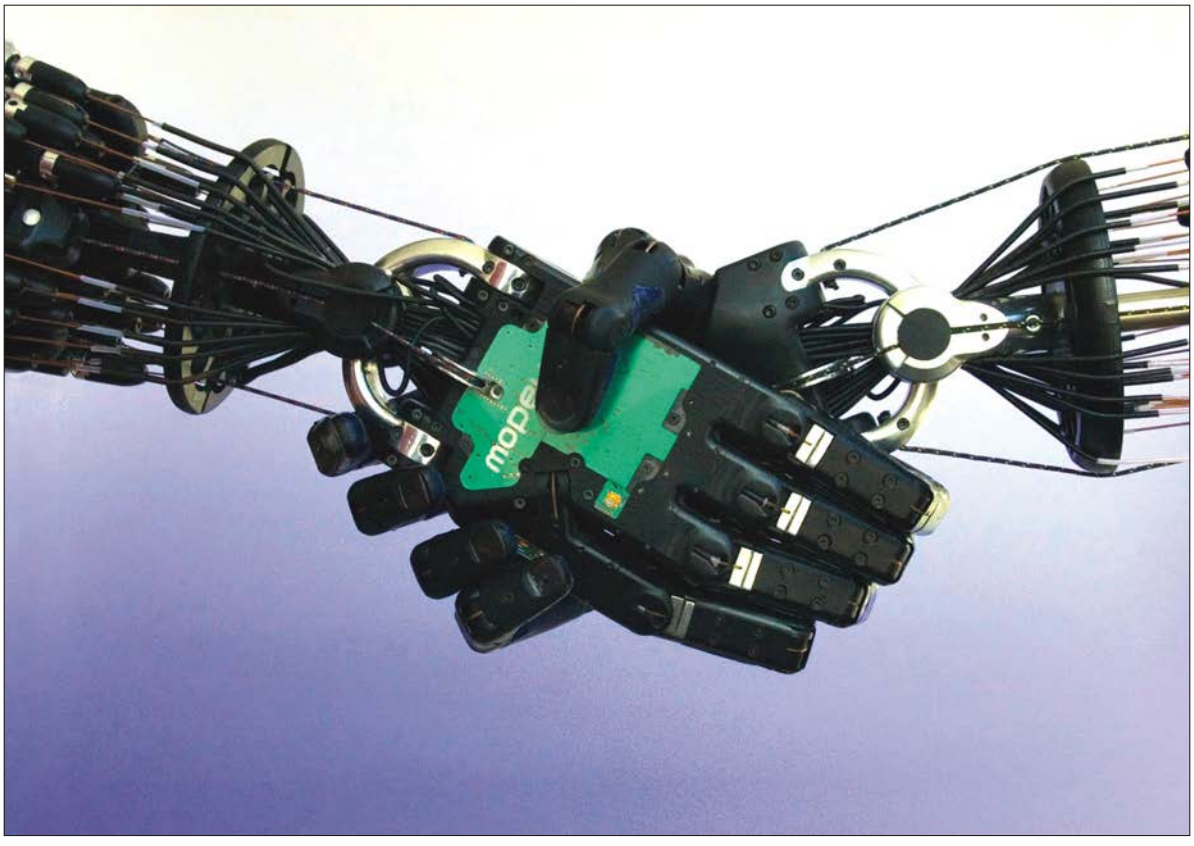

Journal of Artificial Intelligence

ISSN 1994-5450

science alert

ANSInet
an open access publisher
http://ansinet.com

Journal of Artificial Intelligence 6 (1): 22-32, 2013
ISSN 1994-5450 / DOI: 10.3923/jai.2013.22.32

FPGA Implementation of Turbo Decoder for LTE Standard

K. Kalyani, A. Sakthi Amutha Vardhini and S. Rajaram
Department of Electronics and Communication Engineering, Thiagarajar College of Engineering, Madurai, India

Corresponding Author: K. Kalyani, Department of Electronics and Communication Engineering, Thiagarajar College of Engineering, Madurai, India

ABSTRACT

The data rate of 100 Mbps will be supported by upcoming 3G Long Term Evolution (LTE) standard. In 20 MHZ of bandwidth, this data rate will be attained. For the arrival of high data rate of the 3G LTE systems, there is an essential requirement of turbo decoder implementation. In this work, Log-MAP algorithm based turbo decoder for LTE receiver is proposed. The Log-MAP based turbo decoder provides performance nearer to Shannon's limit with reasonable computational complexity. The VHDL coding for parallel architecture of turbo decoder using Log-MAP algorithm for LTE is simulated and synthesized using Xilinx XC3S200-4ft256, Spartan 3 family and the results are verified with the manual calculation.

Key words: Log-MAP algorithm, Long Term Evolution (LTE), turbo decoder

INTRODUCTION

In modern society, Wireless communication has become a key element. From the beginning of mobile phones, mobile devices have prolonged severely. The Long Term Evolution (LTE) can readily co-exist with HSPA and earlier networks. LTE is also used to afford greater radio-access technology which offer vehicular speed mobility. So that Operators are able to move easily around their networks and also switch over from HSPA to LTE over time, because of scalable bandwidth. The LTE receiver systems having number of transmitter and receiver antennas afford an enhanced performance in terms of multiplexing and diversity (Yang and Cavallaro, 2011). Multiple Input and Multiple Output (MIMO) with Orthogonal Frequency Division Multiplexing (OFDM) exploits greater transmission rates, better link robustness, higher coverage and higher spectral efficiencies without rising total bandwidth or transmission power (Gupta and Kumar, 2010). But the turbo decoder in LTE receiver apply different algorithms will vary the BER performance of LTE receiver. So there is a need to identify the receiver architecture with MIMO-OFDM having high BER performance. When turbo decoding is performed with Log-MAP algorithm, it provides enhanced performance compared to Soft Output Viterbi Algorithm (SOVA). So Log-MAP algorithm is considered as best among all other algorithms because of its powerful error correcting ability, flexibility in block lengths and code rates and reasonable complexity. Thus in this work, turbo decoder is proposed using Log-MAP algorithm. This decoder is simulated and the results are verified with manual results. Thus the BER performance of LTE receiver is improved with fewer complexes compared to conventional one.

DESIGN OF TURBO CODER FOR LTE STANDARD

The block diagram representation of the two-antenna receiver for MIMO-OFDM is presented in Fig. 1. Radio-frequency functions are considered as input ports of the receiver. The data rate of

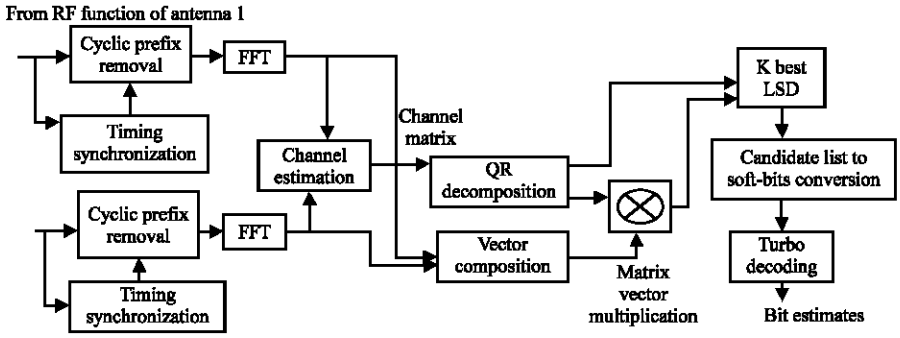

Fig. 1: Simplified diagram of two antenna MIMO OFDM receiver

100 Mbps will be supported by upcoming 3G Long Term Evolution (LTE) standard. This data rate will be attained with transmission techniques like MIMO-OFDM and efficient forward error correction method that is, turbo coding (Perttu *et al*., 2008). However, turbo decoder using hard decision algorithms in 3GPP LTE receiver may decrease the BER performance. Hence, in this work, suitable algorithm has been presented for turbo decoding to have a better BER performance of LTE receiver.

Turbo code is a great achievement in the field of communication system. It can be created by connecting a turbo encoder and a decoder serially. The complementary SISO decoders which are separated by interleaver and de-interleaver are present in turbo decoder. For implementing the turbo decoder, the very first consideration is to select a SISO algorithm which can give efficient performance. There are different decoding algorithms with soft decision and hard decision approaches. Soft-decision decoder is an algorithm to decode the data which will be decoded with use of error correcting code. Soft-decision decoder operates on a range of values between 0 and 1 in a binary code but hard-decision decoder operates on fixed set of values 0 and 1 typically in binary code. So, a soft-decision decoder is better in performance than hard-decision decoder in the presence of corrupted data. There are various decoding algorithms namely soft output Viterbi, MAX Log-MAP algorithm, Max MAP algorithm and Log-MAP algorithm. All the algorithms are based upon the trellis based estimation. The trellis based estimation algorithms are classified into two types. They are sequence estimation algorithms and symbol-by-symbol estimation algorithms. The SOVA is classified as sequence estimation algorithm, whereas MAP, Max Log-MAP and Log-MAP algorithms are classified as symbol-by-symbol estimation algorithms. In general, the symbol-by-symbol estimation algorithms have better BER performance than the sequence estimation algorithms for both fading channels and the AWGN channels. In error correction performance, the Log-MAP algorithm is superior compared to SOVA (Salim *et al*., 2010). Though the SOVA is less computationally complex compared to Log-MAP algorithm, it gives least accurate results and it cannot be used for iterative decoding (Lucian and Stoian, 2008; Karim *et al*., 2011). Among symbol-by-symbol estimation algorithms, Log-MAP algorithm gives better BER performance compared to MAP and Max-Log algorithms. So, in this paper turbo decoder based on Log-MAP algorithm is considered and the simulation is done using Xilinx.

Turbo encoder block: There are two systematic convolutional encoders present in a turbo decoder. Each encoder consists of an interleaver unit with a specified encoding structure (Zeng and Hong, 2002). We send 'n' number of input data bits to the encoder. This data goes to the

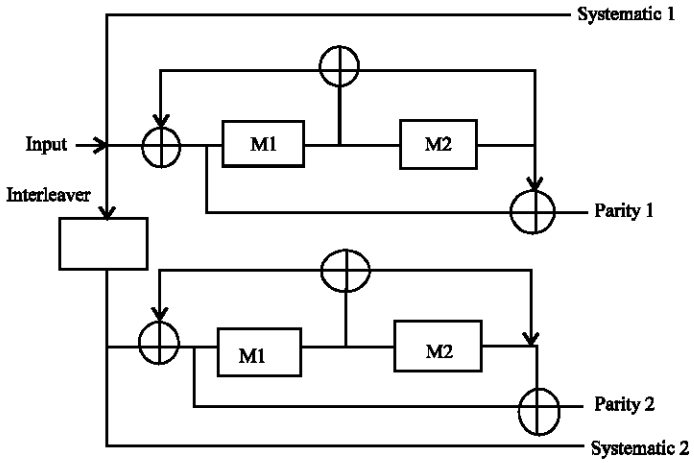

Fig. 2: Simplified diagram of turbo encoder

Interleaver pattern	1-3	2-4	3-1	4-5	5-2	6-6	7-7
Input	1	-1	1	-1	1	-1	-1
Interleaved input	1	1	1	-1	-1	-1	-1

Fig. 3: Interleaver pattern

first convolutional encoder whereas an interleaved data is passed through the second convolutional encoder. Turbo encoded output data bits are passed through AWGN (Additive white Gaussian noise) channel. Due to noise some of data bits may get corrupted and error maybe introduced in the system. These data bits are transferred to the decoder unit where the error is removed and original data is recovered. The turbo encoder block is as shown in Fig. 2.

Interleaver block: In turbo code, interleaver unit is a random block that is used to rearrange the input data bits with no repetition (Vishvaksenan *et al*., 2011). Interleaver unit is used in both encoder and decoder part. At the encoder side it generates a long block of data, whereas in decoder part it correlates the two SISO decoder and helps to correct the error. At the decoder side after passing the encoded data to the first decoder some of the errors may get corrected, then again interleaving this first decoded data and passing it through the second decoder remaining error may get correct. Like this, the process is repeated for more number of times. The interleaving pattern in general is done as shown in Fig. 3.

In this figure, according to the pattern in the first column, the interleaved input is calculated from input seven bits.

Turbo decoder block: Same numbers of decoders are used in the receiver as in encoder. These decoders work on independent set of parity but on the same information bit. This arrangement is said to be Parallel Concatenated Convolutional Code (PCCC) (Hagenauer *et al*., 1996; Oliver *et al*., 2010). This PCCC must be Recursive Systematic Code (RSC). In RSC, the independent parity bits taken from different coders together with systematic bit is used to measure the reliability. This type of decoding frequently used is called as iterative form of decoding. This paper takes two

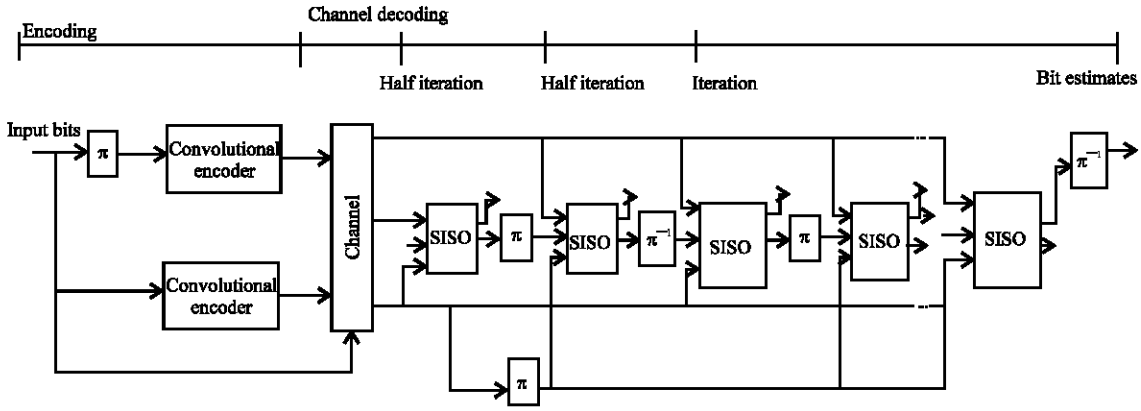

Fig. 4: Schematic for turbo decoder

complementary decoders which are operated in parallel manner to get the original transmitted signal. The schematic for turbo decoder is given in Fig. 4.

DESIGN OF LOG–MAP ALGORITHM BASED TURBO DECODER

In Log-MAP algorithm, soft inputs are taken as inputs to the decoder 1. So, the systematic and the parity outputs taken from the turbo encoder are fed as inputs to the decoder 1. The receive data from turbo encoder output are as shown in Fig. 5.

The SISO component decoders describe the turbo decoding. Logarithms of likelihood ratios present the soft information. All component decoders process parity bit vector y_p and systematic bit vector y_s, simultaneously. Extrinsic information is passed between the SISO decoders. This will describe how new a-posteriori information is generated with use of a-priori information (Han *et al.*, 2005). This generated information will be sent to the next decoder in next cycle while the first decoder receives the new input vectors. Second half of each iteration in a cycle corresponds to the interleaved systematic bits. Basically, there are four computation tasks in Log-MAP algorithm. The four tasks are forward metrics computation, branch metrics computation, backward metrics computation and computation of soft or hard soft bit estimate with new extrinsic information. Then decision is found using:

$$L(u_k) = [L^e(u_k) + Lc.y_k^{1,s}] + \log \frac{\sum_{u^+} \tilde{\alpha}_{k-1}(S').\tilde{\beta}_{k-1}(S').\gamma_k^e(S', S)}{\sum_{u^-} \tilde{\alpha}_{k-1}(S').\tilde{\beta}_{k-1}(S').\gamma_k^e(S', S)}$$

where, $L(u_k)$ is the channel L-value, $L^S(u_k)$ is the Apriori value from the previous decoder, L_c is the measure of signal SNR. $\gamma_k^{1,S}$ is the systematic bit. The third long term is the extrinsic value which is calculated using forward, backward and partial branch metrics values. The forward metrics is given by:

$$\overline{\alpha_k}(S) = \frac{\Sigma_{s'} \overline{\alpha_{k-1}}(s').\gamma_k(s', s)}{\Sigma_s \Sigma_{s'} \overline{\alpha_{k-1}}(s').\gamma_k(s', s)}$$

where, $\overline{\alpha_k}(S)$ is the forward metrics value, $\gamma_k(s', s)$ is the full branch metrics value. Similarly the backward metrics is calculated using:

DECI received data	k = 1	k = 2	k = 3	k = 4	k = 5	k = 6	k = 7
$Lc.y_k^s$	2	1	3	-2	2	-4	-5
$Lc.y_k^p$	-5	2	-1	-2	1	-2	-1
DEC2 received data	**k = 1**	**k = 2**	**k = 3**	**k = 4**	**k = 5**	**k = 6**	**k = 7**
$Lc.y_k^s$	3	2	2	-1	2	-4	-5
$Lc.y_k^p$	6	-1	2	2	5	5	6

Fig. 5: Received data from turbo encoder

$$\beta_{k-1}(S') = \frac{\Sigma_s \cdot \overline{\beta_k}(s).\gamma_k(s', s)}{\Sigma_s \Sigma_{s'} \overline{\alpha_{k-2}}(s').\gamma_{k-1}(s', s)}$$

Here, the full branch metrics is calculated using the formula:

$$\gamma(s', s) \propto \exp\left[\frac{1}{2}.L^e(C_k^1).C_k^1 + L_c.\frac{1}{2}.\gamma_k^{1,s}.C_k^1\right] \text{Exp}\left[\Sigma_{i=2}^{q}\left(L_c.\frac{1}{2}.\gamma_k^{i,p}.C_k^i\right)\right]$$

where, the last term in the above equation is the partial branch metrics and it is denoted by $\gamma_k^e(s', s)$. By multiplying forward, backward and partial branch metrics values, the extrinsic values are obtained. To compute the L-values, the ratio of these numbers for the +1 and -1 branches is needed. This value on taking log and add with l-Apriori value (initialize as 0) and l-channel value (1*systematic bit) gives l-value. Based on the sign of l-value, the decision is taken. If the l-value is positive take it as '1', else if negative then take it as '0'.

SIMULATION RESULTS

Simulation result for turbo encoder: To obtain the simulation for Turbo encoder as shown in Fig. 6, taking 'input [6:0]' as the input message bits, 's1' and 's2' as input systematic bits,'p1' and 'p2' are output parity bits. To compare this simulation result with the manual calculation, the input applied to turbo encoder is input [6:0]=1100100, then output obtained from turbo encoder is p1[6:0]=1011001 for s1[6:0] =1100100, p2[6:0]= 1011001 for s2[6:0] = 1100001.

Simulation result for decoder 1

Simulation result for full branch metrics: The full branch metrics using Log-MAP algorithm is simulated by taking 'y11' as input systematic bits, 'y22' as input parity bits from encoder, 'x1 and x2' are the two bits produced by the two encoders for mapping process, 'par' as the partial branch metrics values for constraint length k = 1-7 which is obtained as the intermediate result,'z' as the output full branch metrics values for k = 1-7. In order to compare this simulation result with manual calculation, Inputs given to this algorithm are 'y11' = [2 1 3 -2 2 -4 -5], 'y22'= [5 2 -1 -2 1 -2 -1], 'x1'= [-1 -1 -1 -1 1 1 1 1], 'x2'=[-1 -1 1 1 1 1 -1 -1], then the intermediate result of partial branch metric is:

$$par[7:1] = \begin{bmatrix} 0.0802085 & 0.0802085 & 12.1825 & 12.1825 & 12.1825 & 12.1825 & 0.0802085 & 0.0802085 \\ 0.367879 & 0.367879 & 2.718 & 2.718 & 2.718 & 2.718 & 0.367879 & 0.367879 \\ 1.6478 & 1.64872 & 0.606531 & 0.606531 & 0.606531 & 0.606531 & 1.64872 & 1.64872 \\ 2.718 & 2.718 & 0.367879 & 0.367879 & 0.367879 & 0.367879 & 2.718 & 2.718 \\ 0.606531 & 0.606531 & 1.64872 & 1.64872 & 1.64872 & 1.64872 & 0.606531 & 0.606531 \\ 2.718 & 2.718 & 0.367879 & 0.367879 & 0.367879 & 0.367879 & 2.718 & 2.718 \\ 1.6478 & 1.64872 & 0.606531 & 0.606531 & 0.606531 & 0.606531 & 1.64872 & 1.64872 \end{bmatrix}$$

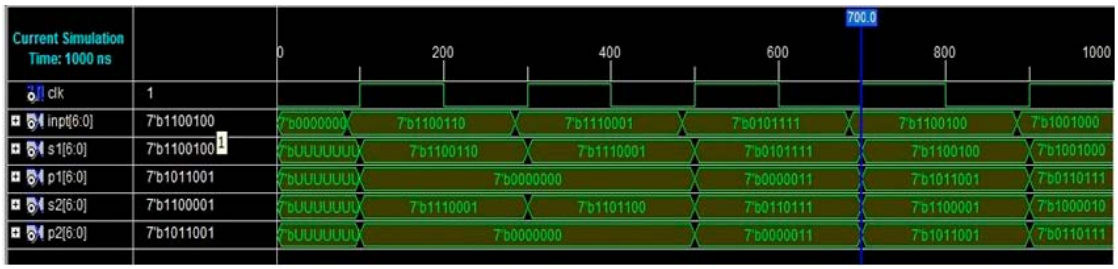

Fig. 6: Simulation result for turbo encoder

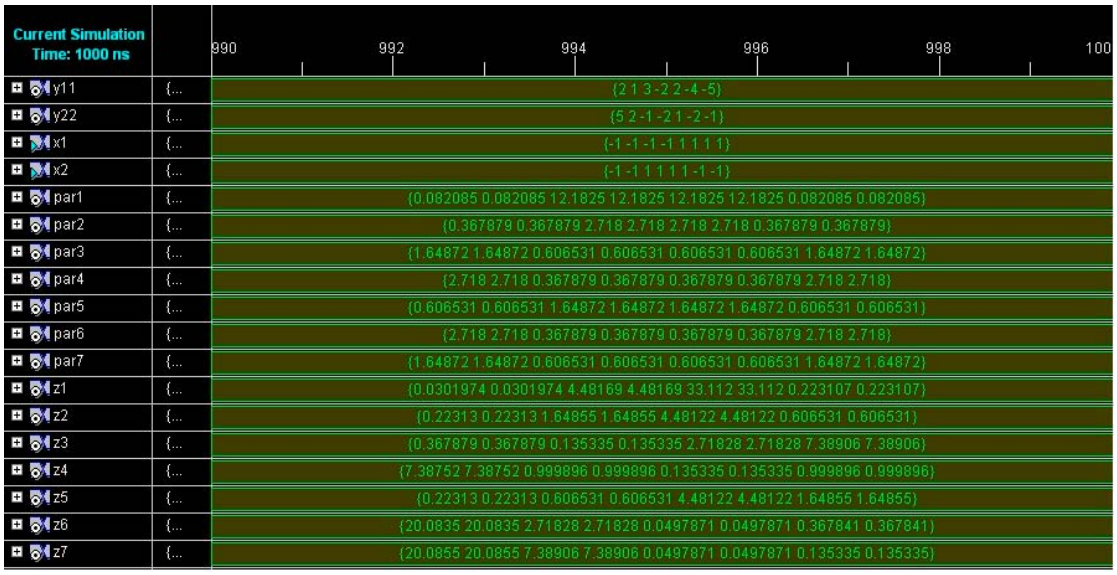

Fig. 7: Simulation result for full branch metrics

then output full branch metrics is:

$$z[7:1] = \begin{bmatrix} 0.0301974 & 0.0301974 & 4.48169 & 4.48169 & 33.112 & 33.112 & 0.223107 & 0.223107 \\ 0.22313 & 0.22313 & 1.64855 & 1.64855 & 4.4.8122 & 4.4.8122 & 0.606531 & 0.606531 \\ 0.367879 & 0.367879 & 0.135335 & 0.135335 & 2.71828 & 2.71828 & 7.38906 & 7.38906 \\ 7.38752 & 7.38752 & 0.999896 & 0.999896 & 0.135335 & 0.135335 & 0.999896 & 0.999896 \\ 0.22313 & 0.22313 & 0.606531 & 0.606531 & 4.48122 & 4.48122 & 1.64855 & 1.64855 \\ 20.0385 & 20.0385 & 2.71828 & 2.71828 & 0.049787 & 0.049787 & 0.367841 & 0.367841 \\ 20.0855 & 20.0855 & 7.38096 & 7.38096 & 0.049787 & 0.049787 & 0.135335 & 0.135335 \end{bmatrix}$$

This result is same as that of result obtained from manual calculation of full branch metrics using Log-MAP algorithm. The simulation result of full branch metrics is given in Fig. 7.

Simulation result for forward metrics: The forward metrics using Log-MAP algorithm is simulated by taking 'z' as the input full branch metrics values for k = 1-7,'d' as output forward metrics value for k = 1-7. In order to compare this simulation result with manual calculation, Inputs given to this algorithm are:

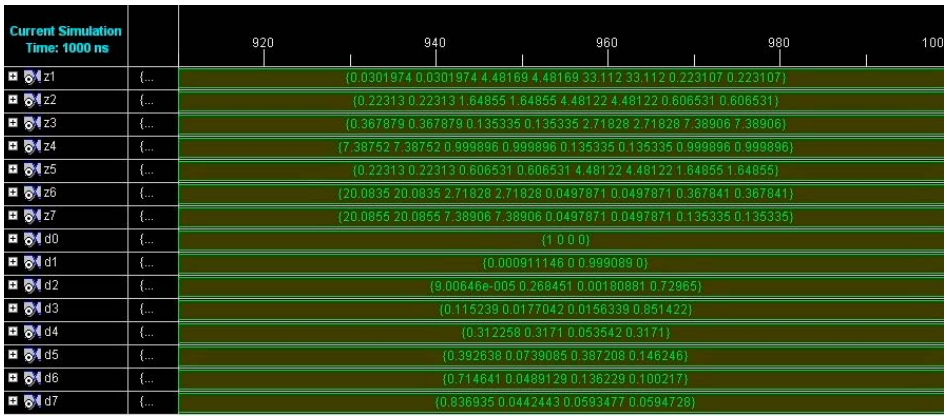

Fig. 8: Simulation result for forward metrics

$$z[7:1] = \begin{bmatrix} 0.0301974 & 0.0301974 & 4.48169 & 4.48169 & 33.112 & 33.112 & 0.223107 & 0.223107 \\ 0.22313 & 0.22313 & 1.64855 & 1.64855 & 4.4.8122 & 4.4.8122 & 0.606531 & 0.606531 \\ 0.367879 & 0.367879 & 0.135335 & 0.135335 & 2.71828 & 2.71828 & 7.38906 & 7.38906 \\ 7.38752 & 7.38752 & 0.999896 & 0.999896 & 0.135335 & 0.135335 & 0.999896 & 0.999896 \\ 0.22313 & 0.22313 & 0.606531 & 0.606531 & 4.48122 & 4.48122 & 1.64855 & 1.64855 \\ 20.0385 & 20.0385 & 2.71828 & 2.71828 & 0.049787 & 0.049787 & 0.367841 & 0.367841 \\ 20.0855 & 20.0855 & 7.38096 & 7.38096 & 0.049787 & 0.049787 & 0.135335 & 0.135335 \end{bmatrix}$$

'd[0]' = {1 0 0 0} for k = 0 then the output forward metrics:

$$d[0:7] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0.0091115 & 0 & 0.999089 & 0 \\ 9.006 & 0.2684 & 0.0018088 & 0.72965 \\ 0.115239 & 0.0177042 & 0.0156339 & 0.851422 \\ 0.312258 & 0.3171 & 0.053542 & 0.3371 \\ 0.392638 & 0.073085 & 0.387208 & 0.146246 \\ 0.714641 & 0.0489129 & 0.136229 & 0.100217 \\ 0.836935 & 0.0442443 & 0.0593477 & 0.0594728 \end{bmatrix}$$

This result is same as that of result obtained from manual calculation of forward metrics using Log-MAP algorithm. The simulation result of forward metrics is given in Fig. 8.

Simulation result for backward metrics: The backward metrics using Log-MAP algorithm is simulated by taking 'z' as input full branch metrics values for k = 1-7, 'b' as output the backward metrics value for k = 1-7. In order to compare this simulation result with manual calculation, Inputs given to this algorithm are:

$$z[7:1] = \begin{bmatrix} 0.0301974 & 0.0301974 & 4.48169 & 4.48169 & 33.112 & 33.112 & 0.223107 & 0.223107 \\ 0.22313 & 0.22313 & 1.64855 & 1.64855 & 4.4.8122 & 4.4.8122 & 0.606531 & 0.606531 \\ 0.367879 & 0.367879 & 0.135335 & 0.135335 & 2.71828 & 2.71828 & 7.38906 & 7.38906 \\ 7.38752 & 7.38752 & 0.999896 & 0.999896 & 0.135335 & 0.135335 & 0.999896 & 0.999896 \\ 0.22313 & 0.22313 & 0.606531 & 0.606531 & 4.48122 & 4.48122 & 1.64855 & 1.64855 \\ 20.0385 & 20.0385 & 2.71828 & 2.71828 & 0.049787 & 0.049787 & 0.367841 & 0.367841 \\ 20.0855 & 20.0855 & 7.38096 & 7.38096 & 0.049787 & 0.049787 & 0.135335 & 0.135335 \end{bmatrix}$$

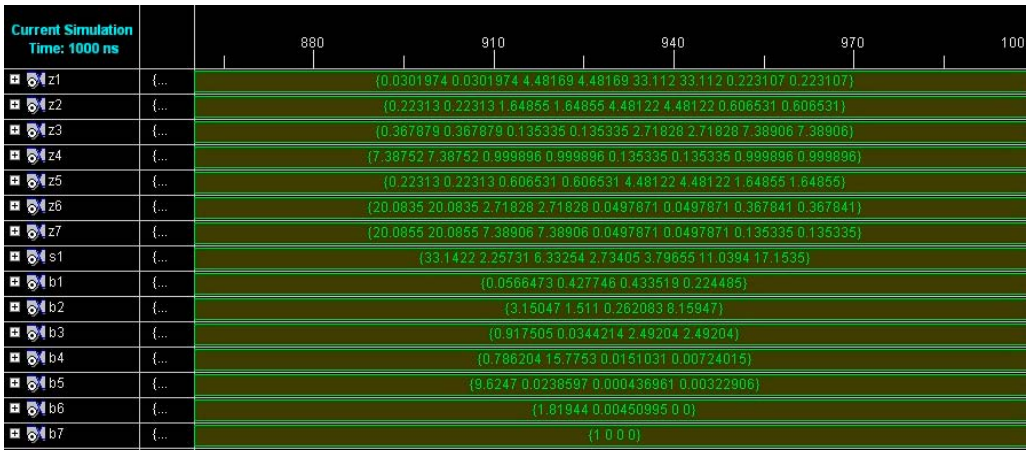

Fig. 9: Simulation result for backward metrics

'b[7]'={1 0 0 0} for k=7, then output backward metrics is:

$$d[0:7]=\begin{bmatrix} 0.0566473 & 0.427746 & 0.433519 & 0.224485 \\ 3.15047 & 1.511 & 0.262083 & 8.15947 \\ 0.917505 & 0.0344214 & 2.49204 & 2.49204 \\ 0.786204 & 15.773 & 0.0151031 & 0.0072402 \\ 9.6247 & 0.0238597 & 0.000437 & 0.0032291 \\ 1.81944 & 0.00450995 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

This result is same as that of result obtained from manual calculation of backward metrics using Log-MAP algorithm. The simulation result of backward metrics is given in Fig. 9.

The l-value using Log-MAP algorithm is calculated by taking 'ext' and 'extr' as the input extrinsic values for k = 1-7, 'zz' as the output l-value for k = 1-7. In order to compare this simulation result with manual calculation, Inputs given to this algorithm are:

$$ext=\begin{bmatrix} 0.00464989 & 0 & 0 & 0 \\ 0.00105601 & 0 & 22.1572 & 0 \\ 0.00013624 & 1.10298 & 0.002734 & 0.0152334 \\ 0.246255 & 0.00072676 & 4.16\,E-05 & 4.94116 \\ 1.82286 & 8.40\,E-05 & 2.85\,E-04 & 0.0124741 \\ 1.94169 & 0 & 0 & 0.0002426 \\ 1.17824 & 0 & 0 & 0 \end{bmatrix} \quad extr=\begin{bmatrix} 0.00464989 & 0 & 0 & 0 \\ 0.00105601 & 0 & 22.1572 & 0 \\ 0.00013624 & 1.10298 & 0.002734 & 0.0152334 \\ 0.246255 & 0.00072676 & 4.16\,E-05 & 4.94116 \\ 1.82286 & 8.40\,E-05 & 2.85\,E-04 & 0.0124741 \\ 1.94169 & 0 & 0 & 0.0002426 \\ 1.17824 & 0 & 0 & 0 \end{bmatrix}$$

then output l-value is 'zz' = {7.0346 -3.68516 1.03232 -2.01331 1.00869 -3.57845 -3.68172}. This result is same as that of result obtained from manual calculation of extrinsic values using Log-MAP algorithm.

Simulation result for finding decision: The decision using Log-MAP algorithm is simulated by taking 'zz' as the input l-value for k = 1-7, 'lext1' as the input l-extrinsic value for decoder1, 'dec1' as output decision for decoder 1. In order to compare the simulation result with manual calculation.

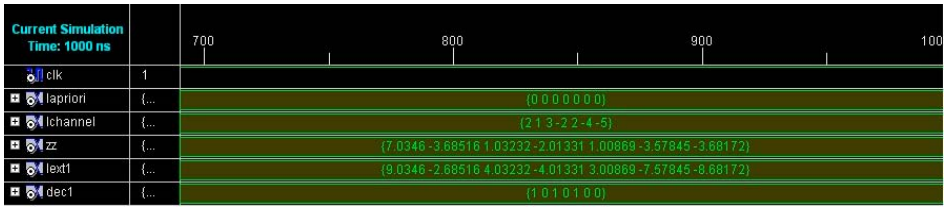

Fig. 10: Simulation result for finding decision

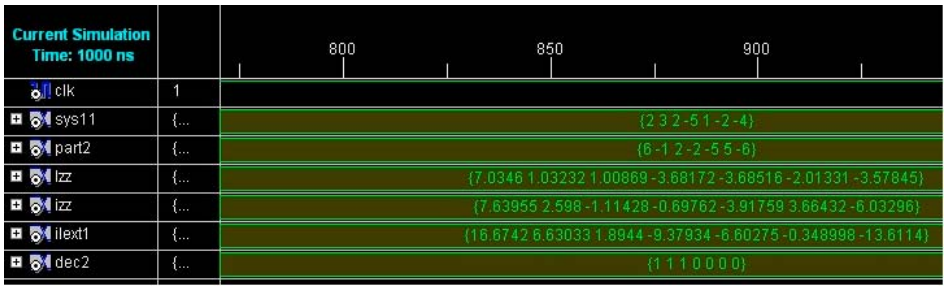

Fig. 11: Simulation result for decoder 2 decision

Inputs given to this algorithm are, Initializing input 'lapriori' as {0 0 0 0 0 0 0}, input 'lchannel' as {2 1 3 -2 2 -4 -5}, 'zz' = {7.0346 -3.68516 1.03232 -2.01331 1.00869 -3.57845 -3.68172} then the output is 'lext1' = {9.0346 -2.68516 4.03232 -4.01331 3.00869 -7.57845 -8.68171} and decision taken by decoder1 is 'dec1' = {1 0 1 0 1 0 0}. This result is same as that of result obtained from manual calculation of decision finding for first decoder using Log-MAP algorithm. The simulation result of for finding decision is given in Fig. 10.

Simulation result for decoder 2 decision: The same algorithm is repeated for decoder 2 by giving interleaved systematic bits, second parity bits and interleaved l-extrinsic which is l-value of decoder1 as input, to obtain the second decision value which then de-interleaved to get the original message bit. The decision using Log-MAP algorithm for decoder2 is simulated by taking 'lzz' as input interleaved l-value from decoder 1,'sys11' as the input interleaved systematic bits,'part2' as input parity bits then the intermediate results obtained from decoder 2 are 'izz' as the l-value for k = 1-7 from decoder2, 'ilext1' as the output l-extrinsic value for decoder 2, then 'dec2' as final output decision for decoder 2. In order to compare the simulation result with manual calculation, Inputs given to this algorithm are 'lzz[7:1]' = {7.0346, 1.03232, 1.00869, -3.68172, -3.68516, -2.01331, -3.57845}, 'sys11' = {2, 3, 2, -5, 1, -2, -4} and 'part2' = {6 -1 2 -2 -5 5 -6} then intermediate results obtained from decoder2 are izz[1:7]' = {7.63955, 2.598, -1.11428, -0.69762, -3.91759, 3.66432, -6.03296}, 'ilext1[1:7]' = {16.6742, 6.63033, 1.894, -9.37934, -6.60275, -0.348998, -13.6114}, then decision from decoder2 is 'dec2[1:7]' = {1, 1, 1, 0, 0, 0, 0}. This result is same as that of result obtained from manual calculation of decision finding for decoder 2 using Log-MAP algorithm. The simulation result of decoder 2 decision is given in Fig. 11.

Simulation result for de-interleaver: The final decision using Log-MAP algorithm is simulated by taking 'dec 2' as the input for de-interleaver then 'deci' as the output from de-interleaver. In order to compare the simulation result with manual calculation, Inputs given to the de-interleaver

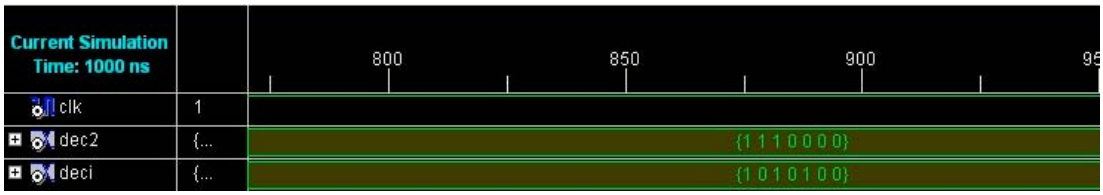

Fig. 12: Simulation result for de-interleaver

Table 1: Summary of device utilization for turbo encoder

Resources	Available	Used	Utilization (%)
No. of Slices	7680	6	1
No. of four input LUTs	15360	11	1
No. of bonded IOBs	333	21	6

Table 2: Summary of device utilization for turbo decoder

Resources	Available	Used	Utilization (%)
No. of Slices	7680	528	6
No. of four input LUTs	15360	1008	6
No. of bonded IOBs	333	56	16
No. of MULT 18×18 sec	24	7	29

of Turbo decoder is dec 2 = {1, 1, 1, 0, 0, 0, 0} and output is deci = {1, 0, 1, 0, 1, 0, 0} which is same as that of the original message bits. The output from this is decoder is applied to next decoder, this process prolong until to obtain the original message bits in the turbo decoder output. The simulation result of De-interleaver is given in Fig. 12.

SYNTHESIS REPORT

The VHDL coding for turbo coder and encoder are synthesized and downloaded into Xilinx XC3S200-4ft256, Spartan 3 family. The synthesis result of turbo encoder is given in Table 1 and synthesis result of turbo decoder is given in Table 2.

CONCLUSION

This paper mainly targets the simulation and synthesis of parallel architecture for turbo decoder using Log-MAP algorithm which has better BER performance with reasonable computational complexity. The synthesis report shows that 4 input LUTs, number of slices and Multipliers utilized for implementation of turbo decoder on FPGA using Log-MAP algorithm is less. Thus remaining slices of the same FPGA utilized for implementation of other blocks of LTE receiver in future. Therefore the proposed design becomes cost effective. Because of parallel processing, the proposed work meets the requirement of LTE standard. By this way, 3GPP Long Term Evolution standard become efficient. Thus proposed turbo decoder can also assure the specifications of OFDM systems which includes DAB, DVB, VDSL and 802.16.

REFERENCES

Gupta, A.K. and S. Kumar, 2010. VHDL Implementation of different turbo encoder using Log-MAP decoder. J. Telecommun., 2: 49-53.

Hagenauer, J., E. Offer and L. Papke, 1996. Iterative decoding of binary block and convolutional codes. IEEE Trans. Inform. Theory, 42: 429-445.

Han, J.H., A.T. Erdogan and T. Arslan, 2005. A power efficient re-configurable Max-Log-MAP turbo decoder for wireless communication systems. Proceedings of the IEEE International SOC Conference, September 19-23, 2005, Herndon, VA, pp: 247-250.

Karim, S.M., G. Mahale and I. Chakrabart, 2011. A pipelined architecture for high throughput efficient turbo decoder. Proceedings of the International Conference on Electronics, Information and Communication Engineering, December 12-16, 2011, Foundation of Computer Science, New York, USA.

Lucian, A.P. and R. Stoian, 2008. The decision reliability of MAP, Log-MAP,Max-Log-MAP and SOVA algorithms for turbo codes. Int. J. Commun., 2: 65-74.

Oliver, M., A. Baghdadi and M. Jezequel, 2010. Parallelism efficiency in convolutional turbo decoding. EURASIP J. Adv. Signal Process., Vol. 2010. 10.1155/2010/927920

Perttu, S., H. Sorokin and J. Takala, 2008. A programmable Max-Log-MAP turbo decoder implementation. VLSI Des., Vol. 2008 10.1155/2008/319095.

Salim, M., R.P. Yadav and S.R. Kanth, 2010. Performance analysis of Log-MAP, SOVA and modified SOVA algorithm for turbo decoder. Int. J. Comput. Appl., 9: 29-32.

Vishvaksenan, K.S., R. Seshasayanan and D.V. Hariprasad, 2011. FPGA implementation of turbo decoder for IDMA scheme. Res. J. Inform. Technol., 3: 104-113.

Yang, S. and J.R. Cavallaro, 2011. Efficient hardware implementation of highly parallel 3GPP LTE/LTE advance turbo decoder. Int. VLSI J., 44: 305-315.

Zeng, X.J. and Z.L. Hong, 2002. Design and implementation of a turbo decoder for 3GW-CDMA systems. IEEE Trans. Consum. Electron., 48: 284-291.