Implementation of Turbo Code with Early Iteration Termination in GNU Radio

Salija P, Yamuna B
Department of Electronics and Communication Engineering,
Amrita School of Engineering, Coimbatore,
Amrita Vishwa Vidyapeetham,
Amrita University, India.
p_salija@cb.amrita.edu

***Abstract*—Wireless communication systems demand energy efficient and performance optimized error correction scheme. Turbo code, an iterative error correction code, shows strong error correction capability. Many wireless communication systems use Turbo code in their standards due to its near ideal performance. The iterative nature of Turbo decoder introduces additional computations, decoding delay, and power consumption. The number of iterations required to obtain the desired output varies with the channel conditions. Early iteration termination at appropriate time reduces the computational complexity without performance degradation. An early iteration termination based on the absolute value of the mean of extrinsic information has been proposed recently. This technique efficiently terminates the iteration at low and high SNR conditions and also minimizes the half iterations. Software Defined Radio (SDR), a communication system technology, is a common platform that supports various standards. GNU Radio is the software part of SDR that allows implementing various features of communication systems. A low complex Turbo decoder in GNU Radio along with Universal Software Radio Peripheral (USRP) helps to implement real time applications with low decoding delay and reduced complexity. In this paper, Turbo CODEC with early iteration termination has been implemented in GNU Radio platform.**

***Index Terms*—Early Iteration Termination; Turbo Codes; GNU Radio.**

I. INTRODUCTION

Turbo code is an attractive error correction code with wide applications in communication systems because of its near Shannon's limit performance [1]. The strong error correcting performance of Turbo code is due to the Recursive Systematic Convolutional (RSC) encoders separated by an interleaver and the iterative Turbo decoding operation [2], [3], [4]. Iterative nature of Turbo code introduces additional computational complexity and leads to decoding delay and power consumption [5]. This is because the decoder needs to perform a certain number of iterations to obtain a tolerable error rate; the number of iterations required depends upon the channel conditions and the block size [6]. In a standard Turbo decoder, the process of decoding is iteratively carried out for a fixed number of iterations. As the number of iterations increases, the error correction performance also increases, but with a corresponding increase in computational complexity. The selection of the number of iterations involves a tradeoff between performance and complexity. The necessity of early iteration termination arises when the desired performance is reached well before the final iteration or when there is no scope of performance improvement even with infinite iterations [7]. Early iteration termination in such scenario reduces decoding delay and power consumption.

The importance of SDR in wireless communication has been realized in recent years. SDR implements significant aspects of physical layer functionality in software rather than hardware [8]. It is easier to modify and upgrade the functionality in software than hardware. SDR supports multiple modes, multiple bands and different standards in a single platform and allows the dynamic selection of parameters for the modules such as channel coding, modulator/demodulator, multiplexing/de-multiplexing and signal generation. The wireless communication applications demand an integration of various features and a reduction in power consumption. Many wireless communication standards include Turbo codes are used in real time and for broad band communication applications [9],[10],[11]. Turbo decoder consumes most of the power and is responsible for the decoding delay due to the iterative nature of decoding operation. Functionality in most of the devices is limited by the available power especially in resource constrained networks [12]. A suitable iteration termination technique reduces the computational complexity, and hence power consumption and decoding delay. Early iteration termination based on the absolute value of the mean of extrinsic information is a simple and efficient method to reduce the additional complexity without performance degradation. This technique is applicable at low and high SNR conditions and minimizes the half iteration computations. There is a clear need for Turbo decoder block with reduced complexity and acceptable performance level in GNU Radio, which can be used for real time application [13].

In this paper, we focus on the creation and implementation of a Turbo CODEC block with early iteration termination in GNU Radio platform. The paper is organized as follows: Section II presents the overview of the Software Defined Radio (SDR) Platform. Section III deals with the Turbo code operation and MAP algorithm. The concept of early iteration termination in Turbo code and the early iteration termination based on the absolute value of the mean of extrinsic information is presented in section IV. Section V shows the results and discussion followed by conclusion in section V.

II. SOFTWARE DEFINED RADIO PLATFORM

In wireless communication systems, the size, cost, and competitions introduce limitations in implementing new systems on hardware in order to support various standards. The proliferation of wireless communication standards requires more flexible designs [5]. SDR implements most of the physical layer functionality in software, and hence helps to upgrade the system by software modifications. SDR is a communication system technology that supports various standards, different functionalities like modulation/ demodulation, encoding/decoding, multiplexing/ demultiplexing etc., all with great implementation flexibility [14]. System upgradation with reduced cost, reduced hardware dependency and compatibility issue, and implementation of different functionalities are the facilities of SDR. Hardware and software co-designs are necessary to implement SDR applications [15]. The block diagram of SDR is shown in Figure 1.

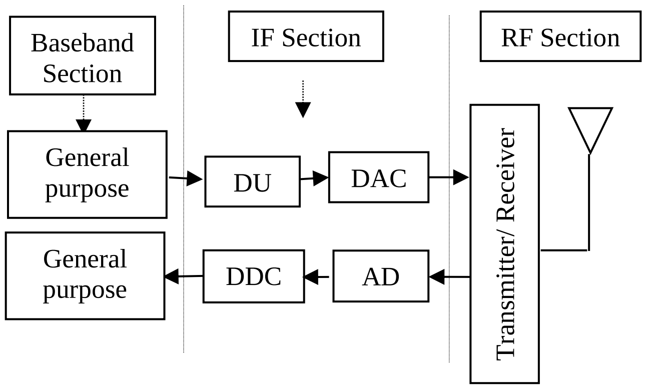


Figure 1: SDR Block diagram [5]

The Radio Frequency (RF) section is responsible for two tasks namely, the transmission and reception of radio signals and the conversion of the RF signals to the Intermediate Frequency (IF) at the receiver side and IF signals to RF signals at the transmitter side. IF section performs the Digital Up Conversion (DUC) and Digital to Analog Conversion (DAC) at the transmitter side and the Digital Down conversion (DDC) and Analog to the Digital conversion(ADC) at the receiver side. Base band section incorporates the features of blocks on software and performs according to the demand of the applications.

GNU Radio is an open source and free software package consisting of various blocks and allows the creation of new blocks. Various signal processing blocks can be built and modified on GNU Radio platform depending on the application. GNU Radio consists of different signal processing blocks, and flow graph connects the different signal processing blocks to implement different features according to the demand [16]. Blocks are interconnected using Python flow. It can be interfaced with the external hardware component in order to transmit/receive signal. USRP - the hardware component - used in SDR allows the real time implementation of any communication system [13].

III. TURBO CODE

Turbo code is a practical Forward Error Correction Code (FEC) with strong error correction capability. Turbo code forms a class of iterative channel codes with wide applications in communication systems despite its computational complexity.

The major applications of Turbo codes are in mobile communication, deep space communication, wireless sensor networks, video conferencing [17], Orthogonal Frequency Division Multiplexing (OFDM), and Wireless Metropolitan Area Networks (WMAN) [18],[19],[11]. Turbo code has improved channel quality interaction in satellite communication and is adapted in Mars Reconnaissance Orbiter [20]. Digital Video Broadcast (DVB-T) standard, telemetry coding standard by the CCSDS, High-Speed Downlink Packet Access (HSDPA), WiMax, and IEEE 802.11n are the key associated standards. W-CDMA (3rd Generation Partnership Project (3GPP)), and Universal Mobile Telecommunications System (UMTS), Long Term Evolution (LTE) and CDMA2000 ([21], [22]) standards also incorporate Turbo code.

A. Turbo Encoder

Turbo encoder requires at least two RSC encoders with an interleaver separation [23]. The Turbo encoder produces systematic output and parity outputs. If necessary, data rate can be changed by applying puncturing at the output of the encoder. Since the probability of the two RSC encoders producing a low weight codeword is low, the concatenated codeword would be a high weight codeword [24]. This contributes to the superior performance of Turbo code.

B. Turbo Decoder

Soft Input Soft Output (SISO) algorithms like Maximum A Posteriori (MAP) or Soft Output Viterbi Algorithm (SOVA) of the Turbo decoder computes the Log Likelihood Ratios (LLR) of each of the received bits. The generic block diagram of Turbo decoder is shown in Figure 2.

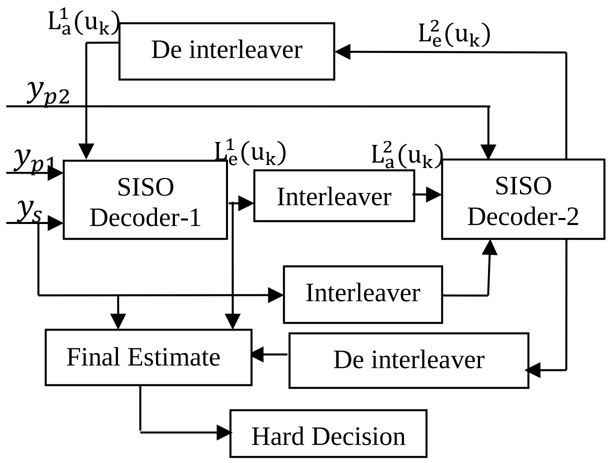


Figure 2: Turbo Decoder [24]

The inputs to the SISO decoders are the received systematic information, received parity information and apriori information from the other component decoder [25]. SISO decoders generate extrinsic aposteriori information from the received information and apriori information. During the first iteration, apriori information for SISO Decoder-1 was set to zero. The first SISO decoder generates extrinsic aposteriori information from the received systematic and parity information and passes this extrinsic information to the next component decoder as apriori information after interleaving. Similarly, the aposteriori information from the second SISO decoder is passed to the first SISO decoder after de-interleaving. This process continues till the maximum number of iteration is reached [23].

C. MAP Algorithm

MAP algorithm also known as BCJR algorithm [23], outperforms SOVA under low noise conditions and has excellent error correction capability with reasonable complexity. Log Likelihood Ratio (LLR) for each of the received information bit was calculated using MAP algorithm.

The procedure for calculating the LLR values is given in Table 1. Let $u = [u_1, u_2 \dots u_n]$ be the data sequence encoded and transmitted through the noisy channel. Let $y = [y_1, y_2 \dots y_n]$ be the corresponding received information from the channel at the receiver. The procedure includes the calculation of branch metric, forward metric and backward metric.

Table 1.
MAP Algorithm [23]

1. Initialize the forward metric value and backward metric value as follows,

$$\alpha_0(s) = \begin{cases} 1, s' = 0 \\ 0, s' \neq 0 \end{cases} \qquad \beta_n(s) = \begin{cases} 1, s' = 0 \\ 0, s' \neq 0 \end{cases}$$

2. The branch metric from one state to next state transition is calculated as follows:

$$\gamma_k(s', s) = e^{\frac{u_k L_a(u_k)}{2}} e^{\left(\frac{L_c}{2}\right) y_k . x_k} \tag{1}$$

Where: $L_c = \frac{4E_c}{N_0}$, is the channel reliability factor

s' and s represent the previous state and present state respectively.

3. The forward metric for the kth node is calculated as given below,

$$\alpha_{k+1}(s) = \sum_{s \in \omega k} \gamma_k(s', s) \alpha_k(s') \tag{2}$$

4. The backward metric for the kth node is calculated as given below:

$$\beta_k(s') = \sum_{s' \epsilon \omega k+1} \gamma_k(s', s) \beta_{k+1}(s) \tag{3}$$

5. Aposteriori Probability LLR values related to each received information bit can now be calculated as:

$$L_{uk} = ln\left(\frac{\sum_{(s',s) \in \sum_k^+ .} \alpha_k(s') \gamma_k(s', s) \beta_{k+1}(s)}{\sum_{(s',s) \in \sum_k^- .} \alpha_k(s') \gamma_k(s', s) \beta_{k+1}(s)}\right) \tag{4}$$

MAP algorithm is associated with several computations; one variant of MAP algorithm to reduce the intensive mathematical calculations is the LOG-MAP algorithm. This uses the following approximation function to perform the decoding operation.

$$\max{}^*(x, y) = \ln(e^x + e^y) = \max(x, y) + \ln\left(1 + e^{-|x-y|}\right) \tag{5}$$

Another popular variant of MAP algorithm is the MAX-LOG-MAP algorithm which uses the approximations;

$$\max{}^*(x, y) = \ln(e^x + e^y) = \max(x, y) \tag{6}$$

MAX-LOG-MAP algorithm reduces the computational complexity with performance degradation when compared to the MAP and LOG-MAP algorithm.

IV. EARLY ITERATION TERMINATION

Turbo decoding algorithm is associated with the large computational complexity and each iteration introduces additional decoding complexity, decoding delay and power consumption ([5],[26]). One of the major challenges in wireless communication is to establish energy efficient transmission without performance degradation. In resource-constrained networks, functionality is limited by the available power. Real time applications and multimedia applications require Turbo decoding with reduced latency. One efficient and simple technique to reduce the computational complexity is to terminate the iteration at the appropriate time, when further iteration provides little or no improvement. MAP or LOG-MAP algorithm itself is computationally complex and each iteration introduces additional computational complexity to Turbo decoder. The iteration number required to attain the desirable result depends on the channel conditions. If the channel conditions are good, only a few iterations are required for the decoding to converge, but if the channel conditions are bad further improvement is negligible even after infinite iterations. Sometimes errors will reduce to an acceptable level and again appear to increase with additional iterations. If the iteration is terminated at appropriate time, reduction in computational complexity can be achieved without performance degradation [27]. An efficient early iteration termination technique reduces the decoding delay, power consumption, error accumulation, computation complexity associated with the decoding and improves the throughput by decoding more number of data blocks [6].

The objective of the early termination is to reduce the unnecessary computations which does not contribute to further improvement in error correction performance. Standard Turbo code performs fixed number of iterations at any channel conditions. Under low noise conditions error correction performance improvement between successive iterations is negligible. At high SNR conditions, only a few iterations are required to obtain the desired output and further iterations provide no improvement in error correction. An efficient early termination technique capable of terminating iterations at low and high SNR conditions would be optimum in terms of performance and complexity. Figure 3 shows the plot of average number of iterations for different SNRs [28].

The horizontal line in Figure 3(a) shows that the decoder goes through fixed number of iteration as in a standard Turbo code irrespective of channel conditions. Whereas curved line shows the actual number of iterations required to obtain desired output depending on the channel conditions. However, at low SNR the desired output is never reached even after infinite iterations; at high SNR less number of iterations are required than the fixed number to obtain the desired output. Figure 3(b) shows the iteration termination at high SNR conditions, without going up to the maximum number of iterations. Figure 3(c) shows the iteration termination at low and high SNR conditions. At low SNR conditions, decoding is terminated after a few iterations since the desired output is never reached even after the fixed number of iterations. As SNR increases, the iterations stop at proper time because the desired output is reached early. Iteration termination at low and high SNR conditions reduces the computational complexity without performance degradation.

Appropriate iteration termination leads to reduction in half iteration computations. An early iteration termination technique for Turbo code at proper time, which saves the half iteration computation at low and high SNR conditions without degrading the performance remains a challenge. An efficient and simple early iteration termination technique based on absolute value of the mean of extrinsic information [27] is implemented in GNU Radio.

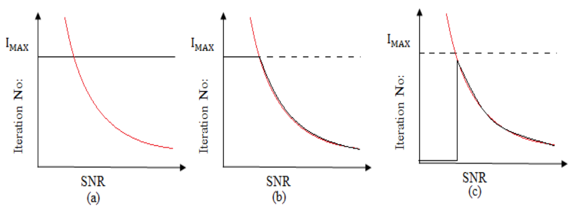


Figure 3: Average iterations required for, a) Fixed number of iteration, b) high SNR conditions, c) high and low SNR conditions.[28]

A. *Absolute value of the Mean of Extrinsic Information Based Iteration Termination in GNU Radio*

An efficient early iteration termination technique reduces additional decoding delay and computational complexity. This paper focuses on implementing an efficient early iteration termination in GNU Radio. An efficient early iteration termination technique, which is capable of terminating at low and high SNR conditions and also save half iteration is an important requirement in GNU Radio to minimize the decoding delay and computational complexity in real time implementations using USRP. Existing Turbo decoder block in GNU Radio does not incorporate early iteration termination. Early iteration termination technique based on absolute mean value of extrinsic information is implemented in GNU Radio. The early iteration termination technique uses the extrinsic information output from both component decoders to stop the iteration at appropriate time. The early iteration termination based on absolute value of the mean of extrinsic information terminates when further iteration provides no improvement or when the desired output is obtained from any of the component decoders. The Turbo CODEC with early iteration termination implemented in GNU Radio will reduce the decoding delay as well as computational complexity and can integrate with USRP for real time applications.

The early iteration termination implemented in GNU Radio is based on the extrinsic information [27]. Extrinsic information greatly influences the final decoding decision. At low SNR conditions, the extrinsic information at both component decoders does not vary between iterations and for high SNR conditions, the extrinsic information at both component decoder output increases with iterations. The mean value of the extrinsic information also varies according to that of extrinsic information. Mean value of the extrinsic information is calculated at the output of both component decoders and decision on early iteration termination is made based on the absolute value of the calculated mean.

Absolute value of the mean is calculated at the output of both the component decoders at ith iteration and compared with a predefined threshold Th1 as in Equation (7):

$$abs(MEAN(i)) \geq Th1 \tag{7}$$

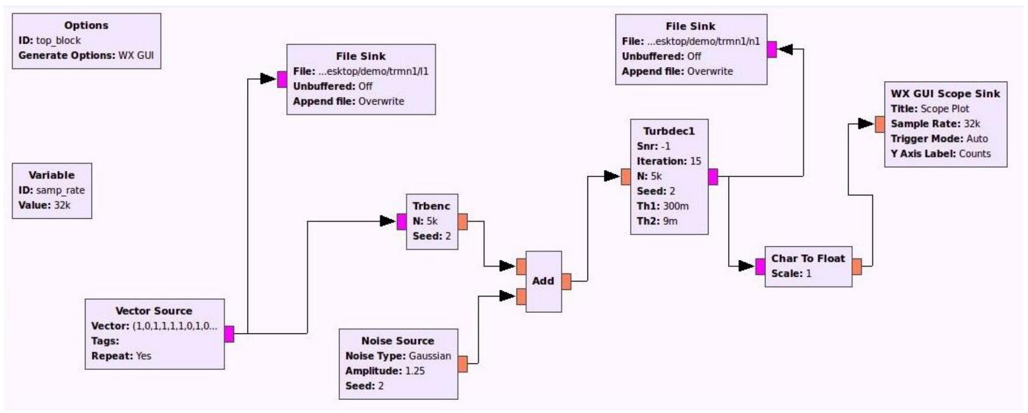
If the condition in Equation 7 is satisfied, iteration stops; otherwise, it compares the calculated mean value at ith iteration with the previous one as in Equation (8):

$$abs\ (MEAN(i) - MEAN(i-1)) \leq Th2 \tag{8}$$

If the condition in Equation 8 is satisfied, the iteration stops. There is a tradeoff between performance and complexity in choosing these threshold values. As Th1 value increases, complexity increases correspondingly with the improvement in performance. Similarly, as Th2 reduces, complexity increases and performance improves. The algorithm for early iteration termination based on absolute value of the mean of extrinsic information is given in Table 2 [27].

Table 2.
Algorithm [27]

```
Initialization: $i$=1; STOP=1; Set $imax$
WHILE (STOP OR $i \leq imax$)
Perform the $i$th iteration for component decoder 1
Calculate, MEAN1, mean of extrinsic information at the output of component decoder-1;
    IF abs(MEAN1)(i) ≥ Th1
    STOP = 0;
    END IF
    IF i >1 && abs(MEAN1(i) - MEAN1(i-1)) ≤ Th2
    STOP = 0;
    END IF
Perform the $i$th iteration for Turbo decoder 2
Calculate, MEAN2, mean of extrinsic information at the output of component decoder-2;
    IF abs( MEAN2(i)) ≥ Th1
    STOP = 0;
    END IF
    IF abs(MEAN2(i) - MEAN2(i-1)) ≤ Th2
    STOP = 0;
    END IF
    $i=i+1;$
    END WHILE
Final Output
```


Options
ID: top_block
Generate Options: WX GUI

File Sink
File: ...esktop/demo/trmn1/l1
Unbuffered: Off
Append file: Overwrite

File Sink
File: ...esktop/demo/trmn1/n1
Unbuffered: Off
Append file: Overwrite

WX GUI Scope Sink
Title: Scope Plot
Sample Rate: 32k
Trigger Mode: Auto
Y Axis Label: Counts

Turbdec1
Snr: -1
Iteration: 15
N: 5k
Seed: 2
Th1: 300m
Th2: 9m

Variable
ID: samp_rate
Value: 32k

Trbenc
N: 5k
Seed: 2

Add

Char To Float
Scale: 1

Vector Source
Vector: (1,0,1,1,1,1,0,1,0...
Tags:
Repeat: Yes

Noise Source
Noise Type: Gaussian
Amplitude: 1.25
Seed: 2

Figure 4: Early iteration termination block for Turbo code in GNU Radio

V. Results and Discussions

Turbo CODEC with early iteration termination technique based on absolute value of the mean of extrinsic information has been created as a block in GNU Radio. Turbo code with generator polynomial $[1, \frac{1+D+D^3}{1+D^2+D^3}]$ and overall coding rate of 1/3 is considered. The experimental setup for the Turbo code with early iteration termination is shown in Figure 4.

Trbenc and Turbdec1 in Figure 4 are the newly created blocks. ‘Trbenc’ block is a Turbo encoder with Binary Phase Shift Keying (BPSK) modulation. ‘Turbdec1’ block is a Turbo decoder with early iteration termination based on absolute value of the mean of extrinsic information. The algorithm is capable of reducing the average number of iterations of Turbo decoder, and hence the computational complexity [27]. This technique calculates the mean of the extrinsic information at the output of the component decoder after each half iteration. Then, if the absolute of mean value is greater than a predefined threshold or the difference in the mean value between iterations lies within a predefined threshold, the iteration stops. Reduction in iteration number achieved with the technique is as shown in Figure 5 [27]. LOG-MAP algorithm has been considered for implementing Turbo decoder. Vector Source block generates binary data continuously; File Sink saves the input and output information and Scope sink is used to observe the decoded output. The Noise Source block is used to add noise to the encoded data.

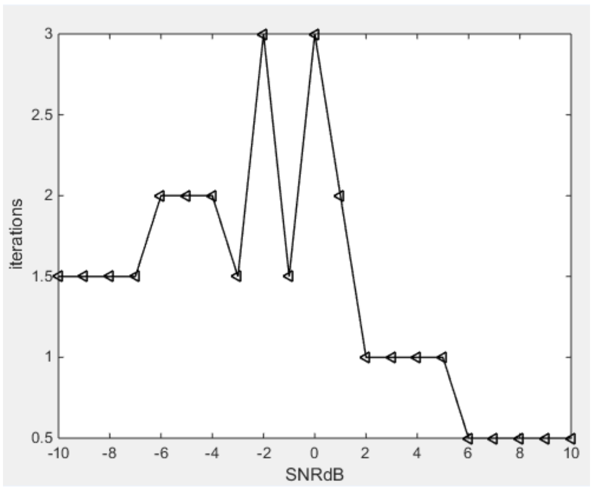


Figure 5: Number of iterations as a function of SNR [27]

Details of parameters of Turbo CODEC with early iteration termination technique based on absolute value of the mean of extrinsic information implemented in GNU Radio are given in Table 3.

Figure 6 shows the WX-GUI Scope Sink output of Turbo decoder at SNR of 10 dB and Figure 7 shows the binary inputs that have been generated using vector source. Figure 8 shows the corresponding expected output from Turbo decoder that has been written on text editor using file sink at SNR of 10 dB. Both Figure 7 and Figure 8 clearly indicate that the decoder has correctly decoded at 10dB. Early iteration termination based on the absolute value of the mean of extrinsic information reduces the average number of iterations at low and high SNRs [27].

Table 3
Parameters related to Turbo CODEC with early termination

Parameter	Details
'Snr'	Signal to Noise Ratio value used to calculate channel reliability.
'Iteration'	Maximum iteration number.
'N'	Number of message bits.
'Seed'	Interleaving and de-interleaving operation based on the 'Seed' provided.
'Th1' & Th2'	Threshold values for early iteration termination of Turbo code corresponding to the Equations (7) and (8) respectively.

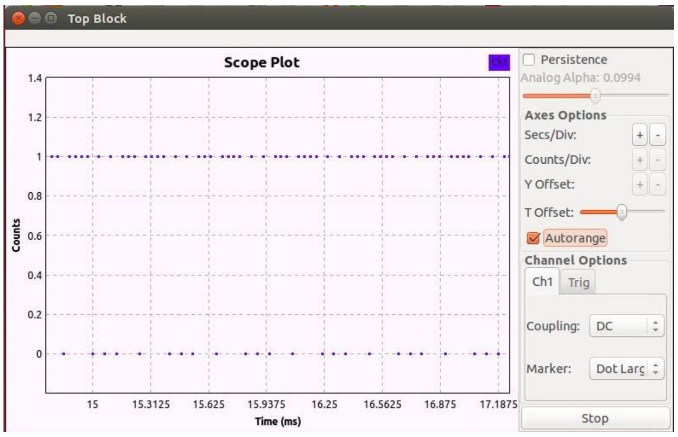
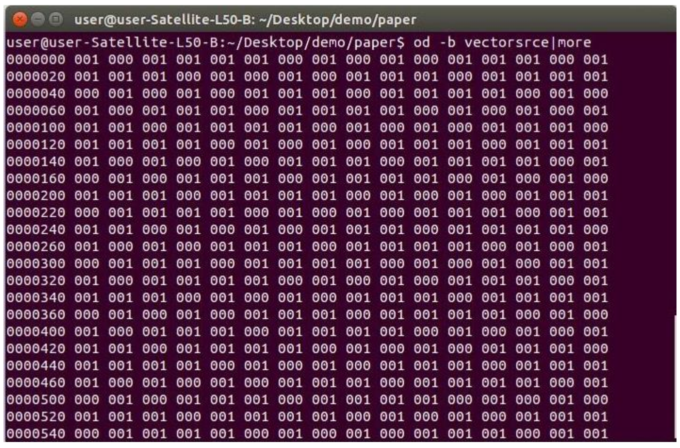
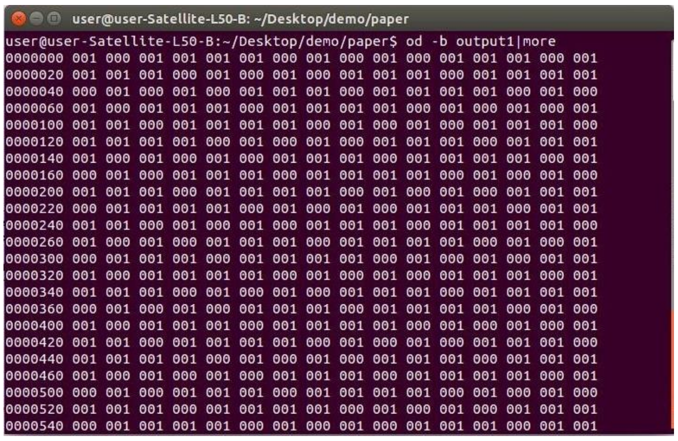


Figure 6: WX-GUI Scope Sink output of Turbo decoder



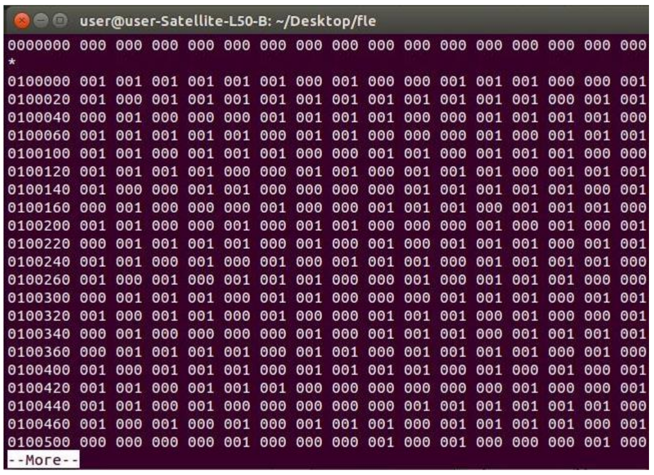
```
user@user-Satellite-L50-B: ~/Desktop/demo/paper
user@user-Satellite-L50-B:~/Desktop/demo/paper$ od -b vectorsrce|more
0000000 001 000 001 001 001 001 000 001 000 001 000 001 001 001 000 001
0000020 001 001 001 000 001 000 001 000 001 001 001 000 001 001 001 001
0000040 000 001 000 001 000 001 001 001 000 001 001 001 001 000 001 000
0000060 001 000 001 001 001 000 001 001 001 001 000 001 000 001 000 001
0000100 001 001 000 001 001 001 001 000 001 000 001 000 001 001 001 000
0000120 001 001 001 001 000 001 000 001 000 001 001 001 000 001 001 001
0000140 001 000 001 000 001 000 001 001 001 000 001 001 001 001 000 001
0000160 000 001 000 001 001 001 000 001 001 001 001 000 001 000 001 000
0000200 001 001 001 000 001 001 001 001 000 001 000 001 000 001 001 001
0000220 000 001 001 001 001 000 001 000 001 000 001 001 001 000 001 001
0000240 001 001 000 001 000 001 000 001 001 001 000 001 001 001 001 000
0000260 001 000 001 000 001 001 001 000 001 001 001 001 000 001 000 001
0000300 000 001 001 001 000 001 001 001 001 000 001 000 001 000 001 001
0000320 001 000 001 001 001 001 000 001 000 001 000 001 001 001 000 001
0000340 001 001 001 000 001 000 001 000 001 001 001 000 001 001 001 001
0000360 000 001 000 001 000 001 001 001 000 001 001 001 001 000 001 000
0000400 001 000 001 001 001 000 001 001 001 001 000 001 000 001 000 001
0000420 001 001 000 001 001 001 001 000 001 000 001 000 001 001 001 000
0000440 001 001 001 001 000 001 000 001 000 001 001 001 000 001 001 001
0000460 001 000 001 000 001 000 001 001 001 000 001 001 001 001 000 001
0000500 000 001 000 001 001 001 000 001 001 001 001 000 001 000 001 000
0000520 001 001 001 000 001 001 001 001 000 001 000 001 000 001 001 001
0000540 000 001 001 001 001 000 001 000 001 000 001 001 001 000 001 001
```

Figure 7: Binary input from the vector source.



```
user@user-Satellite-L50-B: ~/Desktop/demo/paper
user@user-Satellite-L50-B:~/Desktop/demo/paper$ od -b output1|more
0000000 001 000 001 001 001 001 000 001 000 001 000 001 001 001 000 001
0000020 001 001 001 000 001 000 001 000 001 001 001 000 001 001 001 001
0000040 000 001 000 001 000 001 001 001 000 001 001 001 001 000 001 000
0000060 001 000 001 001 001 000 001 001 001 001 000 001 000 001 000 001
0000100 001 001 000 001 001 001 001 000 001 000 001 000 001 001 001 000
0000120 001 001 001 001 000 001 000 001 000 001 001 001 000 001 001 001
0000140 001 000 001 000 001 000 001 001 001 000 001 001 001 001 000 001
0000160 000 001 000 001 001 001 000 001 001 001 001 000 001 000 001 000
0000200 001 001 001 000 001 001 001 001 000 001 000 001 000 001 001 001
0000220 000 001 001 001 001 000 001 000 001 000 001 001 001 000 001 001
0000240 001 001 000 001 000 001 000 001 001 001 000 001 001 001 001 000
0000260 001 000 001 000 001 001 001 000 001 001 001 001 000 001 000 001
0000300 000 001 001 001 000 001 001 001 001 000 001 000 001 000 001 001
0000320 001 000 001 001 001 001 000 001 000 001 000 001 001 001 000 001
0000340 001 001 001 000 001 000 001 000 001 001 001 000 001 001 001 001
0000360 000 001 000 001 000 001 001 001 000 001 001 001 001 000 001 000
0000400 001 000 001 001 001 000 001 001 001 001 000 001 000 001 000 001
0000420 001 001 000 001 001 001 001 000 001 000 001 000 001 001 001 000
0000440 001 001 001 001 000 001 000 001 000 001 001 001 000 001 001 001
0000460 001 000 001 000 001 000 001 001 001 000 001 001 001 001 000 001
0000500 000 001 000 001 001 001 000 001 001 001 001 000 001 000 001 000
0000520 001 001 001 000 001 001 001 001 000 001 000 001 000 001 001 001
0000540 000 001 001 001 001 000 001 000 001 000 001 001 001 000 001 001
```

Figure 8: Decoder output from file sink at SNR of 10dB.

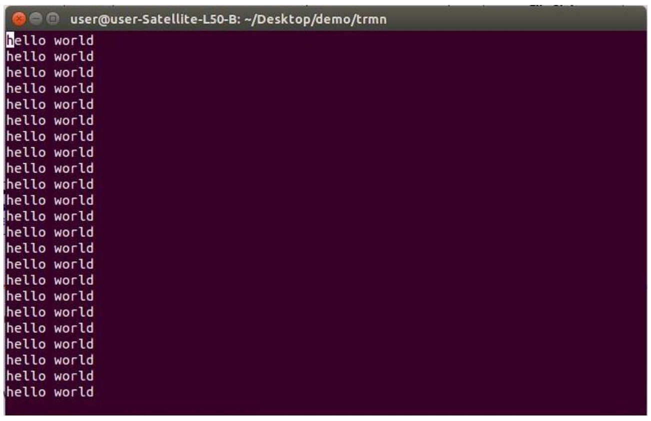


```
user@user-Satellite-L50-B: ~/Desktop/fle
0000000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000
*
0100000 001 001 001 001 001 001 000 001 000 000 001 001 001 000 000 001
0100020 001 000 001 001 001 001 001 001 001 001 001 001 001 000 001 001
0100040 000 001 000 000 000 001 001 001 001 000 000 001 001 001 001 000
0100060 001 001 001 001 001 000 001 001 000 000 000 001 000 001 001 001
0100100 001 001 000 001 001 001 000 000 001 001 000 001 001 001 000 000
0100120 001 001 001 001 000 000 001 001 000 001 001 001 000 001 001 001
0100140 001 000 000 001 001 000 000 000 000 001 001 001 001 001 000 001
0100160 000 001 000 000 000 001 000 000 001 001 001 000 001 001 001 000
0100200 001 001 000 000 001 000 001 001 000 000 000 001 000 001 000 001
0100220 000 001 001 001 001 000 001 000 001 000 001 001 001 000 001 001
0100240 001 001 000 001 000 001 000 001 001 001 000 001 001 001 001 000
0100260 001 000 001 000 001 001 001 000 000 001 000 001 001 001 000 000
0100300 000 001 001 001 000 001 001 000 000 000 001 001 001 000 001 001
0100320 001 000 001 001 000 001 000 000 001 001 001 000 001 000 000 001
0100340 000 001 000 000 000 000 001 000 001 001 001 000 001 001 001 001
0100360 000 001 001 001 001 000 001 001 000 001 001 001 001 000 001 000
0100400 001 000 001 001 001 000 001 001 001 001 000 001 000 001 000 001
0100420 001 001 000 001 001 001 000 000 000 000 000 000 001 000 000 001
0100440 001 001 000 001 000 000 000 000 000 001 001 001 001 001 001 000
0100460 001 000 001 000 001 000 001 001 001 000 001 001 001 001 000 001
0100500 000 000 000 000 001 000 000 000 001 000 001 000 000 000 001 000
--More--
```

Figure 9: Binary input and output at SNR of -1dB.

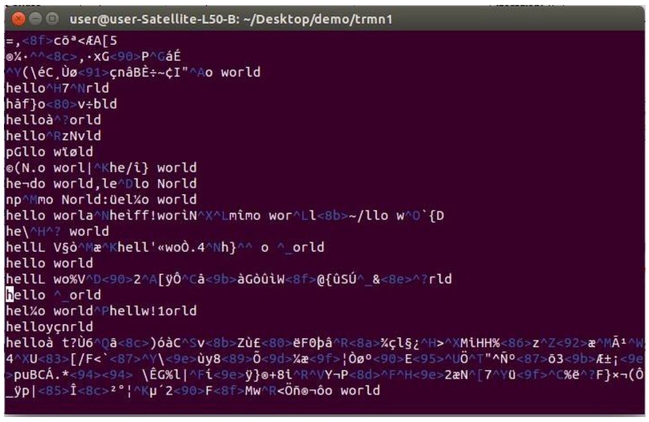
At high SNR conditions, the desired output may be reached at any of the component decoders. The CODEC block created with early iteration termination technique based on absolute value of the mean of extrinsic information in GNU Radio stops the iteration in any of the component decoder, after the desired output is reached. Similarly, at low SNR conditions, the block terminates early in order to reduce the unnecessary computations. The 'Turbdec1' block in GNU Radio is capable of terminating the iteration at low and high SNR conditions and saves half iteration without performance degradation. Figure 9 shows the binary input and corresponding decoder output at SNR of -1dB.

Figure 10 shows the transmission of a single word 'hello world' repeatedly. File source transmits the text repeatedly and 'Packed to Unpacked' block converts the text into binary form. After decoding, 'Unpacked to Packed' block converts the binary data back to text. The decoded text output is saved using file sink and verified to be correct. Figure 11 shows the decoded output at SNR of 10 dB. Figure 12 and 13 show the decoder output of the single text input of 'hello world' repeatedly at SNR of -1dB and 5dB. From these figures, it is clear that the decoder decodes correctly at high SNR conditions, and it is not able to make correct decoding decisions at low SNR conditions. If the SNR is low, output is never reached even after infinite iterations, and if the SNR is high the desired output may be reached after a few iterations either at output of SISO decoder 1 or SISO decoder-2. The implemented early iteration termination algorithm in GNU Radio is capable of stopping the iteration at low and high SNR and also save the half iteration.



```
user@user-Satellite-L50-B: ~/Desktop/demo/trmn
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
```

Figure 11: Text output from the Turbo code at SNR of 10dB.



```
user@user-Satellite-L50-B: ~/Desktop/demo/trmn1
=,<8f>cōª<ÆA[5
®¼·^^<8c>,·xG<90>P^GáÉ
^Y(\éC¸Ùø<91>çnâBÈ÷~¢I"^Ao world
hello^H7^Nrld
håf}o<80>v÷bld
helloà^?orld
hello^RzNvld
pGllo wïøld
©(N.o worl|^Khe/î} world
he¬do world,le^Dlo Norld
np^Mmo Norld:üel¼o world
hello worla^Nheìff!worìN^X^Lmîmo wor^Ll<8b>~/llo w^O`{D
he\^H^? world
hellL V§ò^Mæ^Khell'«woÒ.4^Nh}^^ o ^_orld
hello world
hellL wo%V^D<90>2^A[ÿÔ^Cå<9b>àGòûìW<8f>@{ûSÚ^_&<8e>^?rld
hello ^_orld
hel¼o world^Phellw!1orld
helloyçnrld
helloà t?Ù6^Qā<8c>)óàC^Sv<8b>Zù£<80>ëF0þâ^R<8a>¾çl§¿^H>^XMìHH%<86>z^Z<92>æ^MĀ¹^W
4^XU<83>[/F<`<87>^Y\<9e>ùy8<89>Õ<9d>¼æ<9f>¦Òø°<90>E<95>^UÖ^T"^Ñº<87>ō3<9b>Æ±¡<9e
>puBCÁ.*<94><94> \ÊG%l|^Fí<9e>ÿ}®+8ì^R^VY¬P<8d>^F^H<9e>2æN^[7^YÜ<9f>^C%ë^?F}×¬(Ô
_ÿp|<85>Î<8c>²°¦^Kµ´2<90>F<8f>Mw^R<Öñ®¬ôo world
```

Figure 12: Text output from the Turbo code at SNR of -1dB.

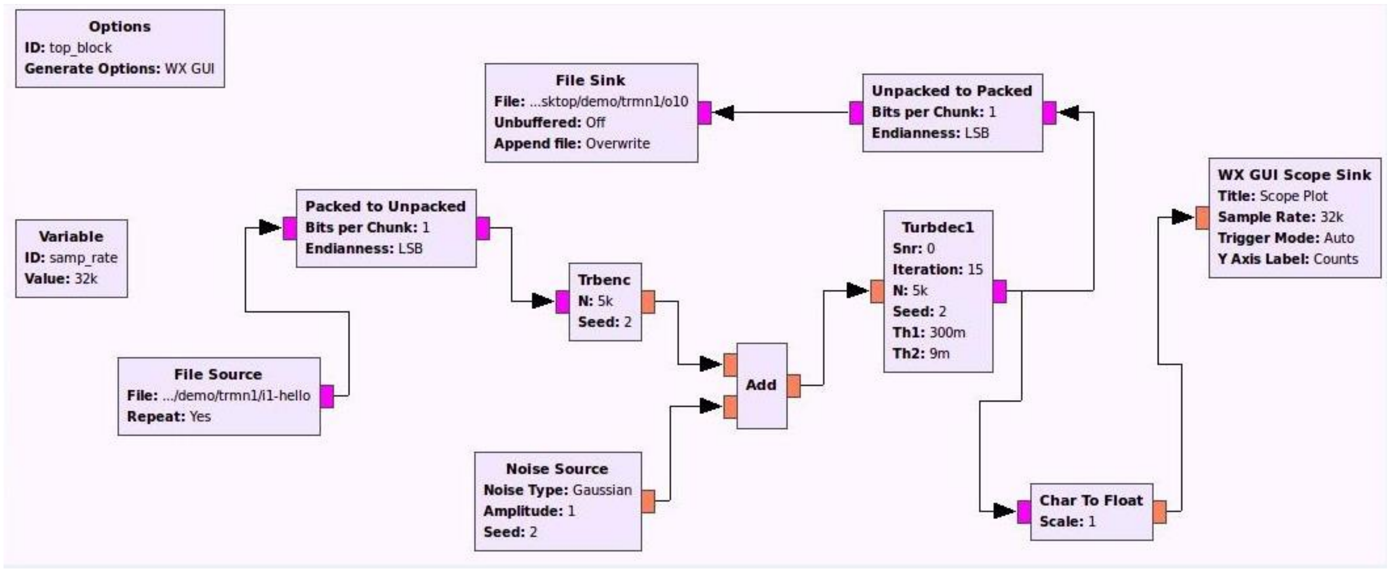
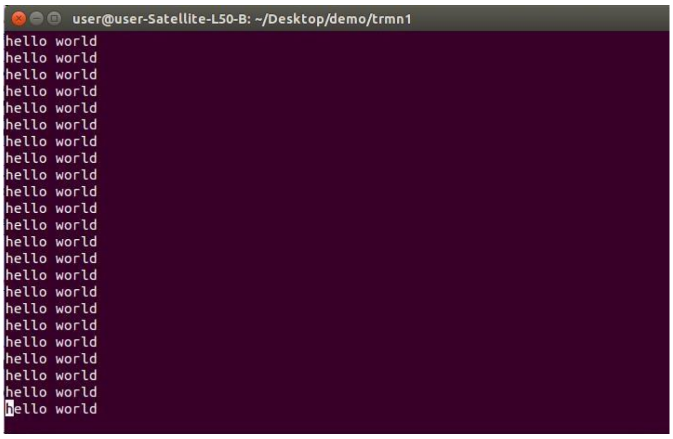


Figure 10: Transmission of text using Turbo code with early iteration termination technique based on absolute value of the mean of extrinsic information in GNU Radio.

Turbo CODEC with early iteration termination is implemented on LINUX 14.04 operating system and with GNU Radio version number 3.7.7.



```
user@user-Satellite-L50-B: ~/Desktop/demo/trmn1
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
```

Figure 13: Text output from the Turbo code at SNR of 5dB.

VI. Conclusion

A simple method to reduce the computational complexity and decoding delay in Turbo code is to terminate the iteration at the proper time. Early iteration termination technique based on absolute value of the mean of extrinsic information is an efficient termination method in terms of error correction, iteration number and processing time [27]. The technique is applicable at any channel conditions to terminate iteration at the proper time and also capable of reducing half iteration computations. The technique is a simple and useful technique for applications requiring reduced decoding delay, low computational complexity and error correction. There is no Turbo decoder block with early iteration termination in GNU Radio. In this paper, we have implemented Turbo CODEC with early iteration termination technique based on absolute value of the mean of extrinsic information in GNU Radio platform. Integration of USRP with newly created CODEC block in GNU Radio allows the real-time implementations.

References

[1] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, no. 1, pp. 379–423, 1948.

[2] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon Limit Error - Correcting Coding and Decoding : Turbo-Codes ," in *IEEE Int.Conf. Commun, . ICC '93 Geneva. Technical Program,* 1993, vol. 2, no. 1, pp. 1064–1070.

[3] C. Berrou and a Glavieux, "Near optimum error correcting coding and decoding: Turbo-codes," *IEEE Trans. Commun.*, vol. 44, no. 10, pp. 1261–1271, 1996.

[4] L. A. Perisoara and R. Stoian, "The Decision Reliability of MAP , Log-MAP , Max-Log-MAP and SOVA Algorithms for Turbo Codes," *Int. J. Commun.*, vol. 2, no. 1, pp. 65–74, 2008.

[5] A. Imran, "Software Implementation and Performance of UMTS Turbo Code," Tampere University of Technology, 2012.

[6] F. Gilbert, F. Kienle, and N. Wehn, "Low Complexity Stopping Criteria for UMTS Turbo-Decoders," *57th IEEE Semiannual* conf. in *Vehicular Technlo -VTC 2003-Spring.*, 2003, pp. 2376–2380.

[7] M. Moher, "Decoding via cross-entropy minimization," in *Proceedings of GLOBECOM '93. IEEE Global Telecommun.. Conf.*, 1993, pp. 809–813.

[8] Harold A. Haldren, "Studies in Software-Defined Radio System Implementationitle," Spring, 2014.

[9] C. Condo, M. R. Roch, M. Martina, and G. Masera, "Computation reduction for turbo decoding through window skipping," *Electron. Lett.*, vol. 52, no. 3, pp. 202–204, 2016.

[10] M. Martina, G. Masera, S. Papaharalabos, P. T. Mathiopoulos, and F. Gioulekas, "On practical implementation and generalizations of max * Operator for turbo and LDPC decoders," *IEEE Trans. Instrum. Meas.*, vol. 61, no. 4, pp. 888–895, 2012.

[11] P. Chhabra and V. Nath, "An efficient high performance turbo code implementation in Rayleigh fading channel," in *IEEE Symp. on Wireless Technol. and Applications, ISWTA*, 2014, pp. 47–52.

[12] P. Salija and B. Yamuna, "Optimum energy efficient error control techniques in wireless systems: a survey," *J. Commun. Technol. Electron.*, vol. 60, no. 11, pp. 1257–1263, 2015.

[13] Y. Ren, D. Yao, and X. Zhang, "The implementation of TETRA using GNU Radio and USRP," in *Proceedings - 2011 4th IEEE Int. Symp. Microwave. Antenna. Propagation. EMC. Technol.Wireless. Commu.*, 2011, pp. 363–366.

[14] A. M. Lalge, M. S. Karpe, and S. U. Bhandari, "Software Defined Radio Principles and Platforms," *Int. J. Adv. Comput. Res.*, vol. 3, no. 11, pp. 133–138, 2013.

[15] M. Talasila, "Implementation of Turbo codes on GNU Radiotle," 2010.

[16] "Guided Tutorials - GNU Radio - gnuradio.org." [Online]. Available: http://gnuradio.org/redmine/projects/gnuradio/wiki/Guided_Tutorials.

[17] A. Burr, "Turbo-codes: the ultimate error control codes?," *Electron. Commun. Eng. J.*, vol. 13, no. 4, pp. 155–165, 2001.

[18] I. Kaur and Y. K. Mathur, "Improving BER using turbo codes in OFDM systems," *Int. J. Sci. Eng. Res.*, vol. 3, no. 7, pp. 1–5, 2012.

[19] P. Ituero and M. López-Vallejo, "Further specialization of clustered VLIW processors: A MAP decoder for software defined radio," *ETRI J.*, vol. 30, no. 1, pp. 113–128, 2008.

[20] R. Mohamad, H. Harun, M. Mokhtar, W. A. W. Adnan, and K. Dimyati, "Performance analysis of stopping turbo decoder iteration criteria," in *Proc. IEEE. Int. Colloq. Signal Process. Its. Appl*, 2014, no. 1, pp. 5–9.

[21] V. P. Patil, "Implementation of efficient Turbo Code Encoder-Decoder with MAX- Log-MAP Algorithm," in proc. *Int. e-Conf. Emerging. Trends. Technol.* 2014, pp. 73–77.

[22] J. Kaza and C. Chakrabarti, "Design and implementation of low-energy turbo decoders," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 12, no. 9, pp. 968–977, 2004.

[23] S. Lin and D. J. Costello, *Error control coding : fundamentals and applications*. Pearson-Prentice Hall, 2004.

[24] Ranjan Bose, *Information Theory, Coding and Cryptography*, Second edi. New Delhi: Tata McGraw Hill, 2008.

[25] H. Wang, H. Yang, and D. Yang, "Improved Log-MAP decoding algorithm for turbo-like codes," *IEEE Commun. Lett.*, vol. 10, no. 3, pp. 186–188, 2006.

[26] P. Reddy, F. Clermidy, R. A. Khayat, A. Baghdadi, and M. Jezequel, "Power Consumption Analysis and Energy Efficient Optimization for Turbo Decoder Implementation," Proc. *Int.l Symp. Syst.on. Chip. (SoC)*, 2010, p. 12.

[27] P. Salija and B. Yamuna, "An Efficient Early Iteration Termination for Turbo Decoder," *J. Telecommun. Inf. Technol.*, no. 2, pp. 1–10, 2016.

[28] J. Geldmacher, K. Hueske, J. Götze, and M. Kosakowski, "Hard decision based low SNR early termination for LTE Turbo decoding," *Proc. Int. Symp. Wireless. Commun. Syst.*, 2011, pp. 26–30.