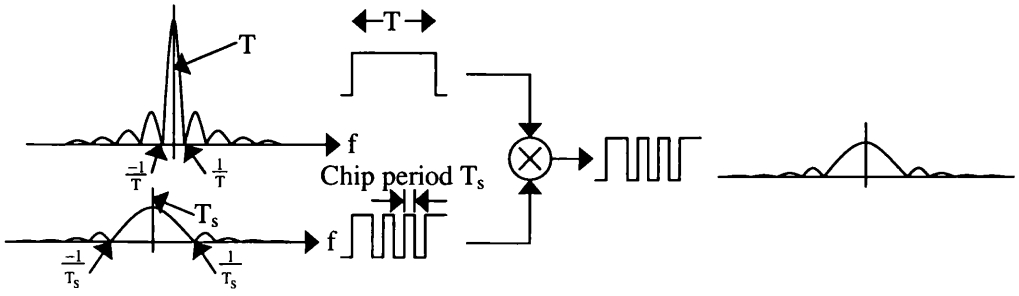

Figure 19: Spread spectrum signal

[1] Actual chip period used is usually 1·2288Mchip/sec, to match IS-95 standard.

The multipath delays inherent to mobile channels can cause problems in spread spectrum systems as they reduce orthogonality. Figure 20 shows a multipath channel between a user and a base station, Figure 21 shows the affect this has on the signal received at the base station.

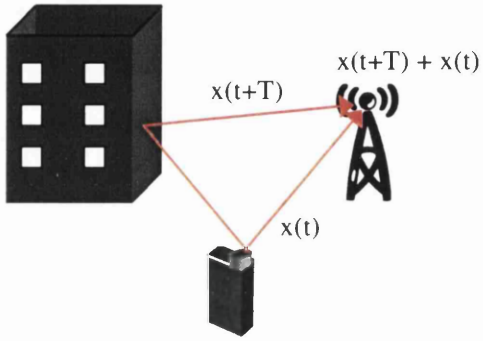

Figure 20: Multipath channel

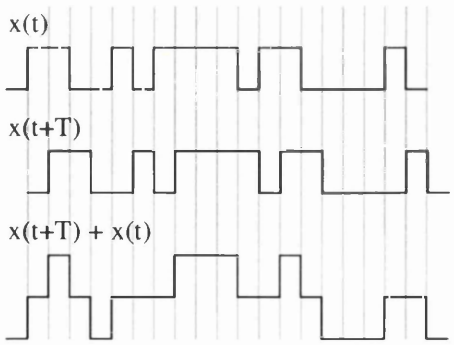

Figure 21: Signal received at base station

A simple solution could be to use the strongest signal received from each user. However, to obtain the best possible signal the base station must extract all signals received from each user. This is achieved in spread spectrum systems by using a rake receiver [48] at the base station. A pilot sequence is used to determine the characteristics of the channel, the pilot sequence is known by both the mobile and the base station. This pilot sequence can be used by the rake receiver to determine the different components of a user's transmitted signal. A rake receiver is shown in Figure 22, where Δ_x represents the delay between each received signal.

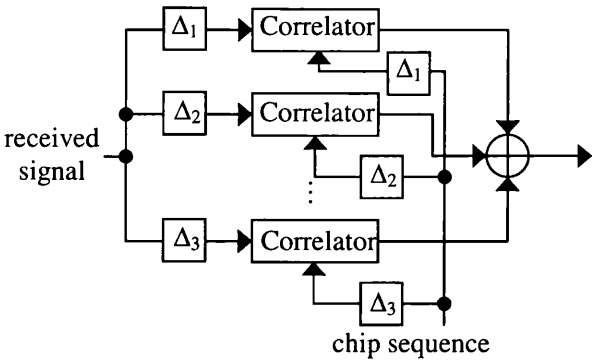

Figure 22: Rake receiver

Figure 23 shows a typical 3G system architecture, highlighting where channel coding takes place within the transport channel. Turbo codes are used in 3G for coding data traffic, the inherent latency associated with the recursive Turbo decoder means currently they are unlikely to be used for voice traffic.

As Figure 23 shows multiple users data are transmitted by one base station, each user is assigned a channel to transmit on. The ability of FPGAs to process multiple channels, or users, at once is one major advantage that FPGAs have over DSPs for implementing components required by base stations. This advantage is gained because of the parallel nature of FPGA architectures. Although the cost of an FPGA is significantly higher than the cost of a DSP, the cost per channel is significantly lower [24]. The savings of using an FPGA rather than a DSP are estimated to be $490 per channel [24]. Some DSPs now contain dedicated hardware to perform tasks such as Viterbi and Turbo decoding [49]. The disadvantage of using these hard wired components is that they cannot be reconfigured when a particular standard changes and become obsolete when a new standard needs to be implemented.

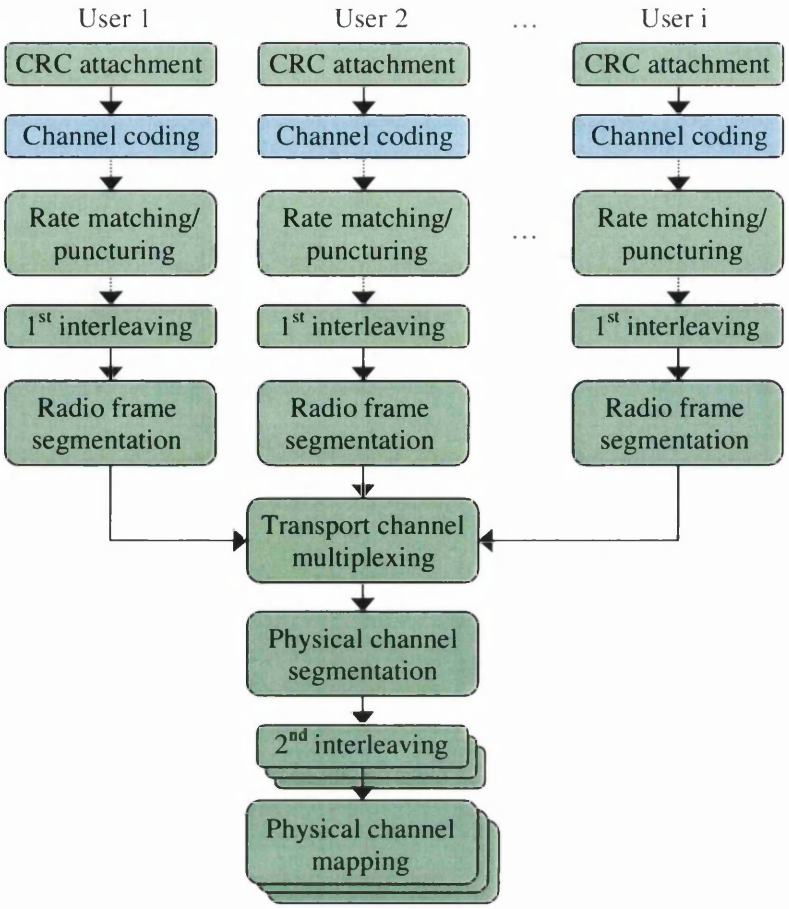

Figure 23: Transport channel of a typical 3G system

The changing nature of 3G standards is one reason for choosing an FPGA implementation over an ASIC implementation. One example of this was the fifth release of the UMTS standard which contained standards for High-speed Downlink Packet Access (HSDPA). It is estimated that future UMTS systems will transmit 80% of data using HSDPA [50]. Therefore, HSDPA will provide a lot of revenue for network providers. Any UMTS base station using a combination of ASICs and DSPs designed before release 5 would have to redesign their base station, again causing lost revenue. If a UMTS base station was designed using FPGAs the release 5 standard could be implemented by reconfiguring the FPGAs in the base station. A base station designed using FPGAs gives the network provider confidence that future standards can be adhered to without expensive re-designs. This is important as network standards are sure to evolve even when standards are set. Network providers will want to implement systems such as multiple-in, multiple-out

(MIMO) [51] when they become available to increase data rates and improve network efficiency.

One other important future development in mobile technology is software defined radio (SDR) [52]. The main idea behind SDR is that mobiles can be reconfigured depending upon the environment in which it is being used. For example, a user moving from a country using a UMTS standard to a country using a cdma2000 standard would have their handset reconfigured so that it could be used in a cdma2000 environment. When the user returns to a UMTS environment the handset would again be reconfigured. SDR could also be used for entertainment purposes, a handset could be reconfigured to play various types of media when required. Current ASICs cannot be used to implement SDR systems, FPGAs and DSPs can. However, as has already been mentioned, DSPs cannot compete with the speed of FPGAs and do not have the same parallel computing power that FPGAs have.

7. Results and Discussion

Results in this Chapter show how the cdma2000 and UMTS Turbo codecs perform when certain parameters and channel conditions are altered. All results discussed were produced using the System Generator BERT and in all cases an additive white Gaussian noise (AWGN) channel is used. Section 7.1 discusses the System Generator BERT and its novelty. Results produced by the BERT platform are then shown and discussed for both the cdma2000 and UMTS Turbo code standards. These results show how each of the standards perform when basic inputs such as block size, code rate and the number of iterations to be performed are altered. The UMTS Turbo decoder included a fast termination algorithm, this algorithm is presented and results obtained when using different fast termination thresholds (FTTs) are shown. The BERT was also used to produce novel BER results. One value that could be changed in the BERT platform was channel variance. The RE used this to find a novel channel variance value that optimised the cdma2000 Turbo decoder core. A parameterisable Turbo decoder is also presented, by altering the parameters of the decoder core the amount of FPGA resources used can be determined by the user. This is done by allowing the user to change parameters such as the SISO implementation algorithm, the sliding window size, the width of input data metric and the width of the internal data that represent the α, β and γ metrics. Results obtained when these parameters are changed and the impact on the resources used are discussed. Another novel result produced by the RE shows that the cdma2000 Turbo code standard outperforms the UMTS Turbo code standard. Reasons for the difference in performance are presented.

7.1. Bit Error Rate Test Platform

A top level view of the Bit Error Rate Test (BERT) Platform is shown in Figure 24. Random data from the BERT platform is input to the encoder under test. Encoded data is then passed to an additive white Gaussian noise channel, composed of a white Gaussian noise (WGN) generator and a soft converter. The soft converter converts a binary 0 to a -1 and a binary 1 to a +1. After the encoded data is soft converted the WGN block adds noise. The noisy data is then passed to the decoder. Decoded data is passed to the BER calculator along with the original input data to the encoder. The two values are compared and the BER can be calculated from this comparison, the result is then plotted by a Matlab script. A BERT is started by running a Matlab script, the script contains all information needed by

the BERT to run a test, i.e., the number of iterations to be performed, block size values and the code rate of the encoder. Figure 25 shows the procedure of running a test on the BERT.

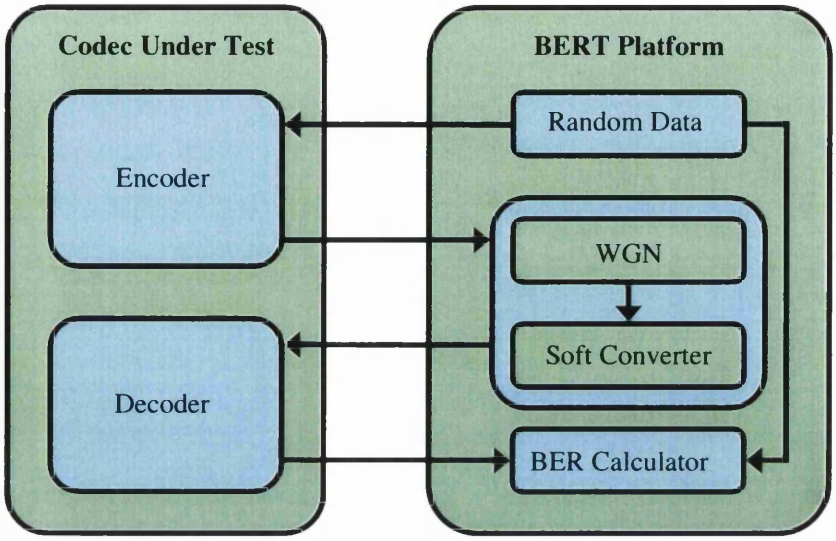

Figure 24: BERT Platform top-level

The Matlab script controlling the BERT platform can contain multiple tests, each test containing a different value for block size, number of iterations to be performed and code rate. The script automatically begins a test once the previous test has finished. If there are no more tests to be performed then the simulation is stopped. The results presented in this thesis were for two different Turbo codecs. However, the BERT platform could be adopted to test other codecs.

For a point to be plotted one of two conditions must be met. The number of errors detected must be equal to or greater than x, a value input by the user, or the decoder must have output y bits, again this value is determined by the user. If neither of these events occurs another packet of data, whose length is equal to the block size stated in the Matlab script is encoded. If either of the conditions is met the BER value to be plotted is checked against a minimum BER value, if the BER value to be plotted is below the minimum BER value it is discarded and the next test is executed. The minimum BER value is determined by the user. If the BER value to be plotted is greater than the minimum BER value it is plotted by a Matlab script, the $\frac{E_b}{N_0}$ value is then incremented and the process of plotting another point is restarted. If a point is plotted because y bits are received then another user defined input,

minimum errors, is used to ensure that the result to be plotted was within confidence intervals. If the Turbo decoder processes y bits, at least z of these bits have to be in error for a point to be plotted. If they are not the next test is executed without that point being plotted. Normally for statistical reliability z would be set to 100 bits [53], meaning that to meet a certain minimum BER y has to be chosen accordingly. For example, if z=100 and minimum BER=1×10^{-7} then y must be greater than 1×10^{9}.

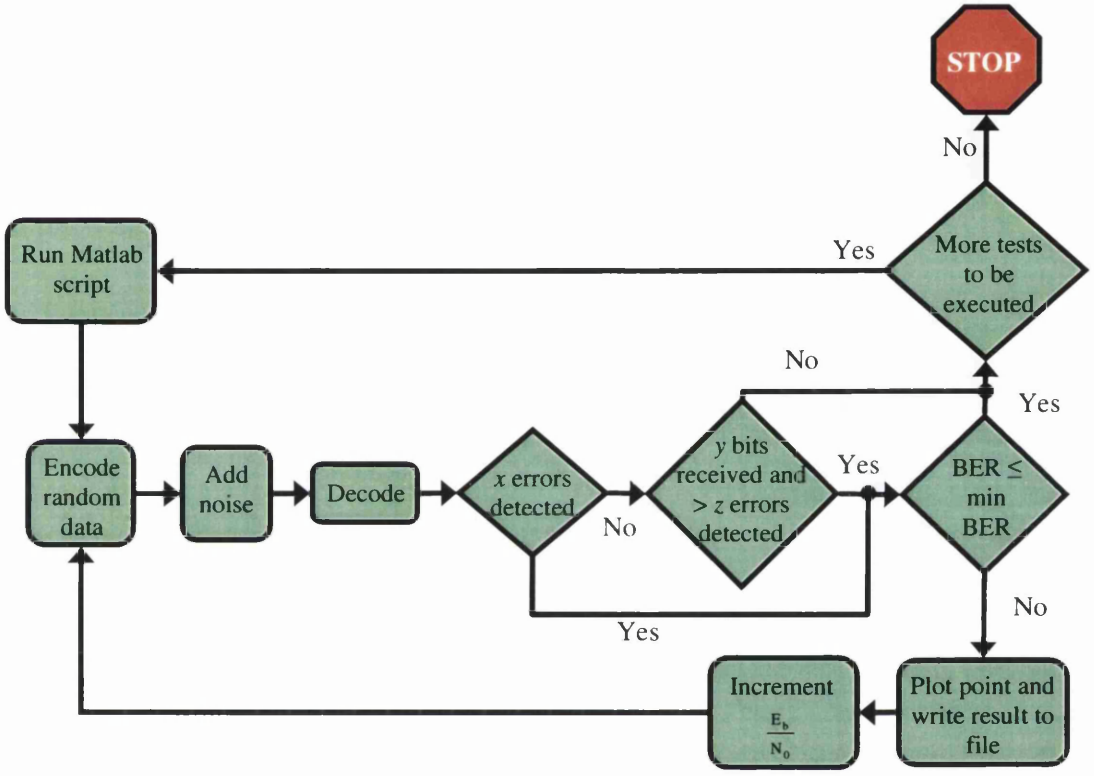

Figure 25: Procedure for running BERT

The BERT platform implemented in the System Generator tool is shown in Figure 26. The control logic in the BERT contains components that compares the encoder input with the decoder output and decides when a BER point should be plotted. The advantage of the BERT being implemented in System Generator is that it allows non-FPGA designers to use and upgrade a BERT platform in an FPGA environment without using standard FPGA design techniques such as HDLs. One example of this is the implementation of a puncturing block for both the cdma2000 and UMTS Turbo codecs. Neither of these cores originally contained a puncturing block. The RE created puncturing blocks for both the cdma2000 and UMTS Turbo codecs using System Generator.

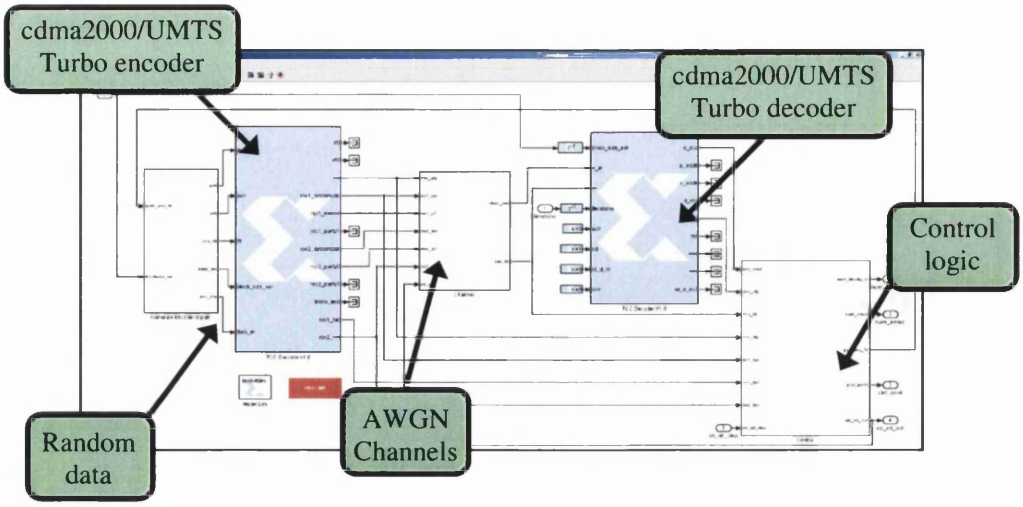

Figure 26: System Generator implementation of BERT platform

Although another BERT platform has been designed in System Generator [44], the BERT platform presented in this thesis is almost entirely user programmable whereas the BERT previously implemented in System generator was not. The following elements of the BERT platform are user programmable:

- Number of errors to be detected before a point is plotted
- Number of bits to be processed before a point is plotted
- Minimum BER value to be plotted
- Number of iterations to be performed
- Block size of data to be encoded
- Code rate of data to be encoded

Having a user programmable BERT has a two-fold advantage for the sponsoring company. It gives them a test platform that is tuneable to any FEC system. Also, having the ability to change the test parameters, e.g. number of errors to be detected before a point is plotted, highlights the reconfigurability of FPGAs.

Originally the BERT platform contained 5 AWGN channels, one for each encoder output in the cdma2000 standard. The smallest FPGA device the BERT platform plus the target Turbo codec could be implemented on was a Xilinx Virtex-II 3000. The Virtex-II 3000 contains a total of 14,336 slices. Although the BERT and Turbo codec only occupy around 3000 slices the number of block RAMs it requires means that it has to be implemented on

the Xilinx Virtex-II 3000. One advancement made by the RE was to reduce the number of channels in the BERT to one, this reduces the number of block RAMs required and means the BERT platform and Turbo codec being tested could be implemented on a Xilinx Virtex-II 2000 device. This is highlighted in Table 6 which shows the resources available on both the Virtex-II 2000 and the Virtex-II 3000 and the resources used by the BERT when implemented with five channels and with only one channel. When the BERT was being tested with five channels a flaw was discovered with the Xilinx Synthesis Tool (XST) that caused it to crash. The tool would crash because of the number of files generated by the BERT when five channels were included in the design. The only solution available at this time was to use a different synthesis tool. This flaw was not found when using only a single channel. Using the BERT with only one channel therefore had more advantages than just reducing the resources required. It also allowed Xilinx customers to take the BERT through the System Generator design flow using only Xilinx software tools, meaning Xilinx customers would not have to spend extra money on licenses for synthesis tools.

Table 6: Resources available and resources required for various BERT implementations

	Slices	**EMults**	**BRAMs**
Device Resources Available			
Virtex II 2000	10,752	56	56
Virtex II 3000	14,336	96	96
BERT Implementation Resources Required			
Single Channel	3,481	10	54
Five Channels	7,181	34	86

The main advantage of implementing the BERT platform in hardware instead of software was the speed up obtained. It was shown [15] showed that the speedup obtained when using a hardware implementation of the BERT platform gave a speedup of 17,500 over a software implementation of the BERT platform.

7.2. Block Size

When the block size of the data input to a Turbo encoder is increased the BER performance of the Turbo codec improves. This is shown in Figure 27 and Figure 28. The latter shows how BER performance improves as block size increases for the UMTS Turbo codec. Figure 27 shows how BER performance improves as block size increases for the cdma2000

Turbo codec. All BER curves in Figure 27 were produced using the Max Scale SISO algorithm, a code rate of $\frac{1}{5}$ and 5 iterations of the Turbo decoder.

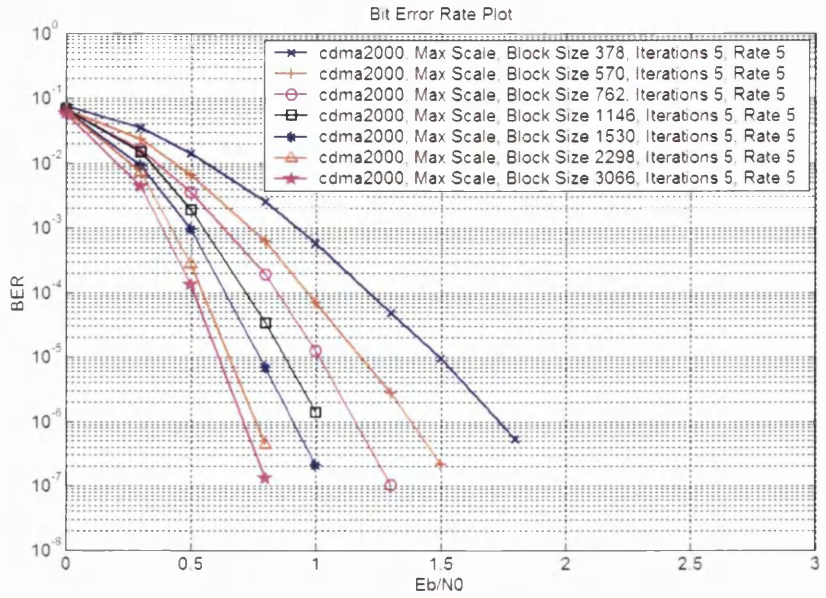

Figure 27: BER performance of cdma2000 Turbo codec as block size changes

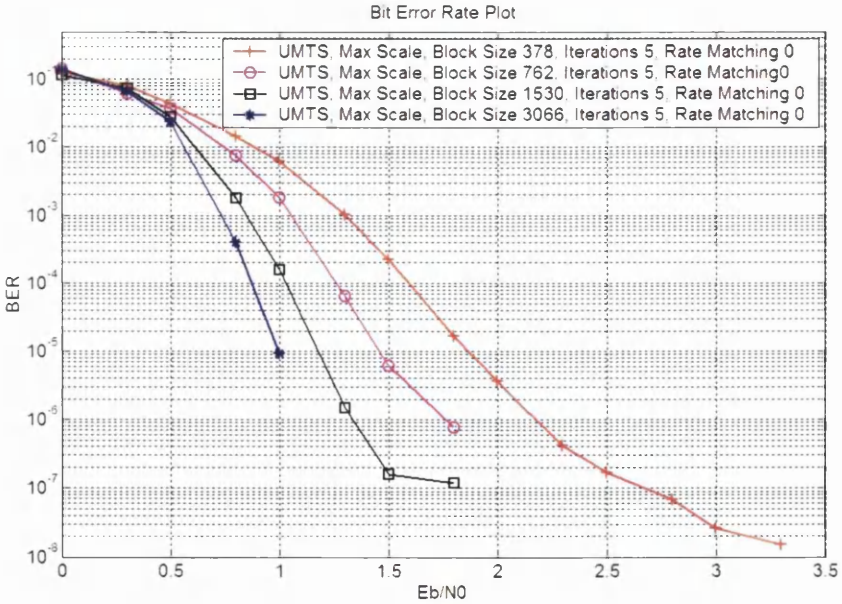

Figure 28 BER performance of UMTS Turbo codec as block size changes

BER performance improves as block size increases because of the interleaver in the Turbo encoder. As the number of bits input to the Turbo interleaver increases the correlation between the data input to RSC1 and RSC2 decreases. Decreasing the correlation of the RSC inputs allows each decoder to output a vastly different representation of the data set input to the Turbo encoder, allowing each SISO decoder to create a more independent estimate of the original input data set. Figure 28 shows the error floor for the UMTS Turbo decoder being reached, as will be shown in Section 7.6 the error floor for the cdma2000 Turbo code implementation is approached later than the UMTS standard Turbo code, leading to the cdma2000 standard having a slightly better performance than the UMTS standard.

7.3. Iterations

Increasing the number of iterations performed by the Turbo decoder improves the BER performance. Figure 29 and Figure 30 show how the number of iterations performed affect the BER performance of the cdma2000 and UMTS Turbo codecs respectively.

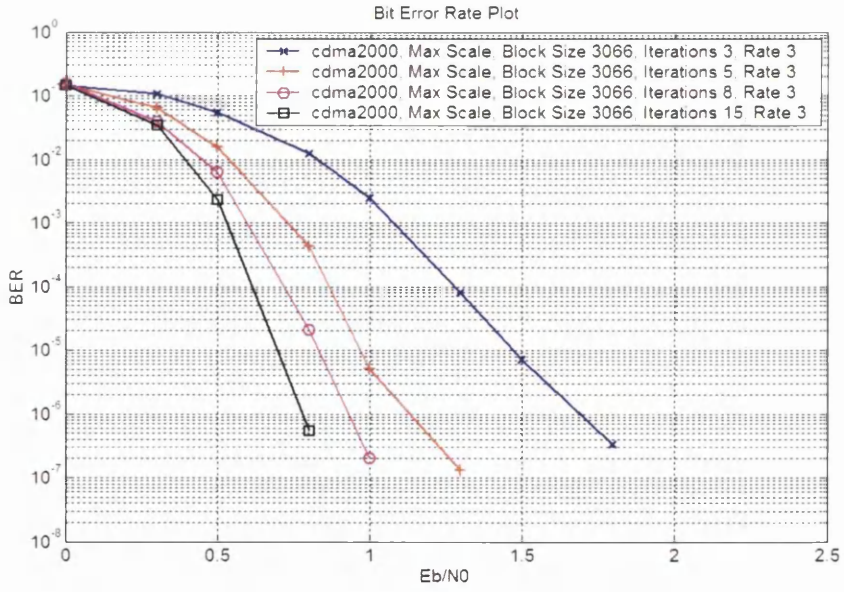

Figure 29: BER performance of cdma2000 Turbo codec as number of iterations performed increases

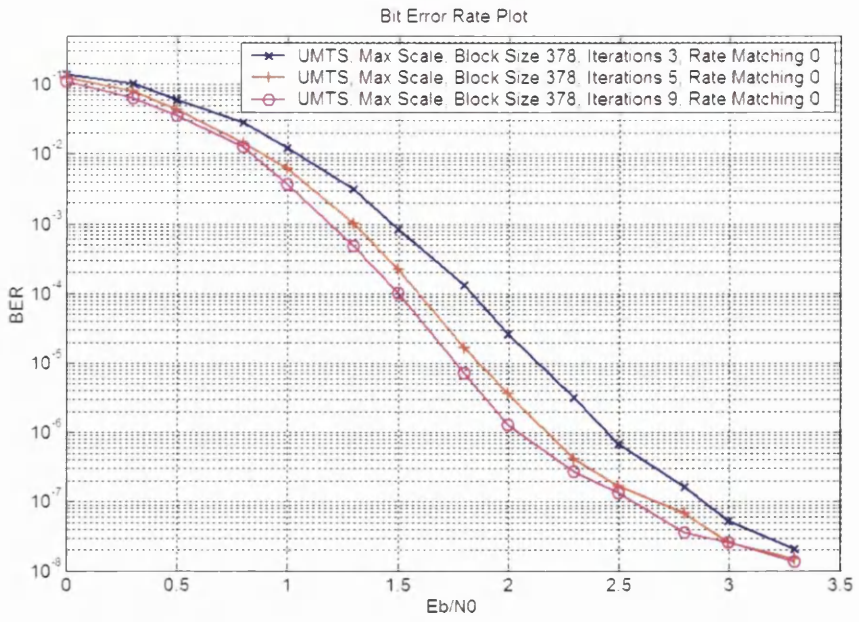

Figure 30: BER performance of UMTS Turbo codec as number of iterations performed increases

As the number of iterations increases the coding gain between each single increment decreases. This can be seen in Figure 29. The coding gain between the BER curves for iterations equal to 3 and 5 is 0·5dB at a BER of 1×10^{-6}. At the same BER the coding gain between iterations equal to 5 and 15 is only 0·4dB.

The UMTS Turbo codec also contained the option of including a fast termination algorithm. Fast termination allows the Turbo decoder to stop before the specified number of iterations to be performed is reached. There are a number of different algorithms available to achieve this [54, 55]. The algorithm implemented for the UMTS Turbo decoder [54] monitors hard outputs from the SISO decoders and is one of the simplest algorithms to implement. When a certain number of consecutive SISO decoder outputs are equal for a particular decoder input at time *t* the decoder outputs data. The number of consecutive SISO decoder outputs that have to be equal is determined by the user. The user can specify a maximum number of iterations, if this maximum is reached before a consecutive number of SISOs are determined to be equal the decoder begins to output data. The advantage of fast termination is that the average number of iterations is reduced, thereby reducing the average latency of the Turbo decoder. As the number of iterations is reduced the power consumed by the FPGA per decoder cycle is also reduced. Given that

power consumption is a major consideration for base station designers this is a relatively important result.

The FTT determines how many consecutive SISOs must be equal for a particular decoder input at time *t*. If the FTT is set to zero, fast termination is switched off. If the FTT is set to one, two consecutive SISO decoder outputs must be equal before the decoded data is output. If the FTT is set to two, three consecutive SISO decoder outputs must be equal before the decoded data is output. This process continues up to a maximum FTT of seven, in this instance eight consecutive SISO decoder outputs must be equal before the decoded data is output. Figure 31 shows a BER plot for different FTTs. Figure 32 shows the average number of iterations performed for each FTT. The disadvantage of using a low FTT is shown between $\frac{E_b}{N_0}$ values 1·5dB and 2·25dB in Figure 31. At these $\frac{E_b}{N_0}$ values the curve with a FTT of 3 outperforms the FTT of 1. The main advantage of using a low FTT is that the average number of iterations is lower with respect to higher FTTs. Up until 1dB the BER curves are relatively similar due to the average number of iterations being performed being relatively close together. After 1dB, Figure 31 shows that the two BER curves start to diverge slightly. The divergence is most obvious above 1·5dB.

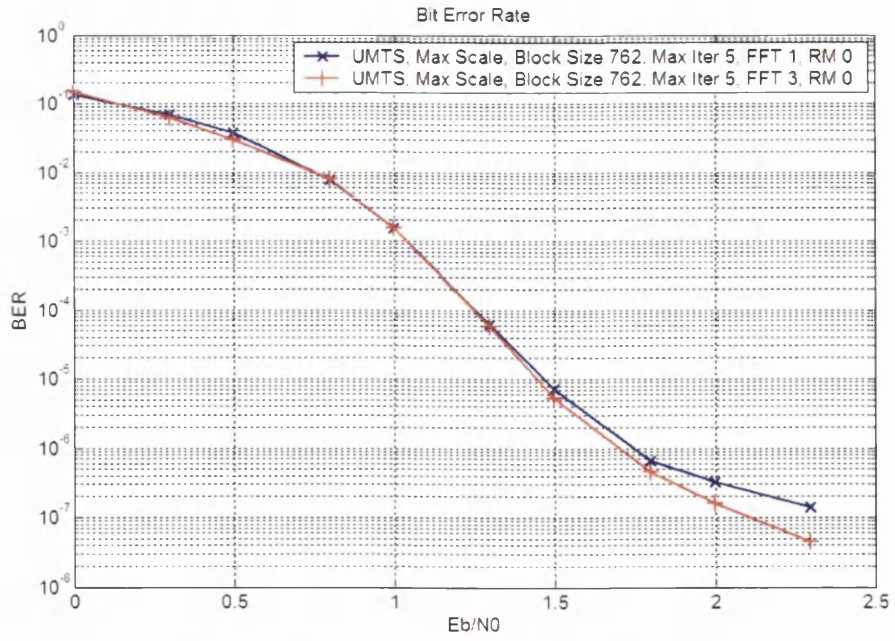

Figure 31: BER plot of different fast termination thresholds

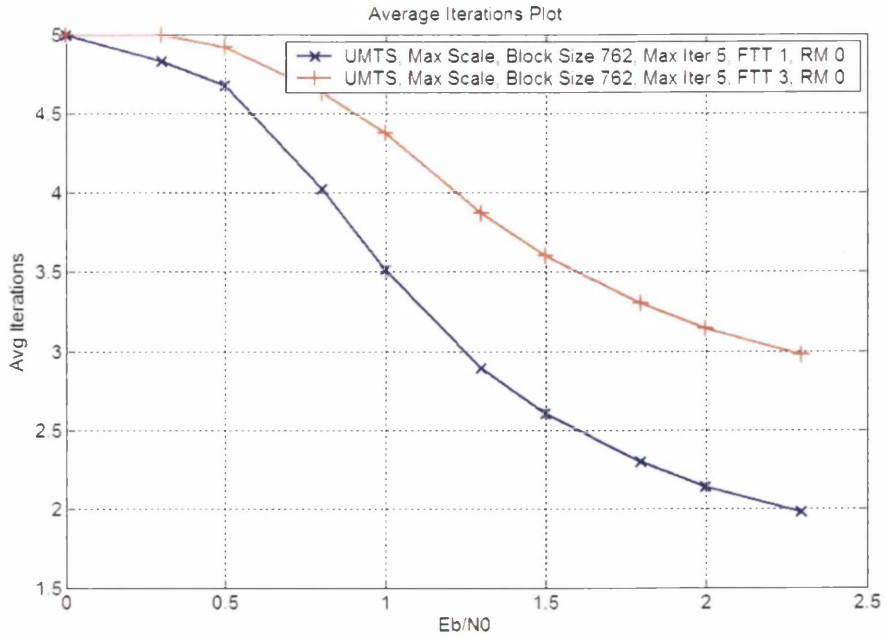

Figure 32: Average iterations plot for different fast termination thresholds

The fast termination algorithm implemented uses hard outputs from each SISO decoder. As each SISO decoder is initially not configured to output hard values the Turbo decoder structure must be changed. The Turbo decoder structure for implementing the particular fast termination algorithm described in this section is shown in Figure 33.

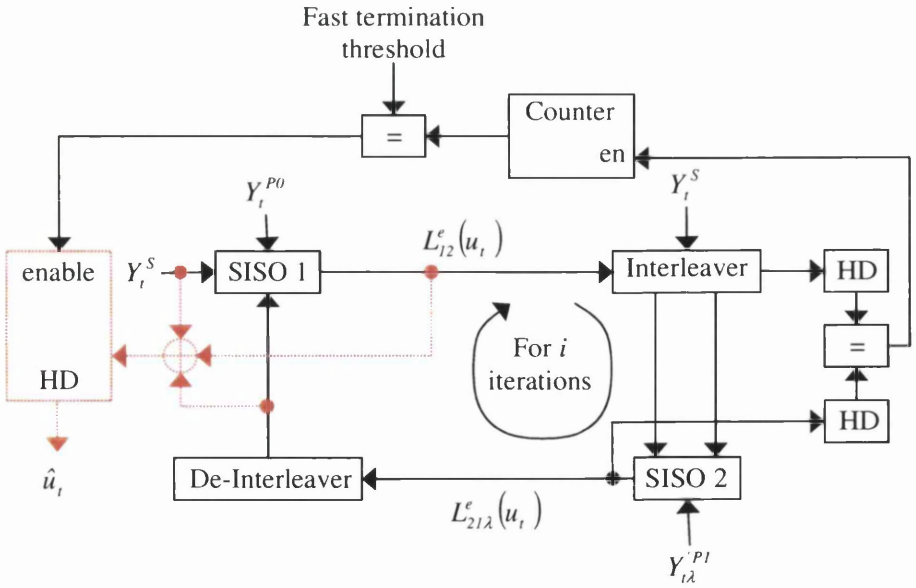

Figure 33: Turbo decoder with fast termination implemented

The extra resources required for the Turbo decoder with fast termination is one block RAM and approximately thirty slices.

7.4. Code Rate

Decreasing the Turbo encoder code rate has the advantage of improving BER performance, as shown in Figure 34. However, it also increases the amount of redundant information sent by the Turbo encoder. The cdma2000 standard Turbo encoder's puncturing pattern was discussed in Section 6.2.1. Results obtained when changing the code rate in the cdma2000 Turbo codec are shown in Figure 34. The drop in performance is due to the punctured bits being replaced with an all zero codeword at the decoder input.

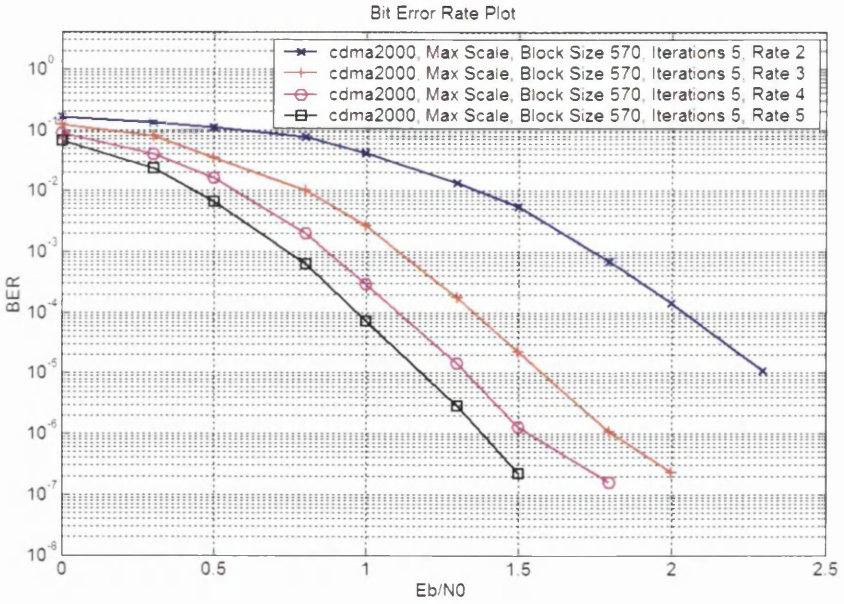

Figure 34: Effect of using different code rates on BER performance in cdma2000 Turbo codec

7.5. Parameterisable Turbo Decoder

The object of the novel parameterisable Turbo decoder is to offer the user of the decoder various options that they can use to balance performance and the resources used by the FPGA to suit their particular need. The parameterisable options available are:

- Input data width: the input to the Turbo decoder can be anywhere between 3 and 7 bits in length. The integer part of this input can be either 2 or 3 bits. The fractional part can be between 1 and 4 bits in length.

- Internal metric width: The integer part of the α, β, γ and log-likelihood metrics can be either 6 or 7 bits. The fractional part of these metrics can be between 1 and 4 bits.
- Sliding window size: The size of the sliding window can be either 32 or 64.
- SISO algorithm: The SISO algorithm can be either the log-MAP, Max-log-MAP or Max Scale algorithms.
- External RAM: The data processed and produced by the Turbo decoder can use memory available on the FPGA or external RAM.

The sliding window technique [56] is a way of reducing the amount of memory required by the Turbo decoder. If the sliding window technique was not used the number of β and γ metrics required to be stored would be directly proportional to the block size of the packet being decoded. If the sliding window technique is used the number of β and γ metrics that need to be stored is directly proportional to the sliding window size. Depending on the sliding window size the amount of memory needed for the β and γ metrics is reduced by a factor of 159 for the UMTS standard and 647 for the cdma2000 standard. Figure 35 shows how the sliding window technique works. The packet to be decoded is split into windows of the same size. In decoding stage 1 an estimate of the starting β metric for window 1 is gained by performing a backwards recursion on window 2. The initial values of all β metrics in window 2 is an all zero symbol, in stage one the β values calculated for window 2 are discarded. At the same time a forward recursion is performed on window 1. All α metrics calculated for window 1 are stored. In stage 2 the β values for window 1 are calculated. These values and the α metrics calculated in stage 1 can then be used to calculate the log-likelihood ratios for window 1. At the same time an initial backward recursion is performed on window 3 to gain a starting point for the β metric calculations in window 2. A forward recursion on window 2 is also performed at this time.

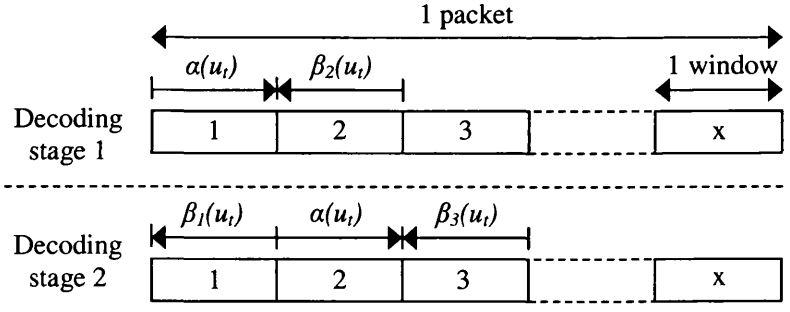

Figure 35: Sliding window technique

As mentioned previously the sliding window technique decreases the memory required by the Turbo decoder, it also decreases the clock cycle latency of the Turbo decoder. However, the BER performance is compromised as the β metrics calculated using the sliding window technique will have an error associated with them. Traditional Turbo decoders require a latency that is at least twice the block size of the packet being decoded. Appendix D {vol. 2/pp. 26-40}, presents an architecture that requires a latency of only one times the block size of a single packet while still retaining the correct β metrics. Although this architecture was never implemented, theoretically it should reduce the latency of one Turbo decoder iteration by $\frac{1}{3}$. Where one Turbo decoder iteration includes two SISO decoder operations, one interleaver operation and one de-interleaver operation. In a traditional Turbo decoder a SISO must perform both a forward and backward traversal of the trellis, this will take approximately twice as many clock cycles to complete one SISO operation as it would to perform an interleaver or de-interleaver operation. If it takes c clock cycles to interleave or de-interleave a data set, then it will take 2c clock cycles to complete one SISO operation on the same data set, giving a total of 6c clock cycles to complete one Turbo decoder iteration on any given data set for a traditional Turbo decoder. The architecture proposed halves the number of clock cycles required to complete one SISO operation to c, giving the total number of clock cycles to complete one Turbo decoder iteration to be 4c for the new architecture proposed.

The advantage of using a larger sliding window size is that a more accurate value for the β metrics is obtained. However, the latency required by a larger sliding window size is relatively larger. This is shown in Figure 36 which also shows that the latency for a traditional Turbo decoder is almost twice the latency for a Turbo decoder using the sliding window technique.

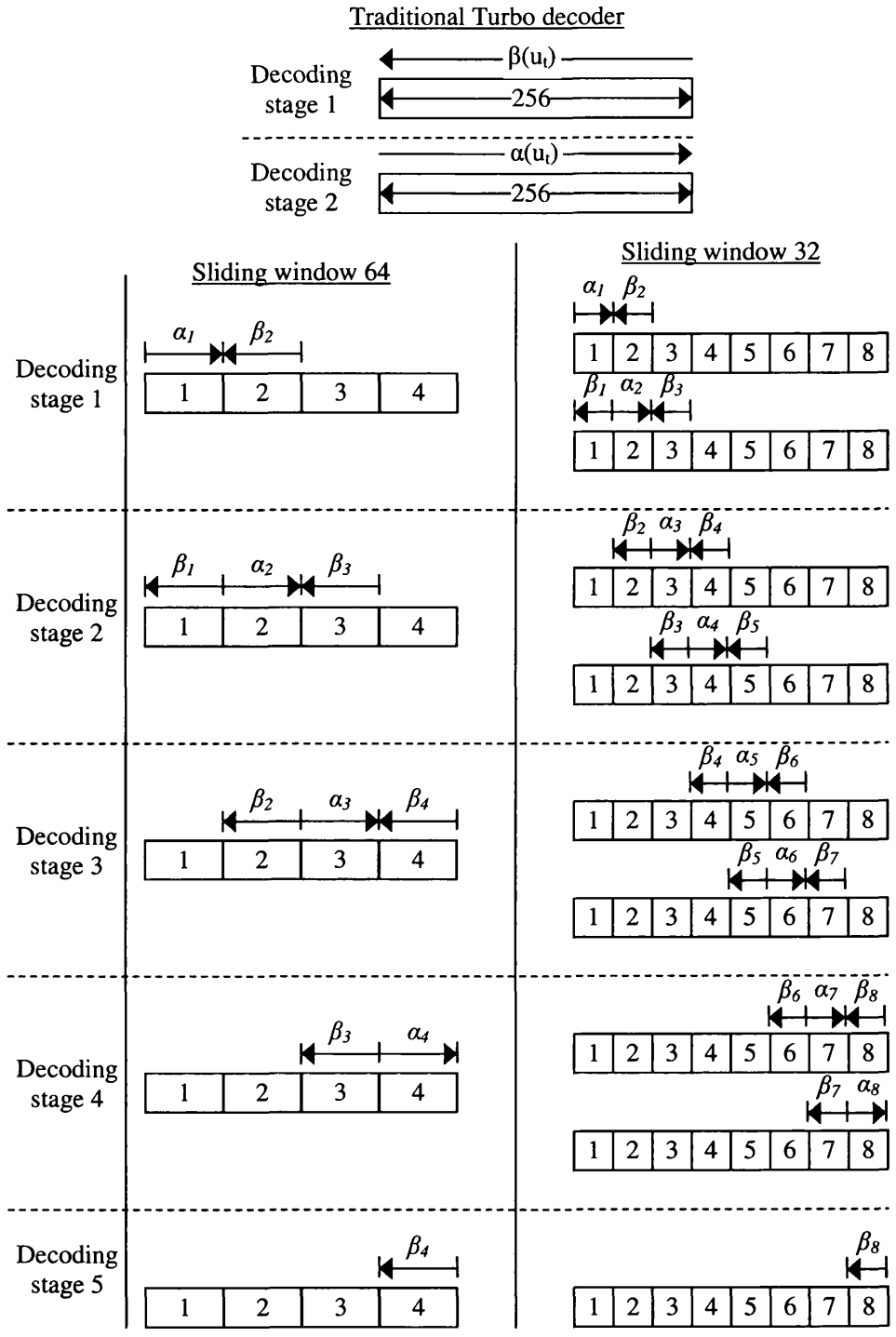

Figure 36: Decoding of packet with block size 256 using traditional Turbo decoder, Turbo decoder with sliding window 64 and Turbo decoder with sliding window 32

A sliding window of size 32 can operate on two windows in the same amount of time that it takes a sliding window of size 64 to operate on 1. Hence, each decoding stage contains two operations for every sliding window of size 32 and only one operation for a sliding window of size 64. Each decoding stage for the traditional Turbo decoder takes a total of 256 clock cycles, one clock cycle per input symbol. Therefore, a total of 512 clock cycles are required to output all log-likelihood metrics. The decoding stage for each of the sliding window implementations takes 64 clock cycles to complete. The only exception is for the sliding window of size 32 in decoding stage 5, as Figure 36 shows, this takes only 32 clock cycles. By adding the number of clock cycles for both sliding window sizes in Figure 36, it can be shown that a sliding window of size 64 will take 320 clock cycles to decode a set of data containing 256 symbols. A sliding window of size 32 will take 288 clock cycles to decode the same data set, both of these results conform to (20). Therefore (20) can be used to estimate the total clock cycles required to decode a data set, given the number of symbols in the data set and the sliding window size being used to decode the data set.

$$\text{clock cycles} = \text{block size} + \text{sliding window size} \tag{20}$$

The BERT platform is also capable of producing frame error rate (FER) plots as well as BER plots. The FER is the number of errors occurring in one packet of data. Figure 37 shows the FER for a sliding window of size 64 and a sliding window of size 32, highlighting the advantage of using a relatively larger sliding window size. However, as Section 7.8 will show, a standard Turbo decoder outperforms a Turbo decoder using the sliding window technique.

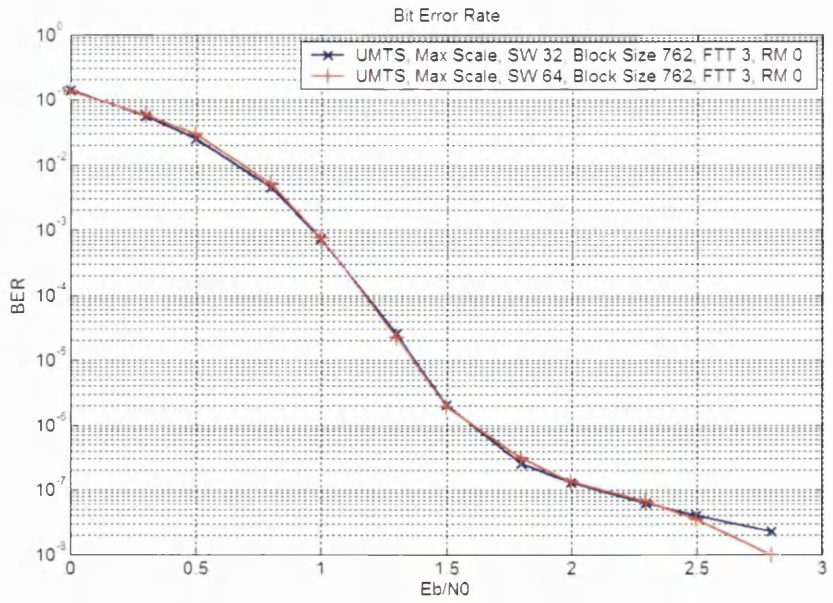

Figure 37: FER plot showing different sliding window sizes

Implementing a sliding window of size 64 will require more resources than implementing a sliding window of size 32. A Turbo decoder using the Max Scale algorithm with 5 bits representing input to the decoder, 9 bits representing internal metrics and a sliding window of size 32 will require 3,401 LUTs, 1,668 registers and 47 block memory components. The same configuration, with a sliding window size of 64 uses 3,697 LUTs, 1,883 registers and 47 block memory components.

Sliding window size is not the only parameter that can be used to alter the memory required a Turbo decoder. The input and internal metric widths also have an impact on both memory and performance. As the width of both the internal metric and input data increase, the BER performance of the Turbo decoder improves. Increasing the width of these values essentially decreases quantization noise. The input data comes from a demodulator, as Figure 3 shows; there will be a conversion between the data coming from the demodulator into the Turbo decoder. Allowing the turbo decoder to use more bits to represent the input data means that the quantization error will be reduced. The internal metric represents the α, β, γ and log-likelihood metrics. Increasing the width of these metrics allows the decoder to produce a more accurate representation of the internal metrics, increasing the BER performance of the decoder. This is shown in Figure 38 for different internal metric fractional widths; the integer width of all of these BER plots is 6.

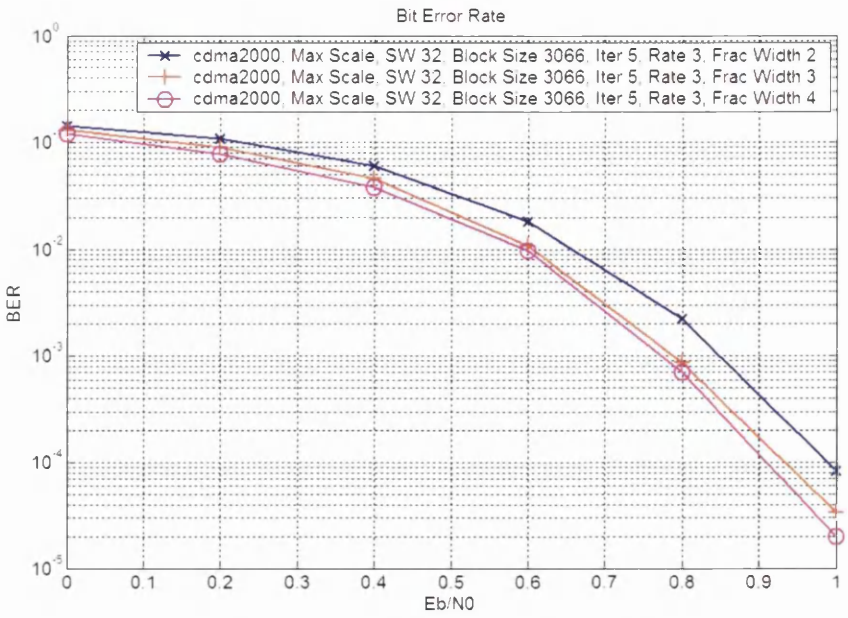

Figure 38: BER plot for different internal metrics

The Turbo decoder used in Figure 38 uses the Max Scale algorithm with an input data width of 4 bits and a sliding window of size 32. A fractional width of 3 and 4 produces very similar results, however both are a great improvement over a fractional width of 2. At a BER of 1×10^{-4} the coding gain of using 4 fractional bits compared with 2 fractional bits is 0·07dB. Using 3 fractional bits gives a coding gain of 0·05dB compared with 2 fractional bits at the same BER. The resources required in implementing the Turbo decoder with different internal fractional widths is shown in Table 7.

Table 7: FPGA resources required for different internal fractional widths

Fractional width	Slices	EMults	BRAMs
2	1800	2	40
3	1920	2	42
4	2066	2	43

As Table 7 shows the resources required by a Turbo decoder using 4 internal fractional bits is slightly more than a Turbo decoder using 2 or 3 internal fractional bits. If the amount of resources used by the Turbo decoder is not an issue, then the core with an internal fractional width of 4 bits should always be used. Just over 10% more resources are required, when

compared with a Turbo decoder using 2 internal fractional bits. However, if the resources used were an issue then a compromise would be to use a Turbo decoder with only 3 fractional bits. At a BER of 1×10^{-4} the coding gain of using 4 fractional bits compared with 3 fractional bits is only 0·02dB. Each state in Turbo decoder is associated with 4 internal metrics, an α, β and two γ metrics. Each time instance is also associated with a log-likelihood metric. Each time instance has 8 states and therefore in total 33 metrics are associated with a single time instance. Increasing the internal metric width by a single bit means that the memory required increases by 33 bits per time instance.

The SISO algorithm chosen has a significant bearing on the performance of the Turbo decoder. Figure 39 highlights the differences between the three algorithms described in Section 6.2.3. The Turbo decoder used to create the plots in Figure 39 had an input data width of 5 bits, an internal metric width of 9 bits and a sliding window of size 32. Table 8 shows the resources used for the Turbo decoder implemented with different SISO algorithms.

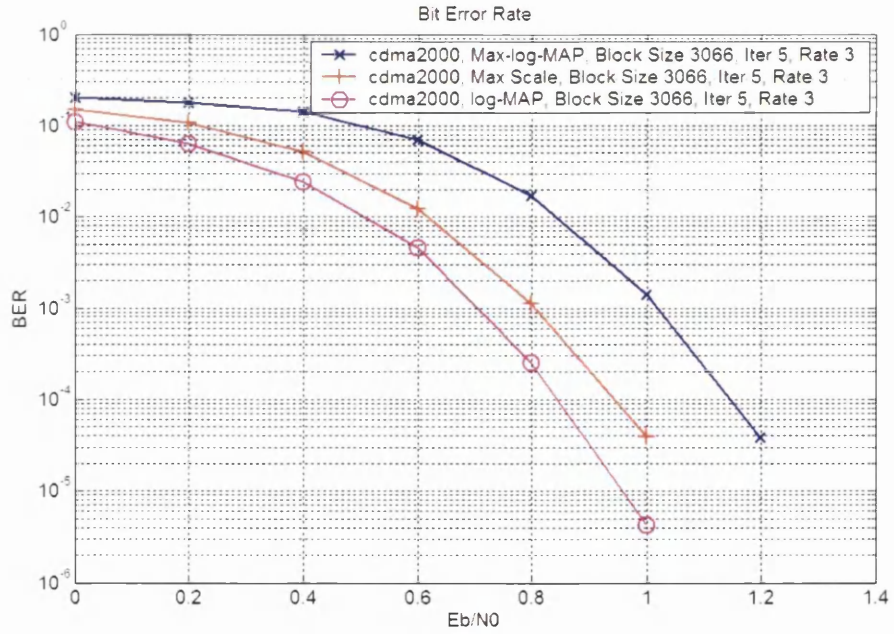

Figure 39: BER performance of log-MAP, Max-log-MAP and Max Scale algorithm

Table 8: FPGA resources used for different SISO algorithms

SISO Algorithm	Slices	EMults	BRAMs
Max-log-MAP	1971	2	47
Max Scale	1942	2	47
log-MAP	2137	2	47

As mentioned in Section 6.2.3 the Max Scale and Max-log-MAP algorithms are extremely similar, hence the resources required are very similar. However, Figure 39 shows that the Max Scale gives a significant coding gain relative to the Max-log-MAP algorithm. The coding gain for the Max Scale algorithm compared with the Max-log-MAP algorithm is almost 0·3dB at a BER of 1×10^{-4}. Figure 39 also shows the coding gain that is achieved by the more complex log-MAP algorithm. This has a coding gain of approximately 0·1dB compared with the Max Scale algorithm. The extra resources required by the log-MAP algorithm over the other two SISO algorithms are used to implement the look up tables used in the log-MAP algorithm. Again, if the amount of resources required by the Turbo decoder were not an issue, then the log-MAP algorithm should always be chosen as the SISO decoder implementation, as only around 10% more resources are required by the log-MAP implementation, when compared with the Max-log-MAP implementation.

7.6. cdma2000 vs. UMTS

Using the System Generator BERT the RE was able to compare the cdma2000 and UMTS Turbo code standards. Figure 40 shows the BER plots for the cdma2000 and Turbo decoders at various block sizes. The UMTS results are shown in red, the cdma2000 results are shown in blue.

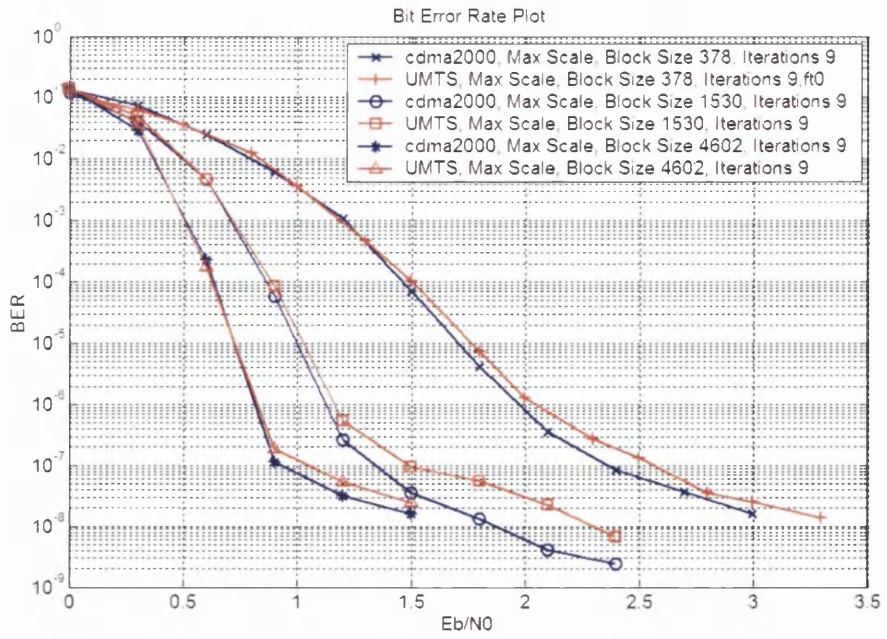

Figure 40: Comparison of cdma2000 and UMTS Turbo decoders

Figure 40 shows that the cdma2000 Turbo decoder is better than the UMTS Turbo decoder at all block sizes tested for the Max Scale implementation, this result was also observed for the log-MAP algorithm. In some instances the variation in performance could be attributed to different design techniques or design flows. However, both decoders were implemented using the same design tools and design flow which would discount this argument. The most likely reason for the difference in performance is either the Turbo encoder or Turbo interleaver. Section 6.2.2 discussed the differences between the cdma2000 and UMTS Turbo interleavers. This is likely to be a contributing factor to the coding gain obtained by the cdma2000 Turbo interleaver relative to the UMTS Turbo interleaver. Although the UMTS Turbo interleaver is more complex and therefore more difficult to implement in hardware it does have some advantages over the cdma2000 Turbo interleaver. The main advantage is that the cdma2000 Turbo interleaver can only process discrete block sizes whereas the UMTS Turbo interleaver can process a packet of any width between 40 bits and 5114 bits.

Figure 40 also reveals more advantages of using hardware rather than software for implementing the BERT. Using hardware means that BER plots can be generated at very low BER values. A software implementation would have to run for weeks or even months

to get to these BER levels. At the low BER levels it can be seen the BER plot is approaching the Turbo error floor. The error floor is an inherent problem with Turbo interleavers. The error floor occurs after the so-called waterfall region of the BER plot. The waterfall region of the BER plot is where the BER curve has a very steep gradient, i.e. between 0·3dB and 0·8dB for a block size of 4602 in Figure 40. The error floor of this BER curve occurs above 0·8dB. One point of note from Figure 40 is that as the plot showing a block size of 4602 approaches the error floor it has a shallower gradient than the other two BER curves. By referring to Figure 40 it can be seen that as block size increases the gradient of the BER plot as it approaches error floor becomes shallower. Had the BER curve for a block size of 4602 continued it would have crossed the BER curve for a block size of 1530. Section 7.7 discusses one particular situation where a BER curve for a code rate of $\frac{1}{5}$ cross over the BER curves of a higher code rate. A novel solution to this problem is given.

7.7. Channel Variance

The channel variance value, σ^2, of an AWGN channel has an impact on the input to the Turbo decoder, which is in the form of a log-likelihood ratio, $L(u_t)$, the effect is shown in (21), where μ is the mean.

$$L(u_t) = \frac{2 \times \mu \times u_t}{\sigma^2} \tag{21}$$

The data input to the decoder would usually take the form of $\frac{L(u_t)}{2}$ as both the mean and variance would be estimated to be 1.

Channel variance is also linked to $\frac{E_b}{N_0}$ by (22).

$$\frac{E_b}{N_0} = \frac{1}{2 \times rate \times \sigma^2} \tag{22}$$

A plot of (22) is shown in Figure 41. If a channel variance estimate of 1 was used then it means that the Turbo decoder is optimal at $\frac{E_b}{N_0}$ values of 0dB, 1·8dB and 3dB for code rates of $\frac{1}{2}$, $\frac{1}{3}$ and $\frac{1}{4}$ respectively.

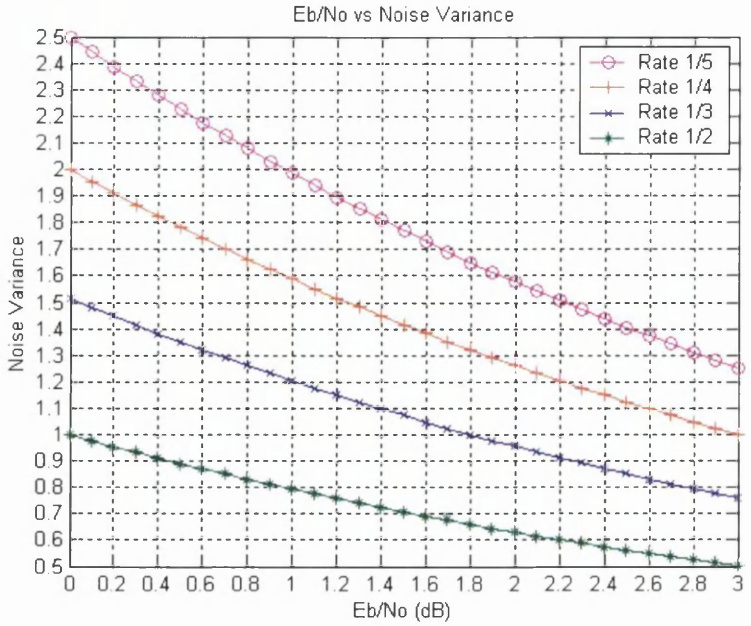

Figure 41: Plot of $\frac{E_b}{N_0}$ against noise variance

It would be expected that if a number of BER plots were generated, each with a different code rate, the BER curve with the lowest code rate would have the best BER performance. However, one result produced by the BERT platform, shown in Figure 42, shows that the BER curves for code rates of $\frac{1}{3}$ and $\frac{1}{4}$ both have a better BER performance than a code rate of $\frac{1}{5}$ above a $\frac{E_b}{N_0}$ value of 1·8dB. This result could have occurred because of the Turbo interleaver as discussed in Section 7.6. However, after referring to Figure 41 the RE decided to investigate changing the noise variance value to see how it affected the BER plot shown in Figure 42. A channel variance value of 1 was used in the channel to generate the plots in Figure 42.

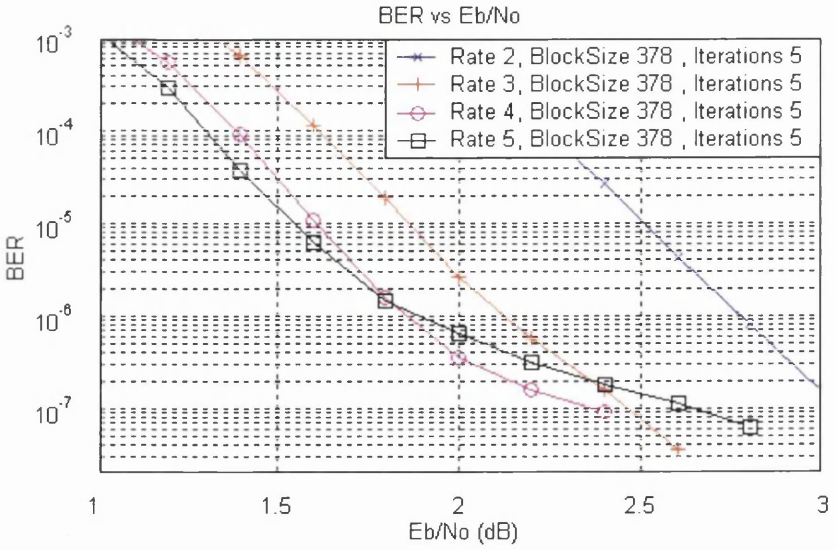

Figure 42: Non-optimal BER performance of code rate $\frac{1}{5}$ using σ^2 value of 1

The RE used the BERT platform to alter the channel variance value in (21). A number of channel variance values were tested until an optimum value was found for the code rates in Figure 42. Noise variance was incremented from a value of 1 to a value of 1·35 in increments of 0·05 to find the optimal noise variance value. Figure 43 shows the results obtained when using a code rate of $\frac{1}{5}$ zoomed into the region of interest. Values of 1·1 and 1·15 were found to be optimal for a code rate of $\frac{1}{5}$. Although variance values of 1, 1·05 and 1·3 seem to perform better at the highest $\frac{E_b}{N_0}$ value they suffer from the crossover problem highlighted earlier. For a code rate of $\frac{1}{4}$ noise variance values of 1·1, 1·2 and 1·25 were found to be most optimal. Results obtained for code rates of $\frac{1}{3}$ and $\frac{1}{2}$ showed a similar pattern. As a value of 1·1 is optimal for all code rates, it was chosen as the preferred σ_n^2 value. Figure 12 compares the results obtained when the value of noise variance was set to 1 and 1·1.

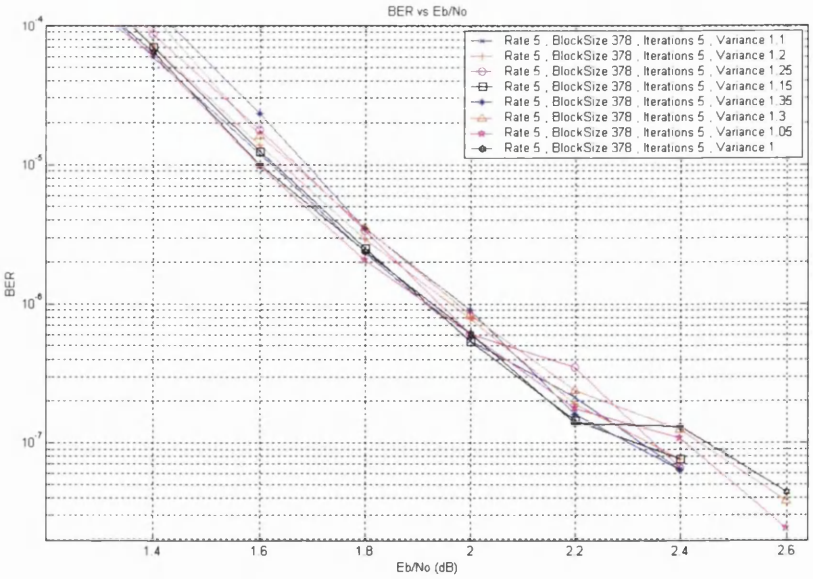

Figure 43: Code rate $\frac{1}{5}$ BER plots for noise variance values between 1 and 1·35

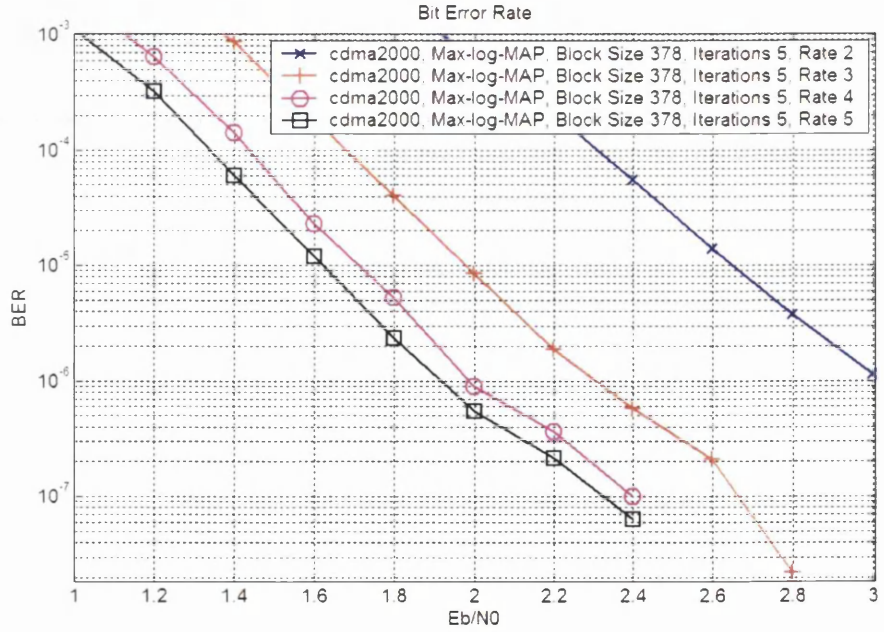

Figure 44: Optimal BER performance of code rate $\frac{1}{5}$ using σ^2 value of 1·1

As Figure 44 shows, the problem of crossover has been resolved, giving a more preferable result. However, at the expense of solving the crossover problem, the BER performance of all code rates is compromised, relative to BER plots where the variance value is calculated

using (22). This is shown in Figure 45. BER plots where σ^2 is known are shown in red, BER plots where σ^2=1·1 are shown in blue.

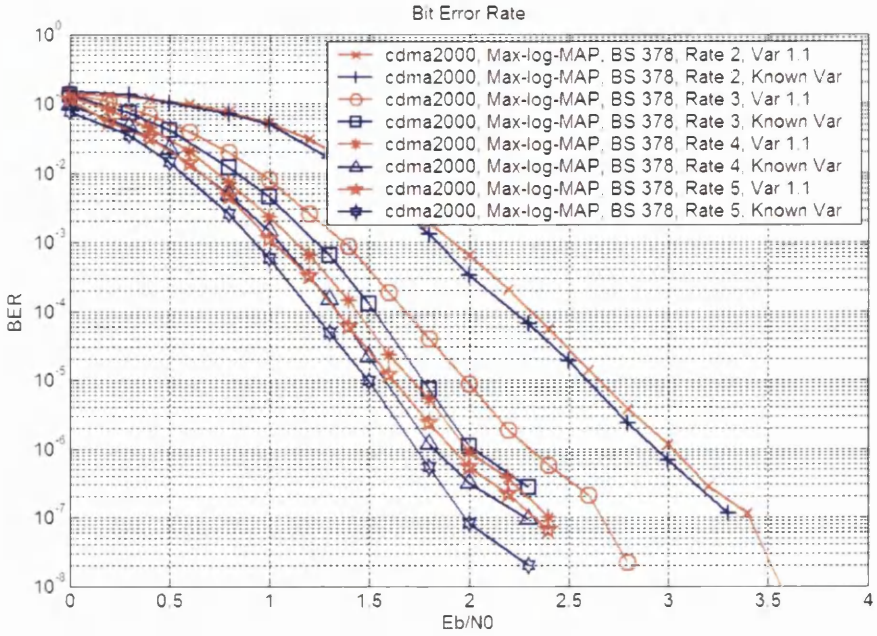

Figure 45: Comparison of BER plots with known σ^2 and BER plots using σ^2=1·1

As Figure 45 shows the BER plots with a known σ^2 value clearly outperforms BER plots whose σ^2=1·1. At a BER of 10^{-6}, the BER plots using a known σ^2 value outperform the BER plots whose σ^2=1·1 at code rates of $\frac{1}{3}$, $\frac{1}{4}$ and $\frac{1}{5}$ by around 0·25dB.

7.8. Comparison With Published Results

To qualify the results published in this portfolio, BER plots generated using the System generator BERT were compared to results already published by other researchers.

Figure 46 shows results produced by the RE for the cdma2000 decoder. This plot shows how the log-MAP algorithm performs when using a block size of 1530 at all available puncturing rates. The results shown in Figure 46 are comparable with those published by Valenti and Sun [57]. Table 9 shows how both sets of results compare at a BER of 10^{-6}, at all rates the results shown in [57] are, at maximum, within 0·1dB of the results produced by the RE.

Table 9: Comparison of Valenti and Sun [57] with results produced by the RE at BER of 10^{-6}

Rate	$\frac{1}{2}$	$\frac{1}{3}$	$\frac{1}{4}$	$\frac{1}{5}$
Valenti & Sun [57]	1·7dB	1dB	0·75dB	0·6dB
RE	1·75dB	1·1dB	0·7dB	0·65dB

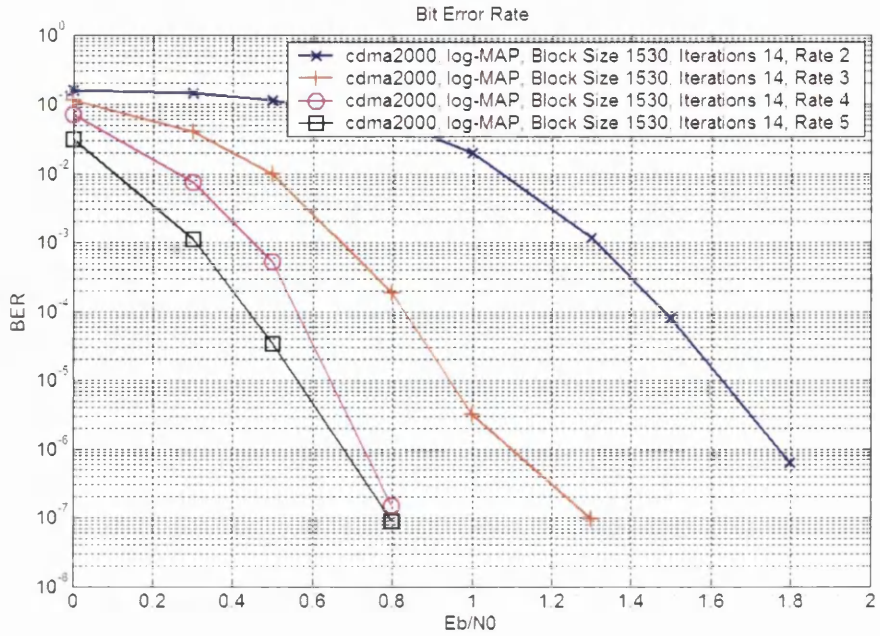

Figure 46: BER plot for comparison with Valenti and Sun [57]

Figure 47 shows results produced by the RE using the cdma2000 Turbo decoder for a block size of 378 and iterations between 1 and 8. Qi [58], reveals results for these decoder settings at a $\frac{E_b}{N_0}$ of 1·2dB. Table 10 compares the BER results for [58] with the results shown in Figure 47. The improvement seen in the results published by Qi can be explained by the fact that Qi used the traditional method of decoding Turbo codes, whereas the RE used a sliding window of size 32. The improvement is most evident when either 5, 6 or 7 iterations are performed. Table 10 shows when these number of iterations are performed the standard Turbo decoder has BER that is almost one decade better than the Turbo decoder using the sliding window technique.

Table 10: Comparison of Qi [58] with results produced by the RE

Iterations	1	2	3	4	5	6	7	8
Qi [58]	6×10^{-1}	3×10^{-2}	9×10^{-2}	2×10^{-3}	6×10^{-3}	7×10^{-3}	8×10^{-3}	7×10^{-3}
RE	$3{\cdot}5\times10^{-1}$	$9{\cdot}5\times10^{-1}$	4×10^{-2}	$8{\cdot}5\times10^{-2}$	$9{\cdot}25\times10^{-2}$	$9{\cdot}75\times10^{-2}$	1×10^{-3}	2×10^{-3}

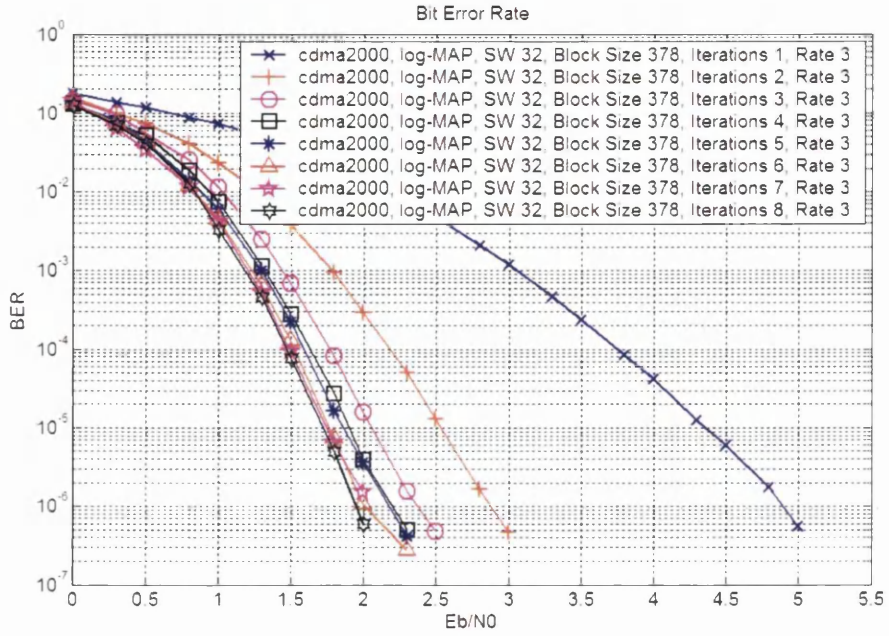

Figure 47: BER plot for comparison with Qi [58]

Figure 48 shows how the results produced by the RE compare with Sugimoto et al [59] for a block size of 378, 6 iterations and a code rate of $\frac{1}{4}$ using the cdma2000 decoder. Table 11 compares the results produced by the RE and by Sugimoto et al for BER plots at 10^{-3}, 10^{-4} and 10^{-5}. The algorithm used in both cases is the Max Scale algorithm, called the NSubMap in [59].

Table 11: Comparison of results produced by RE and Sugimoto et al [59]

BER	10^{-3}	10^{-4}	10^{-5}
Sugimoto et al [59]	0·9dB	1·2dB	1·4dB
RE	0·9dB	1·15dB	1·4dB

Table 11 shows that the results shown in [59] are very comparable with those produced by the RE, reinforcing the quality of the results produced in this portfolio.

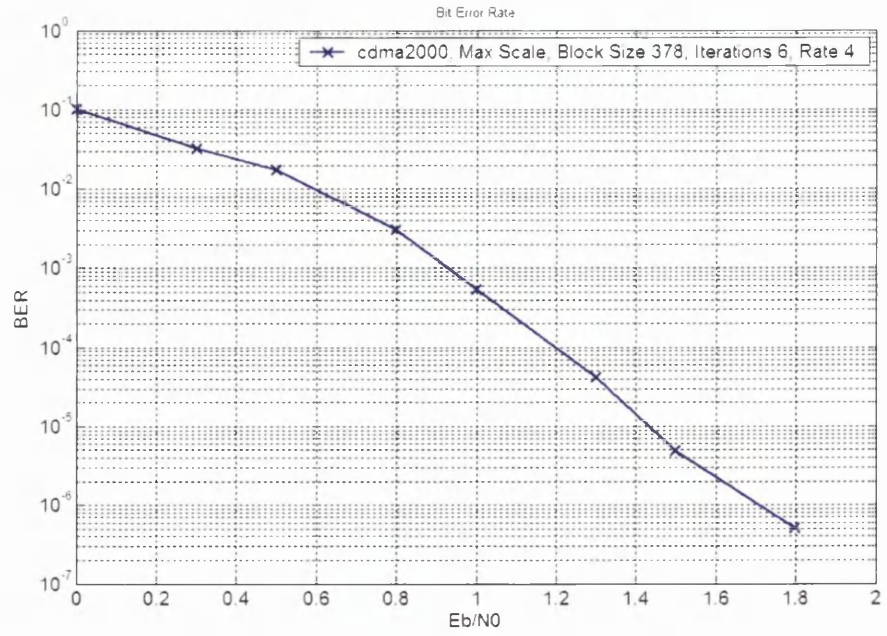

Figure 48: BER plot for comparison with Sugimoto et al [59]

Sagimoto et al [59] was also used to compare the results produced by the RE shown in Figure 49, this plot shows the BER performance of the cdma2000 decoder with a block size of 6138 at code rates of $\frac{1}{2}$ and $\frac{1}{3}$, performing 6 iterations. Table 12 compares the results produced by the RE and those published in [59] at BERs of 10^{-3}, 10^{-4} and 10^{-5}. The results shown in [59] and in Figure 49, both use the Max Scale algorithm.

Table 12: Comparison of results produced by RE and Sugimoto et al [59]

Rate	$\frac{1}{2}$			$\frac{1}{3}$		
BER	**10^{-3}**	**10^{-4}**	**10^{-5}**	**10^{-3}**	**10^{-4}**	**10^{-5}**
Sugimoto et al [59]	0·95dB	1·2dB	1·45dB	0·5dB	0·6dB	0·675dB
RE	1·1dB	1·2dB	1·3dB	0·5dB	0·6dB	0·7dB

Table 12 again shows that the results produced by the RE are comparable by those shown in [59]. The difference between the BER curves at the selected BERs is, at most, 0·15dB.

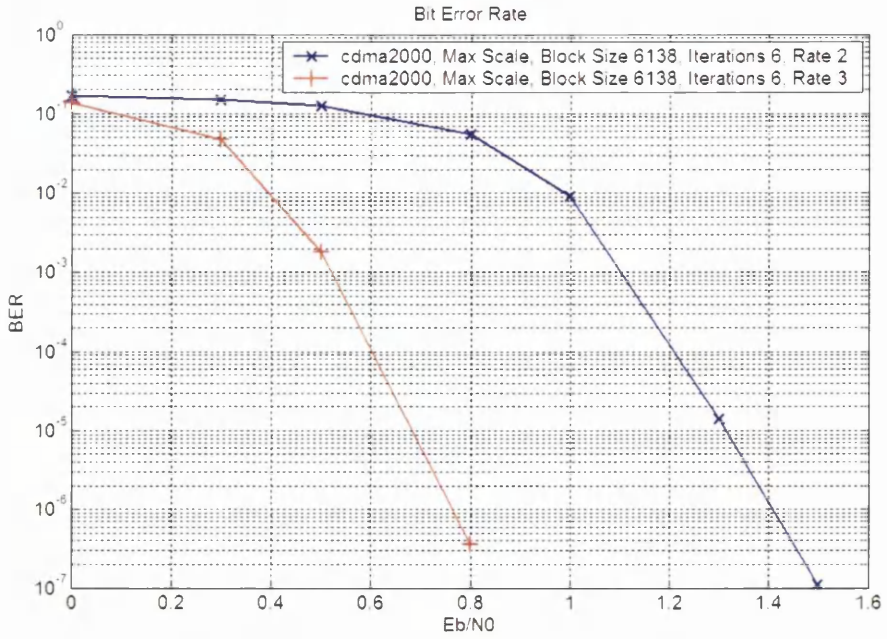

Figure 49: BER plot for comparison with Sugimoto et al [59]

The UMTS Turbo decoder was used to produce the results shown in Figure 50. Table 13 compares the results shown in [57] and with those produced by the RE for a block size of 1530, a decoder using the log-MAP algorithm, a code rate of $\frac{1}{3}$ and iterations of 5, 7 and 9 at a BER of 10^{-6}.

Table 13: Comparison of Valenti and Sun [57] with results produced by the RE at BER of 10^{-6}

Iterations	**5**	**7**	**9**
Valenti & Sun [57]	1·2dB	1·1dB	1dB
RE	1·25dB	1·15dB	1·05dB

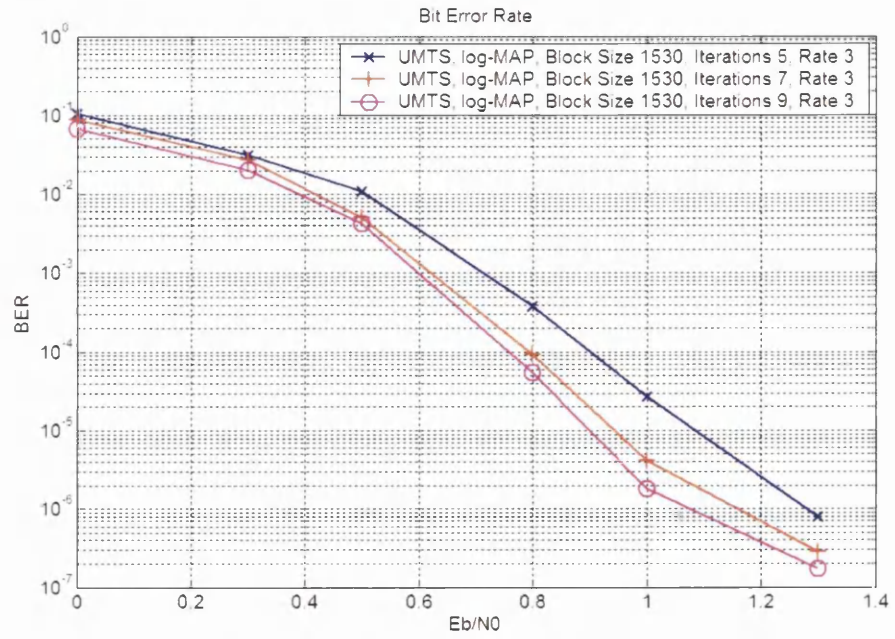

Figure 50: BER plot for comparison with Valenti and Sun [57]

As Table 13 shows, the RE and [57] produce results that are, at most, 0·05dB apart, showing that the results produced by the RE using the UMTS Turbo decoder are comparable with those published by other researchers.

8. Conclusion

In this portfolio thesis the RE has introduced the subject of Turbo codes, FPGAs and 3G mobile technology in general. FPGAs were evaluated against other silicon devices such as ASICs and DSPs. Presently, the main competitor for FPGAs in the 3G market is DSPs. This thesis has shown that FPGAs have both more processing power than DSPs and that they are cheaper per channel than DSPs. The main advantage over ASICs for FPGAs is their reconfigurability. Being able to reconfigure FPGAs means that new standards, or upgrades to standards, can be implemented remotely, potentially saving the network provider money. The case of the UMTS standard release 5 was highlighted, release 5 standardised the HSDPA architecture for UMTS. Any network provider who had implemented an earlier release of the UMTS standard in an ASIC would have to have respun their design, absorbing large non-recurring engineering costs in the process. If the design was implemented in an FPGA the new standard could be downloaded remotely.

The design of Turbo codes in FPGAs was also covered, the RE proposed a novel hardware architecture that halved the latency compared to a traditional Turbo decoder. One of the main advantages of implementing a Turbo decoder in an FPGA rather than a DSP is that the FPGA can process multiple channels simultaneously. The maximum number of Turbo decoders that can be implemented on a Xilinx Virtex-II 3000 FPGA is six: this figure is for a cdma2000 Turbo decoder using the Max Scale algorithm with a data input width of 5 bits, an internal metric width of 9 bits, a sliding window size of 32 and using external RAM. The number of cores that can be implemented on a single device is restricted by the number of block RAMs used. When external RAM is used only 15 BRAMs are required; if external RAM is not used 47 BRAMs are required. This highlights the large memory requirements of Turbo decoders.

A novel parameterisable Turbo decoder was described. The novelty of this decoder is that the user can specify a number of parameters that allow them to offset performance for implementation resources. The portfolio thesis highlighted the performance gains of altering certain parameters and revealed the extra resources that these performance gains would require.

Implementing the BERT in System Generator produced a novel design that the sponsoring company could use to evaluate all FEC cores. It also helped the sponsoring company evaluate using System Generator for core development. Using System Generator allowed the RE to create a unique design that could be easily used and could also be configured to the users need. The configurable options in the System Generator BERT allowed the user to specify how many errors should be detected before plotting a point, at the same time the BERT monitored how many bits were processed by the decoder. Once a certain threshold was reached a point would be plotted. The advantage of this technique is that at low $\frac{E_b}{N_0}$ errors are accumulated rapidly, however at relatively higher $\frac{E_b}{N_0}$ values it takes longer for the error threshold to be reached. When this occurs the number of bits processed threshold is used to plot a point, which speeds up the BERT for any given test. The user can also specify the minimum BER point to be plotted. When this threshold is reached the BERT is stopped. A Matlab script is used to automate the BERT platform, in the script a user can specify a number of code rates, block sizes and iterations to be performed so that a number of tests can be performed consecutively. The RE was able to reduce the resources used by the System Generator BERT. This allowed the BERT to fit onto a smaller device and allowed it to be taken through the System Generator design flow using only Xilinx software tools.

The BERT allowed the RE to produce a number of novel results. These were a comparison of UMTS and cdma2000 Turbo codecs and a channel variance value that optimised the cdma2000 Turbo decoder.

The BERT was also a great commercial tool for the sponsoring company. It allowed the sponsoring company to evaluate their FEC cores using their own tools rather than third party software such as a C simulator while obtaining a relatively large speed up. It also allowed them to give their customers a tool that they could easily use even if they were not familiar with the FPGA design process. The BERT system is also used extensively as a company demonstration of the System Generator hardware emulation.

Other work completed by the RE was also of great commercial relevance to Xilinx. The RE designed a Turbo encoder behavioural model that could be used by Xilinx customers

who purchased their UMTS Turbo encoder core. A cdma2000 standard RSC encoder was also designed by the RE.

A number of research reports produced by the RE allowed Xilinx to evaluate different proposals for their Turbo decoder designs. These included reports on subjects such as Turbo decoder hardware architectures, allowing Xilinx to decide which hardware architecture to use in their cdma2000 decoder. The RE also evaluated how the DVB standard Turbo codec could be designed, taking note of the main differences between this duo-binary Turbo codec and standard binary Turbo codecs. A report on how to calculate the input values to Turbo decoders led to a novel variance value that improved the cdma2000 Turbo decoder core. Finally a number of papers were published by the RE; these papers highlighted the performance and parameterisablity of the Turbo codecs presented in this thesis.

8.1. Future Direction

The System Generator BERT described in this thesis used an AWGN channel to evaluate the Turbo codecs presented. An AWGN channel allows the user to evaluate the Turbo codecs presented in this thesis against other implementations, however AWGN channels do not represent a real mobile channel. The AWGN channel could be replaced by one which is more like a real mobile environment such as a Rayleigh fading or Rician fading channel.

Another department in the sponsoring company produced a cdma searcher in System Generator. If more 3G components were implemented in System Generator the sponsoring company could build a demonstration of a 3G base station. The System generator BERT would play an integral part in this model. Using such a model would be of great commercial advantage to the sponsoring company.

Some of the architectures proposed by the RE could be implemented and tested against the sliding window architectures already implemented. The most potentially interesting could be the novel architecture proposed by the RE, although the memory usage would be high it would be interesting to see the performance advantages gained.

The RE implemented one novel channel variance value to improve the cdma2000 Turbo decoder core. A future development would be to extend this so that each code rate at

discrete $\frac{E_b}{N_0}$ values were associated with a certain variance value. These values could be stored as a look up table on the FPGA. Alternatively, (21) could be implemented. Both of these implementations would require an estimate of the channel to be input to the decoder.

In terms of future coding developments, low-density parity check (LDPC) codes [60] are emerging as a challenge to Turbo codes in future mobile standards. LDPC codes should be evaluated against current Turbo code implementations to determine what improvements are possible.

9. References

[1] C. Dick, J. Steensma. FPGA Implementation of a 3GPP Turbo codec. Signals, Systems and Computers, 2001. Conference Record of the Thirty-Fifth Asilomar Conference on ,Volume: 1 , 4-7 Nov. 2001
Pages:61 - 65 vol.1

[2] 3rd Generation Partnership Program (3GPP). Technical Specification Group Radio Access Network; Multiplexing and Channel Coding (FDD) 3GPP TS 25.212 V5.6.0, 2003.

[3] D. Johnson. Programming a Xilinx FPGA in "C". Xilinx Xcell Journal No. 34, pp. 26-30. December 1999.

[4] D. Johnson. Architectural Synthesis from Behavioral Code to Implementation in a Xilinx FPGA. Xilinx Xcell Journal No. 36, pp. 23-25. June 2000.

[5] 3rd Generation Partnership Program 2 (3GPP2). C.S0002-C Physical Layer Standard for Spread Spectrum Systems, Version 1.0, Release C, May 2002.

[6] Nallatech. BERT Test Platfrom User Guide Issue 1·0. May 2002, www.nallatech.com, last accessed July 2002.

[7] E. Brown. Reconfigurable cores for wireless appliances: turbo codes. Poster presentation, SET for Europe, London , UK , December 2002.

[8] Digital Video Broadcasting. Interaction Channel for Satellite Distribution Systems, ETSI EN 301 790, Version 1.3.1, April 2003.

[9] E. Brown, J. Irvine, B. Wilkie. A memory-efficient parameterisable FPGA implementation of the cdma2000 codec. Colloquium on DSPenabled Radio, Livingston , Scotland , 22-23 Sep 2003.

[10] E. Brown, J. Irvine, B. Wilkie. A memory-efficient parameterisable FPGA implementation of the cdma2000 codec. World Wireless Congress, San Francisco, CA, 25-28 May 2004.

[11] Xilinx. System Generator User Guide. http://www.xilinx.com/products/software/sysgen/app_docs/user_guide.htm, last accessed June 2004.

[12] Nallatech. Virtex-II XtremeDSP Development Kit, http://www.xilinx.com/ipcenter/dsp/XtremeDSP_Development_Kit-II_User_Guide.pdf, last accessed, June 2004.

[13] Annapolis. Annapolis Wildcard II Data Sheet. http://www.annapmicro.com/datasheets/wcii_markdata_12969_1_2.pdf, last accessed June 2004.

[14] Alpha Data. ADM-XRC-II PC Mezzanine Card User Guide v1·9. http://www.alpha-data.com/pdf/adm-xp.pdf, last accessed June 2004.

[15] E. Brown, J. Irvine, B. Wilkie. Rapid Prototyping of an Automated Test Harness for Forward Error Correcting Codes, 11th European Wireless Conference, pp 79-83, Nicosia, Cyprus, 10-13 April 2005

[16] E. Brown, J. Irvine, B. Wilkie. Rapid Prototyping of a Test Harness for Forward Error Correcting Codes. ACM/SIGDA thirteenth International Symposium on Field Programmable Gate Arrays (FPGA 2005), Monterey, CA, 20-22 Feb 2005.

[17] BBC. 3G Goes Live in the UK. http://news.bbc.co.uk/1/hi/technology/2808761.stm, last accessed February 2005.

[18] M. Benson, H.J. Thomas. Investigation of the UMTS to GSM Handover Procedure. IEEE 55th Vehicular Technology Conference, VTC Spring 2002, Volume 4, pp. 1829-1833. Birmingham, USA, 6-9 May 2002.

[19] E. Jugl, U. Bernhard, H. Pampel. Strategy and Performance on UMTS-GSM Handover. 4th International Conference on 3G Mobile Communication Technologies, 3G 2003, Volume 2, pp.1307-1312. London, UK, 25-27 June 2003.

[20] D. Lugara. J. Tartiere. L. Girard. Performance of UMTS to GSM Handover Algorithms. 15th International Symposium on Personal , Indoor and Mobile Radio Communications, PIMRC 2004. Volume 1, pp. 444-448. Barcelona, Spain, 5-8 September 2004.

[21] BBC. Video Phones Show Slow Take Off. http://news.bbc.co.uk/1/hi/technology/3322359.stm, last accessed February 2005.

[22] BBC. UK Gearing Up For 3G Christmas, http://news.bbc.co.uk/1/hi/business/3739834.stm, last accessed February 2005.

[23] ZDNet UK, 3 Leans on LG as 3G Gathers Momentum. http://news.zdnet.co.uk/communications/3ggprs/0,39020339,39164118,00.htm, last accessed February 2005.

[24] BDTI. Alternatives to DSP: What and Why? www.bdti.com/articles/20030527_Alternatives_to_DSPs.pdf, last accesses April 2005.

[25] www.sdrforum.com

[26] Xilinx. Xilinx Virtex-II Data Sheet. http://direct.xilinx.com/bvdocs/publications/ds031.pdf, last accessed February 2005.

[27] C.E. Shannon. A Mathematical Theory of Communication. Bell System Technical Journal, Vol 27, pp. 379-423, July 1948, pp. 623-656, October 1948.

[28] S. Lin, D.J. Costello Jr. Error Control Coding: Fundamentals and Applications. Prentice-Hall, 1983, ISBN 0-13-283796-X, pp. 51-84

[29] S. Benedetto, E. Biglieri. Principles of Digital Transmission With Wireless Applications. Kluwer Academic/Plenum Publishers, 1999, ISBN 0-306-45753-9, pp. 452-527.

[30] D. Divsalar, F. Pollara. Turbo Codes for PCS Applications. IEEE Proceedings 1995 International Communications Conference Seattle, USA, pp. 54-59, June 1995.

[31] C. Berrou, A. Glavieux, P. Thitimajshima. Near Shannon Limit Error-correcting Coding and Decoding Turbo codes. IEEE Proceedings 1993 International Communications Conference, pp. 1064-1070, 1993.

[32] S. Benedetto, G. Montorsi. Unveiling Turbo Codes: Some Results on Parallel Concatenated Coding Schemes. IEEE Transactions on Information Theory, Vol. 42, No. 2, pp. 409-428, March 1996.

[33] D. Divsalar, F. Pollara. Turbo Codes for Deep-Space Communications. http://tmo.jpl.nasa.gov/tmo/progress_report/42-120/120D.pdf, last accessed April 2005.

[34] Consultive Committee for Space Data Systems (CCSDS). Recommendation for space data system standards: Telemtry and channel coding. CCSDS 101.0-B-6, Blue Book, October 2002.

[35] A.J. Viterbi. Convolutional Codes and Their Performance in Communication Systems. IEEE Transactions on Communications, Vol 19, No 5, pp. 751-772, October 1971.

[36] J. Hagenauer and L. Papke. Decoding Turbo Codes With The Soft Output Viterbi Algorithm (SOVA). Proceedings of the International Symposium on Information Theory, pp. 164, 1994.

[37] L. R. Bahl, J.Cocke, F.Jelink and J.Reviv. Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate. IEEE Transactions on Information Theory, pp. 284-287, March 1974.

[38] P. Robertson, E. Villebrun and P. Hoeher. A Comparison of Optimal and Sub-Optimal MAP Decoding Algorithms Operating in the Log Domain. IEEE International Conference on Communications. Vol. 2, pp. 1009-1013, 18-22 June 1995.

[39] W. Ryan. A Turbo Code Turtorial. http://www.ee.udel.edu/~fu/images/tc_tutorial.pdf, last accessed April 2005.

[40] P. Robertson and P. Hoeher. Optimal and sub-optimal maximum a posteriori algorithms suitable for Turbo decoding. European Trans. on Telecommunications, Vol. 8, No. 2, 1997, pp. 119-125.

[41] E. Boutillon, W.J. Gross, G. Gulak. VLSI Architectures for the MAP Algorithm. IEEE Transactions on Communications, Vol. 51, No. 2, pp. 175-185, February 2003.

[42] J. Vogt, A. Finger. Improving the MAX Log MAP Turbo Decoder. IEEE Electronics Letters, Vol 36, No 23, pp. 1937-1939, November 2000.

[43] Xilinx. Xilinx Virtex-II Pro Data Sheet. http://direct.xilinx.com/bvdocs/publications/ds083.pdf, last accessed April 2005.

[44] V. Singh, A. Root, E. Hemphill, N. Shirazi, J. Hwang. Accelerating Bit Error Rate Testing Using a System Level Design Tool. 11th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, 9-11 April 2003, pp. 62-68.

[45] G.R. Cooper, R.W. Nettleton, D.P. Grybos. Cellular Land-Mobile Radio: Why Spread Spectrum. IEEE Communications Magazine, Vol. 17, No.2, March 1979, pp. 17-23

[46] H. Holma, A. Toskala. WCDMA for UMTS-Second Edition. J. Wiley and Sons, ISBN 0470844671, 2002.

[47] A.J. Viterbi. CDMA-Principles of Spread Spectrum Communications. Addison-Wesley, ISBN 0201633744, 1995.

[48] R. Price, P.E. Green Jr. A Communication Technique for Multipath Channels. Proceedings of Institute of Radio Engineers, Vol. 46, March 1958, pp. 555-570.

[49] Texas Instruments. TMS320TCI100 Fixed-Point Digital Signal Processor Datasheet. http://focus.ti.com/general/docs/lit/getliterature.tsp?baseLiteratureNumber=sprs218, last accessed April 2005.

[50] Xilinx. Using FPGAs in Wireless Base Station Designs. http://www.xilinx.com/publications/xcellonline/xcell_52/xc_pdf/xc_v4wireless52.pdf , last accessed April 2005.

[51] X. Li, H. Huang, G.J. Foschini, R.A. Valenzuela. Effects of Iterative Detection and Decoding on the Performance of BLAST. IEEE Global Telecommunications Conference: GLOBECOM2000, 27 November – 1 December, Vol. 2, pp. 1061-1066.

[52] IEEE. Software Radios. IEEE Communications Magazine, Vol. 3, No. 5, May 1995.

[53] M.C. Jeruchim, P. Balaban, and K.S. Shanmugan. Simulation of Communication Systems. Plenum Press, New York, 1992

[54] A. Matache, S. Dolinar, F. Pollara. Stopping Rules for Turbo Decoders. TMO Progress Report 42-142, http://tmo.jpl.nasa.gov/tmo/progress_report/42-142/142J.pdf, August 15, 2000, last accessed, April 2005.

[55] D. Garrett. Xu Bing. C. Nicol. Energy efficient turbo decoding for 3G mobile. International Symposium on Low-Power Electronics and Design, 6-7 August 2001, pp. 328-333

[56] S.A. Barbulescu. Sliding Window and Interleaver Design. Electronics Letters, Vol. 37, Issue 21, 11 October 2001, pp. 1299-1300.

[57] F. Dowla. Handbook of RF and Wireless Technologies. Newes, 2003. ISBN 0750676957. pp. 375-400.

[58] J. Qi. Turbo Code in IS-2000 CDMA Communications Under Fading. Wichita State University, MSc Thesis. October 1999, pp. 28.

[59] H. Sugimoto, T. Wang, X. Ping. A Sub-MAP Decoder Performance in IMT2000 Mobile Environment. 2nd International Symposium on turbo Codes & Related Topics, 4-7 September, 2000.

[60] R.G. Gallager. Low Density Parity Check Codes. IRE Transactions on Information Theory, Vol IT-8, pp.21-28, January 1962.

Reconfigurable Cores for Wireless Appliances: Turbo Codes

VOLUME 2 (OF 2)

Edward Brown

A themed portfolio submitted to
The Universities of

Edinburgh
Glasgow
Heriot Watt
Strathclyde

for the Degree of
Doctor of Engineering in System Level Integration

Contents