SYMBOL LEVEL DECODING FOR DUO-BINARY TURBO CODES

YOGESH BEEHARRY*, TULSI PAWAN FOWDUR AND KRISHNARAJ MADHAVJEE SUNJIV SOYJAUDAH

[1]*Electrical and Electronics Engineering Department,*
Faculty of Engineering, University of Mauritius, Réduit, Mauritius.

**Corresponding author: y.beeharry@uom.ac.mu*

(Received: 28th Mar. 2016; Accepted: 27th Sep. 2016; Published online: 30th May 2017)

ABSTRACT: This paper investigates the performance of three different symbol level decoding algorithms for Duo-Binary Turbo codes. Explicit details of the computations involved in the three decoding techniques, and a computational complexity analysis are given. Simulation results with different couple lengths, code-rates, and QPSK modulation reveal that the symbol level decoding with bit-level information outperforms the symbol level decoding by 0.1 dB on average in the error floor region. Moreover, a complexity analysis reveals that symbol level decoding with bit-level information reduces the decoding complexity by 19.6 % in terms of the total number of computations required for each half-iteration as compared to symbol level decoding.

ABSTRAK: Kertas ini mengkaji prestasi tiga algoritma dekoding aras simbol yang berbeza untuk kod Turbo Duo-Binari. Butiran jelas mengenai komputasi yang terlibat dalam tiga teknik dekoding dan analisis kompleksiti komputasi adalah dibentangkan. Hasil simulasi yang menggunakan panjang ganding yang berbeza, kadar-kod, dan modulasi QPSK mendapati bahawa dekoding aras simbol dengan maklumat aras-bit melebihi prestasi dekoding aras simbol sebanyak 0.1 dB purata rantau ralat lantai. Selain itu, analisis kompleksiti menunjukkan bahawa dekoding aras simbol dengan maklumat aras-bit mengurangkan kompleksiti dekoding sebanyak 19.6% dari segi jumlah komputasi yang diperlukan bagi setiap setengah-lelaran berbanding dekoding aras simbol.

KEYWORDS: *duo-binary turbo code; max-log MAP turbo decoding*

1. INTRODUCTION

Since its inception in 1993, Turbo code [1], which allowed the bound established by Shannon in 1948 [2] to be approached significantly, have been the center of attention of researchers. The Turbo code community has recently been conducting a lot of research on non-binary Turbo codes. With a comparable implementation complexity, duo-binary Turbo codes can provide better performance in terms of error correction than binary Turbo codes [3]. The excellent performance of duo-binary Circular Recursive Systematic Convolutional (CRSC) codes [4] has led to their adoption in Digital Video Broadcasting with Return Channel via Satellite (DVB-RCS) [5] replacing the conventional scheme that consisted of serial concatenation of a Reed Solomon (RS) code and a convolutional code. The DVB-RCS standard specifies an air-interface where many small terminals send return signals via satellite to a central gateway [6, 7].

There are several advantages of non-binary turbo codes, for example: better convergence of iterative decoding, low latency, and reduced sensitivity to puncturing

patterns [13]. Puncturing and the sub-optimal Max-Log-MAP decoding algorithm have a less significant influence with duo-binary Turbo codes than with binary Turbo codes [6]. Also, duo-binary Turbo codes allow for the latency of the decoder to be halved [7]. Recently, in [14], the performance of transmission systems with Duo-binary Turbo Codes (DBTC) and 16-QAM square modulation in Additive White Gaussian Noise (AWGN) channel have been investigated for various allocation modes. The author of [15], has investigated the input quantization of low complexity decoding algorithms and proposed an algorithm for an effective decoder quantization with the introduction of a scale factor into the decoding algorithm so as to achieve significant improvement in the hardware implementation of the decoder architecture. In [16], a duo-binary turbo code incorporating the Quadratic Permutation Polynomial (QPP) interleaver rather than the one defined in the DVB-RCS standard has been proposed. A complete detailing has been performed on the parameters and performance of the proposed scheme. In [12], a low-memory intensive decoding architecture has been proposed for a double binary convolutional Turbo code. The scheme is based on an improved decoding algorithm storing part of state metrics in the state metrics cache. In [13], the algorithm for double-binary Turbo decoding is studied using QPSK modulation over Rayleigh Fading channel. The authors of [17], have made a performance analysis between Turbo-φ codes and 3D-Turbo codes for the next generation DVB-RCS system in terms of error performance and decoder complexities. The decoding algorithms for Turbo codes are the Maximum A-Posteriori Probability (MAP), Logarithmic MAP (Log-MAP) and the Maximum Log-MAP (Max Log-MAP) algorithms. Due to the extensive computational complexity and numerical instability of the MAP algorithm, researchers have proposed the Log-MAP algorithm [18]. In order to further reduce computational complexity, the Max Log-MAP algorithm was brought forward. This reduction in computational complexity comes with a slight degradation in error performance as trade-off.

In [19] and [20], the authors have presented a symbol level decoding scheme for duo-binary and triple-binary Turbo codes. This comprehensive study has been carried out over an AWGN channel to demonstrate the good performance of the proposed schemes. In [6], a variant of the symbol level decoding algorithm for duo-binary Turbo codes has been presented. The performance of the decoding scheme is compared to the Turbo code standard for DVB-RCS over a Gaussian channel at Frame Error Rate (FER) of 10^{-4}. The results demonstrate the performance of the proposed scheme is almost similar to that used in the DVB-RCS standard. In [21], an investigation of bit-wise and symbol-wise decoding for the case of multi-binary convolutional Turbo codes employing the MAP algorithms has been performed. The symbol-wise decoding algorithm presented in this work operates on bit-level LLRs as input and is shown to outperform the bit-wise decoding. The advantage of this technique is that the limitation of using only QPSK modulation with the duo-binary Turbo codes [19, 20] can be overcome.

Different equations have been used in Turbo decoding algorithms. As such, this survey paper presents the existing Max Log-MAP Turbo decoding algorithm with the different equations used for duo-binary Turbo codes. Explicit details of the computations involved in the three decoding techniques as well as a complexity analysis have been provided. Simulation results with different couple lengths, code-rates and QPSK modulation reveal that symbol level decoding with bit-level information outperforms symbol level decoding by 0.1 dB on average in the error floor region. Moreover, a complexity analysis reveals that symbol level decoding with bit-level information reduces the complexity by 19.6 % as compared to symbol level decoding.

The paper is organized as follows. Section 2 presents the different methods for Max Log-MAP decoding algorithm for duo-binary Turbo codes. Section 3 presents the simulation results and Section 4 concludes the paper.

2. DUO-BINARY TURBO CODES

The encoding structure for duo-binary Turbo codes employed in the DVB-RCS standard is shown in Fig. 1.

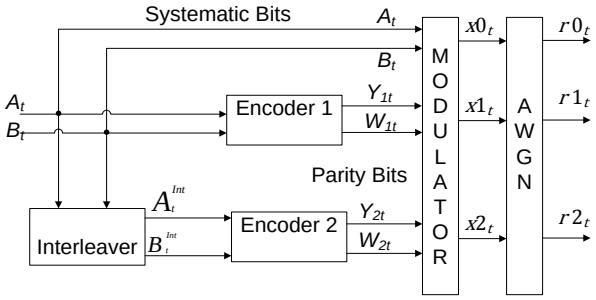

Fig. 1: Outputs at time *t* of the duo-binary encoder with QPSK modulation [14].

The DVB-RCS standard uses a parallel concatenation of two Circular Recursive Systematic Convolutional (CRSC) codes as the encoder [13-15] separated by an interleaver which uses a two-level interleaving. Let N_c be the size of each couple at the input of the duo-binary Turbo encoder. At the first level, an intra-symbol permutation takes place and at the second level an inter-symbol permutation takes place. The two levels of interleaving are well described in [5].

The modulation scheme used in duo-binary Turbo codes for the DVB-RCS standard is gray-coded Quadrature Phase Shift Keying (QPSK) modulation as depicted in Fig. 1. The constellation mapping for the gray-coded QPSK modulation used is shown in Fig. 2. The couples $\{A_t, B_t\}$, $\{Y_{1t}, W_{1t}\}$ and $\{Y_{2t}, W_{2t}\}$ are mapped onto the modulated symbols $x0_t$, $x1_t$and $x2_t$ respectively.

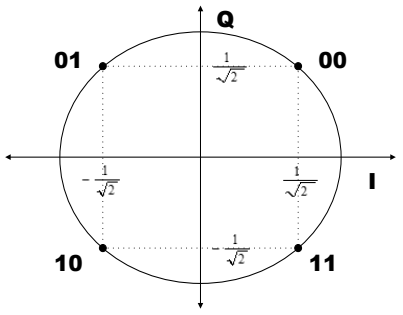

Fig. 2: Bit-mapping of Gray-coded QPSK modulation [20, 22].

After modulation of the symbols, the stream is multiplexed and transmitted over a complex AWGN channel. The received noisy symbol vectors are intercepted at the receiver side and fed to the Turbo decoder. $r0$, $r1$ and $r2$ are the received noisy vectors of the systematic and parity information. $\overline{r0}$ is the interleaved version of the received noisy vector of the systematic information.

The conventional decoding process for duo-binary Turbo codes is performed with the exchange of symbol-level extrinsic information in an iterative manner between the two turbo decoders after each half-iteration as depicted in Fig. 3. In this work, the Max-Log MAP algorithm has been used for decoding. Let the branch transition probability associated with input symbol $C_t = i$, (where i can take values 0, 1, 2 and 3 for duo binary) from state $S_{t-1} = l'$ to $S_t = l$ and at time t be denoted by $\gamma_t^{1,i}(l', l)$ for decoder 1.

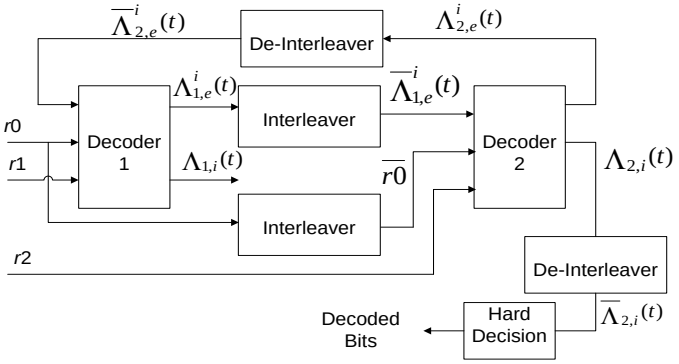

Fig. 3: Non-binary Turbo decoder.

In Fig. 3, $\Lambda_{1,i}(t)$ is the a-posteriori LLR for decoder 1, $\Lambda_{2,i}(t)$ is the a-posteriori LLR for decoder 2, $\Lambda^{i}_{1,e}(t)$is the extrinsic LLR for decoder 1, $\bar{\Lambda}^{i}_{1,e}(t)$ is the interleaved extrinsic LLR from decoder 1, $\Lambda^{i}_{2,e}(t)$is the extrinsic LLR from decoder 2 and $\bar{\Lambda}^{i}_{2,e}(t)$is the de-interleaved extrinsic LLR from decoder 2.

The Trellis diagrams for each symbol input of the duo-binary Turbo codes are shown in Fig. 4.

$x0_{t1}$ is the noiseless modulated symbol of the systematic information at time instant $t1$.

$x1_{t1}$ is the noiseless modulated symbol of the parity information from the first encoder at time instant $t1$.

$r0_{t1}$ is the noisy received symbol of the systematic information from the transmission of $x0_{t1}$ through the AWGN channel at time instant $t1$.

$r1_{t1}$ is the noisy received symbol of the parity information from the first encoder after the transmission of $x1_{t1}$ through the AWGN channel at time instant $t1$.

The components which are used in the decoding equations are the in-phase and quadrature phase of these complex symbols which are described as follows:

$x0_t^{I(i)}(l)$ and $x0_t^{\mathrm{Q}(i)}(l)$ are the modulated in-phase and quadrature components of the complex systematic symbol $x0$ at time t which is associated with the transition $S_{t-1} = l'$ to $S_t = l$ and input symbol i

$x1_t^{I(i)}(l)$ and $x1_t^{\mathrm{Q}(i)}(l)$ are the modulated in-phase and quadrature components of the complex parity symbol $x1$ at time t which is associated with the transition $S_{t-1} = l'$ to $S_t = l$ and input symbol i

$x2_t^{I(i)}(l)$ and $x2_t^{\mathrm{Q}(i)}(l)$ are the modulated in-phase and quadrature components of the complex parity symbol $x2$ at time t which is associated with the transition $S_{t-1} = l'$ to $S_t = l$ and input symbol i

$r0_t^I$ and $r0_t^{\mathrm{Q}}$ are the in-phase and quadrature components of the complex received symbol $r0$ at time t

$r1_t^I$ and $r1_t^{\mathrm{Q}}$ are the in-phase and quadrature components of the complex received symbol $r1$ at time t

$r2_t^I$ and $r2_t^{\mathrm{Q}}$ are the in-phase and quadrature components of the complex received symbol $r2$ at time t.

Note that the trellis diagram for the second decoder is similar except that the decoder uses $\overline{r0}$ and $r2$ as channel inputs. The following subsections describe three decoding algorithms that can be used with duo-binary Turbo codes. The first two are variants of symbol level decoding algorithms while the third one is a symbol-level decoding scheme with bit-level LLRs as inputs.

The parameters shown in Fig. 4 can be defined as follows:

α_t^1 is the forward recursive variable for decoder 1 and β_t^1 is the backward recursive variable of decoder 1.

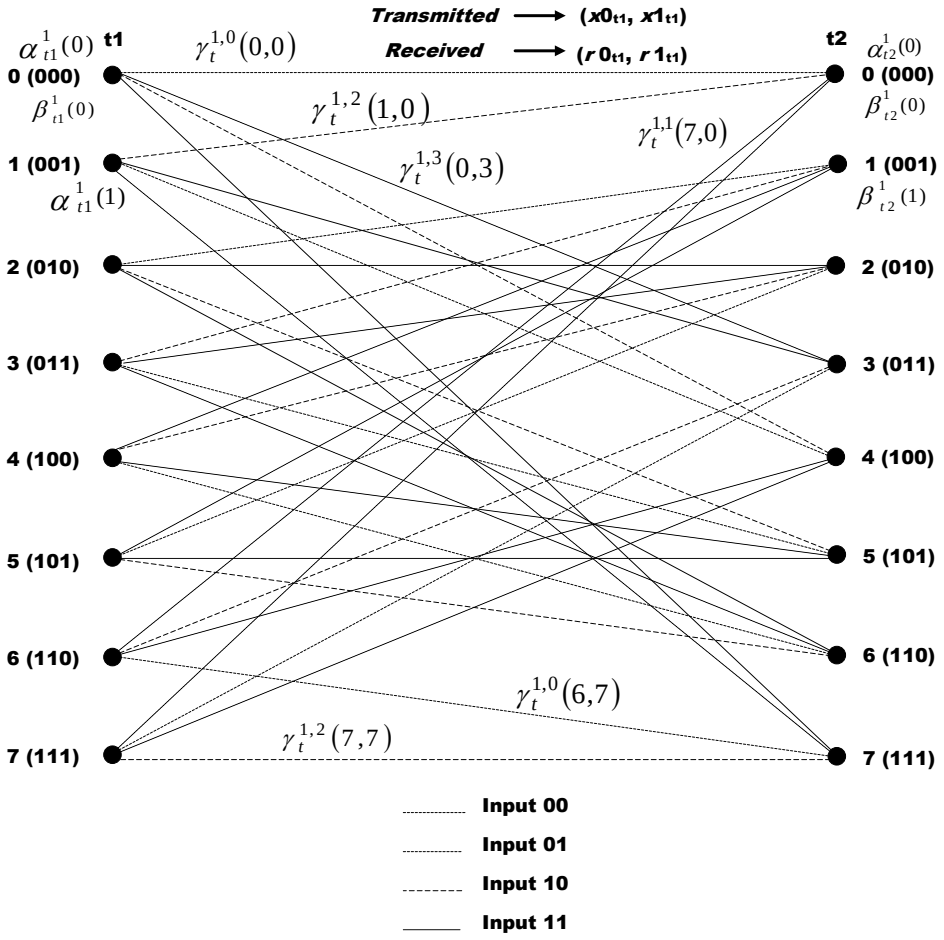

Fig. 4: Trellis diagram.

2.1 Decoding for Duo-Binary Turbo Codes using Method 1

The decoding Method 1 from [20] is a symbol level decoding scheme used in the DVB-RCS standard. The decoding equations are presented next. The first decoder's branch metric is given as:

$$\gamma_t^{1,i}(l', l) = log\big(P(u_t^2 = i)\big) - \left(\left[r0_t^I - x0_t^{I(i)}(l)\right]^2 + \left[r0_t^Q - x0_t^{Q(i)}(l)\right]^2 + \left[r1_t^I - x1_t^{I(i)}(l)\right]^2 + \left[r1_t^Q - x1_t^{Q(i)}(l)\right]^2\right) \quad (1)$$

Where, $log\big(P(u_t^2 = i)\big)$ is the a-priori logarithmic symbol probability of symbol i obtained from the second decoder and the value is set to zero at the beginning of the decoding process.

The number of computations required for the branch metric given in Eqn. (1) is shown in Table 1. The information provided in Table 1 will be used in the analysis part of section 3.

Table 1: Number of computations for a branch transition metric of Decoder 1 with Method 1.

	Logarithm Operations	Additions	Subtractions	Multiplications	Total
Branch Transition Metric	1	3	5	4	13

The forward recursive variable for the first decoder is computed as follows [7]:

$$\alpha_t^1(l) = \max\left(\alpha_{t-1}^1 + \gamma_t^{1,i}(l', l)\right), \text{for } 0 \le l' \le \mathrm{M}_\mathrm{S}^1 - 1 \quad (2)$$

Where, M_S^1 is the total number of states for decoder 1. The number of computations required for the forward recursive variable shown in equation (2) is shown in Table 2. The information provided in Table 2 will be used in the analysis part of section 3.

Table 2: Number of computations for a forward recursive Variable in Decoder 1 with Method 1.

	Maximum Operations	Additions	Total
Forward Recursive Variable	1	4	5

The backward recursive variable for the first decoder is computed as follows [7]:

$$\beta_t^1(l) = \max\left(\beta_{t+1}^1 + \gamma_{t+1}^{1,i}(l', l)\right), \text{for } 0 \le l' \le \mathrm{M}_\mathrm{S}^1 - 1 \quad (3)$$

The number of computations required for the backward recursive variable shown in equation (3) is shown in Table 3. The information provided in Table 3 will be used in the analysis part of section 3.

Table 3: Number of computations for a backward recursive Variable in Decoder 1 with Method 1.

	Maximum Operations	Additions	Total
Backward Recursive Variable	1	4	5

The equation for the log likelihood ratio is as follows [7]:

$$\Lambda_{1,i}(t) = \max\left(\alpha_{t-1}^{1} + \gamma_t^{1,i}(l', l) + \beta_t^1(l)\right) - \max\left(\alpha_{t-1}^{1} + \gamma_t^{1,0}(l', l) + \beta_t^1(l)\right), \quad \text{for } 0 \le l' \le \mathrm{M}_{\mathrm{S}}^{1} - 1 \tag{4}$$

Where, $\Lambda_{1,i}(t)$is the Log-Likelihood Ratio (LLR) of symbol *i* where, *i* $\in$ {1, 2, and 3} for duo binary turbo codes. The number of computations required for the Log-Likelihood Ratio of symbol i at time instant t shown in equation (4) is shown in Table 4. The information provided in Table 4 will be used in the analysis part of section 3.

Table 4: Number of computations for a LLR of symbol $\boldsymbol{i}$ at time instant $\boldsymbol{t}$ for Decoder 1 with Method 1.

	Maximum Operations	Additions	Subtractions	Total
Log-Likelihood Ratio	2	32	1	35

The a-posteriori LLR comprises of 3 LLRs, namely, the a-priori LLR, the intrinsic LLR and the extrinsic LLR related as follows [23]:

A-posteriori LLR = A-priori LLR + Intrinsic LLR + Extrinsic LLR (5)

$$\Lambda_{1,i}(t) = \bar{\Lambda}_{2,e}^{i}(t) + \Lambda_{1,in}^{i}(t) + \Lambda_{1,e}^{i}(t) \tag{6}$$

Where, $\Lambda_{1,in}^{i}(t)$ is the intrinsic LLR for decoder 1. The extrinsic LLR for the decoder 1 is thus calculated as follows:

$$\Lambda_{1,e}^{i}(t) = \Lambda_{1,i}(t) - \Lambda_{1,in}^{i}(t) - \bar{\Lambda}_{2,e}^{i}(t) \tag{7}$$

The number of computations required for the extrinsic LLR of symbol i at time instant t shown in equation (7) is shown in Table 5. The information provided in Table 5 will be used in the analysis part of section 3.

Table 5: Number of computations for an extrinsic LLR of symbol $\boldsymbol{i}$ at time $\boldsymbol{t}$ for Decoder 1 with Method 1.

	Subtractions	Total
Extrinsic Log-Likelihood Ratio	2	2

The intrinsic LLR of decoder 1 associated with the systematic bits:$(\mathrm{A_t}, \mathrm{B_t})$ can be represented as:

$$\Lambda_{1,in}^{i}(t) = log\left(\frac{P(r0_t \mid u_t^1 = i)}{P(r0_t \mid u_t^1 = 00)}\right) \tag{8}$$

Considering for example the intrinsic LLR $\Lambda_{1,in}^{i=01}(t)$ which can be expressed as:

$$\Lambda_{1,in}^{i=01} = \left(\frac{P(r0_t \mid u_t^1 = 01)}{P(r0_t \mid u_t^1 = 00)}\right) = log\left(\frac{P\left(r0_t \mid x_t = +\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}\right)}{P\left(r0_t \mid x_t = +\frac{1}{\sqrt{2}}, +\frac{1}{\sqrt{2}}\right)}\right)$$

$$= \frac{1}{2\sigma^2}\left[-\frac{4}{\sqrt{2}}\left(r0_t^Q\right)\right] = \left[-\frac{2}{\sqrt{2}\sigma^2}\left(r0_t^Q\right)\right] \tag{9}$$

Likewise, similar expressions can be obtained for the intrinsic LLRs for the other symbols.

$$\Lambda_{1,in}^{i=10} = \left[-\frac{2}{\sqrt{2}\sigma^2}(r0_t^I)\right]; \Lambda_{1,in}^{i=11} = \left[-\frac{2}{\sqrt{2}\sigma^2}\left(r0_t^I + r0_t^Q\right)\right]; \Lambda_{1,in}^{i=00} = 0 \quad (10)$$

The number of computations required for the intrinsic LLR of symbol i at time instant t shown in equations (9) and (10) is shown in Table 6. The information provided in Table 6 will be used in the analysis part of section 3.

Table 6: Number of computations for intrinsic LLRs of symbol $\boldsymbol{i}$ at time instant $\boldsymbol{t}$.

	Additions	Multiplications	Divisions	Total
$\Lambda_{1,in}^{i=01}(t)$		2	1	3
$\Lambda_{1,in}^{i=10}(t)$		2	1	3
$\Lambda_{1,in}^{i=11}(t)$	1	2	1	4

The probability computation to be fed to next decoder is as follows:

$$P(u_t^1 = 00) + P(u_t^1 = 01) + P(u_t^1 = 10) + P(u_t^1 = 11) = 1 \quad (11)$$

$$P(u_t^1 = i) = P(u_t^1 = 00) \cdot e^{\Lambda_{1,e}^i(t)} \quad (12)$$

Therefore,

$$P(u_t^1 = 00) = \frac{1}{1 + e^{\Lambda_{1,e}^{01}(t)} + e^{\Lambda_{1,e}^{10}(t)} + e^{\Lambda_{1,e}^{11}(t)}}; P(u_t^1 = 01) = \frac{e^{\Lambda_{1,e}^{01}(t)}}{1 + e^{\Lambda_{1,e}^{01}(t)} + e^{\Lambda_{1,e}^{10}(t)} + e^{\Lambda_{1,e}^{11}(t)}} \quad (13)$$

$$P(u_t^1 = 10) = \frac{e^{\Lambda_{1,e}^{10}(t)}}{1 + e^{\Lambda_{1,e}^{01}(t)} + e^{\Lambda_{1,e}^{10}(t)} + e^{\Lambda_{1,e}^{11}(t)}}; \qquad P(u_t^1 = 11) = \frac{e^{\Lambda_{1,e}^{11}(t)}}{1 + e^{\Lambda_{1,e}^{01}(t)} + e^{\Lambda_{1,e}^{10}(t)} + e^{\Lambda_{1,e}^{11}(t)}} \quad (14)$$

The max approximation is defined as:

$$log \sum_j e^{a_j} = \max_j(a_j) \quad (15)$$

Applying the max approximation to the computation of the log probabilities:

$$log\left(P(u_t^1 = 00)\right) = -max\left(0, \Lambda_{1,e}^{01}(t), \Lambda_{1,e}^{10}(t), \Lambda_{1,e}^{11}(t)\right) \quad (16)$$

$$log\left(P(u_t^1 = 01)\right) = \Lambda_{1,e}^{01}(t) - max\left(0, \Lambda_{1,e}^{01}(t), \Lambda_{1,e}^{10}(t), \Lambda_{1,e}^{11}(t)\right) \quad (17)$$

$$log\left(P(u_t^1 = 10)\right) = \Lambda_{1,e}^{10}(t) - max\left(0, \Lambda_{1,e}^{01}(t), \Lambda_{1,e}^{10}(t), \Lambda_{1,e}^{11}(t)\right) \quad (18)$$

$$log\left(P(u_t^1 = 11)\right) = \Lambda_{1,e}^{11}(t) - max\left(0, \Lambda_{1,e}^{01}(t), \Lambda_{1,e}^{10}(t), \Lambda_{1,e}^{11}(t)\right) \quad (19)$$

The number of computations required for the a-posteriori probabilities of symbol i at time instant t shown in equations (16 - 19) is shown in Table 7. The information provided in Table 7 will be used in the analysis part of section 3.

Table 7: Number of computations for a-posteriori probabilities for Decoder 1 with Method 1.

	Maximum Operations	Subtractions	Total
A-posteriori Probabilities	4	4	8

The equations for decoder 2 are now presented. The branch metric of the second decoder is given as follows:

$$\gamma_t^{2,i} = log\big(\mathrm{P}(u_t^1 = i)\big) - \left(\left[\overline{r0}_t^I - x0_t^{I(i)}(l)\right]^2 + \left[\overline{r0}_t^{\mathrm{Q}} - x0_t^{\mathrm{Q}(i)}(l)\right]^2 + \left[r2_t^I - x2_t^{I(i)}(l)\right]^2 + \left[r2_t^{\mathrm{Q}} - x2_t^{\mathrm{Q}(i)}(l)\right]^2\right) \quad (20)$$

The number of computations involved in calculating the branch metric for each transition in the second decoder is similar to that shown in Table 1.

The computation of the forward and backward recursive variables is done as follows:

$$\alpha_t^2(l) = \max\left(\alpha_{t-1}^2 + \gamma_t^{2,i}(l', l)\right), \text{for } 0 \le l' \le \mathrm{M}_\mathrm{S}^2 - 1 \quad (21)$$

The number of computations involved in calculating each forward transition metric at each time instant in the second decoder is similar to that shown in Table 2.

The backward recursive variables for the first decoder are computed as follows:

$$\beta_t^2(l) = \max\left(\beta_{t+1}^2 + \gamma_{t+1}^{2,i}(l', l)\right), \text{for } 0 \le l' \le \mathrm{M}_\mathrm{S}^2 - 1 \quad (22)$$

The number of computations involved in calculating each backward transition metric at each time instant in the second decoder is similar to that shown in Table 3. The equation for the log likelihood ratio is as follows:

$$\Lambda_{2,i}(t) = \max\left(\bar{\alpha}_{t-1}^2 + \overline{\gamma_t^{2,i}}(l', l) + \bar{\beta}_t^2(l)\right) - \max\left(\bar{\alpha}_{t-1}^2 + \overline{\gamma_t^{2,0}}(l', l) + \bar{\beta}_t^2(l)\right), \text{for } 0 \le l' \le \mathrm{M}_\mathrm{S}^2 - 1 \quad (23)$$

Where, M_S^2 denotes the number of states for the second decoder.

The equations for the extrinsic information output from both decoders for double binary and triple binary are explained in [23]. The number of computations involved in calculating each LLR at each time instant in the second decoder is similar to that shown in Table 4. The extrinsic LLR for the decoder 2 is thus calculated as follows:

$$\Lambda_{2,e}^i(t) = \Lambda_{2,i}(t) - \Lambda_{2,in}^i(t) - \bar{\Lambda}_{1,e}^i(t) \quad (24)$$

The number of computations involved in calculating each extrinsic LLR at each time instant in the second decoder is similar to that shown in Table 5. The number of computations involved in calculating each intrinsic LLR at each time instant in the second decoder is similar to that shown in Table 6.

The probability computation to be fed to next decoder is [21]:

$$P(u_t^2 = 00) + P(u_t^2 = 01) + P(u_t^2 = 10) + P(u_t^2 = 11) = 1 \quad (25)$$

$$P(u_t^2 = sym) = P(u_t^2 = 00)\,.\,e^{\Lambda_{2,e}^i(t)} \quad (26)$$

Therefore,

$$P(u_t^2 = 00) = \frac{1}{1 + e^{\Lambda_{2,e}^{01}(t)} + e^{\Lambda_{2,e}^{10}(t)} + e^{\Lambda_{2,e}^{11}(t)}}\,;\; P(u_t^2 = 01) = \frac{e^{\Lambda_{2,e}^{01}(t)}}{1 + e^{\Lambda_{2,e}^{01}(t)} + e^{\Lambda_{2,e}^{10}(t)} + e^{\Lambda_{2,e}^{11}(t)}} \quad (27)$$

$$P(u_t^2 = 10) = \frac{e^{\Lambda_{2,e}^{10}(t)}}{1 + e^{\Lambda_{2,e}^{01}(t)} + e^{\Lambda_{2,e}^{10}(t)} + e^{\Lambda_{2,e}^{11}(t)}};\; P(u_t^2 = 11) = \frac{e^{\Lambda_{2,e}^{11}(t)}}{1 + e^{\Lambda_{2,e}^{01}(t)} + e^{\Lambda_{2,e}^{10}(t)} + e^{\Lambda_{2,e}^{11}(t)}} \quad (28)$$

Applying the max approximation to the computation of the log probabilities:

$$log\big(P(u_t^2 = 00)\big) = -max\left(0, \Lambda_{2,e}^{01}(t), \Lambda_{2,e}^{10}(t), \Lambda_{2,e}^{11}(t)\right) \quad (29)$$

$$log\big(P(u_t^2 = 01)\big) = \Lambda_{2,e}^{01}(t) - max\left(0, \Lambda_{2,e}^{01}(t), \Lambda_{2,e}^{10}(t), \Lambda_{2,e}^{11}(t)\right) \quad (30)$$

$$log\big(P(u_t^2 = 10)\big) = \Lambda_{2,e}^{10}(t) - max\left(0, \Lambda_{2,e}^{01}(t), \Lambda_{2,e}^{10}(t), \Lambda_{2,e}^{11}(t)\right) \quad (31)$$

$$log\big(P(u_t^2 = 11)\big) = \Lambda_{2,e}^{11}(t) - max\left(0, \Lambda_{2,e}^{01}(t), \Lambda_{2,e}^{10}(t), \Lambda_{2,e}^{11}(t)\right) \quad (32)$$

The number of computations involved in calculating each a-posteriori probability at each time instant in the second decoder is similar to that shown in Table 7.

2.2 Decoding for Duo-Binary Turbo Codes: Method 2 [6]

A variant of the decoding algorithm for duo-binary Turbo codes is described in [6]. The main difference as compared to Method 1 lies in the computations of the branch transition metrics and the intrinsic LLRs. The equations for the decoding algorithm are presented next. The branch transition metric is computed as:

$$\gamma_t^{1,i}(l', l) = \bar{\Lambda}_{2,e}^{i}(t) + \frac{L_c}{2}\Big(\Big[(r0_t^{\mathrm{I}}) \,.\, \Big(x0_t^{\mathrm{I}(i)}(l)\Big)\Big] + \Big[(r0_t^{\mathrm{Q}}) \,.\, \Big(x0_t^{\mathrm{Q}(i)}(l)\Big)\Big] + \Big[(r1_t^{\mathrm{I}}) \,.\, \Big(x1_t^{\mathrm{I}(i)}(l)\Big)\Big] + \Big[(r1_t^{\mathrm{Q}}) \,.\, \Big(x1_t^{\mathrm{Q}(i)}(l)\Big)\Big]\Big) \quad (33)$$

Where, L_c is the channel reliability estimate. The number of computations required for the branch metric shown in Eqn. (33) is shown in Table 8. The information provided in Table 8 will be used in the analysis part of section 3.

Table 8: Number of computations for a branch transition metric in Decoder 1 with Method 2.

	Additions	Multiplications	Divisions	Total
Branch Transition Metric	4	5	1	10

The forward recursive variables for the first decoder are computed in the same way as for Method 1, which is shown in Eqn. (2). The number of computations involved in each forward recursive variable at each time instant is as shown in Table 2. The backward recursive variables for the first decoder are computed as shown in Eqn. (3). The number of computations involved in each backward recursive variable at each time instant is as shown in Table 3. The equation for the log likelihood ratio is the same as shown in Eqn. (4). The number of computations involved in each log likelihood ratio at each time instant is as shown in Table 4.

The a-posteriori LLR comprises of 3 LLRs, namely, the a-priori LLR, the intrinsic LLR and the extrinsic LLR are related as shown in Eqns. (4), (5) and (6) [23]. The intrinsic LLRs are computed as follows:

$$\Lambda_{1,in}^{i=00} = \frac{L_c}{2}\Big(\Big[(r0_t^{\mathrm{I}}).\Big(x0_t^{\mathrm{I}(i=00)}(l)\Big)\Big] + \Big[(r0_t^{\mathrm{Q}}).\Big(x0_t^{\mathrm{Q}(i=00)}(l)\Big)\Big]\Big) - \frac{L_c}{2}\Big(\Big[(r0_t^{\mathrm{I}}).\Big(x0_t^{\mathrm{I}(i=00)}(l)\Big)\Big] + \Big[(r0_t^{\mathrm{Q}}).\Big(x0_t^{\mathrm{Q}(i=00)}(l)\Big)\Big]\Big) \quad (34)$$

$$\Lambda_{1,in}^{i=01} = \frac{L_c}{2}\left(\left[\left(r0_t^{\mathrm{I}}\right) . \left(x0_t^{\mathrm{I}(i=01)}(l)\right)\right] + \left[\left(r0_t^{\mathrm{Q}}\right) . \left(x0_t^{\mathrm{Q}(i=01)}(l)\right)\right]\right) - \frac{L_c}{2}\left(\left[\left(r0_t^{\mathrm{I}}\right) . \left(x0_t^{\mathrm{I}(i=00)}(l)\right)\right] + \left[\left(r0_t^{\mathrm{Q}}\right) . \left(x0_t^{\mathrm{Q}(i=00)}(l)\right)\right]\right) \quad (35)$$

$$\Lambda_{1,in}^{i=10} = \frac{L_c}{2}\left(\left[\left(r0_t^{\mathrm{I}}\right) . \left(x0_t^{\mathrm{I}(i=10)}(l)\right)\right] + \left[\left(r0_t^{\mathrm{Q}}\right) . \left(x0_t^{\mathrm{Q}(i=10)}(l)\right)\right]\right) - \frac{L_c}{2}\left(\left[\left(r0_t^{\mathrm{I}}\right) . \left(x0_t^{\mathrm{I}(i=00)}(l)\right)\right] + \left[\left(r0_t^{\mathrm{Q}}\right) . \left(x0_t^{\mathrm{Q}(i=00)}(l)\right)\right]\right) \quad (36)$$

$$\Lambda_{1,in}^{i=11} = \frac{L_c}{2}\left(\left[\left(r0_t^{\mathrm{I}}\right) . \left(x0_t^{\mathrm{I}(i=11)}(l)\right)\right] + \left[\left(r0_t^{\mathrm{Q}}\right) . \left(x0_t^{\mathrm{Q}(i=11)}(l)\right)\right]\right) - \frac{L_c}{2}\left(\left[\left(r0_t^{\mathrm{I}}\right) . \left(x0_t^{\mathrm{I}(i=00)}(l)\right)\right] + \left[\left(r0_t^{\mathrm{Q}}\right) . \left(x0_t^{\mathrm{Q}(i=00)}(l)\right)\right]\right) \quad (37)$$

The number of computations required for each intrinsic LLR of symbol i at time instant t shown in Eqn. (34) to (37) is shown in Table 9. The information provided in Table 9 will be used in the analysis part of section 3.

Table 9: Number of computations for intrinsic LLRs of symbol $\boldsymbol{i}$ at time $\boldsymbol{t}$ for Decoder 1 with Method 2.

	Additions	Multiplications	Divisions	Subtractions	Total
Intrinsic LLRs	2	4	1	1	8

The branch transition metric for the second decoder is computed as:

$$\gamma_t^{2,i}(l', l) = \bar{\Lambda}_{1,e}^{i}(t) + \frac{L_c}{2}\left(\left[\left(\overline{r0_t^{\mathrm{T}}}\right) . \left(x0_t^{\mathrm{T}(i)}(l)\right)\right] + \left[\left(\overline{r0_t^{\mathrm{Q}}}\right) . \left(x0_t^{\mathrm{Q}(i)}(l)\right)\right] + \left[\left(r2_t^{\mathrm{T}}\right) . \left(x2_t^{\mathrm{T}(i)}(l)\right)\right] + \left[\left(r2_t^{\mathrm{Q}}\right) . \left(x2_t^{\mathrm{Q}(i)}(l)\right)\right]\right) \quad (38)$$

The number of computations for the branch transition metrics of the second decoder is similar to that for decoder 1, as shown in Table 8. The forward recursive variables for the second decoder are computed in the same way as for Method 1 which is shown in Eqn. (2). The number of computations for the forward recursive variable is the same as shown in Table 2. The backward recursive variables for the second decoder are computed as shown in Eqn. (3). The number of computations for the backward recursive variable is the same as shown in Table 3. The equation for the log likelihood ratio is the same as shown in Eqn. (4). The number of computations for the a-posteriori LLR is the same as shown in Table 4.

The a-posteriori LLR comprises of 3 LLRs, namely, the a-priori LLR, the intrinsic LLR and the extrinsic LLR are related as shown in Eqn. (32) [23].

The number of computations for the extrinsic LLR is the same as shown in Table 5. The intrinsic LLR of decoder 2 associated with the interleaved systematic bits (A_t^{Int}, B_t^{Int}) can be represented as:

$$\Lambda_{2,in}^{i=00} = \frac{L_c}{2}\left(\left[\left(\overline{r0_t^{\mathrm{I}}}\right).\left(x0_t^{\mathrm{I}(i=00)}(l)\right)\right] + \left[\left(\overline{r0_t^{\mathrm{Q}}}\right).\left(x0_t^{\mathrm{Q}(i=00)}(l)\right)\right]\right) - \frac{L_c}{2}\left(\left[\left(\overline{r0_t^{\mathrm{I}}}\right).\left(x0_t^{\mathrm{I}(i=00)}(l)\right)\right] + \left[\left(\overline{r0_t^{\mathrm{Q}}}\right).\left(x0_t^{\mathrm{Q}(i=00)}(l)\right)\right]\right) \quad (39)$$

$$\Lambda_{2,in}^{i=01} = \frac{L_c}{2}\left(\left[\left(\overline{r0_t^{\mathrm{I}}}\right) . \left(x0_t^{\mathrm{I}(i=01)}(l)\right)\right] + \left[\left(\overline{r0_t^{\mathrm{Q}}}\right) . \left(x0_t^{\mathrm{Q}(i=01)}(l)\right)\right]\right) - \frac{L_c}{2}\left(\left[\left(\overline{r0_t^{\mathrm{I}}}\right).\left(x0_t^{\mathrm{I}(i=00)}(l)\right)\right] + \left[\left(\overline{r0_t^{\mathrm{Q}}}\right).\left(x0_t^{\mathrm{Q}(i=00)}(l)\right)\right]\right) \quad (40)$$

$$\Lambda_{2,in}^{i=10} = \frac{L_c}{2}\left(\left[\left(\overline{r0_t^{\mathrm{I}}}\right).\left(x0_t^{\mathrm{I}(i=10)}(l)\right)\right] + \left[\left(\overline{r0_t^{\mathrm{Q}}}\right).\left(x0_t^{\mathrm{Q}(i=10)}(l)\right)\right]\right) - \frac{L_c}{2}\left(\left[\left(\overline{r0_t^{\mathrm{I}}}\right).\left(x0_t^{\mathrm{I}(i=00)}(l)\right)\right] + \left[\left(\overline{r0_t^{\mathrm{Q}}}\right).\left(x0_t^{\mathrm{Q}(i=00)}(l)\right)\right]\right) \quad (41)$$

$$\Lambda_{2,in}^{i=11} = \frac{L_c}{2}\left(\left[\left(\overline{r0_t^{\mathrm{I}}}\right).\left(x0_t^{\mathrm{I}(i=11)}(l)\right)\right] + \left[\left(\overline{r0_t^{\mathrm{Q}}}\right).\left(x0_t^{\mathrm{Q}(i=11)}(l)\right)\right]\right) - \frac{L_c}{2}\left(\left[\left(\overline{r0_t^{\mathrm{I}}}\right).\left(x0_t^{\mathrm{I}(i=00)}(l)\right)\right] + \left[\left(\overline{r0_t^{\mathrm{Q}}}\right).\left(x0_t^{\mathrm{Q}(i=00)}(l)\right)\right]\right) \quad (42)$$

The number of computations for the intrinsic LLR is the same as shown in Table 9.

2.3 Decoding for Duo-Binary Turbo Codes using Method 3 [24]

The Turbo decoding equations to perform symbol-level decoding with bit-level LLRs as input was proposed by the authors of [24]. The encoding structure and gray coded modulation for the double binary Turbo codes are as shown in Fig. 1 and Fig. 2 respectively. The systematic and parity information are modulated based on the Gray-coded QPSK bit-mapping constellation of Fig. 2. The modulated symbols are then transmitted over an AWGN channel. The noisy versions of $x0_t$, $x1_t$ and $x2_t$ are intercepted at the receiver side as $r0_t$, $r1_t$ and $r2_t$ respectively. A soft de-mapping algorithm is employed to extract the bit-level LLRs from the received noisy symbols. The modulation and soft-demapping processes are as shown in Fig. 5.

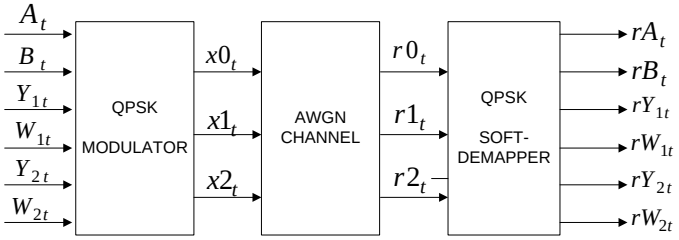

Fig. 5 Modulation and Soft De-mapping

The soft-demapping equations are follows:

$$rA_t = \mathrm{real}(r0_t); \quad rB_t = \mathrm{imag}(r0_t); rY_{1t} = real(r1_t); \quad rW_{1t} = \mathrm{imag}(r1_t); \quad rY_{2t} = real(r2_t); rW_{2t} = \mathrm{imag}(r2_t) \quad (43)$$

Where, real() and imag() denote the real and imaginary parts of the input complex arguments: $r0_t$, $r1_t$ and $r2_t$. The Turbo decoder is as shown in Fig. 6.

The parameters shown in Fig. 6 are defined as:

rA_t and rB_t are the soft-bit de-mapped LLRs of the received complex systematic symbol $r0_t$ at time *t*,

$\overline{rA_t}$ and $\overline{rB_t}$ are the soft-bit de-mapped LLRs of the received complex interleaved systematic symbol $\overline{r0_t}$ at *time t*,

rY_{1t} and rW_{1t} are the soft-bit de-mapped LLRs of the received complex systematic symbol $r1_t$ at *time t*,

rY_{2t} and rW_{2t} are the soft-bit de-mapped LLRs of the received complex systematic symbol $r2_t$ at time *t* and

$xA_t, xB_t, xY_{1t}, xW_{1t}, xY_{2t}\ and\ xW_{2t} \in \{+1, -1\}$ depending on whether the bits $A_t, B_t, Y_{1t}, W_{1t}, Y_{2t}\ and\ W_{2t} \in \{1, 0\}$ respectively.

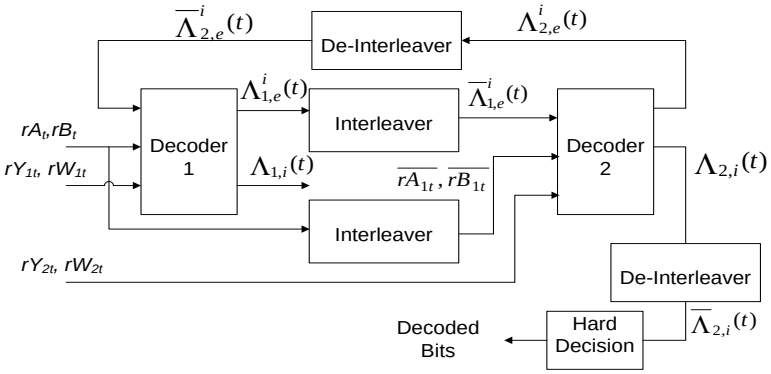

Fig. 6: Turbo Decoder with bit-level input LLRs.

The first decoder's branch transition metric is given as:

$$\gamma_t^{1,i}(l',l) = \bar{\Lambda}^i_{2,e}(t) + \frac{L_c}{2}([(rA_t)\,.\,(xA_t)] + [(rB_t)\,.\,(xB_t)] + [(rY_{1t})\,.\,(xY_{1t})] + [(rW_{1t})\,.\,(xW_{1t})]) \quad (44)$$

The number of computations required for the branch metric shown in Eqn. (58) is shown in Table 10. The information provided in Table 10 will be used in the analysis part of section 3.

Table 10: Number of computations for intrinsic LLRs of symbol $\boldsymbol{i}$ at time $\boldsymbol{t}$ for Decoder 1 with Method 2.

	Additions	Divisions	Multiplications	Total
Branch Transition Metric	2	1	5	10

The forward recursive variables for the first decoder are computed in the same way as for Method 1 which is shown in Eqns. (3) and (4). The number of computations for the forward recursive variable is the same as shown in Table 2. The backward recursive variables for the first decoder are computed in the same way as for Method 1, which is shown in Eqns. (5) and (6). The number of computations for the backward recursive variable is the same as shown in Table 3. The log-likelihood ratios for the first decoder are computed in the same way as for Method 1, which is shown in Eqn. (4). The number of computations for the a-posteriori LLRs is the same as shown in Table 4. The a-posteriori LLR comprises of 3 LLRs, namely, the a-priori LLR, the intrinsic LLR and the extrinsic LLR are related as shown in Eqns. (4), (5) and (6) [23]. The number of computations for the extrinsic LLR is the same as shown in Table 5. The intrinsic LLRs are computed as follows:

$$\Lambda^i_{1,in} = \frac{L_c}{2}\left(\left[(rA_t).\left(xA_t^{(i)}\right)\right] + \left[(rB_t).\left(xB_t^{(i)}\right)\right]\right) \quad (45)$$

The number of computations required for each intrinsic LLR of symbol i at time instant t shown in Eqn. (45) is shown in Table 11. The information provided in Table 11 will be used in the analysis part of section 3.

The operation of the second decoder can now be started. The branch transition metric for the second decoder is computed as:

$$\gamma_t^{2,i}(l',l) = \bar{\Lambda}_{1,e}^{i}(t) + \frac{L_c}{2}([(\overline{rA_t}) \,.\, (xA_t)] + [(\overline{rB_t}) \,.\, (xB_t)] + [(rY_{2t}) \,.\, (xY_{2t})] + [(rW_{2t}) \,.\, (xW_{2t})]) \quad (46)$$

The number of computations for the branch transition metrics of decoder 2 is the same as shown in Table 10.

The forward recursive variables for the first decoder are computed in the same way as for Method 1, which is shown in Eqn. (2). The number of computations for the forward recursive variable of decoder 2 is the same as shown in Table 2. The backward recursive variables for the second decoder are computed as shown in Eqn. (3). The number of computations for the backward recursive variable of decoder 2 is the same as shown in Table 3. The equation for the log likelihood ratio is the same as shown in Eqn. (4). The number of computations for the a-posteriori LLR of decoder 2 is the same as shown in Table 4. The a-posteriori LLR comprises of 3 LLRs, namely, the a-priori LLR, the intrinsic LLR and the extrinsic LLR are related as shown in Eqn. (5) [23]. The number of computations for the extrinsic LLRs of decoder 2 is the same as shown in Table 5. The intrinsic LLR of decoder 2 associated with the interleaved systematic bits: (A_t^{Int}, B_t^{Int}) can be represented as:

$$\Lambda_{2,in}^{i} = \frac{L_c}{2}\left(\left[(\overline{rA_t}).\left(xA_t^{(i)}\right)\right] + \left[(\overline{rB_t}).\left(xB_t^{(i)}\right)\right]\right) \quad (47)$$

The computational equations for the intrinsic LLRs are as follows:

$$\Lambda_{2,in}^{0} = \frac{L_c}{2}([(\overline{rA_t}).(-1)] + [(\overline{rB_t}).(-1)]) \quad (48)$$

$$\Lambda_{2,in}^{1} = \frac{L_c}{2}([(\overline{rA_t}).(-1)] + [(\overline{rB_t}).(+1)]) \quad (49)$$

$$\Lambda_{2,in}^{2} = \frac{L_c}{2}([(\overline{rA_t}).(+1)] + [(\overline{rB_t}).(-1)]) \quad (50)$$

$$\Lambda_{2,in}^{3} = \frac{L_c}{2}([(\overline{rA_t}).(+1)] + [(\overline{rB_t}).(+1)]) \quad (51)$$

The number of computations for the intrinsic LLRs of decoder 2 is the same as shown in Table 11.

2.4 Computational Complexity Analysis

In this section, the computational complexities for the three decoding methods have been compared. The break-down of the number of computations at each half-iteration for Methods 1, 2, and 3 are shown in Tables 12, 13, and 14 respectively. The values obtained for the metrics computed over one half-iteration for Method 1 are explained next. The number of computations for the branch transition metric of Eqn. (1) shown in Table 1 is for a single branch transition metric. Each transition in the trellis consists of 32 branch transition metrics and in all, there are N_c transitions in the trellis. These explain the multiplication by 32 and N_c for the computations of the branch transition metrics. The number of computations for the forward recursive variable of Eqn. (2) and backward recursive variable of Eqn. (3), as shown in Tables 2 and 3 respectively, are multiplied by 8 and N_c since there are 8 states for each transition over a total couple length of N_c.The computations for the a-posteriori LLRs of Eqn. (4) as shown in Table 4 are multiplied by 4 and N_c for the 4 symbols over the whole couple length. The computations of the extrinsic LLRs of Eqn. (7) as shown in Table 5 are also multiplied by 4 and N_c. The computations of the intrinsic LLRs of Eqns. (9) and (10) as shown in Table 6 cater for the 4 symbols and thus are only multiplied by the couple length, N_c. Similarly, the computations of the a-

posteriori probabilities of Eqns. (11 - 14), as shown in Table 7, are only multiplied by the couple length, N_c.

Table 12: Breakdown of the number of computations at one half-iteration for Method 1.

	Log Operations	Maximum Operations	Additions	Sub-tractions	Multipli-cations	Divisions	Total
Branch Transition Metric	$1\times32\times\mathbf{N_c}$		$3\times32\times\mathbf{N_c}$	$5\times32\times N_c$	$4\times32\times N_c$		$\mathbf{416N_c}$
Forward Metric		$1\times8\times\mathbf{N_c}$	$4\times8\times N_c$				$\mathbf{40N_c}$
Backward Metric		$1\times8\times\mathbf{N_c}$	$4\times8\times N_c$				$\mathbf{40N_c}$
A-Posteriori LLR		$2\times4\times\mathbf{N_c}$	$32\times4\times N_c$	$1\times4\times N_c$			$\mathbf{140N_c}$
Extrinsic LLR				$2\times4\times\mathbf{N_c}$			$\mathbf{8N_c}$
Intrinsic LLR			$1\times\mathbf{N_c}$		$6\times N_c$	$3\times N_c$	$\mathbf{10N_c}$
A-Posteriori Probabilities		$4\times N_c$		$4\times N_c$			$\mathbf{8N_c}$
TOTAL	$\mathbf{32N_c}$	$\mathbf{28N_c}$	$\mathbf{289N_c}$	$\mathbf{176N_c}$	$\mathbf{134N_c}$	$\mathbf{3N_c}$	$\mathbf{662N_c}$

The values obtained for the metrics computed over one half-iteration for Method 2 are explained next. The number of computations for the branch transition metric of Eqn. (33) shown in Table 8 are multiplied by 32 and N_c as explained for Method 1. The number of computations for the forward recursive variable, backward recursive variable, a-posteriori LLRs and extrinsic LLRs are exactly as explained for Method 1. The computations of the intrinsic LLRs of Eqns. (34 - 37), as shown in Table 9, are multiplied by 4 and N_c to cater to the 4 symbols and the couple length.

Table 13: Breakdown of the number of computations at one half-iteration for Method 2.

	Maximum Operations	Additions	Subtractions	Multiplications	Total
Branch Transition Metric		$4\times32\times N_c$		$5\times32\times N_c$	$\mathbf{288N_c}$
Forward Metric	$1\times8\times N_c$	$4\times8\times N_c$			$\mathbf{40N_c}$
Backward Metric	$1\times8\times N_c$	$4\times8\times N_c$			$\mathbf{40N_c}$
A-Posteriori LLR	$2\times4\times N_c$	$32\times4\times N_c$	$1\times4\times N_c$		$\mathbf{140N_c}$
Extrinsic LLR			$2\times4\times N_c$		$\mathbf{8N_c}$
Intrinsic LLR		$2\times4\times N_c$	$1\times4\times N_c$	$4\times4\times N_c$	$\mathbf{28N_c}$
TOTAL	$\mathbf{24N_c}$	$\mathbf{328N_c}$	$\mathbf{16N_c}$	$\mathbf{176N_c}$	$\mathbf{544N_c}$

The values obtained for the metrics computed over one half-iteration for Method 3 are explained next. The number of computations for the branch transition metric of Eqn. (44) shown in Table 10 are multiplied by 32 and N_c as explained for Method 1. The computations for the forward recursive variable, backward recursive variable, a-posteriori

LLRs and extrinsic LLRs are exactly as explained for Method 1. The computations of the intrinsic LLRs of Eqn. (45) as shown in Table 11 are multiplied by 4 and N_c to cater for the 4 symbols and the couple length.

Table 14: Breakdown of the number of computations at one half-iteration for Method 3.

	Maximum Operations	**Additions**	**Subtractions**	**Multiplications**	**Total**
Branch Transition Metric		$4\times32\times N_c$		$5\times32\times N_c$	$\mathbf{288N_c}$
Forward Metric	$1\times8\times N_c$	$4\times8\times N_c$			$\mathbf{40N_c}$
Backward Metric	$1\times8\times N_c$	$4\times8\times N_c$			$\mathbf{40N_c}$
A-Posteriori LLR	$2\times4\times N_c$	$32\times4\times N_c$	$1\times4\times N_c$		$\mathbf{140N_c}$
Extrinsic LLR			$2\times4\times N_c$		$\mathbf{8N_c}$
Intrinsic LLR		$1\times4\times N_c$		$3\times4\times N_c$	$\mathbf{16N_c}$
TOTAL	$\mathbf{24N_c}$	$\mathbf{324N_c}$	$\mathbf{12N_c}$	$\mathbf{172N_c}$	$\mathbf{532N_c}$

The total number of computations for each half-iteration for Method 1 is given as:

$$C_{M1} = (416 + 40 + 40 + 140 + 8 + 10 + 8) \text{ x } N_c = 662N_c \quad (52)$$

Where, C_{M1} is the total number computations per half-iteration for Method 1.

The total number of computations for each half-iteration for Method 2 is given as:

$$C_{M2} = (288 + 40 + 40 + 140 + 8 + 28) \text{ x } N_c = 544N_c \quad (53)$$

Where, C_{M2} is the total number computations per half-iteration for Method 2.

The total number of computations for each half-iteration for Method 3 is given as:

$$C_{M3} = (288 + 40 + 40 + 140 + 8 + 16) \text{ x } N_c = 532N_c \quad (54)$$

Where, C_{M3} is the total number computations per half-iteration for Method 3.

It can be observed that the total number of computations for Method 3 per half-iteration is lower in comparison to that required for both Methods 1 and 2. In total, Method 2 and Method 3 require 118 and 130 fewer computations than Method 1, respectively. From the percentage aspect, Method 2 and Method 3 require 17.8 % and 19.6 % fewer computations than Method 1, respectively, at each half-iteration. However, it is assumed that an addition and a multiplication have the same complexity of one computation.

3. SIMULATION RESULTS

In this section, the performances of the three different decoding methods for Duo-Binary Turbo codes both, with and without circular states are compared.

Q-PSK modulation has been used in all the simulations.

An interleaver size of couple length N_c has been used in all the simulations. The parameters for the duo-binary Turbo code used are as follows [19, 20, 22]:

Feedback branch: 1 + D + D3

First set of parity bits: 1 + D2 + D3

Second set of parity bits: 1 + D3

N_c = 48, 212 and 752

Maximum number of iterations, T = 12.

Code-rates = 1/3,1/2 and 2/3

Channel model: Complex AWGN.

The BER performances for Duo-Binary Turbo codes with N_c = {48, 212 and 752}, code-rate = 1/3, Q-PSK modulation and the different Decoding Techniques with and without circular states are shown in Fig. 7.

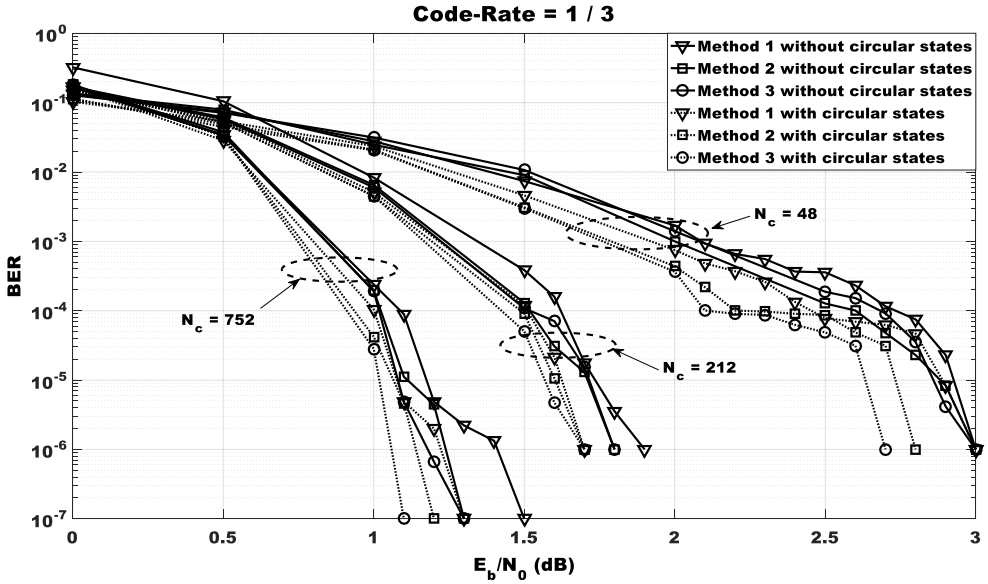

Fig. 7: BER performance for duo-binary Turbo codes with Q-PSK Modulation, N_c = {48, 212 and 752} and rate=1/3.

It can be observed for N_c = 48 from Fig. 7, that without circular states, decoding Methods 2 and 3 have better error performance than decoding Method 1 for BERs below 10^{-4}. For a BER of 10^{-5}, Methods 2 and 3 provide a gain of 0.1 dB compared to Method 1. It can be observed from the results with circular states, that decoding Method 3 has better error performance than decoding Methods 1 and 2 for BERs below 10^{-4}. For a BER of 10^{-5}, Method 3 provides a gain of 0.1 dB compared to Methods 1 and 2.

It can be observed for N_c = 212 from Fig. 7, that without circular states, decoding Methods 2 and 3 have better error performance than decoding Method 1 for the whole BER range. For a BER of 10^{-6}, Methods 2 and 3 provide a gain of 0.1 dB compared to Method 1. It can be observed from the results with circular states that decoding Methods 2 and 3 have better error performance than decoding Method 1 for almost the whole BER range. For a BER of 10^{-5}, Method 3 provides a gain of 0.1 dB compared to Method 1.

It can be observed for N_c = 752 from Fig. 7, that without circular states, decoding Methods 2 and 3 have better error performance than decoding Method 1 for almost the whole BER range. For a BER of 10^{-7}, Methods 2 and 3 provide a gain of 0.2 dB compared to Method 1. It can be observed from the results with circular states that decoding Methods 2 and 3 have better error performance than decoding Method 1 for almost the whole BER range. For a BER of 10^{-7}, Methods 2 and 3 provide a gain of 0.1 dB and 0.2 dB respectively as compared to Method 1. The BER performances for duo-binary Turbo

codes with N_c = {48, 212 and 752}, code-rate = 1/2, Q-PSK modulation and the different decoding techniques with and without circular states are shown in Fig. 8.

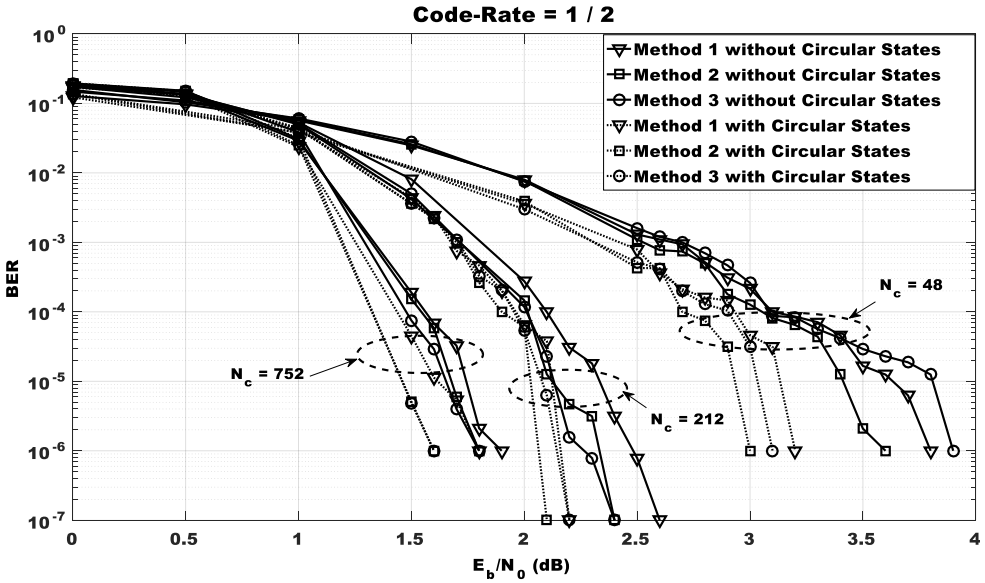

Fig. 8: BER performance for duo-binary Turbo codes with Q-PSK modulation, N_c = {48, 212 and 752} and rate=1/2.

It can be observed for N_c = 48 from Fig. 8, that without circular states, decoding Method 2 has better error performance than decoding Methods 1 and 3 for BERs below 10^{-4}. For a BER of 10^{-6}, Method 2 provides a gain of 0.2 dB as compared to Method 1. It can be observed from the results with circular states that decoding Methods 2 and 3 have better error performance than decoding Method 1 for BERs below 10^{-4}. For a BER of 10^{-6}, Methods 2 and 3 provide a gain of 0.1 dB and 0.2 dB respectively compared to Method 1.

It can be observed for N_c = 212 from Fig. 8, that without circular states, decoding Methods 2 and 3 have better error performance than decoding Method 1 for BERs below 10^{-2}. For a BER of 10^{-7}, Methods 2 and 3 provide a gain of 0.2 dB as compared to Method 1. It can be observed from the results with circular states that decoding Methods 2 and 3 have better error performance than decoding Method 1 for BERs below 10^{-4}. For a BER of 10^{-7}, Method 2 provides a gain of 0.1 dB as compared to Methods 1 and 3.

It can be observed for N_c = 752 from Fig. 8, that without circular states, decoding Methods 2 and 3 have better error performance than decoding Method 1 for BERs below 10^{-4}. For a BER of 10^{-6}, Methods 2 and 3 provide a gain of 0.1 dB as compared to Method 1. It can be observed from the results with circular states that decoding Methods 2 and 3 have better error performance than decoding Method 1 for BERs below 10^{-2}. For a BER of 10^{-6}, Methods 2 and 3 provide a gain of 0.2 dB as compared to Method 1. The BER performances for duo-binary Turbo codes with N_c = {48, 212 and 752}, code-rate = 2/3, Q-PSK modulation and the different Decoding Techniques without circular states are shown in Fig. 9.

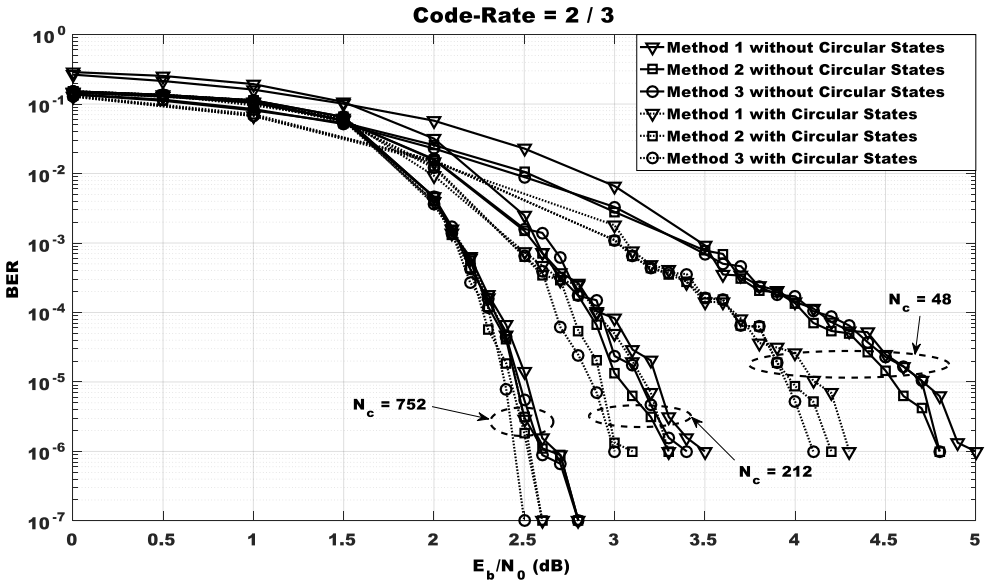

Fig. 9: BER performance for duo-binary Turbo codes with Q-PSK modulation, N_c = {48, 212 and 752} and rate=2/3.

It can be observed for N_c = 48 from Fig. 9, that without circular states, decoding Methods 2 and 3 have better error performance than decoding Method 1 for the whole BER range. For a BER of 10^{-6}, Methods 2 and 3 provide a gain of 0.2 dB compared to Method 1. It can be observed from the results with circular states that decoding Method 3 has a better error performance than both decoding Methods 1 and 2 for BERs below 10^{-4}. For a BER of 10^{-6}, Method 3 provides a gain of 0.2 dB and 0.1 dB as compared to Methods 1 and 2 respectively.

It can be observed for N_c = 212 from Fig. 9, that without circular states, decoding Methods 2 and 3 have better error performance than decoding Method 1 for almost the whole BER range. For a BER of 10^{-6}, Method 2 provides a gain of 0.1 dB and 0.2 dB as compared to Methods 3 and 1 respectively. It can be observed from the results with circular states that decoding Methods 2 and 3 have better error performance than decoding Method 1 for BERs below 10^{-4}. For a BER of 10^{-6}, Methods 2 and 3 provide a gain of 0.3 dB as compared to Method 1.

It can be observed for N_c = 752 from Fig. 9, that without circular states, decoding Methods 2 and 3 have almost the same error performance as decoding Method 1 for the whole BER range. It can be observed from the results with circular states that at a BER of 10^{-7}, decoding Method 3 outperforms the decoding Methods 1 and 2 by 0.1 dB.

4. CONCLUSION

In this paper, an investigation of the state of the art of different iterative decoding techniques for the Max-Log MAP algorithm have been presented for duo-binary Turbo codes. Different couple lengths and code-rates have been employed with duo-binary Turbo codes with and without the incorporation of circular states have been used for this work. Essentially, three different decoding techniques have been identified for this work. The identified schemes were implemented in Matlab and all the simulations were carried out using an AWGN channel, QPSK modulation and the eight-state double binary turbo encoder of the DVB-RCS standard. The computational complexities of the 3 methods

were analyzed for one half-iteration. It can be observed that Method 2 and Method 3 require 17.8 % and 19.6 % fewer computations than Method 1 respectively at each half-iteration with the assumption that an addition and a multiplication have the same complexity of one computation. Intensive simulations were then carried out for duo-binary Turbo codes with and without circular states for couple lengths: 48, 212 and 752; and code-rates: 1/3, ½ and 2/3. In most results, Methods 2 and 3 outperform Method 1 for the whole BER range. These results are important when low-complexity decoding algorithms for non-binary Turbo codes need to be considered. Compared to previous work, the investigation in this paper is geared towards analyzing three different sets of decoding equations for the Max Log MAP algorithm used with Duo-Binary Turbo codes. Additionally, based on the equations used to compute the different parameters of the iterative process, a computational complexity analysis was also performed. Methods 1 and 2 are variants of a symbol-level decoding mechanism with the symbol-level a-priori LLRs as input. However, Method 3 performs symbol-level decoding using bit-level LLRs and the results become significant in the sense that the limitation of using only QPSK modulation with duo-binary Turbo codes is overcome with this technique. Using higher order modulations with duo-binary Turbo codes help achieving higher spectral efficiencies. This possibility of using higher order modulations with non-binary Turbo codes opens avenues for the incorporation of prioritization constellation mapping and other schemes in view to enhance error performance with decoders having low computational complexities.

ACKNOWLEDGEMENT

The technical support of the University of Mauritius and the financial support of the Tertiary Education Commission are duly acknowledged.

REFERENCES

[1] Berrou C, Glavieux A, Thitimajshima P. (1993) Near Shannon limit error-correcting coding and decoding: Turbo-codes. IEEE Iinternational Conference on Communications, ICC 93, Geneva.

[2] Shannon C E. (1948) A mathematical theory of communications, part I. Bell System Technical Journal, 27:379-423.

[3] Berrou C, Jézéquel M. (1999) Non Binary Convolutional Codes for Turbo Coding. IEEE Electronic Letters, 35(1):39-40.

[4] Berrou C, Douillard C, Jézéquel M. (1999) Multiple Parallel Concatenation of Circular Recursive Convolutional (CRSC) codes. Annals of Telecommunications, 54(3-4):166-172.

[5] Giancristofaro D, Bartolazzi A. (2001) Novel DVB-RCS standard turbo codes: details and performance of a decoding algorithm. Seventh International Workshop in Digital Signal Processing Techniques for Space Communications.

[6] Soleymani M R, Gao Y, Vilaipornsawai U. (2002) Turbo Coding for Satellite and Wireless Communications. Kluwer Academic Publishers, New York.

[7] Soleymani M R, Gao Y. (2002) Spectrally efficient non-binary turbo codes: Beyond DVB-RCS standard. Proceedings of 21st Biennial Symposium on Communications, 3:951-955.

[8] Kalama M, Acar G, Evans B, Isoard A. VoIP over DVB-RCS Satellite Systems: Trial Results and the Impact of Adaptive Speech Coding using Cross-Layer Design. International journal of Computer and Telecommunications Networking, 52(13):2461-2742.

[9] Berrou C, Jezequel M, Douillard C, Kerouedan S. (2001) The advantages of non-binary turbo codes. Information Theory Workshop, IEEE proc.

[10] Balta H, Alexa F, Vesa A. (2014) On the allocation of double-binary turbo coded bits in the case of 16-QAM modulation. 11th International Symposium on Electronics, Timisoara.

[11] Shaker S W. (2014) DVB-RCS: Efficiently quantized turbo decoder. 16th International Conference on Advanced Communication Technology (ICACT), Pyeongchang.
[12] Zhan M, Zhou L, Wu J. (2014) A low-memory intensive decoding architecture for double-binary convolutional turbo code. Turkish Journal of Electrical Engineering & Computer Sciences, pp. 202-213.
[13] Zheng J Y, Li L, Zhu Y S. (2015) The Performance of Double-Binary Turbo Codes in the HAP-Based Communication System. Applied Mechanics and Materials, 713-715:1141-1144.
[14] Balta H, Lucaciu R, Gajtzki P, Isar A. (2012) QPP Interleaver based on Turbo Code for DVB-RCS Standard. 4th International Conference on Computer Modeling, Singapore.
[15] Park T, Kim M, Kim C, Jung J. (2010) Analysis of Turbo codes for Next Generation DVB-RCS system. 28th American Institute of Aeronautics and Astronautics International Communications Satellite Systems Conference (ICSSC), Anaheim, California.
[16] Vucetic B, Yuan J. (2000) Turbo Codes: Principles and Applications. Kluwer Academic Publishers.
[17] Gao Y, Soleymani MR. (2002) Triple binary circular recursive systematic convolutional turbo codes. IEEE, pp 1.
[18] Gao Y, Soleymani MR. (2002) Spectrally efficient non-binary turbo codes: Beyond DVB-RCS standard. Proceedings of 21st Biennial Symposium on Communications, 3:951-955.
[19] Fowdur T P, Beni Y. (2012) Reliable JPEG image transmission using unequal error protection with modified non-binary turbo codes. IARIA, 5:284-296.
[20] Mouhamedou Y O C. (2005) [Online]. Available: http://www-mmsp.ece.mcgill.ca/MMSP/Theses/2005/Ould-Cheikh-MouhamedouT2005.pdf. [Zugriff am December 2012].
[21] Balta H, Kovaci M, Botiz C, Poenaru C. (2009) Bit Decoding versus Symbol Decoding in Multi-Binary Turbo Decoders. 4th International Conference of Engineering Technologies, ICET 2009, Novi, Sad.