TELKOMNIKA, Vol. 11, No. 4, April 2013, pp. 2116~2123
ISSN: 2302-4046

Turbo Code Design and Implementation of High-Speed Parallel Decoder

Yi Bo-nian
The Technology Institute of automation of Wuhan University, Wuhan Hubei 430070
Corresponding author, e-mail: 2462158049@qq.com

Abstract

Due to the high complexity Turbo decoding algorithm, implemented the hardware decoder logic of consumes more resources and storage resources, decode delay larger. In order to meet LTE systems, high reliable transmission of high data rate needs high speed Turbo decoder design faces enormous challenges. This topics select based on LTE system of parallel Turbo Code decoders hardware design and achieved as research direction, select FPGA (Virtex-6) as hardware design and achieved of platform, from Log-MAP algorithm of Sentinel of, and State measure value handed owned calculation in the of owned a of processing, and Diego generation decoding algorithm of parallel of, and parallel interwoven Manager design, and key path optimization technology, aspects analysis, design achieved has LTE Turbo Code high-speed parallel decoders.

Keywords: *LTE turbo code, log-map, parallel, FPGA, iterative decoding algorithm*

1. Introduction

Due to its good performance, after the introduction of Turbo Code [1, 2], it has attracted many researchers. There are mainly two kinds of algorithms in Turbo iterative decoding algorithms. One is Log-Map and Max-Log-MAP algorithm, the other is SOVA algorithm with lower complexity. All the researches mainly center on the decoding performance, implementation complexity, decoding suboptimal simplify, parallel decoding implementation, fixed point quantify on the performance of decoding.

The encoding of Turbo code is very simple, no matter software implementation or hardware implementation. However, the decoding process is rather complex. The long decoding delay makes it a challenge to use it in practical engineering applications. FPGA manufactures like Xilinx and Altera have targeted IP core to solve this problem but they are too expensive, more than 15000 dollars. Furthermore, the targeted IP core can not be directly used in the design. Therefore, it is important to make Turbo decoding faster and more generic on FPGA platform.

2. Turbo Decoding Algorithm

Turbo decoding algorithm [3] mainly consists of two types: one is called SOVA evolved from Viterbi algorithm, the other is Log-MAP and Max-Log-Map evolved from MAP algorithm which achieves high performance. Due to its poor performance, we will not talk about the SOVA algorithm.

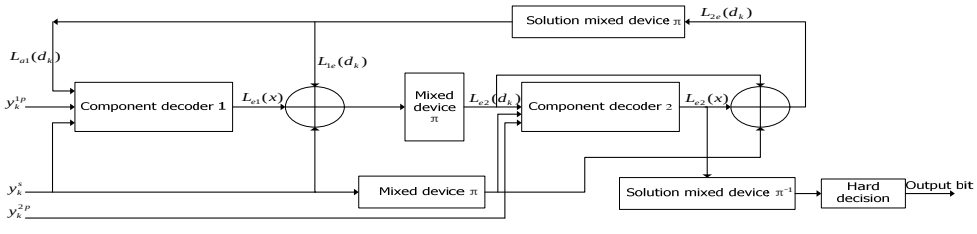

Figure 1. The Structure of Turbo Iterative Decoding Algorithm

Received January 14, 2013; Revised February 25, 2013; Accepted March 5, 2013

The two component decoder in Figure 1 all use SISO. SISO use the soft bit sequence to calculate the posterior probability of each bit. The most common SISO algorithm are MAP, Log-MAP and MAX-Log-MAP algorithm. MAP algorithm is based on the symbol maximum posteriori probability decoding algorithm. For linear block coding is convolutional code, it can reduce the bit error rate to minimum. The BCJR algorithm can calculate the posteriori probability of Markov source going through discrete memoryless channel. Since the output of the convolutional encoder through a discrete memoryless channel constitute a Markov source, therefore the BCJR algorithm can be the maximum posteriori probability decoding algorithm for convolutional code [4].

Deduce the MAP decoding algorithm of Turbo code in detail. There are lots of multiplication in the MAP, therefore implement it directly on the hardware system is rather complex. While if implement it in the Logarithmic domain, the multiplication can be converted into addition and without any harm to the decoding performance.

The following give the derivation of the algorithm. Denote x_t as the encoding output of system bit at time t , $a_t^i, i = 1,2$ as the CRC at time t , $r_t^i, i = 0,1,2$ as the received data at time t, while r_t^0 denoted as system bit, r_t^1 and r_t^2 stand for CRC bit.

$$\Gamma_t(p,q) = \log \gamma_t(p,q) \tag{1}$$

$$A_t(p) = \log \alpha_t(p) \tag{2}$$

$$B_{t+1}(q) = \log \beta_{t+1}(q) \tag{3}$$

The calculation steps of Log-MAP is as follows:

start from t=1 , use the following formula (4) to calculate $\Gamma_t(p,q)$

$$\begin{aligned}\Gamma_t(p,q) &= \log \gamma_t(p,q) \\ &= \log p(r_t^0 \mid x_t) + \log p(r_t^1 \mid a_t^1 = a^{(1,p,q)}) + \log p(x_t = x^{(p,q)}) \\ &= \frac{L_c}{2} x_t^{(p,q)} r_t^0 + \frac{L_c}{2} v_t^{(1,p,q)} r_t^1 + \frac{1}{2} x_t^{(p,q)} L_t^a\end{aligned} \tag{4}$$

When t=1, initialize the foward branch metric A and using formula(5) to calculate $A_t(p)$, then store them.

$$\begin{aligned}A_{t+1}(q) &= \log \alpha_{t+1}(q) = \log \sum_p \alpha_t(p)\gamma_t(p,q) \\ &= \log \sum_p \exp[A_t(p) + \Gamma_t(p,q)]\end{aligned} \tag{5}$$

When t=N+1, initialize the backward branch metric B and using formula(6) to calculate $B_t(p)$ from t=N to t=1.

$$\begin{aligned}B_t(p) &= \log \beta_t(p) = \log \sum_q \beta_{t+1}(q)\gamma_t(p,q) \\ &= \log \sum_q \exp[B_{t+1}(q) + \Gamma_t(p,q)]\end{aligned} \tag{6}$$

Calculate LLR from K=0 to K=N-1

$$\begin{aligned}LLR_t &= \log \frac{\sum_{(p,q)\in S_1} \exp[A_t(p) + \Gamma_t(p,q) + B_{t+1}(q)]}{\sum_{(p,q)\in S_0} \exp[A_t(p) + \Gamma_t(p,q) + B_{t+1}(q)]} \\ &= \log \sum_{(p,q)\in S_1} \exp[A_t(p) + \Gamma_t(p,q) + B_{t+1}(q)] - \log \sum_{(p,q)\in S_0} \exp[A_t(p) + \Gamma_t(p,q) + B_{t+1}(q)]\end{aligned} \tag{7}$$

Then calculate $L_e(t)$ using the calculated results LLR_t

$$L_t^e = LLR_t - (L_t^a + L_c r_t^0) \quad (8)$$

Use L_t^e as the priori information of $\Gamma_t(p,q)$, then go to step (1) until the iteration of loop operation reach the maximum number. Make decision according to the LLR_t in the final iteration process.

Function $\ln(e^x + e^y) = \max(x,y) + \ln(1+e^{-|x-y|})$ involves seeking maximum function and calibration ion function. Function f(x) can be implemented using lookup table.

The Log-MAP algorithm only has addition operation, therefore it is easy to implement it on the hardware. The calculation of state metric in Log-MAP is an iteration process. The feedback path in the hardware is significant but also influence the maximum clock frequency. The calibration function need an adder to modify it, which contribute to the path delay. The Max-Log-MAP algorithm doesn't need any calibration function or lookup table, so it can significantly reduce the path delay, increase the maximum frequency clock. But this is at the cost of performance reduction due to approximate calculation.

3. System Design

3.1. Minimum Bit Width Quantify

(1) Different Iteration Times

Figure 2 show us the performance of Log-MAP algorithm under different conditions with code length 1024. We vary the test condition of float point, 6bit fixed point [5], 8bit fixed point and 4 iteration times, 6 iteration times, 8 iteration times.

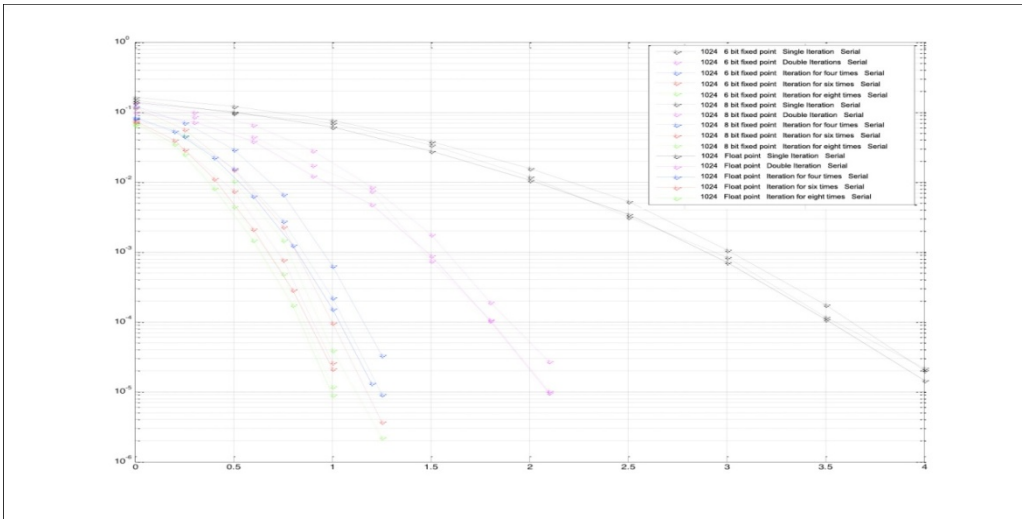

Figure 2. Performance of Log-MAP under Different Iteration Times and Bit Width

According to Figure 2, under the condition of the same code length, the performance difference of two quantization methods is not affected much by the iteration times. With code length 1024, the performance of the 8bit quantization is almost equal to the float point pattern. The 6bit quantization is only 0.1dB reduction compared to the float point quantization.

(2) Different Code Length

This time, the iteration times is confirmed as 8. Under this condition, we vary the code length with 512,1024, and 6144. The quantization methods are the same with the above.

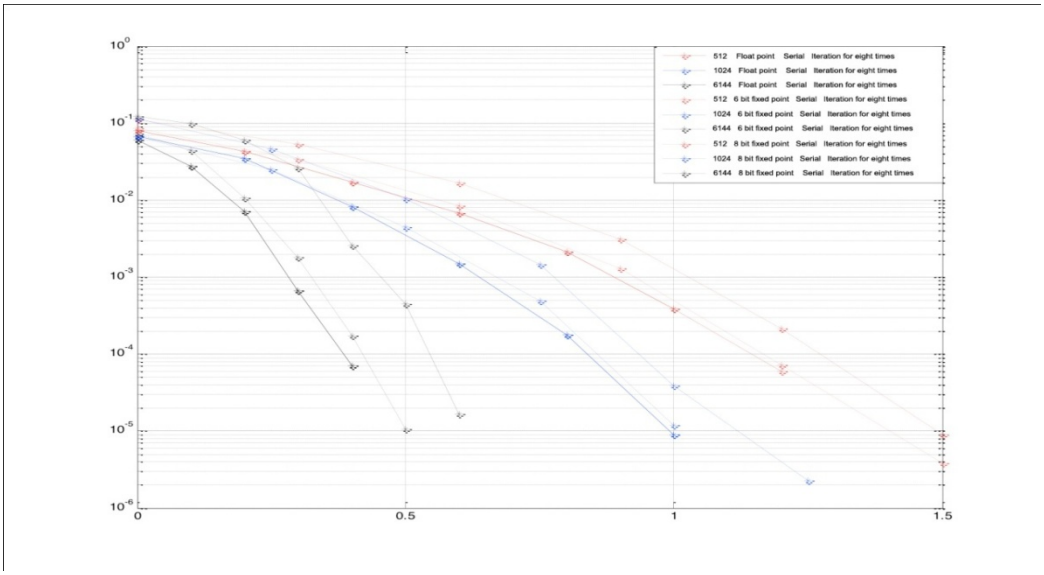

Figure 3. Performance of Log-MAP under Different Code Length and Bit Width

From Figure 3, we can see that performance penalty of two different quantization is almost the same. The performance loss worsen as code length increases. The performance loss is about 0.1dB.

In summary, we can see that under different iteration times or different code length, the differences between fixed point quantization and floating point quantization can be regarded as fixed. The 8bit quantization almost has no performance loss while the 6bit has 0.1dB. So in you design, if you attach great importance to the decoding performance, you can choose the 8bit quantization method. In this paper we choose the 6bit quantization method due to the only 8 or 9 bit width in Xilinx FPGA RAM, regardless of the 0.1dB loss.

3.2. Implementation of Lookup Table in Calibration Function

In Log-MAP algorithm, the recursive formula α 、 β of is:

$$\max^*(x_1, x_2) = \max(x_1, x_2) + fun\_corr(|x_1 - x_2|) \tag{9}$$

The calculation of LLR need to nest this formula 7 times, so it is one of the most basic arithmetic unit in Log-MAP. For the recursive of α ,

$$x_1 = \alpha_1 + \gamma_1$$
$$x_2 = \alpha_2 + \gamma_2$$

$$fun\_corr(|x_1 - x_2|) = \ln(1 + \exp(-|x_1 - x_2|)) \tag{10}$$

Formula (10) is a calibration function. It need lookup table to implement it. Under the condition of 3bit fractional bit quantized input channel information, the number of available values are 22, fun_corr_table = [6,5,5,4,4,3,3,3,3,2,2,2,2,1,1,1,1,1,1,1,1,1]; while 2bit are 9,fun_corr_table = [3,2,2,2,1,1,1,1,1];

4. FPGA Implementation

LTE Turbo code interleaver is a QPP interleave. When the code block is equally divided into N = 0,1, ... code segments, each code segment can be achieved entirely without conflict interleaving, thereby performing parallel decoding. Wherein, parameters f_1 and f_2 are determined by the code block length K. Parallel decoder modules mainly includes input buffer

module, parallel PU module, interleaving/deinterleaving module. Parallel decoding unit PU and interleaving/deinterleaving module in the decoder are used to implement the interleaver and deinterleaver function. The input and output buffer module is used to complete parallel-serial conversion or serial-parallel conversion and to improve throughput.

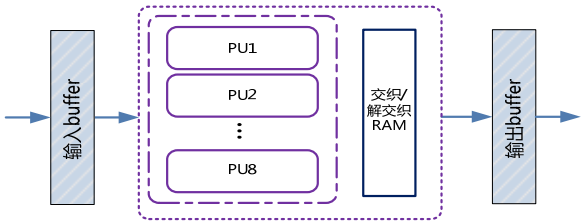

Figure 4. Function Diagram of Parallel Decoder

In order to further improve the data throughput, pipelining structure can be used in the interior of each PU to implement iterative decoding [6]. For example, we choose four pipeline in each PU, each sub-decoder contains basic SISO unit to complete the basic operation process, external information storage RAM for interleaving and deinterleave also included. The first level sub-decoder complete the previous It/4 times iteration of one sub-block. Here 'It' is denoted as the maximum iteration times. Post-stage sub-decoder sequentially complete the computation. But there are problems in the process: (1) As a result of the pipeline design, the throughput increases P times. But it still can not reduce the decoding delay of every single code block. (2) There are Le information transmission in the junction of pipeline, the complexity of which is very high. Therefore, how to coordinate every sub-decoder is a major problem. To sum up, it is uneconomical to adopt the pipeline structure.

From another point of view, we can use another structure to achieve the same performance compared with the pipeline structure. The new structure is illustrated in Figure 5.

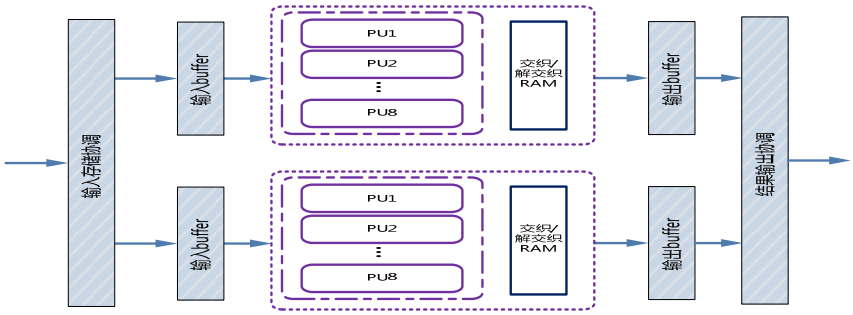

Figure 5. The Structure of A New Way To Improve Throughout

Figure 5, the new structure is consist of two identical full-featured decoder in Figure 4. The code block under decoding is allocated to the two decoder through input storage coordinator with the method of operation in accordance with the ping-pong [7]. The result output coordinator is responsible to output the decoding result with the correct sequence . Compared with the pipeline structure, this new structure can achieve equal performance. But the this new structure is more easy to control, and more scalable. Thus, in this paper, we use this new structure to achieve the throughout scalability .

The SISO unit uses the Log-MAP algorithm[8] to calculate each status quantity, Gamma, Alpha, Beta and Le, and store Le for iteration operation. The calculation process of these parameters consists of two steps – forward recursive process and backward recursive process. Forward recursive process need N/M cycles to get the result of Alpha and Gamma.

Here N denote code length, M denote the number of PU. Next, the backward recursive process also use N/M cycles to calculate Beta and Le[9]. The forward and backward recursive process are conducted in succession. To get the result of Alpha and Beta, you must use Gamma. But Gamma can not be available in both the recursive process at the same time. The functional diagram of SISO is illustrated in Figure 6 .

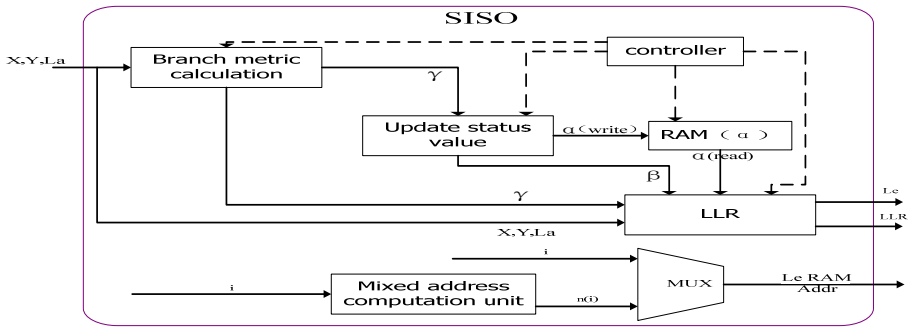

Figure 6. Function Diagram of SISO

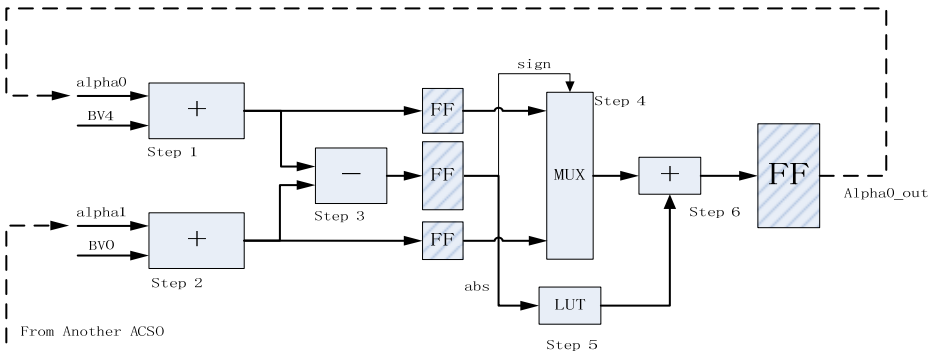

Figure 7. Improved Status Update Calculation Circuit Diagram

In Figure 7, a register is inserted on the critical path by which the path delay greatly reduced, and operation clock frequency improved. However, due to the calculation of a status metric consumes two clock cycles ,making the above recursive process has to wait one clock. Therefore, one waiting clock is wasted. Assuming that, after inserting a register, the operation clock frequency doubles. But when taking the waiting clock into consideration, the overall decoding speed only get a little raise[10].

5. Decoder performance

Under the condition of 8 PU structure, 6 iteration times, 6144 code length, each iteration process consume (6144/8 * 4)=768*4 cycles. One SISO operation need full forward and backward recursive process, consuming 768*2 cycles. One iteration operation containing two SISO operation thus consumes 768*4 cycles. Therefore, the throughout of each PU is :

$$\frac{250\times10^{6}\times768}{768\times4\times6}=10.4Mbps \tag{11}$$

In this formula, the clock frequency is 250MHz. So decoder with 8 PU can achieve 83.2Mbps throughout. If extend it according to Figure 4 and Figure 5, the throughout can reach up to 166.4 Mbps [11]. The throughout has nothing to do with code segment length, making it unrelated to the overall code segment length.

Considering 6144 code length, handling 10000 bits, then the delay of each decoder is :

$$\frac{100000}{166.4\times10^{6}} \approx 600us$$

Figure 8 compare the performance between 6bit fixed point parallel algorithm and float point serial algorithm under different iteration times, using Log-MAP .

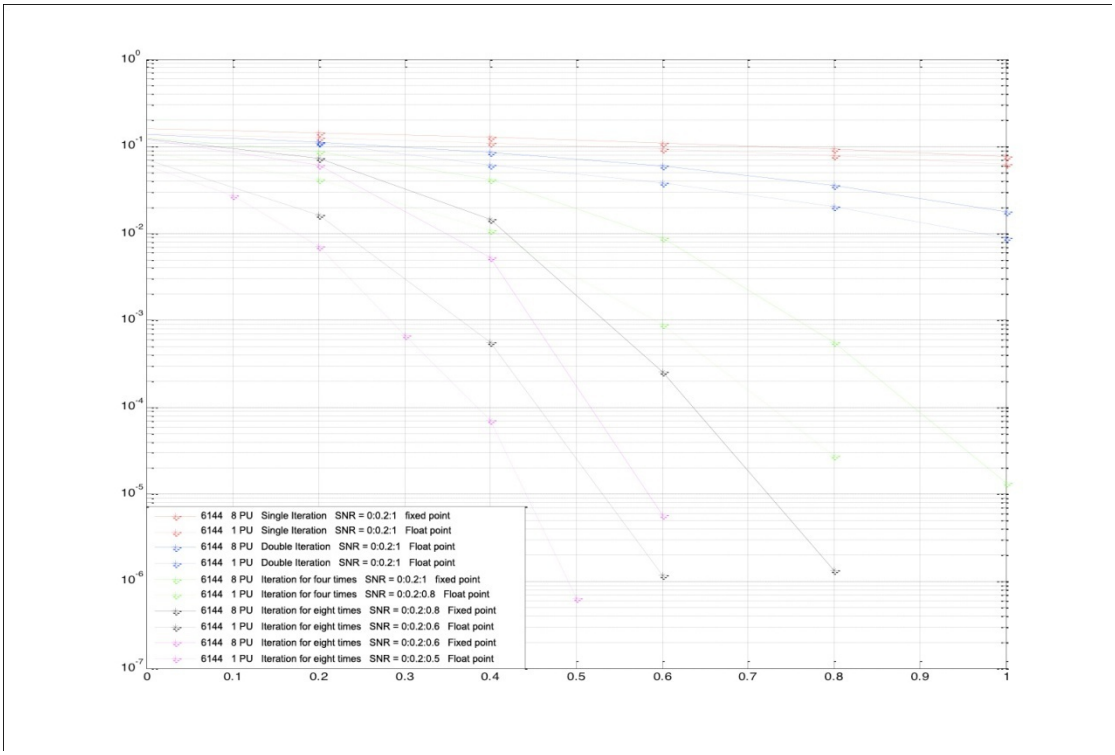

Figure 8. performance Comparison Between Fixed Point Parallel Algorithm and Float Point Serial Algorithm

From the simulation result, we can see fixed point parallel algorithm has about 0.2dB performance loss regardless of iteration times. But it should be noted that in this simulation, the code length is 6144. If the code length is less than that, the performance penalty will worsen. The overall performance loss will largely beyond 0.2dB [12].

6. Conclusion

This paper designs and implements a high-speed Turbo decoder completely meeting the LTE system demand based on the Xilinx Virtex-6 FPGA platform [13-14]. The decoder is completely compatible with the standard LTE Turbo code. The code length and iteration times can be configured. The code length can choose anyone among the 188 types. The iteration times can be conFigured between 1 to 15. The throughout of this decoding system also meet the LTE demands, beyond the maximum 100Mbps .

Reference

[1] Lance C Perez. A Distance Spectrum Interpretation of Turbo Codes. *IEEE Transations On Information Theory*. 1996; 42(6).
[2] Rostislav (Reuven) Dobkin, Michael Peleg. Parallel Interleaver Design and VLSI Architecture for Low-Latency MAP Turbo Decoders. *IEEE Transactions On Very Large Scale Integration (Vlsi) Systems*. 2005; 13(4).

[3] L Bahl, J Cocke, F Jelinek, J Raviv. Optimal decoding of linear codes for minimizing symbol error rate. *IEEE Transactions on Information Theory*. 1974.
[4] Todd K Moon. *Error Correction Coding Mathematical Methods and Algorithms.*Published by John Wiley & Sons Inc. 2005.
[5] G Montorsi, S Benedetto. Design of fixed-point iterative decoders for concatenated codes with interleavers. *IEEE Journal on Selected Areas in Communications.* 2001.
[6] Xilinx, "Xilinx_tcc_decoder_3gpplte_a_ds675". 2009.
[7] Emmanuel Boutillon, Warren J Gross, P Glenn Gulak. VLSI Architectures for the MAP Algorithm. *IEEE Transactions On Communications.* 2003; 51(2).
[8] A Nimbalker, KT Blankenship, B Classon, TE Fuja, DJ Costello. *Inter-Window Shuffle Interleavers for High Throughput Turbo Decoding.* Proc. 3nd International Symposium on Turbo Codes & Related Topics, Brest, France. 2003; 355–358.
[9] Perttu Salmela, Harri Sorokin, Jarmo Takala. A Programmable Max-Log-MAP Turbo Decoder Implementation. *Hindawi Publishing Corporation VLSI Design*. 2008.
[10] A Abbasfar. Turbo-like Codes: Design for High Decoding. *Springer.* 2007.
[11] Byoung-Sup Shim, Seok-Jun Choi, Hyoung-Keun Park, Sun-Youb Kim, Yu-Chan Ra. *A Study on Performance Evaluation of the Asymmetric Turbo Codes.* International Conference on Convergence and Hybrid Information Technology. 2008.
[12] Ajit Nimbalker, T Keith Blankenship. Contention-Free Interleavers for High-Throughput Turbo Decoding. *IEEE Transactions On Communications.* 2008; 56(8).
[13] Hanaa T El-Madany, Faten H Fahmy, Ninet MA El-Rahman, Hassen T. Dorrah Design of FPGA Based Neural Network Controller for Earth Station Power System. *TELKOMNIKA.* 2012; 10(2): 281-290.
[14] Hendra Setiawan, Yuhei Nagao, Masayuki Kurosaki, Hiroshi Ochi. IEEE 802.11n Physical Layer Implementation on Field Programmable Gate Array. *TELKOMNIKA.* 2011; 10(1): 67-74.