High Performance Low Density Parity Check and Turbo Decoding FPGA Implementation

Ryan Shoup
MIT Lincoln Laboratory
Email: Shoup@LL.MIT.EDU

Introduction

In the recent years, near-capacity approaching error correction encoding techniques including turbo codes [1] and Low Density Parity Check (LDPC) codes [2] have gained considerable attention in the open literature. These encoding techniques are typically discussed as being decoded in an iterative manner resulting in near-capacity performance.

The difficulty with these near-capacity codes is the need to perform multiple iterations, limiting the throughput of the decoder. This paper discusses two iterative decoders, a 64-ary orthogonal Serial Concatenated Convolutional Code (SCCC) Turbo decoder and a Low Density Parity Check (LDPC) decoder. These two iterative decoders were designed for FPGA implementation. Increasing the throughput of the decoder implementations was of primary concern.

64-PPM Turbo Decoder

A turbo decoder was designed to decode an optical signal transmitted using 64-PPM modulation. The transmitted data was encoded and then mapped to a 64-PPM symbol as shown in Figure 1.

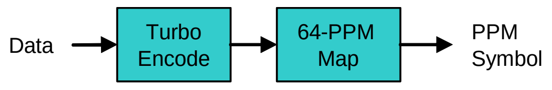

Figure 1: Turbo Encoding and 64-PPM Mapping

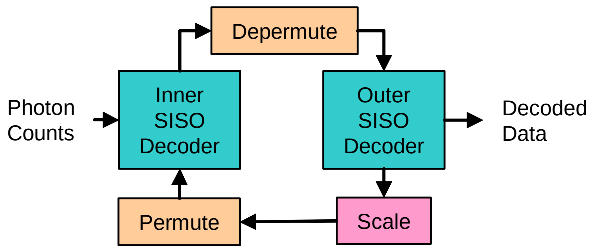

Figure 2: 64-PPM Turbo Decoding

In Figure 1, information bearing data bits are turbo encoded using a SCCC code. The rate of the outer code was ½ with generator polynomials of $\{5,7\}_{octal}$. The inner code was a simple rate-1 accumulator. Six outputs of the inner encoder are then mapped to 1 of 64 PPM modulation symbols and transmitted through the optical channel.

This work was sponsored by NASA under contract FA8721-05-C-0002. Opinions, interpretations, conclusions and recommendations are those of the authors, and are not necessarily endorsed by the United States Government.

The 64-PPM turbo decoder is diagrammed in Figure 2.

The input to the turbo decoder consisted of 64 photon counts per 64-PPM symbol interval. The output of the turbo decoder represented the decoded data.

The first optimization used to improve the turbo decoder was to employ the well-known Max log Map algorithm coupled with scaling of the outer Soft Input Soft Output (SISO) decoder. Scaling of the outputs is known to improve the performance of Max log Map decoding algorithms for standard binary/quaternary modulation schemes [3]. The application of the scaling factor for 64-PPM modulation also seemed to produce desirable results. Figure 3 shows the results of applying the computationally intensive MAP algorithm compared with the less computationally intensive Max Log Map with scaling algorithm, which was implemented on an FPGA. The performance penalty of 0.2 dB trades well for a significant improvement in throughput.

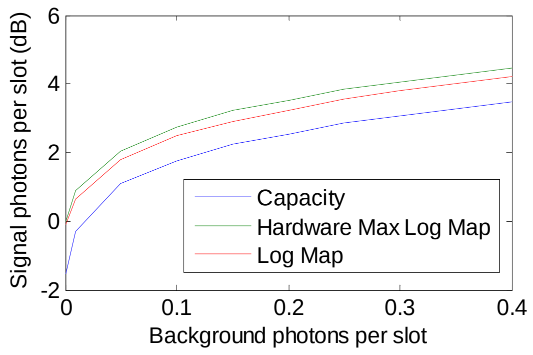

Figure 3: Max Log Map w/ Scaling vs Log Map

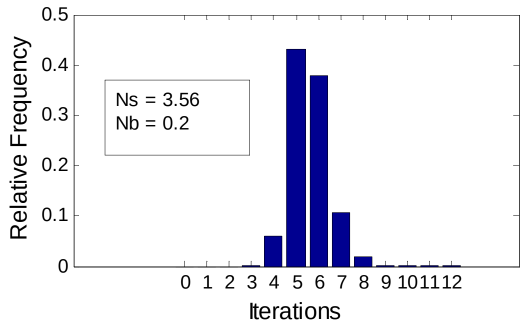

Figure 4: Turbo Decoder Iteration Histogram

A second optimization that can be used to improve the performance of the turbo decoder is to employ a variable number of iterations per codeword. As shown in Figure 4, the number of iterations needed for the turbo decoder to decode a particular codeword is variable. The hardware decoder associated with the results shown in Figure 3 used a fixed number of iterations of 12. By terminating the decoding process when the codeword is finished decoding, the average throughput of the decoder more than doubles. The amount of additional memory needed turns out to relatively small. The ability to pool 10 codewords essentially results in the performance shown in Figure 3 while doubling the throughput of the decoder.

LDPC Decoder

A decoder was developed to decode a (8176,7154) quasi-cyclic code. The details of a quasi-cyclic code and the associated encoding process are not discussed here but can be found in [4]. The corresponding LDPC decoder, which was implemented on an FPGA, is shown in block diagram form in Figure 5.

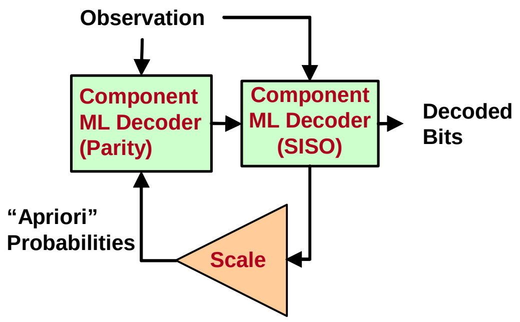

Figure 5: Max Log Map w/ Scaling vs Log Map

All of the optimizations discussed above for the 64-PPM turbo decoder are also applicable for the LDPC decoder. The "apriori" probabilities are scaled in essentially, the same manner as done for the turbo decoder. The performance improvement due to the scaling of the apriori probabilities is shown in Figure 6. Additionally, the max log map algorithm is employed in place of a map algorithm at a negligible loss in performance. Furthermore, the iteration histogram at the waterfall region of the bit error rate curve allows for pooling of the codewords to result in a doubling of the system throughput. The iteration histogram is provided in Figure 7.

A final optimization employed, suitable for 8-PSK, modulation is to choose a constellation mapping that results in the best BER vs E_B/N_0 performance for the LDPC decoder. A non-Grey constellation mapping resulted in a decoding performance improvement shown in Figure 8. Also in the figure, the impact of scaling of the "apriori" probabilities is shown.

Summary

A 64-PPM turbo decoder and LDPC decoder were implemented on an FPGA. Design optimization focused on throughput. These optimizations resulted in a throughput increase while maintaining decoder performance to 1.0 dB from capacity for the 64-PPM turbo decoder and 1.3 dB from capacity for the LDPC decoder for both QPSK and 8-PSK modulation.

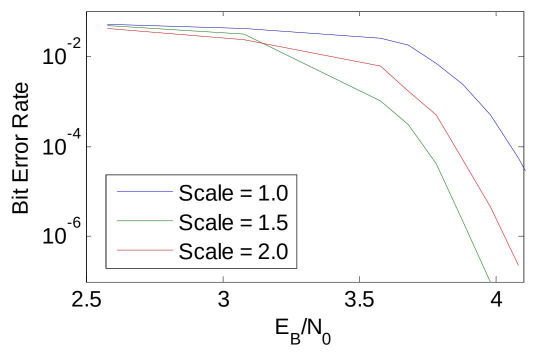

Figure 6: LDPC Decoder Performance with Scaling

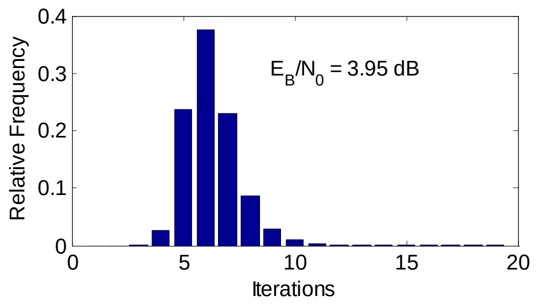

Figure 7: LDPC Decoder Iteration Histogram

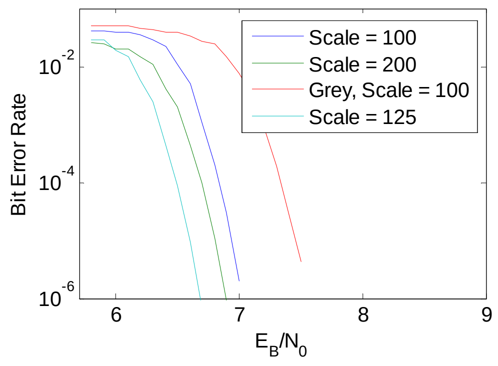

Figure 8: LDPC 8-PSK Performance

References

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon Limit Error-Correcting Coding and Decoding: Turbo-Codes," ICC '93, Geneva, Switzerland, May 1993.

[2] R. G. Gallagher, "Low Density Parity Check Codes," *IRE Transactions on Information Theory*, Vol 8, No 1, Jan 1962.

[3] J. Vogt and A. Finger , "Improving the max-log-MAP turbo decoder," *Electronics Letters*, Vol. 36 No. 23., Nov 2000.

[4] Z. Li, L. Chen, L. Zeng, S. Lin, and W. Fong, "Efficient Encoding of Quasi-Cyclic Low-Density Parity-Check Codes," *IEEE Transactions on Communications*, Vol. 54, No. 1, Jan 2006.