*ICT-MobileSummit 2009 Conference Proceedings*
*Paul Cunningham and Miriam Cunningham (Eds)*
*IIMC International Information Management Corporation, 2009*
*ISBN: 978-1-905824-12-0*

*Poster Paper*

# 3D Duo Binary Turbo Decoder Hardware Implementation

Timo LEHNGIK-EMDEN[1], Matthias ALLES[1], Norbert WEHN[1]
[1]*Microelectronic Systems Design Research Group, University of Kaiserslautern, Erwin-Schroedinger Str., Kaiserslautern, 67663, Germany*
*Tel: +49 631 2053638, Fax: +49 631 2054437,*
*Email: {lehnigk, alles, wehn}@eit.uni-kl.de*

**Abstract:** Each digital communication system needs channel coding to provide a certain quality of service. With the introducation of advanced channel codes like turbo codes and LDPC codes, error correcting near theoretical shannon limit became possible. Many applications require a low error floor in addition. The classical turbo code cannot meet this demand. Increasing the number of components codes, non-binary component codes or code concatenation are solutions for this problem, but come with a large complexity increase. In 2007 a new class of turbo codes, the 3D turbo code, was introduced by Berrou. The 3D turbo code provides a very good convergence and a large minimum distance at a low complexity. To the best of our knowledge this paper presents the first hardware implementation of a 3D turbo decoder. In addition we compare the implementation complexity of the 3D turbo decoder with the 8 and 16-state duo binary turbo decoder on FPGA and in 65nm ASIC technology.

**Keywords:** coding, Channel decoder, duo binary, turbo code, third dimension, 3D, implementation, FPGA, ASIC, complexity

## 1. Introduction

Channel coding is one vital part of each digital communication system. Advanced coding schemes like turbo codes [1] and LDPC codes [2] step towards shannon limit [3][4]. Binary turbo codes [1] were used in many communication standards like, UMTS and LTE [5]. For many current and future applications like DVB-S2, a high minimum hamming distance (MHD) or an error floor below $10^{-9}$ is mandatory which cannot be provided by the classical binary turbo code. With the introduction of the 16-state duo binary turbo code in 1999 [6], the error floor has been dramatically lowered, below FER=$10^{-7}$ for a wide range of code rates [7]. The price to pay is a high implementation complexity of about 2 times compared to a 8-state duo binary turbo code, see Section 4.2 [8]. A further increasing of the MHD can be archived by increasing the number of component encoders. The complexity of such a decoder increases linearely with this number. In 2007 Berrou presented an extension for the turbo code to lower the error floor at a lower complexity. This so called 3D turbo code has an extreme low error floor [9]. In this paper we show the implementation complexity of a decoder for this new class of turbo codes to the best of our knowledge for the first time. In addition we compare the implementation complexity to the 16 and 8-state duo binary turbo code decoder. The paper is structured as follows. Section 2. gives a short introduction to turbo codes and the 3D extension. The next Section 3. presents a hardware architecture of a 3D turbo decoder. The results in Section 4. show simulation on the communications performance and the implementation complexity on a FPGA and ASIC platform in a 65nm technology. The paper is summarized in Section 5.

 

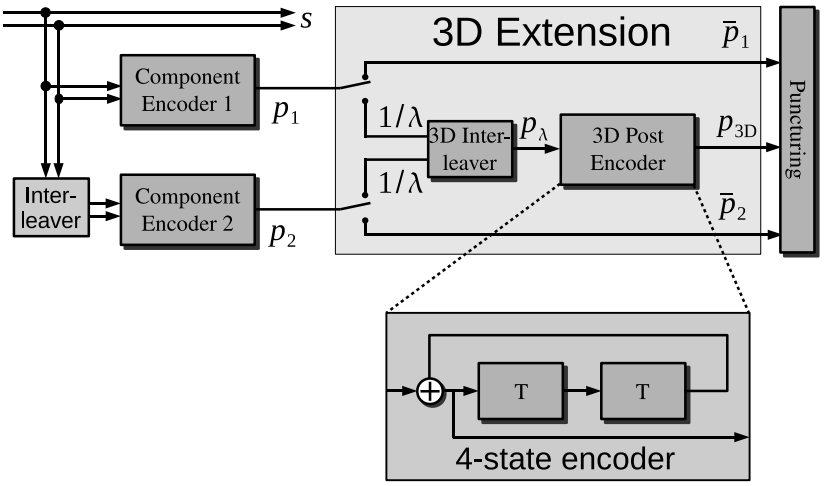


Figure 1: Duo binary turbo encoder with 3D extension

## 2. Turbo Codes

Turbo codes in general consist of a serial or parallel concatenation of two codes, so called component codes, and an interleaver. While the first component code encodes the information in the original order, the second one gets the information in a permuted order, see Figure 1. In all standards convolutional codes are used as component codes.

Decoding of turbo codes is an iterative process where probabilistic information is exchanged between component decoders [10]. Iterative decoding implies a big challenge with respect to low latency and high throughput decoders.

### *2.1 Duo binary turbo codes*

Duo-binary turbo codes [7] were introduced in 1999 and even outperform their binary counterpart in sense of communications performance. While for binary turbo codes these component codes work on single bits only, for duo-binary turbo codes bit couples $s_k = (s_k^{(0)}, s_k^{(1)})$ are used as input at the time step $k$.

### *2.2 3D Extension*

Turbo codes have a convergence near the theoretical limit, but the error floor is still in the region of FER=$10^{-7}$ for the 16-state duo binary turbo code [7]. If an application requires a better error correction capability, code concatenation or increasing the number of component codes [9] is mandatory. In general this solutions comes with a high complexity increasing. To overcome this problem, an extension for the existing turbo codes so called 3D turbo codes were introduced in 2007 by Berrou [9]. The main idea to increase the MHD is to encode parts of the parity bits of the two classical component encoders. This approach can be used for binary and duo binary turbo codes. One example of a 3D turbo encoder is shown in Figure 1. On the left side a state-of-the-art duo binary turbo encoder with two component codes and one interleaver is presented. After the encoding of the systematic information $s = \{s_0, ..., s_k, ..\}$ every $\lambda$-th parity bit of $p_1$ and $p_2$ is taken out from the stream. This new stream $p_\lambda$ goes permutated into the 3D post encoder which calculates the new parity information $p_{3D}$.

More details about which encoder to take as post-encoder the reader is referred to [9]. In our case a 4-state recursive convolutional encoder is taken which is good choice for short and medium blocksizes ($< 5000$) [9]. The turbo interleaver must not be changed, algebraic interleavers like almost regular permutation (ARP) [11] or dither relative

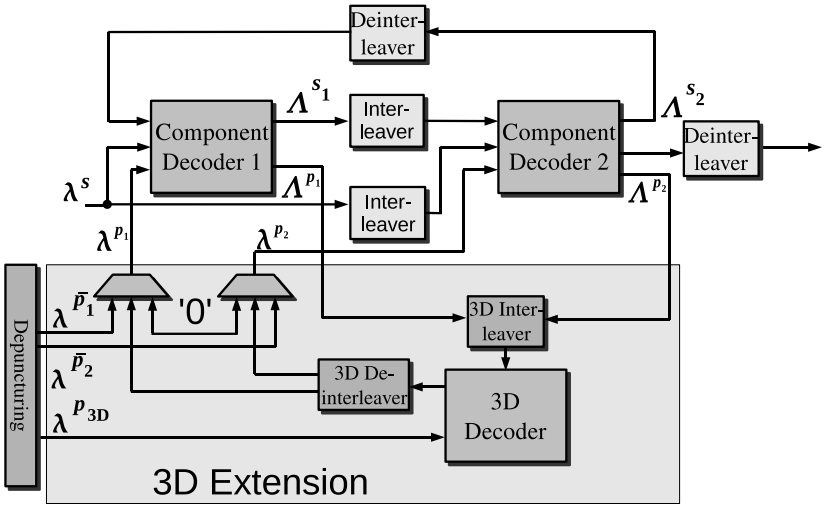


Figure 2: 3D duo binary decoder

prime (DRP) interleavers have an excellent performance and are suitable for parallel hardware implementation. For the 3D interleaver a relative prime (RP) interleaver can be taken with respect to the implementation complexity.

On the decoder side, every component encoder needs one component decoder. These three decoders exchange extrinsic information on the systematic bits $s$ and the parity bits $p_\lambda$. This extrinsic information exchange is referred as turbo principle. Figure 2 shows a possible decoder realization.

At the beginning of the decoding process the two component decoders 1 and 2 calculate extrinsic information $\Lambda^s$ and $\Lambda^p_{1,2}$ on the information bits and the parity bits respectively. In the first iteration no information on the parity bits $p_\lambda$ of the 3D decoder are available, therefor they are taken as equal probable which corresponds to a zero in the logarithmic domain. After that the 3D component decoder estimates new information $\lambda^{p_{1,2}}$ on the punctured parity bits $p_\lambda$ including the channel information $\lambda^{p_{3D}}$ and the a-priori information from the component decoders 1 and 2. This processing is called one iteration. Now the component decoder 1 can can go on with the next iteration.

The three component decoders perform a *maximum a posteriori probability* (MAP) decoding on bit level for the binary turbo code or on symbol level for the duo binary turbo code respectively. They use the BCJR [12] algorithm transformed in the logarithmic domain, the so called Log MAP algorithm [13], to decrease the implementation complexity. However, the suboptimal Max-Log MAP algorithm with *extrinsic scaling factor* (ESF) is more suitable for implementations. In comparison to the optimal algorithm the Max-Log MAP results in a performance loss below 0.2 dB [13][14]. Moreover it was shown in [15] that, when employed in turbo decoding, one does not require knowledge of the SNR. The Max-Log MAP algorithm consists of a forward and a backward recursion. It computes for each possible information and parity symbol an *a posteriori probability* (APP) LLR $\Lambda^s, \Lambda^{p1}, \Lambda^{p2}, \lambda^{p1}$ and $\lambda^{p2}$. All exchanged information is represented as *log likelihood ratio*s (LLR) to decrease the computation complexity.

## 3. 3D Turbo Decoder Hardware Architecture

Figure 3 shows a conventional duo binary turbo decoder with a separate extension for the third component decoder. The standalone duo binary turbo decoder is composed

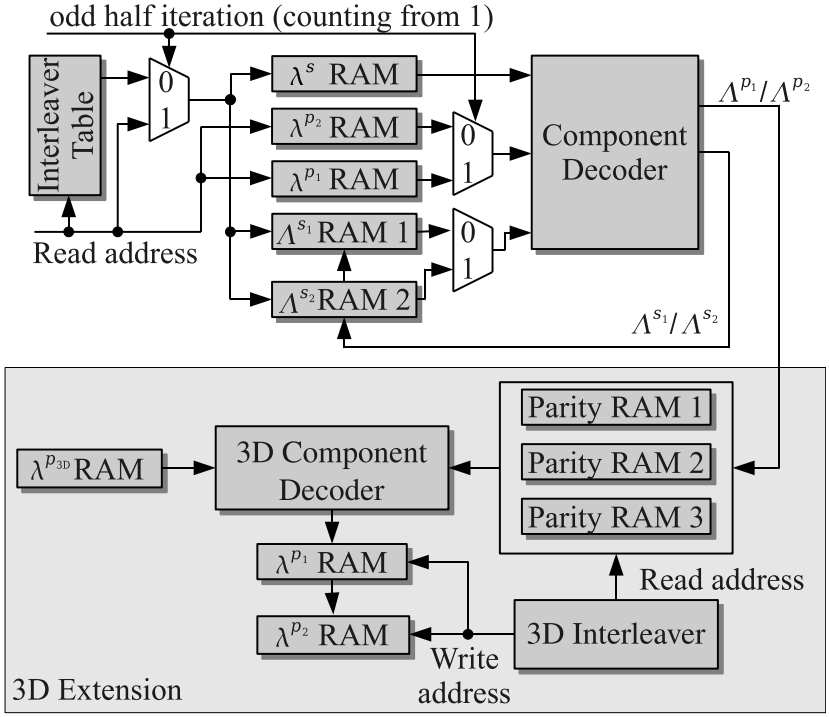


Figure 3: Duo binary turbo decoder with 3D extension

of a component decoder, an interleaver, and several RAMs. These RAMs are needed to store the received channel values ($\lambda^s, \lambda^{p_1}$, and $\lambda^{p_2}$) and the extrinsic information exchanged between both component decoders ($\Lambda^{s_1}, \Lambda^{s_2}$). Only a single component decoder exists. It can either act as component decoder 1 (computing $\Lambda^{s_1}$ and $\Lambda^{p_1}$) or as component decoder 2 (computing $\Lambda^{s_2}$ and $\Lambda^{p_2}$). Depending on the half iteration the decoder is processing, the appropriate input values for the MAP decoding are selected. Interleaving is performed with an interleaver table. When processing component decoder 2 (interleaved parity information), it is necessary to read the systematic and extrinsic information in an interleaved manner to match the order of the interleaved transmitted parity information.

Extending the duo binary turbo decoder to the third dimension is accomplished by adding

- another component decoder (binary 4-state decoder),
- a RAM for the channel values of the postencoded parity bits ($\lambda^{p_{3D}}$),
- a 3D interleaver, and
- three parity RAMs that have to hold the extrinsic values on the parity couples $\Lambda^{p_1}$ and $\Lambda^{p_2}$.

The $\lambda^{p_1}/\lambda^{p_2}$ RAMs are drawn in the extension area for the sake of clarity. They are updated by the results of the 3D component decoder, and are the same as the RAMs drawn above. Note that the information on the postencoded parity bits is initialized to zero in these RAMs. These gaps are filled by the 3D extension during decoding.

Figure 4 shows the scheduling and memory accesses of the decoding process. In each half iteration either component decoder 1 or component decoder 2 is processed. After two half iterations a full decoding iteration is finished. Note that Figure 4 does not

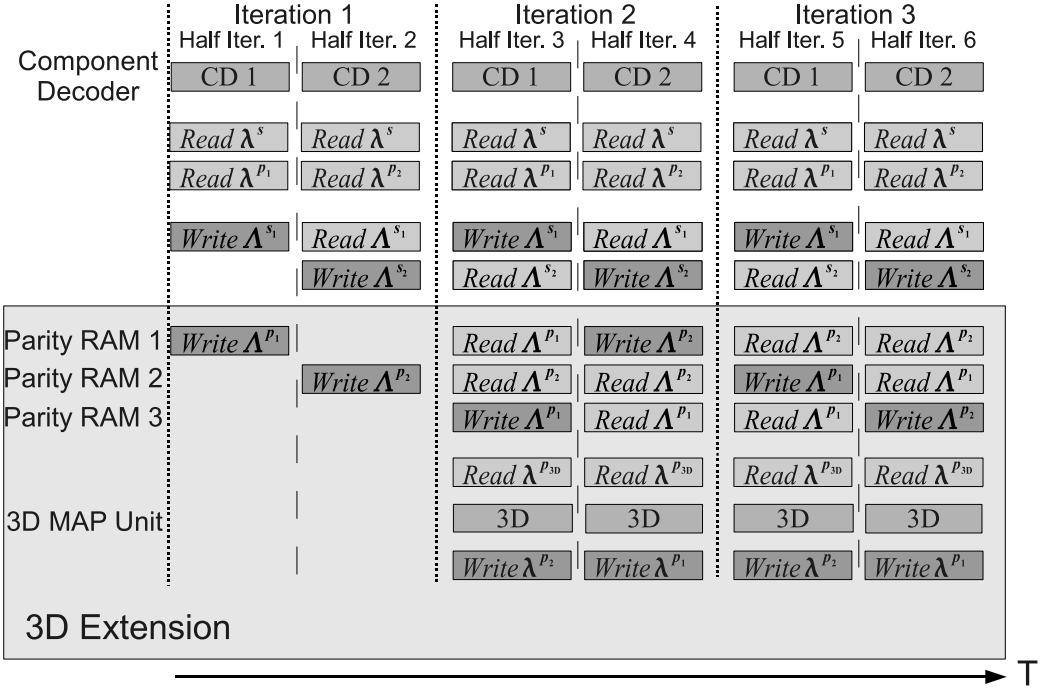


Figure 4: Memory access scheme

show the timing within a single half iteration itself. It is meant to show which memory is read or written in which half iteration.

Conventional turbo decoding will only read the systematic information $\lambda^s$, the parity information $\lambda^{p_1}$, and the extrinsic information $\Lambda^{s_2}$ in odd half iterations or $\lambda^s$, $\lambda^{p_2}$, and $\Lambda^{s_1}$ in even half iterations, respectively. In odd half iterations $\Lambda^{s_1}$ is computed, in even half iteration $\Lambda^{s_2}$, respectively. However, with the 3D extension the memories for $\lambda^{p_1}$ and $\lambda^{p_2}$ are updated during decoding as shown in the following.

Within each half iteration not only the extrinsic values on the systematic information $\Lambda^s$ but also the extrinsic values on the parity information $\Lambda^p$ have to be computed and stored for the extension. The parity RAMs as shown in Figure 3 have to store $\Lambda^p$. In the first half iteration $\Lambda^{s_1}$ and $\Lambda^{p_1}$ are computed. Since there is no extrinsic information on the parities available at this early stage of decoding the 3D component decoder is still idle. In the second half iteration component decoder 2 is processed, taking into account the previously calculated $\Lambda^{s_1}$. At the same time $\Lambda^{s_2}$ and $\Lambda^{p_2}$ are calculated and stored. Only in the third half iteration, when $\Lambda^{p_1}$ and $\Lambda^{p_2}$ are available, the 3D component decoder starts its operation. In order to achieve the same decoding throughputs with the 3D extension, duo binary component decoder and the 3D component decoder work in parallel. No additional decoding latency is introduced. The third parity RAM is needed to store the extrinsic parity information that is calculated while the 3D component decoder is working on the previously calculated $\Lambda^{p_1}$ and $\Lambda^{p_2}$. Depending on the half iteration either $\Lambda^{p_1}$ and $\Lambda^{p_2}$ is updated and stored in the third parity RAM that is not currently read from. The 3D component decoder reads $\lambda^{p_{3D}}$, $\Lambda^{p_1}$, and $\Lambda^{p_2}$ while updating either the parity RAM $\lambda^{p_1}$ (even half iterations) or $\lambda^{p_2}$ (odd half iterations).

## 4. Results

The communications performance of the hardware decoder is simulated with a bit-true software model of the hardware decoder. The implementation complexity is given for

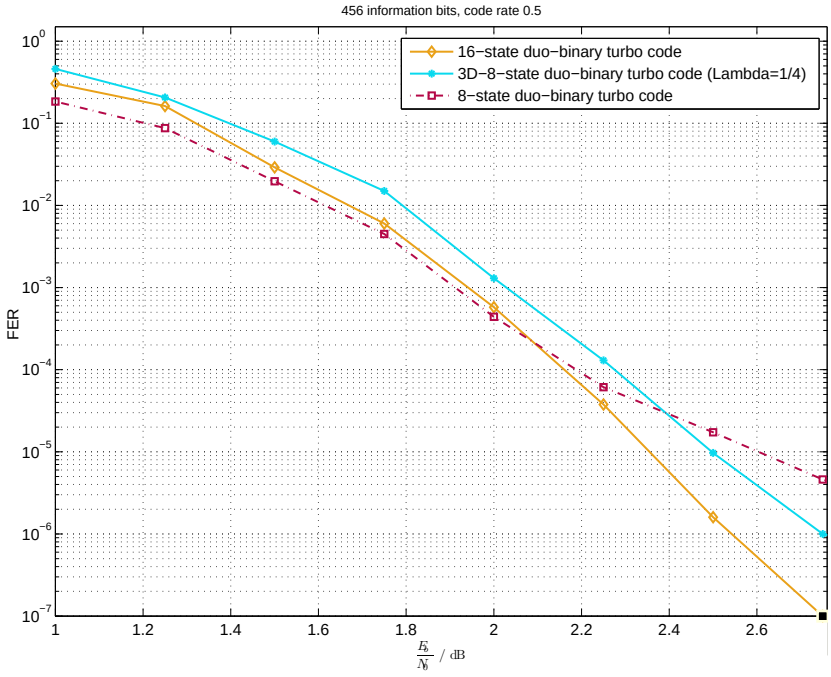


Figure 5: Communications performance comparison - 456 info bits

the FPGA and the ASIC after synthesis before place and route.

### *4.1 Communications Performance*

In Figure 5 a block of 456 information bits and a code rate of 0.5 was simulated taking an 8-state, a 16-state duo binary and an 8-state duo binary with 3D extension code. The 8 and 16-state duo binary codes have the same convergence, see Figure 5. The 8-state duo binary turbo code has a high error floor which begins at 2.1 dB. It can be seen that the 3D code beats the 8-state duo binary code in the error floor and loses only 0.1 dB in convergence. Both 16-state and 3D 8-state duo-binary turbo code show no error floor down to a frame error rate (FER) of $10^{-7}$. This is the expected behavior which is shown in [9].

### *4.2 Implementation Complexity*

Table 1 shows FPGA implementation and ASIC synthesis results for three different duo binary turbo decoders, namely 8-state, 8-state with 3D extension, and 16-state turbo decoders are shown. The 16-state turbo decoder allows for better communications performance, but the implementation complexity is much higher compared to that of a 8-state turbo decoder. 78% more LUTs and 58% more flip-flops (FFs) are required. Furthermore, the maximum clock frequency decreases from 190 MHz to 166 MHz due to increased routing congestions on the FPGA. Hence, the maximum throughput is decreased from 58.5 Mbit/s to 51.1 Mbit/s. Similar results are obtained for the ASIC implementation. The 16-state decoder requires 73% more logic resources yielding an overall area overhead of 39%.

In contrast, the 3D extended turbo decoder with 8-states requires less logic resources than the 16-state decoder. 33% more LUTs and 43% more FFs are required, which is significantly less compared to the 16-state decoder. The reduced logic resources in use furthermore relax the routing congestions which allows for a clock frequency of 190 MHz as in the case of the normal 8-state decoder. Compared to the 16-state decoder

| **Turbo Code** | **Duo Binary** | **Duo Binary** | **3D Duo Binary** |
|---|---|---|---|
| **Trellis States** | **8** | **16** | **8+4** |
| **Infoword Size** | 128-4800 bits | | |
| **Codeword Size** | $\leq$9600 bits | | |
| **Code Rates** | $\geq 1/2$ | | |
| **Input Quantization** | 6 bit | | |
| **Algorithm** | MaxLog-MAP | | |
| **ESF** | 0.0<...<1.0 | | |
| **Max. Iterations** | 6 | | |
| **Technology** | **65nm CMOS** | | |
| **Clock Frequency** | 500 MHz | | |
| **Logic [mm$^2$]** | 0.170 | 0.294 (+73%) | 0.203 (+19%) |
| **Memory [mm$^2$]** | 0.249 | 0.290 (+16%) | 0.269 (+8%) |
| **Overall Area [mm$^2$]** | **0.419** | **0.584 (+39%)** | **0.472 (+13%)** |
| **Payload Thrpt.[Mbit/s]** | 46.3 - 153.9 | | |
| **Technology** | **Xilinx Virtex5 VLX110-3 FPGA** | | |
| **Clock Frequency** | 190 MHz | 166 MHz | 190 MHz |
| **6-Input LUTs** | 9865 | 17619 (+78%) | 13190 (+33%) |
| **Flip-Flops** | 4585 | 7261 (+58%) | 6572 (+43%) |
| **BRAMs** | 12 | 13 (+8%) | 15 (+25%) |
| **DSP48E** | 6 | 6 | 8 (+25%) |
| **Payload Thrpt.[Mbit/s]** | 17.6 - 58.5 | 15.4 - 51.1 | 17.6 - 58.5 |

Table 1: Turbo decoder implementation results

it is only necessary to spend two more block RAMs (BRAMs), and two more DSP48E slices. The latter are used to multiply with the extrinsic scaling factor. The ASIC implementation of the 3D extended turbo decoder needs only 13% more area than the 8-state turbo decoder in contrast to 39% of the 16-state turbo decoder.

## 5. Conclusion

In this paper the implementation complexity of three turbo decoders 8-state duo binary, 16-state duo binary and 8-state with 3D extension is compared. The implementation complexity of the 8-state 3D turbo decoder is significantly smaller than the complexity of the 16-state duo binary decoder by losing 0.1 dB in convergence but getting an error floor at a FER of $10^{-4}$. The 3D code beats the 8-state code in the error floor, increasing the VLSI area of the decoder by only 13%. The 16-state decoder has the best communication performance. The implementation complexity is the price to pay. For future standards which require a very low error floor the 3D turbo code is good choice because of its good communications performance and its moderate implementation complexity.

## References


[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon Limit Error-Correcting Coding and Decoding: Turbo-Codes," in *Proc. 1993 International Conference on Communications (ICC '93)*, (Geneva, Switzerland), pp. 1064–1070, May 1993.

[2] R. G. Gallager, *Low-Density Parity-Check Codes*. Cambridge, Massachusetts: M.I.T. Press, 1963.

[3] S. ten Brink, "Rate one-half code for approaching the Shannon limit by 0.1 dB," in *Electronics Letters*, vol. 36, pp. 1293–1294, 20 July 2000.

[4] S. Chung, G. Forney, T. Richardson, and R. Urbanke, "On the Design of Low-Density Parity-Check Codes Within 0.0045dB of the Shannon Limit," *IEEE Communications Letters*, vol. 5, pp. 58–60, Feb. 2001.

[5] "3GPP LTE (Long Term Evolution) Homepage."

[6] C. Berrou and M. Jezequel, "Non-Binary Convolutional Codes for Turbo Coding," *Electronic Letters*, vol. 35, pp. 39–40, January 1999.

[7] C. Douillard and C. Berrou, "Turbo Codes with rate-m/(m+1) constituent convolutional codes," *IEEE Transactions On Communications*, vol. 53, pp. 1630–1638, oct 2005.

[8] M. Alles, T. Lehnigk-Emden, U. Wasenmller, and N. Wehn, "Implementation Issues of Turbo Synchronization with Duo-Binary Trubo Decoding," in *Proc. 19th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC) 2007*, (Athens, Greece), Sept. 2007.

[9] C. Berrou, A. Graell i Amat, Y. Ould Cheikh Mouhamedou, C. Douillard, and Y. Saouter, "Adding a Rate-1 Third Dimension to Turbo Codes," in *Proc. IEEE Information Theory Workshop ITW '07*, pp. 156–161, 2–6 Sept. 2007.

[10] C. Berrou, "The Ten-Year-Old Turbo Codes are Entering into Service," *IEEE Communications Magazine*, vol. 41, pp. 110–116, Aug. 2003.

[11] C. Berrou, Y. Saouter, C. Douillard, S. Kerouedan, and M. Jezequel, "Designing good permutations for turbo codes: toward a single model," in *Proceedings of ICC 2004*, vol. 1, pp. 341–345, June 2005.

[12] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate," *IEEE Transaction on Information Theory*, vol. IT-20, pp. 284–287, Mar. 1974.

[13] P. Robertson, E. Villebrun, and P. Hoeher, "A Comparison of Optimal and Sub-Optimal MAP decoding Algorithms Operating in the Log-Domain," in *Proc. 1995 International Conference on Communications (ICC '95)*, (Seattle, Washington, USA), pp. 1009–1013, June 1995.

[14] P. Robertson, P. Hoeher, and E. Villebrun, "Optimal and Sub-Optimal Maximum a Posteriori Algorithms Suitable for Turbo Decoding," *European Transactions on Telecommunications (ETT)*, vol. 8, pp. 119–125, March–April 1997.

[15] A. Worm, P. Hoeher, and N. Wehn, "Turbo-Decoding without SNR Estimation," *IEEE Communications Letters*, vol. 4, pp. 193–195, June 2000.