Implementation of an Unequal Error Protection Scheme for Scalable Foveated Image Communication

Muhammad Farooq Sabir and Rashmi Tripathi

Abstract

Data rate requirements may vary in multiuser heterogeneous environment. Hence, scalable image communication is used to allow adaption to a wide range of data rates and network conditions. In this project, we implement an unequal error protection scheme for Embedded Foveation Image Coding (EFIC) compressed images, using rate compatible punctured turbo codes. We implement this scheme as an embedded system for image communication over Rayleigh fading channels, and optimize the system for the TMS320C6701 DSP processor. The results show that this scheme gives better BER performance as compared to the uniform error protection scheme at high compression ratios. The optimization results indicate that a high level of optimization has been obtained by efficient management of data in the memory and writing a few routines in the processor's assembly language.

1 Introduction

In heterogenous environments, different users have different bandwidths and hence different data rate requirements. Scalable Image Communication is useful in such a multiuser heterogenous environments since it allows a different compressed version of the same image to be sent to different users. These different versions are generated by truncating a bitstream at different points. Such image compression techniques, where the bitstream can be truncated at any point to get different compression ratios, are known as scalable image coding

techniques. One example of scalable image coding is Embedded Foveation Image Coding (EFIC) [1], in which the data is arranged according to its importance for human visual system (HVS).

The different parts of the bitstream, having different importance, can be protected using different rate channel codes. Such schemes, used for providing different levels of error protection, are known as unequal error protection (UEP) techniques. In this project, we use UEP for EFIC compressed images, and implement this system over the TMS320C6701 DSP processor.

2 Background

In this section, we present a brief overview of embedded foveation image coding and rate compatible punctured turbo codes.

2.1 Embedded Foveation Image Coding

In embedded foveation image coding [1], bits are arranged such that the wavelet coefficients that contribute greater to the foveated visual quality are encoded and transmitted first. This unequal importance of data can be exploited to provide unequal error protection to different parts of the bitstream, thus increasing the overall data transmission rate. In this project, rate compatible punctured turbo codes have been used to provide unequal error protection to different parts of the bitstream.

2.2 Rate Compatible Punctured Turbo Codes

A turbo encoder [2–4] consists of two *recursive systematic convolutional* component encoders connected in parallel and separated by a random interleaver, as shown in Fig. 3 (a). After encoding, some of the code word bits are removed to achieve a higher rate code such that the higher rate codes are embedded within the lower rate codes. This process is known as rate compatible puncturing. At the receiver end, zeros are inserted at the punctured locations

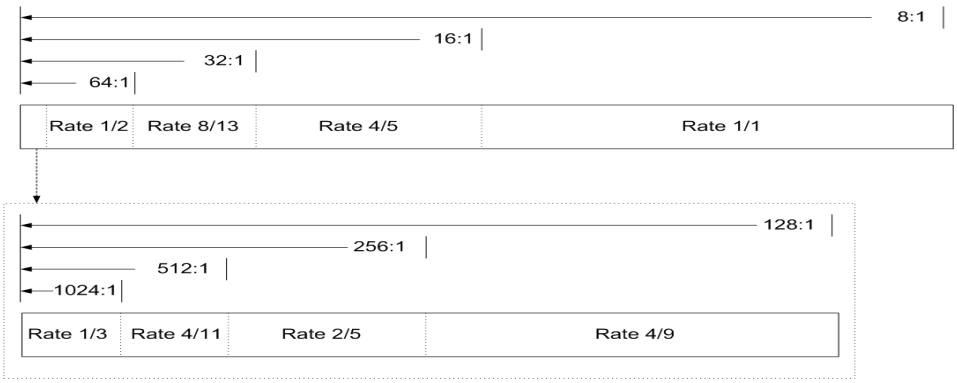


Figure 1: Different compression ratios corresponding to the different truncation points in EFIC bitstream.

before decoding. Either the maximum aposteriori probability (MAP) algorithm or the Soft Output Viterbi Algorithm (SOVA) [4] can be used to decode the bitstream.

2.3 Applications

Since the performance of the turbo codes is very close to the Shannon theoretical limits, they are rapidly being adopted in a number of fields. Turbo codes are being proposed as new coding standard for third generation wireless communications systems, like Code Division Multiple Access (CDMA) 2000, Universal Mobile Telecommunications System (UMTS), and for satellite and deep space applications. A few of the other standard bodies in wireless communications, that have adopted turbo codes, are Third Generation Partnership Project (3GPP), 3GPP2 and Digital Video Broadcasting (DVB) project.

3 Objectives

The primary objective of this project is to implement an unequal error protection scheme for the EFIC compressed images, using rate compatible punctured turbo codes. The puncturing scheme chosen for EFIC is shown in Fig. 1. As shown in this figure, the first block, corresponding to the highest compression ratio, is provided the maximum error protection. The level of error protection is reduced as we move towards the end of the bitstream.

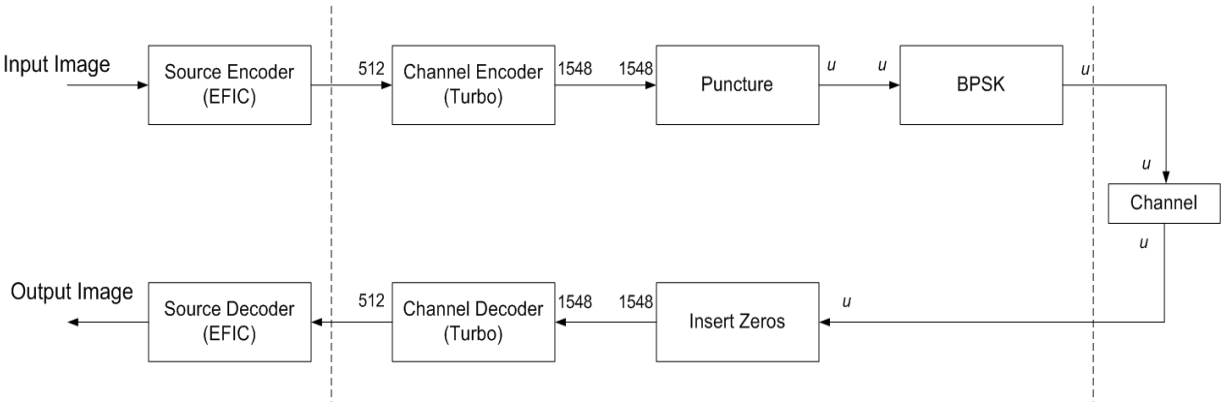


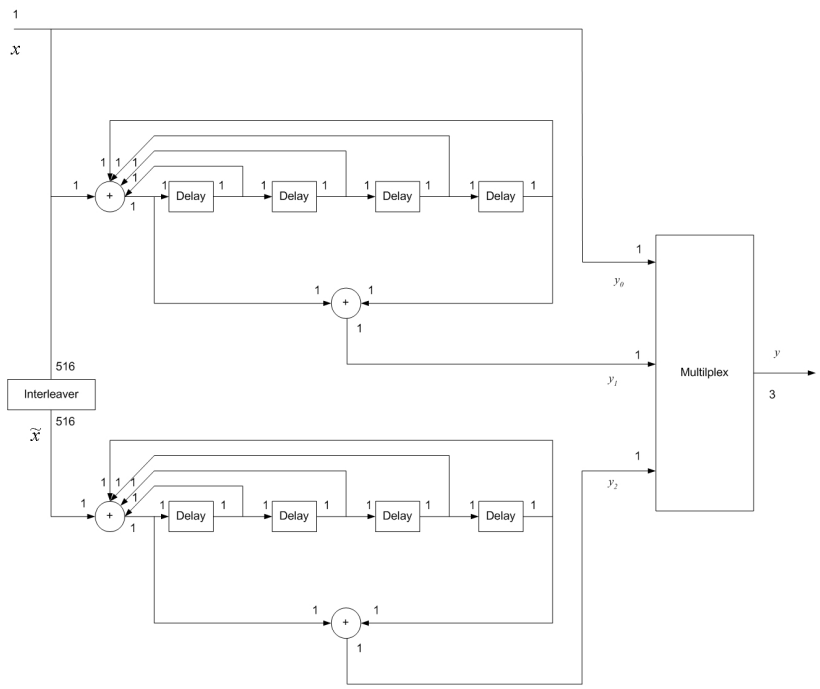
Figure 2: EFIC based image communication system with unequal error protection using Turbo Codes.

This real-time embedded system is implemented on TMS320C6701 floating point DSP processor, using modules written in, both, C and the processor assembly. The SOVA based turbo decoder has not been implemented on this DSP before. As the code generated by the DSP compiler is not completely optimized, we optimized the system with respect to the computation time and memory usage. Since large blocks of data are needed to be stored efficiently in the internal memory for every subsequent iteration, limited on-chip data memory size of 64KB imposes a challenging task on the optimization of highly complex encoder and iterative SOVA decoder.

4 Formal Modelling

The encoder, decoder and the BPSK modulator have been modelled as Synchronous dataflow (SDF) [5] actors. They take fixed number of input tokens to produce a fixed number of outputs. These numbers have been shown on the arcs in Fig. 2. The bitstream is divided into 512 blocks, each of size 512 bytes. The number of bits punctured from each block, depends on the block number, hence the 'puncture' and 'insert zeros' blocks are modelled as Boolean dataflow (BDF) actors. This scheme for puncturing and inserting zeros is shown in Fig. 3 (b) and (c).

The constituent blocks of both the encoder and the decoder can again be modelled as SDF



Block Nos.	Input Tokens	Output Tokens (u)
1-4	1548	1548
5-8	1548	1420
9-16	1548	1292
17-32	1548	1162
33-64	1548	1032
65-128	1548	839
129-256	1548	645
257-512	1548	516

(b)

Block Nos.	Input Tokens (u)	Output Tokens
1-4	1548	1548
5-8	1420	1548
9-16	1292	1548
17-32	1162	1548
33-64	1032	1548
65-128	839	1548
129-256	645	1548
257-512	516	1548

(c)

Figure 3: (a) Turbo Encoder [4]; Number of input and output tokens for (b) Puncture block (c) Insert Zeros block.

actors. The number of tokens consumed and produced by them, have been shown in Fig. 3 (a) and Fig. 4 (a) respectively. However, the generic SOVA decoder can be modelled as a boolean dataflow graph (BDF), to make it compatible with the multirate DSP applications. The schedule for the transmit end is

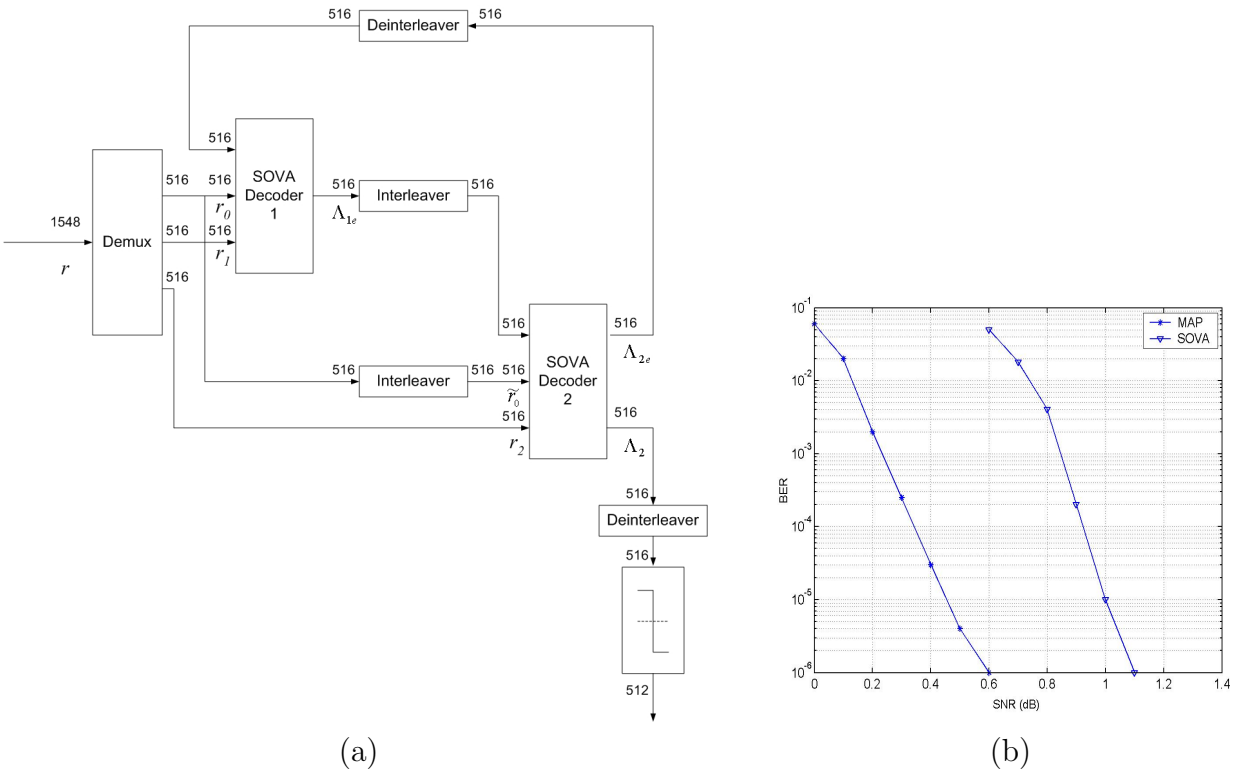
(Channel Encoder)(Puncture)(BPSK)

and the schedule for the receive end is

(Insert Zeros)(Channel Decoder).

5 Implementation

EFIC compressed bitstream was divided into 8 main blocks, each to be protected using different rate codes shown in Fig. 1. Each of these blocks was further divided into sub-blocks of size 512 bytes, due to the memory constraints of the DSP processor. Rate compatible punctured turbo codes were used to provide unequal error protection to these blocks.



(a) (b)

Figure 4: (a) SOVA based Turbo Decoder, (b) Performance comparison between iterative SOVA and iterative MAP decoder [4].

The system consists of two main sections: the transmit end and the receive end. The transmit section consists of the turbo encoder, the 'puncture' block and the binary phase shift keying (BPSK) modulator. The receive section consists of the 'insert zeros' block and the turbo decoder. Rayleigh fading channel was used as the transmission channel and it was simulated in C using the model given in [6].

Iterative SOVA [4, 7] decoder was used to decode the bitstream. Iterative SOVA was chosen because of its 3 times lower complexity as compared to the iterative MAP decoder. However the MAP decoder outperforms the SOVA by a coding gain of 0.55 dB shown in Fig. 4 (b) [4].

All the constituent modules of the system were first written in C and were then ported on the TMS320C6701 DSP processor. The encoder, the 'puncture' block and the modulator

Optimization Stage	Cycle Count (Millions)
Original Code	4.18
Level 3	3.07
Memory Optimization	1.20
Loop Unrolling	0.135
Assembly	0.019

(a)

Optimization Stage	Cycle Count (Millions)
Original Code	170
Level 3	119
Memory Optimization	21.1
Loop Unrolling	18.3
Assembly	10.2

(b)

Optimization Stage	Cycle Count (K)
Original Code	128.7
Level 3	97.3
Memory Optimization	20.1

(c)

Optimization Stage	Cycle Count (K)
Original Code	821.4
Level 3	659.1
Memory Optimization	505.6

(d)

Figure 5: Cycle counts for different optimization stages for (a) Turbo Encoder, (b) SOVA based Turbo Decoder, (c) Puncturing block and (d) Insert Zeros block.

were implemented in fixed point assembly, whereas the decoder and the 'insert zeros' block were implemented in floating point assembly. The code was optimized with respect to, both, execution time and memory usage. The different levels of optimization for encoder, decoder, puncture and insert zeros blocks are shown in Fig. 5. The version of Code Composer used was 1.0.

6 Innovation and Results

We implemented an unequal error protection scheme for EFIC compressed images, as a real-time embedded system, on TMS320C6701 DSP processor. Although the iterative MAP based turbo decoder [8] has been implemented on the TMS320C6x DSP processor, but the SOVA based turbo decoder has not been implemented before. We implemented the five blocks constituting our system, namely 'turbo encoder', 'puncture', 'BPSK modulator', 'insert zeros' and the 'turbo decoder'.

The code was optimized for both memory usage and execution time. As shown in Fig. 5 (a-d), high levels of optimization were achieved. All the steps of optimization shown include level 3 optimization except the original code. Results show that for the encoder, a reduction

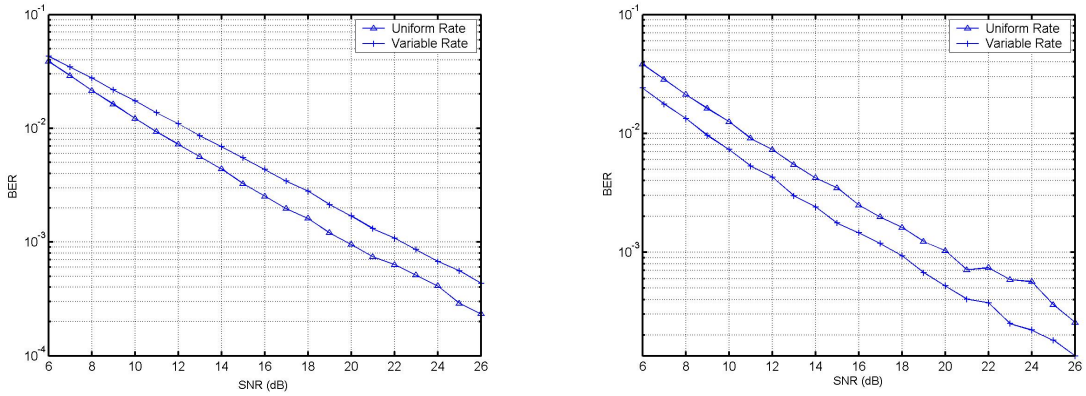
of approximately 63 times in the execution time, was achieved by writing the routine in assembly and loop unrolling as compared to only memory optimization. Similarly, for the decoder, a reduction of approximately 2 times in the execution time was achieved. Assembly routines were not written for the 'puncture' and the 'insert zeros' blocks, however memory optimization was performed. Execution time reduction of around 6 and 1.5 times was achieved, by memory optimization, for the puncture and the insert zeros blocks respectively.

The entire system was simulated for the EFIC coded images over the Rayleigh fading channel. Fig. 6 (a) shows the performance characteristics for EFIC, at a compression ratio of 8:1, in terms of Bit Error Rate (BER) vs. Signal to Noise Ratio (SNR) curves. It can be observed that at a BER of 10^{-2}, 1 dB coding gain is achieved for uniform rate puncturing as compared to the variable rate puncturing. This is so because the former provides larger number of redundant bits and hence offers more error protection.

Fig. 6 (b) shows that at a BER of 10^{-2}, 2 dB coding gain is achieved for variable rate puncturing as compared to the uniform rate puncturing. This is so because the former provides greater protection to the more important bits and truncating results in discarding the less protected, less important bits, that contain more errors. Truncating the uniformly protected bitstream does not reduce the BER significantly because the error is uniformly distributed over the entire bitstream.

7 Conclusions and Future Work

In this project, we implemented an unequal error protection scheme for EFIC compressed images, using rate compatible punctured turbo codes. The system was implemented over TMS320C6701 DSP processor and optimized with respect to execution time and memory. The results show that the unequal error protection scheme, used with EFIC, gives better BER performance as compared to the uniform error protection scheme, at higher compression ratios. The optimization results indicate that a high level of optimization has been obtained



(a) (b)

Figure 6: BER vs. SNR for EFIC (a) at 8:1 compression ratio (b) Bitstream truncated to give 32:1 compression ratio.

by efficient management of data in the memory and writing a few routines in assembly.

Amongst the future work, a natural extension of this project could be a fixed point implementation of the decoder to promote its use in the industry. It could be further extended to provide unequal error protection using spatial diversity. Its implementation as a real-time embedded system, could be very useful in exploiting the advantages associated with the space-time codes.

References

[1] Z. Wang and A. C. Bovik, "Embedded Foveation Image Coding," *IEEE Transactions on Image Processing*, vol. 10, pp. 1397–1410, Oct. 2001.

[2] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: turbo-codes," *IEEE Transactions on Communications*, vol. 44, pp. 1261 –1271, Oct. 1996.

[3] A. Ushirokawa, T. Okamura, N. Kamiya, and B. Vucetic, "Principles of turbo codes and their applications to mobile communciations," *IEICE Transactions on Fundamentals*, vol. E81-A, pp. 1320–1329, July 1998.

[4] B. Vucetic and J. Yuan, "Turbo codes." Kluwer Academic Publishers, 2000.

[5] E. A. Lee and D. G. Messerschmitt, "Synchronous data flow," *Proceedings of the IEEE*, vol. 75, pp. 1235–1245, Sept. 1987.

[6] T. S. Rappaport, "Wireless communications, principles and practice, 2002." Prentice Hall.

[7] J. Hagenauer, E. Offer, and L. Papake, "Iterative decoding of binary block and convolutional codes," *IEEE Transactions on Information Theory*, vol. 42, pp. 429–445, Mar. 1996.

[8] J. Nikolic-Popovic, "Implementing a MAP Decoder for cdma2000 Turbo Codes on a TMS320C62x DSP Device." Texas Instruments Application Report, SPRA629, May 2000.