EFFICIENT FIELD PROGRAMMABLE GATE ARRAY IMPLEMENTATION OF A CONVOLUTIONAL TURBO CODE FOR LONG TERM EVOLUTION SYSTEMS

CRISTIAN ANGHEL, CRISTIAN STANCIU, CONSTANTIN PALEOLOGU

Key words: Long term evolution (LTE), Turbo codes, Field programmable gate array (FPGA) implementation, Maximum logarithmic (Max log) - maximum a posteriori probability (MAP).

This paper describes an efficient Field programmable gate array (FPGA) implementation of a convolutional turbo code (CTC) decoder for long term evolution (LTE) standard, release 8, using maximum logarithmic – maximum a posteriori probability (Max Log MAP) algorithm. The considered coding rate is 1/3 (the native coding rate), the puncturing procedure not being taken into discussion here, and the number of turbo iterations is chosen as 3, without reducing the generality of the reported results. The hardware implementation targets a Xilinx Virtex 5 XC5VFX70T device, from a Xilinx ML507 evaluation board.

1. **INTRODUCTION**

Turbo codes were introduced by Berrou, Glavieux, and Thitimajshima [1–3], but the initial perception of the method was not a promising one, especially because of the good reported results compared with the existing forward error coding (FEC) solutions. Once the authors were able to prove the strengths and the validity of the proposed architecture, more and more standards started to include turbo codes, as recommended in the first phase and fully mandatory afterwards. This evolution was possible as the processing power increased and the complexity of the turbo codes versus the classical convolutional codes was not a bottleneck anymore. Nowadays, the computational power of devices such as digital signal processors (DSPs) or field programmable gate arrays (FPGAs) allows the implementation of turbo encoding/ decoding, but the complexity of the general architecture (for example, the entire digital baseband processing for an long term evolution (LTE) base station) requires further optimization for all the blocks in the scheme.

„Politehnica" University of Bucharest, Iuliu Maniu 1–3, Sect 5, Bucharest, room B102, E:mail: {canghel, cristian, pale}@comm.pub.ro

One of the most important standardization groups which early adopted the turbo codes is the third-generation partnership project (3GPP) [4]. In the first version of Universal Mobile Telecommunications System (UMTS), released in 1999, turbo codes were included for the first time as an FEC solution, in addition to the traditional convolutional codes. Along with the evolution of the UMTS standard, which brought high data throughput once the high speed packet access (HSPA) feature was introduced, the turbo coding architecture remained unchanged because of the elevated performance it provided. Furthermore, once the leap forward was made to the fourth generation LTE [5, 6] technology, turbo codes kept their core structure and key role in reducing transmission errors. In other words, the same constituent encoder is currently used in both UMTS and LTE. The main difference introduced by LTE refers to the interleaving block, which is now a quadratic permutation polynomial (QPP), a block suitable for high data rates obtained especially in parallel decoding architectures.

The arithmetical properties for the QPP interleaver allow the parallelization of the decoding process inside the algorithm, taking advantage on the main principle introduced by turbo decoding, i.e., the usage of extrinsic values from one turbo iteration to another. However, the goal of this paper is to provide an efficient hardware implementation for the main individual components of a turbo decoding architecture, the QPP interleaver, respectively the Soft Input Soft Output (SISO), with a direct impact on both serial and parallel decoding schemes. On the same time, no discussion will be made on the well-known methods [7] used to reduce the decoding latency for the selected algorithm inside a SISO unit.

This paper is organized as follows. Section 2 introduces the LTE turbo coding structure. In Section 3, the selected maximum logarithmic - maximum a posteriori probability (Max Log MAP) decoding algorithm is detailed, all the equations for a binary input being deduced. Section 4 presents the proposed hardware decoding architecture for a serial approach. In Section 5, the decoding performances are discussed in terms of bit error rate (BER) versus signal to noise ratio (SNR) and speed versus required hardware resources when targeting an XC5VFX70T FPGA chip [8] on the Xilinx ML507 [9] evaluation board. Section 6 presents the final conclusions and the perspectives of this study.

2. **LONG TERM EVOLUTION CODING STRUCTURE**

The LTE coding structure is a parallel concatenated convolutional code (PCCC), comprising of two constituent encoders and one interleaving block. Each individual 8-state constituent encoder has the following transfer function:

$$G(D) = \left[1,\ g_1(D)/g_0(D)\right], \tag{1}$$

where D denotes the elemental delay block and:

$$g_0(D) = 1 + D^2 + D^3;\ g_1(D) = 1 + D + D^3. \quad (2)$$

The native coding rate of the encoding structure is 1/3 since the input sequence C_k ($k = 1 \ldots K$, where K denotes the length of the uncoded data block) is sent at the output of the encoder as the systematic sequence X_k and each constituent encoder generates on its own a parity sequence Z_k, respectively $Z_k^{'}$.

The interleaved sequence $C_k^{'}$ is obtained from C_k, after the interleaver module reorganizes the input bits

$$C_k^{'} = C_{\pi(i)}, \quad i = 1 \ldots K, \quad (3)$$

where the length K of the input data and parameters f_1 and f_2 provided in Table 5.1.3–3 in [6] are used to generate addresses:

$$\pi(i) = (f_1 \cdot i + f_2 \cdot i^2) \bmod K. \quad (4)$$

3. **MAX LOG MAXIMUM A POSTERIORI DECODING ALGORITHM**

In terms of turbo decoding algorithms, the classical MAP algorithm provides the reference results, although it is not suitable for practical implementations, its main drawbacks being the large dynamical range of the variables and the prohibitive arithmetic complexity. This is the reason why suboptimal versions of the MAP algorithm were introduced and studied in the literature. Some of the most popular alternatives to the reference MAP are log MAP (logarithmic MAP) [10, 11], lin log MAP (linear logarithmic MAP) [12], const log MAP (constant logarithmic MAP) [13] and max log MAP. All these options use logarithmic representations in order to compensate for the first mentioned drawback. Furthermore, from the Jacobi logarithm expression [14]:

$$\ln(e^x + e^y) = \max(x, y) + \ln(1 + e^{-|y-x|}), \quad (5)$$

the second right term is approximated by the different methods mentioned above, i.e., either as a constant, linear values from a pre-stored table or just ignored (the max log MAP approach).

For the LTE turbo coding, the theoretical decoding scheme is presented in Fig. 1. Same decoding structure was presented in [15] for UMTS. One can notice the input log likelihood ratios (LLRs) for systematic bits $\Lambda^i(X_k)$ and for parity bits $\Lambda^i(Z_k)$ and $\Lambda^i(Z_k^{'})$, the output LLRs for decoding unit 1 (SISO 1) $\Lambda_1^O(X_k)$,

respectively for decoding unit 2 (SISO 2) $\Lambda_2^O(X_k')$, and the *extrinsic* value $W(X_k)$. Also, the main principle of the turbo decoding can be observed, i.e., the input of one decoding unit contains the previously computed output of the other decoding unit.

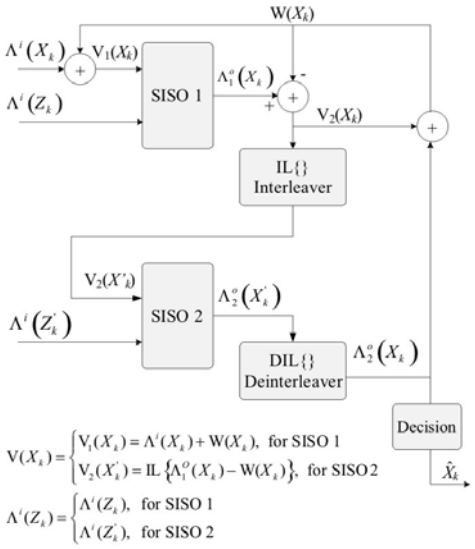


Fig. 1 – LTE turbo decoding scheme.

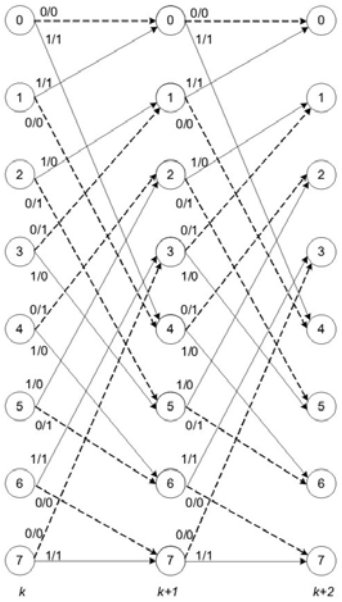


Fig. 2 – LTE turbo coder trellis.

Inside each SISO unit, the max log MAP equations for binary input are deduced from the corresponding constituent encoder trellis depicted in Fig. 2. There are 8 states on each stage of the trellis and each diagram state permits 2 inputs and 2 outputs. First, the branch metrics between states S_i and S_j are computed (2 such metrics for each state from the total of 8 corresponding to each stage of the trellis):

$$\gamma_{ij} = V(X_k)X(i,j) + \Lambda^i(Z_k)Z(i,j). \tag{6}$$

In conclusion, there are 16 branch metrics to be computed at each stage, but in reality there are only 4 possible values for these metrics:

$$\begin{aligned} &\gamma_0 = 0, \qquad \gamma_1 = V(X_k) \\ &\gamma_2 = \Lambda^i(Z_k), \quad \gamma_3 = V(X_k) + \Lambda^i(Z_k) \end{aligned}. \tag{7}$$

The next step in max log MAP algorithm is to execute the *backward recursion* when the backward metrics are computed. The backward metric for the

state S_i at the k^{th} stage is $\beta_k(S_i)$, $2 \le k \le K+3$ and $0 \le i \le 7$. The backward recursion is initialized with $\beta_{K+3}(S_i) = 0,\ 0 \le i \le 7$. From the stage $k = K+2$ until the stage $k = 2$, the computed backward metrics are:

$$\hat{\beta}_k(S_i) = \max\left\{(\beta_{k+1}(S_{j1}) + \gamma_{ij1}), (\beta_{k+1}(S_{j2}) + \gamma_{ij2})\right\}, \tag{8}$$

where $\hat{\beta}_k(S_i)$ represents the un-normalized metric and S_{j1} and S_{j2} are the 2 states from stage k+1 connected to state S_i at stage k. Once that at each stage k the metric $\hat{\beta}_k(S_0)$ is computed, the rest of the 7 backward metrics are normalized and stored:

$$\beta_k(S_i) = \hat{\beta}_k(S_i) - \hat{\beta}_k(S_0). \tag{9}$$

In a similar manner, the *forward recursion* is performed. For stage 0, the forward metrics are initialized $\alpha_0(S_i) = 0,\ 0 \le i \le 7$, and then, from stage $k = 1$ until stage $k = K$ the un-normalized/ normalized forward metrics are computed:

$$\begin{aligned} &\hat{\alpha}_k(S_j) = \max\left\{(\alpha_{k-1}(S_{i1}) + \gamma_{i1j}), (\alpha_{k-1}(S_{i2}) + \gamma_{i2j})\right\}, \\ &\alpha_k(S_j) = \hat{\alpha}_k(S_j) - \hat{\alpha}_k(S_0). \end{aligned} \tag{10}$$

No storing is needed for the forward metrics. Once they are computed for stage k, the decoding algorithm can compute in the same time a LLR estimate for the data bits X_k. This LLR is found the first time by considering that the likelihood of the connection between the state S_i at stage k-1 and the state S_j at stage k is:

$$\lambda_k(i, j) = \alpha_{k-1}(S_i) + \gamma_{ij} + \beta_k(S_j). \tag{11}$$

The likelihood of having a bit equal to 1 (or 0) is when the Jacobi logarithm of all the branch likelihoods corresponds to 1 (or 0) and thus:

$$\Lambda_0(X_k) = \max_{(S_i \to S_j):\, X_i = 1} \{\lambda_k(i, j)\} - \max_{(S_i \to S_j):\, X_i = 0} \{\lambda_k(i, j)\}. \tag{12}$$

4. **PROPOSED HARDWARE DECODING ARCHITECTURE**

The proposed hardware decoding scheme depicted in Fig. 3 represents an adaptation of the theoretical decoding structure presented in Fig. 1. It was introduced by the authors in [16] for an WiMAX CTC turbo decoder. As previously mentioned, the key of the turbo decoding is the iterative usage of the decoding information between the 2 SISO units. A natural conclusion arises, i.e., while one SISO unit is decoding the input information, the second one just waits the finish of the process before starting its own decoding phase. Moreover, since the interleaver/ deinterleaver modules are processing the information in a frame-

based manner, all the decoded data should be available before starting these procedures. In other words, only one SISO unit may be used in the decoding architecture. In Fig. 3, there are 3 dotted-line memory blocks. These are virtual memories, added just for a clear understanding of the scheme. In reality, these memory blocks are not needed since the corresponding data is computed and further used in the same time.

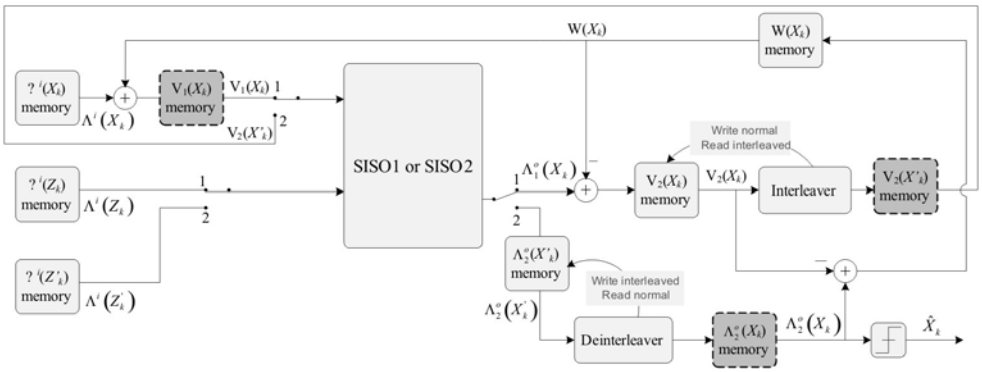
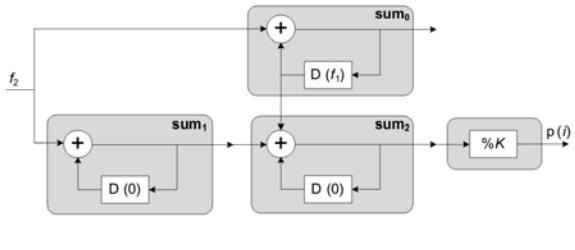


Fig. 3 – Proposed serial turbo decoding scheme.

Also, one point that should be mentioned is that the *interleaver* and *deinterleaver* blocks have the same hardware structure, including a block memory and an interleaver. The memory is written with the interleaved addresses each time a new data block is received. The values are then used as read addresses (when interleaver process is ongoing) or as write addresses (when deinterleaver process is ongoing). More precisely, this memory block used by the interleaver is not a huge pre-stored ROM memory, but a K_{max} (for LTE the value is 6144) location RAM memory, which is written offline each time a new encoded data block is received. This memory block, together with the 3 memory blocks from the left side of the picture (for the input data) are switched-buffers, allowing new data to be written while the previous one is still under decoding process, so that no additional delay to be added in the total decoding latency.

The scheme implements the relations included in Fig. 4. The most complex block of the interleaver remains in this case the modulo K block. One implementation solution for this block is to consider themodulo result as the reminder of a division. In this case, the reminder results correspond to a classic divider-for-integers scheme. The restoring integer division scheme might be a sequential one, i.e., a new set of inputs can be received only after the previous one was processed. This reduced overall processing speed is not attractive, even though the scheme uses few resources (264 Flip Flop registers and 334 LUTs @ 217 MHz for the Virtex 5 targeted device). An accepted solution is a pipe-line radix-2 non-restoring integer division [17]. Such a divider is available in Xilinx Core Generator

13.4 [18], the price for reduced latency being the increased amount of used resources (the complete interleaver with such a divider uses 1578 Flip Flop registers and 1708 LUTs @ 300 MHz for the Virtex 5 targeted device).



$$\begin{aligned}
&\mathrm{sum}_0(0) = f_1, \mathrm{sum}_1(0) = 0, \mathrm{sum}_2(0) = 0\\
&\pi(0) = 0\\
&for\ i = 1 : K-1\\
&\quad \mathrm{sum}_0(i) = \mathrm{sum}_0(i-1) + f_2\\
&\quad \mathrm{sum}_1(i) = \mathrm{sum}_1(i-1) + f_2\\
&\quad \mathrm{sum}_2(i) = \mathrm{sum}_2(i-1) + \mathrm{sum}_1(i) + \mathrm{sum}_0(i-1)\\
&\quad \pi(i) = \mathrm{sum}_2(i) \bmod K\\
&end
\end{aligned}$$

Fig. 4 – Proposed interleaver scheme.

The second block from Fig. 3 that is critical for decoder performances and costs is the SISO decoding unit. The proposed scheme for implementation is described in Fig. 5.

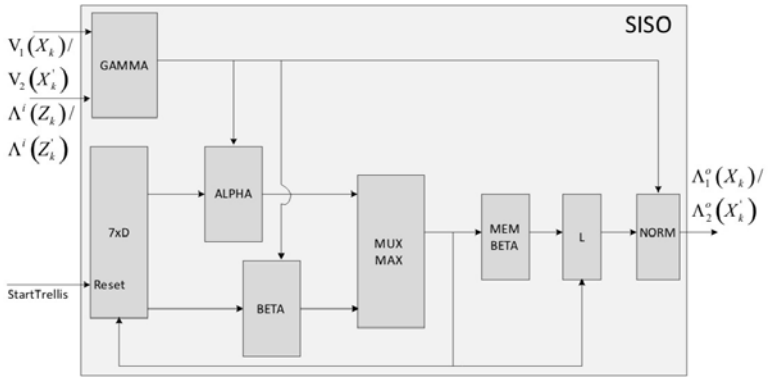


Fig. 5 – Proposed scheme for SISO decoding unit.

All modules are implementing in a dedicated manner the relations provided in Section 3. Each *gamma* (branch metric), *beta* (backward metric) and *alpha* (forward metric) is computed with a dedicated hardware. At each stage, 16 *gamma* values should be theoretically computed, but in reality only 4 possible values exists, one of them being 0. Then the 2 sums from (8) are computed for each of the 8 states in BETA block. The corresponding "max" function is applied in the MUX MAX block. Being a recursive process, after normalization, the 7 obtained *beta* values from one stage are used at the next one after being delayed in the 7xD module and also are stored in the MEM BETA memory. For *alpha* values the procedure is similar, except that no storing is needed since right after the LLRs are computed in the L module. The NORM block performs the final normalization before providing the output LLRs.

5. **PERFORMANCES AND IMPLEMENTATION RESULTS**

This section presents the obtained results for the proposed turbo decoder while simulated in finite/ infinite precision, in different radio environment (AWGN or Rayleigh channels), with different configuration settings (variable data block length *K*) and with different decoding parameters (1 to 5 turbo iterations). All pictures describe BER versus SNR.

Figure 6 depicts the decoding performances degradation when finite precision is used *versus* infinite precision. For finite precision, a 10 bits format is used, one bit for the sign, 6 bits for the integer part and 3 bits for the fractional part. The results are provided for a 512 bits data block, with quadrature phase shift keying (QPSK) modulation, after 3 turbo iterations over an AWGN channel. Figure 7 compares the obtained turbo decoder performances over an AWGN channel, respectively over a Rayleigh channel characterized by slow fading and frequency selective fading. Infinite precision was used for both curves, 512 bits data block, with QPSK and 3 turbo iterations.

Figure 8 presents the dependency between the turbo decoding performances and the turbo iterations number, 512 bits data blocks and QPSK modulation were used. 3 iterations provide best balance between latency and performances. Figure 9 depicts the turbo decoding performances *versus* data block length.

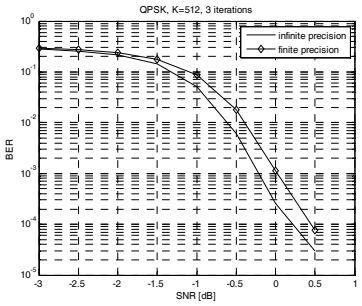


Fig. 6 – Infinite *vs.* finite precision.

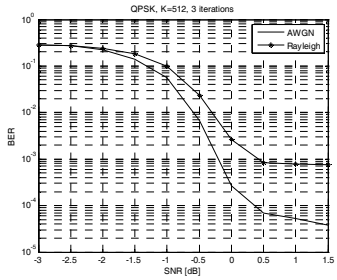


Fig. 7 – AWGN *vs.* Rayleigh channel.

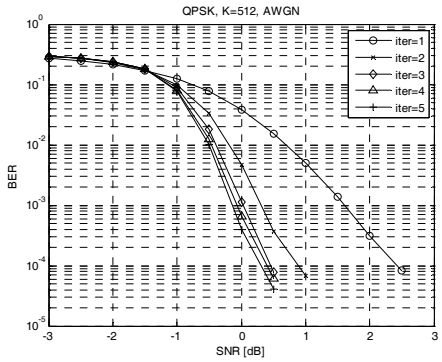


Fig. 8 – Variation *vs*. number of iterations.

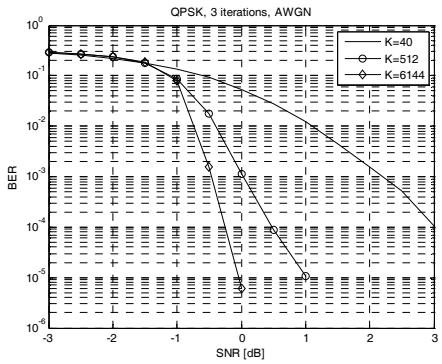


Fig. 9 – Variation *vs*. data block length.

6. CONCLUSION

This paper presented in the first part the principles of LTE turbo coding and the general turbo decoding scheme. For the decoding algorithm, max log MAP equations were then presented. Based on these equations, an efficient FPGA implementation solution for an LTE turbo decoder was proposed. The main two blocks of the architecture were the interleaver and the SISO decoding unit. For the interleaver, a simplified scheme was proposed, based on the usage of 3 similar accumulators and one pipe-line radix-2 non-restoring integer divisor. Even for this simplified scheme, our team continues the efforts to reduce the complexity, mainly provided by the divider, by splitting the related arithmetic so that modulo result to be maximum 2, scenario that requires simplified division scheme. The second block, the SISO decoding unit, was implemented in an efficient manner by taking advantage on the repetitive max log MAP equations for computing branch metrics, the backward metrics and the forward metrics.

The obtained decoding performances were provided, pointing on one hand the small degradation introduced by a 10 bits numerical representation format, and on the other hand comparing the simulation results when the radio environment was changed, when the transmission parameters were modified and when the decoding setting were also changed.

Received on December 6, 2014

ACKNOWLEDGMENTS

The work has been funded by the Sectoral Operational Programme Human Resources Development 2007–2013 of the Ministry of European Funds through the Financial Agreement POSDRU/159/1.5/S/134398.

REFERENCES

1. C. Berrou, A. Glavieux, *Near optimum error correcting coding and decoding: Turbo-Codes*, IEEE Trans. Communications, **44**, *10*, pp. 1261–1271, 1996.
2. C. Berrou, M. Jézéquel, *Non binary convolutional codes for turbo coding*,Electronics Letters, **35**, *1*, pp. 9–40, 1999.
3. C. Berrou, A. Glavieux, P. Thitimajshima, *Near Shannon limit error-correcting coding and decoding: Turbo Codes*, IEEE Proceedings of the Int. Conf. on Communications, Geneva, Switzerland, 1993, pp. 1064–1070.
4.*** *Third Generation Partnership Project*, 3GPP home page www.3gpp.org
5. F. Khan, LTE for 4G Mobile Broadband, Cambridge University Press, New York, 2009.
6.*** *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA); Multiplexing and channel coding* (*Release 8*), Technical Specification, 3GPP TS 36.212 V8.7.0 (2009–05).
7. S. Chae, *A low complexity parallel architecture of turbo decoder based on QPP interleaver for 3GPP-LTE/LTE-A*, http://www.design-reuse.com/articles/31907/turbo-decoder-architecture-qpp-interleaver-3gpp-lte-lte-a.html
8.*** Xilinx Virtex 5 family user guide, www.xilinx.com.
9.*** Xilinx ML507 evaluation platform user guide, www.xilinx.com
10. P. Robertson, E. Villebrun, P. Hoeher, *A Comparison of Optimal and Sub-Optimal MAP Decoding Algorithms Operating in the Log Domain*, Proc. IEEE International Conference on Communications (ICC'95), Seattle, 1995, pp. 1009–1013.
11. C. Vladeanu, S. El Assad, *Hybrid Maximum-Likelihood Detector for Trellis Coded Spatial Modulation*, Rev. Roum. Sci. Techn. – Électrotechn. et Énerg., **57**, *4*, pp. 383–393, 2012.
12. J. F. Cheng, T. Ottosson, *Linearly approximated log-MAP algorithms for turbo decoding*, Vehicular Technology Conference Proceedings, VTC 2000, Tokyo; IEEE 51st, **3**, pp. 2252–2256, 2000.
13. S. Papaharalabos,P. Sweeney, B.G. Evans, *Constant log-MAP decoding algorithm for duo-binary turbo codes*, Electronics Letters, **42**, *12*, pp. 709–710, 2006.
14. J. H. Han, A. T. Erdogan, T. Arslan, *High Speed Max-Log-MAP Turbo SISO Decoder Implementation Using Branch Metric Normalization*, Proceedings of the IEEE Computer Society Annual Symposium on VLSI New Frontiers in VLSI Design, May 2005.

15. M. C. Valenti, J. Sun, *The UMTS Turbo Code and an Efficient Decoder Implementation Suitable for Software-Defined Radios*, International Journal of Wireless Information Networks, **8**, *4*, 2001.
16. C. Anghel, A. A. Enescu, C. Paleologu, S. Ciochina, *CTC Turbo Decoding Architecture for H-ARQ Capable WiMAX Systems Implemented on FPGA*, Ninth International Conference on Networks (ICN 2010), Menuires, France, April 2010.
17. Jen-Shiun Chiang, Eugene Lai, Jun-Yao Liao, *A Radix-2 Non-Restoring 32-b/32-b Ring Divider with Asynchronous Control Scheme*, Tamkang Journal of Science and Engineering, **2**, *1*, pp. 37–43, 1999.
18.*** http://www.xilinx.com/products/intellectual-property/Divider.htm