Wireless Personal Communications (2021) 118:1591–1619
https://doi.org/10.1007/s11277-021-08106-x

On the Logical Computational Complexity Analysis of Turbo Decoding Algorithms for the LTE Standards

Y. Beeharry[1] · T. P. Fowdur[1] · K. M. S. Soyjaudah[2]

Accepted: 14 January 2021 / Published online: 2 February 2021

Abstract
Evaluating the computational complexity of decoders is a very important aspect in the area of Error Control Coding. However, most evaluations have been performed based on hardware implementations. In this paper, different decoding algorithms for binary Turbo codes which are used in LTE standards are investigated. Based on the different mathematical operations in the diverse equations, the computational complexity is derived in terms of the number of binary logical operations. This work is important since it demonstrates the computational complexity breakdown at the binary logic level as it is not always evident to have access to hardware implementations for research purposes. Also, in contrast to comparing different Mathematical operations, comparing binary logic operations provides a standard pedestal in view to achieve a fair comparative analysis for computational complexity. The usage of the decoding method with fewer number of binary logical operations significantly reduces the computational complexity which in turn leads to a more energy efficient/power saving implementation. Results demonstrate the variation in computational complexities when using different algorithms for Turbo decoding as well as with the incorporation of Sign Difference Ratio (SDR) and Regression-based extrinsic information scaling and stopping mechanisms. When considering the conventional decoding mechanisms and streams of 16 bits in length, Method 3 uses 0.0065% more operations in total as compared to Method 1. Furthermore, Method 2 uses only 0.0035% of the total logical complexity required with Method 1. These computational complexity analysis at the binary logical level can be further used with other error correcting codes adopted in different communication standards.

Keywords Computational complexity analysis · Turbo codes · LTE · DVB-RCS · Binary logical operations

✉ Y. Beeharry
y.beeharry@uom.ac.mu

[1] Department of Electrical and Electronic Engineering, Faculty of Engineering, University of Mauritius, Reduit, Mauritius

[2] Port Louis, Mauritius

1 Introduction

Energy efficiency/power saving implementations of decoder systems in telecommunication standards [1, 2] is one of the major concerns in line with the concept of green communications [3, 4] and in parallel to computational methods in the field of mechanics [5–7] and Big Data analytics [8]. Implementations of low-complexity decoders [9, 10] have been brought forward by some researchers and others have performed computational complexity evaluations of the hardware implementations of decoding algorithms [11].

The search for low complexity decoding algorithms is of paramount importance in communication standards. The reason is mainly because reducing the computational complexity engenders a corresponding reduction in the power usage of the electronic devices. Most complexity analysis is performed on Digital Signal Processing hardware and measured in terms of CPU cycles [12–14]. In this work, we propose a computational complexity analysis at the level of binary logical operations by considering three different sets of Turbo decoding equations. The aim behind using binary logical operations is to have a standardized level at which the computational complexity is measured. The different Turbo decoding methods do not have the same sets of Mathematical operations (addition, subtraction, multiplication, etc.) and different Mathematical operations are not necessarily equal in terms of complexity when implemented at the hardware level. However, when converting the different mathematical operations into the number of binary logical operations required, a fair computational complexity analysis can be performed. Additionally, the Turbo decoding mechanisms are equipped with Regression and SDR-based extrinsic information scaling and stopping techniques. A comparative analysis is performed in terms of the error performances as well as the amount of binary logical operations required.

The paper is organized as follows. Section 2 highlights the related-works. Section 3 provides the complexity analysis in terms of binary logical operations for each of the decoding approaches of the binary Turbo codes. Section 4 gives an overview of the different binary Turbo Decoding algorithms and their corresponding complexities in terms of logical operations. Section 5 discusses the performance and complexity analysis. Finally, the work is concluded in Sect. 6.

2 Related Works

Several research works have been initiated and conducted in the quest for low-complexity decoding algorithms having an acceptable trade-off with respect to the corresponding error performances so as to be deployed in relevant communication standards. For example, in [15], the authors have proposed a decoding algorithm for block Turbo codes with low complexity. The algorithm operates on an adaptive application of two different estimation rules and the results demonstrate that the reduction in computational complexity of the proposed algorithm has no significant loss in error performance compared to the conventional one. The authors of [16, 17] have proposed an alternate soft-output decoding mechanism with low complexity for polar codes whose error performance is improved in addition to a significant reduction in terms of storage and processing. The authors of [18] have proposed a normalized Log-MAP (Nor-Log-MAP) decoding algorithm in which the function max* is approximated by using a fixed normalized factor multiplied by the max function. Simulation results show that the proposed algorithm helps in achieving a saving of around 2.1%

in terms of logic resources as compared to the conventional one. Gains of the order of 0.25–0.5 dB in error performance are also realised.

Evaluations of computational complexity have also been performed in the literature. For example, in [19], the authors have analyzed the computational complexity of several turbo decoding algorithms in terms of mathematical operations. The algorithms have also been implemented on a Digital Signal Processor and measurements pertaining to a number of CPU cycles per decoded bit have been taken and analyzed. Results have demonstrated that the Max Log MAP algorithm yields a lower computational complexity than the Viterbi algorithm with no significant loss in error performance. The authors of [20] have proposed two mechanisms which reduce the decoding complexity of Turbo Product Codes using extended Hamming codes as component codes. This reduction in computational complexity is achieved in terms of the Hard Decision Decoding whereby a single component code is required with the proposed algorithm. An additional early termination technique is also proposed for un-decodable blocks which aid in the complexity reduction. The simulation results show that the error performance remains fairly unchanged as compared to the conventional decoding algorithms in addition to the significant reduction in overall computational complexity. In [21], the comparison of performance and computational complexity for two different decoding mechanisms was performed. In this work, the complexity was measured using the number of clock cycles needed to complete the different decoding algorithms. Results demonstrated that Turbo codes were recommended to be used with moderate code-rate and LDPC codes were recommended to be used with high code-rates. Furthermore, the work of [22] investigated three different and efficient error control codes. A derivation of the total number of operations used by the different algorithms has been performed and an evaluation of the results obtained has been compared with benchmarks of state-of-the-art SDR platforms.

3 Logical Complexity Analysis

The assumptions made in the derivation of the total number of computations in this work are as follows:

1. One bitwise logical operation would be either a shift (left or right), or a Boolean operation (OR, NOR, AND, NAND, XOR, XNOR, and NOT).
2. Each value computed would be represented by K bits in general on the binary scale.

The computations for the different operations are shown next.

3.1 Derivation of Complexity in Terms of Logical Operations

In this section, a detailed breakdown of the different mathematical operations in terms of logical operations has been shown.

3.1.1 Addition

The electronic circuit of a Half-Adder [23] which takes as input two bits (BIT 1 and BIT 2) and outputs a SUM BIT and a CARRY BIT is shown in Fig. 1.

One Half-Addition, therefore requires 2 logical operations (1 XOR and 1 AND).

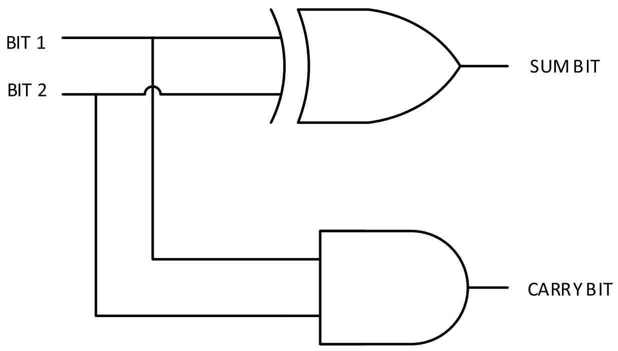

Fig. 1 Half-Adder representation with Binary Logic Gates

The electronic circuit of a Full-Adder [23] which takes as input three bits (BIT 1, BIT 2, and INPUT CARRY BIT) and outputs a SUM BIT and a CARRY BIT is shown in Fig. 2.

One Full-Addition requires 5 logical operators (2 XOR, 2 AND, and 1 OR). The addition of a K-bit stream to another K-bit stream requires: 1 Half-Adder and (K – 1) Full-Adders.

The circuit for performing the addition in parallel is shown in Fig. 3.

The total number of logical operations required is:(1 × 2) + ((K – 1) × 5) = 2 + 5K – 5 and can be represented as:

$$T_L^{Add} = 5\text{K} - 3 \tag{1}$$

3.1.2 Subtraction

The algorithm for subtraction of bit streams (STREAM_1–STREAM_2) is as follows [23]:

1. Perform 2′s complement of STREAM_2
2. Add the 2′s complement of STREAM_2 to STREAM_1.

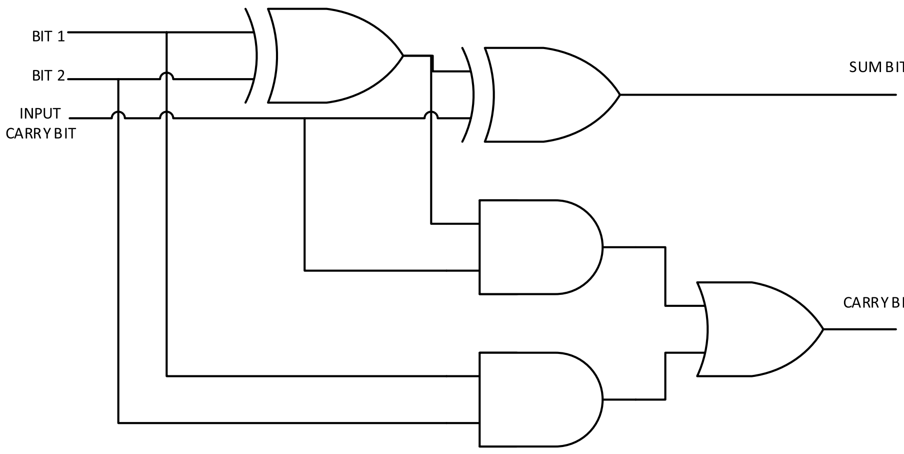

Fig. 2 Full-Adder representation with Binary Logic Gates

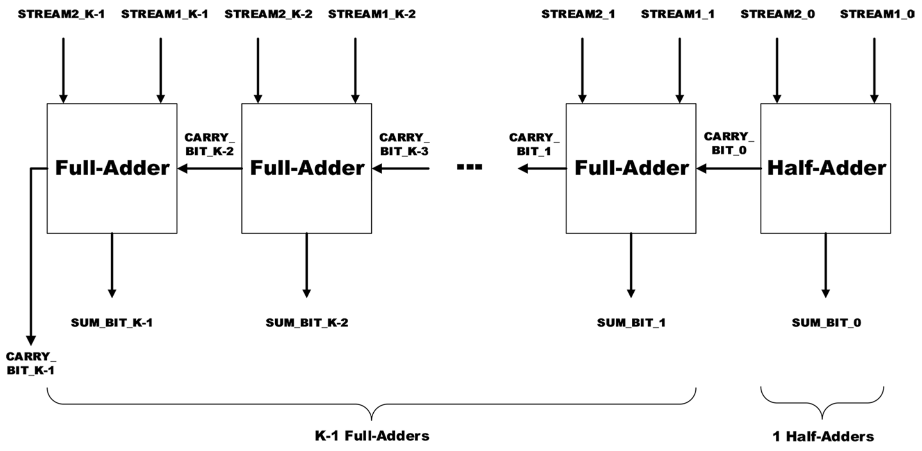

Fig. 3 Representation of Addition in parallel

The 2′complement operation requires K NOT Gates, (K − 1) Full-Adders, and 1 Half-Adder. The total number of logical operations in this case is: K + 5K − 5 + 2 = 6K − 3. The addition operation requires (5K − 3) logical operations. The circuit for subtraction in parallel would be very similar to that for addition with the only difference that there would be two levels instead (One for performing the 2′s complement and the other for the addition of the two streams). The total number of logical operations required for the subtraction is: (6K − 3) + (5K − 3) and can be represented as:

$$T_L^{Sub} = 11\mathrm{K} - 6 \tag{2a}$$

3.1.3 Comparison

A comparison operation can be performed in terms of subtraction operations as demonstrated above, whereby the overflow bit determines the decision of the comparison (whether greater or smaller than). The total number of logical operations required for the comparison operation can be represented as:

$$T_L^{Comp} = 11\mathrm{K} - 6 \tag{2b}$$

3.1.4 Multiplication

The algorithm for binary Multiplication of a multiplicand with a multiplier is as follows:

1. Fix the multiplicand.
2. For each bit in the multiplier.

    a. Shift the multiplicand one bit to the left.

3. End For Loop.
4. Sum all the shifted versions of the multiplicand to obtain the result of the multiplication.

The multiplication of bit streams (STREAM_1 and STREAM_2) requires the following operations [23]:

(K – 1) left shifts and (K – 1) total additions of (2K – 2) bits.
(K – 1) additions of (2K – 2) bits => (K–1) Half-Adders and ((2K–2) × (K–1)) Full-Adders.

- (K–1) Half-Adders
- ($2K^2$–4K + 2) Full-Adders

Total number of operations => (K–1) + (2 × (K–1)) + (5 × ($2K^2$–4K + 2)) = K – 1 + 2K – 2 + $10K^2$ – 20K + 10 and can be represented as:

$$T_L^{Mult} = 10K^2 + 23K + 7 \tag{3}$$

3.1.5 Division

The algorithm for the binary Division of a dividend with a divisor is as follows:

```
1. Set quotient to 0
2. Align leftmost digits in dividend and divisor
3. Repeat
    a. If that portion of the dividend above the divisor is greater than or equal to the divisor
        i. Then subtract divisor from that portion of the dividend and
        ii. Concatenate 1 to the right hand end of the quotient
        iii. Else concatenate 0 to the right hand end of the quotient
    b. Shift the divisor one place right
4. Until dividend is less than the divisor
5. Quotient is correct, dividend is remainder
6. STOP
```

Assuming that the dividend is a K-bit stream, the divisor is an m-bit stream and the binary division by shift and subtract algorithm is used [24], the following number of operations are required:

(i) K shifts
(ii) $\left\lfloor \frac{K}{m+1} \right\rfloor$ shifts consisting of subtractions with (m + 1) bits

Total number of operations => K + $\left\lfloor \frac{K}{m+1} \right\rfloor \times (11(m+1) - 6)$ = K + $\left\lfloor \frac{K}{m+1} \right\rfloor \times (11m - 5)$.

Taking the upper bound, where m is minimum and is taken to be equal to 1, the total number of operations can be represented as:

$$T_L^{Div} = \mathrm{K} + \left(17 \times \left\lfloor \frac{\mathrm{K}}{2} \right\rfloor\right) \tag{4}$$

3.1.6 Logarithm

A logarithm can be bounded based on its properties [25]. Consider the basic inequality:

$$\frac{x-1}{x} \le ln(x) \le x-1, \quad for \quad x > 0 \tag{5}$$

Assuming the upper bound value is computed for the Natural logarithms, the total number of computations would result into:

(i) K NOT gates => K logical operations
(ii) (k – 1) Full-Adders => (5K – 5) logical operations
(iii) 1 Half-Adder => 2 logical operations
(iv) x – 1 operation => (k – 1) full-adders; 1 half-adder => (5K – 3) logical operations.

The total number of logical operations would be: K + 5K – 5 + 2 + 5K – 3 and can be represented as:

$$T_L^{Log} = 11\mathrm{K} - 6 \tag{6}$$

3.1.7 Exponential

An exponential e^τ can be considered as multiplications of the constant value, e by itself τ times, such that:

$$e^\tau = e \times e \times e \times \cdots \times e \tag{7}$$

With e and τ being represented by K bits at the binary level, the exponential would consist of $(2^\mathrm{K} - 1)$ multiplications of e (K-bits). Therefore, the total number of computations would be represented as:

$$T_L^{Exp} = \left(2^\mathrm{K} - 1\right) \times \left(10\mathrm{K}^2 + 23\mathrm{K} + 7\right) \tag{8}$$

3.1.8 Maximum Operation

The selection of the maximum of bit streams (STREAM_1–STREAM_2) requires the following operations [23]:

Treating STREAM_1 and STREAM_2 as signed K-bit integers, then

1. Invert STREAM_2 to its – STRE_2 representation;
2. Sum STREAM_1 to – STREAM_2;
3. Use the sign of the result as a selector variable of a 2-input, K-bit multiplexer.

Converting STREAM_2 to – STREAM_2 by performing 2′s complement. The logical operations required are:

(i) K NOT gates => K logical operations
(ii) (K − 1) Full-Adder => 5 × (K − 1) logical operations
(iii) 1 Half-Adder => 2 logical operations
(iv) The addition of STREAM_1 to − STREAM_2 would require the following operations:
(v) (K − 1) Full-Adder => 5 × (K − 1) logical operations
(vi) 1 Half-Adder => 2 logical operations.

Total number of logical operations => K + 5K − 5 + 2 + 5K − 3 = 11K − 6.

The Selector or Multiplexor would require a separate digital circuit system. Consider a 2:1 MUX as shown in Fig. 4.

Extending from the above logic, 2K-bit inputs through a 2:1 MUX would require K × K binary AND Operations; K × (K − 1) binary OR operations; K × log_2(K) NOT operations. The number of maximum operations would also be impacted by the number of symbols in non-binary Turbo codes. To generalize, the total number of operations required with a maximum of M_s states would be represented as:

$$T_L^{Max} = \left(M_s - 1\right) \times \left(2\mathrm{K}^2 + 10\mathrm{K}{-}6 + \left(\mathrm{K}\ \times log_2(\mathrm{K})\right)\right) \tag{9}$$

4 Logical Complexity of Binary LTE Turbo Codes

The dissimilar decoding techniques and logical complexities for binary LTE Turbo codes are shown in the following sub-sections. The equations of the total computational complexities in terms of binary logical operations for the different decoding approaches of Binary LTE Turbo codes are obtained based on the analysis in Sect. 2.

4.1 Background: Binary Turbo Codes

Figure 5 depicts a classic framework for Binary Turbo decoding. The decoding process is described subsequently. The aggregation of an interleaver together with two decoders make up the Turbo decoder. The first Decoder takes r_0 and r_1 which correspond to the corrupted forms of S_0 and P_1, which are intercepted at the receiver end. Decoder 2 accepts $\overline{r_0}$ which is the interleaved version is of r_0 and r_2, which is the noisy version of P_2. Λ_{1e} and Λ_{2e} are the extrinsic information generated by the decoders. Λ_2 is the Log

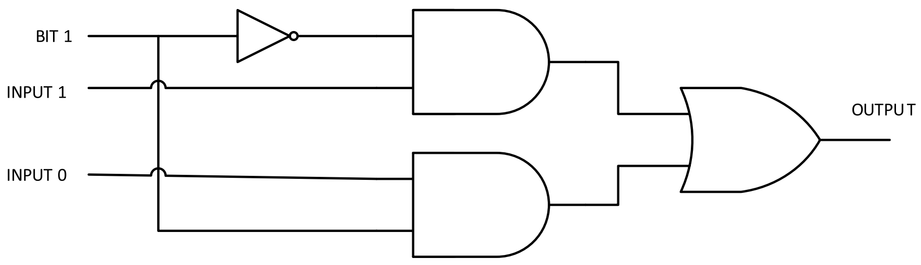

Fig. 4. 2:1 MUX representation with Binary Logic Gates

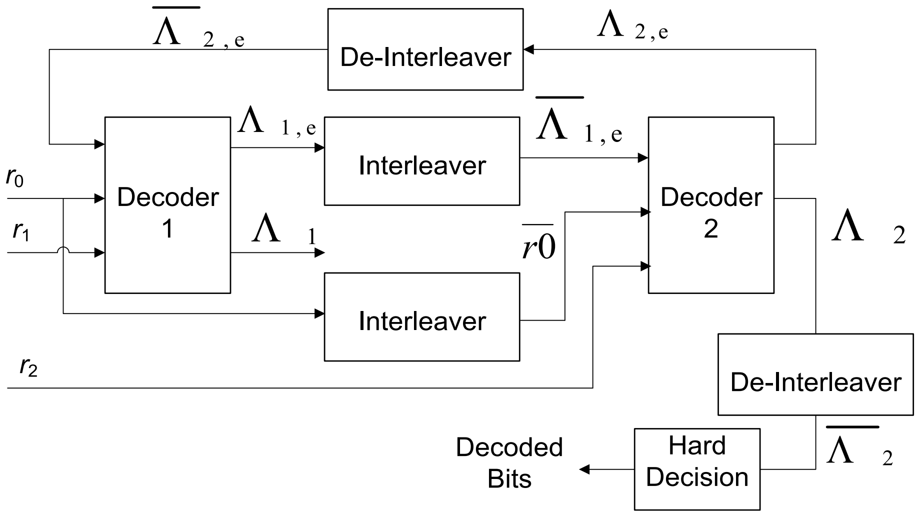

Fig. 5 Generic Turbo decoding structure. Source: [26]

Likelihood Ratio (LLR) output from Decoder 2 and Λ_t is the final hard output obtained after the hard-limiting operation. Standards like CDMA-2000 and Long Term Evolution (LTE) have adopted Turbo codes with the aim to attain higher data rates.

Several equations have been proposed for the diverse Turbo decoding algorithms. As such, the existing Maximum Logarithmic Maximum A-posteriori Probability (Max Log-MAP) binary Turbo decoding system employing different sets of equations are presented in the following sub-sections. The decoding mechanisms are explained in details in [27].

In view to enhance the iterative decoding performance in terms of Bit Error Rate (BER), extrinsic information scaling mechanisms such as Sign Difference Ratio (SDR) [28] and Regression-based [29] have been developed. In addition to improving the error performance, these works also demonstrate the techniques for early-stopping of the decoding mechanisms by employing the computed scale factors. The concept of early-stopping helps in reducing the computational complexity without extensive trade-offs in terms of the error performance. These improvements are however obtained at the expense of additional computations included in the decoding process for computing the scale factor, scaling the extrinsic information and performing comparisons with a set threshold to halt further unnecessary iterations. The decoding frameworks with SDR and Regression-based scaling and stopping are depicted in Figs. 6 and 7 respectively. The detailed operating principles of these algorithms can be obtained from [28, 29] respectively.

4.2 Logical Complexity with Decoding Methods

This section details the complexity breakdowns for each Turbo decoder using each of the Turbo decoding algorithms. The complexity analysis in this sub-section does not involve the incorporation of extrinsic information scaling and early stopping mechanisms.

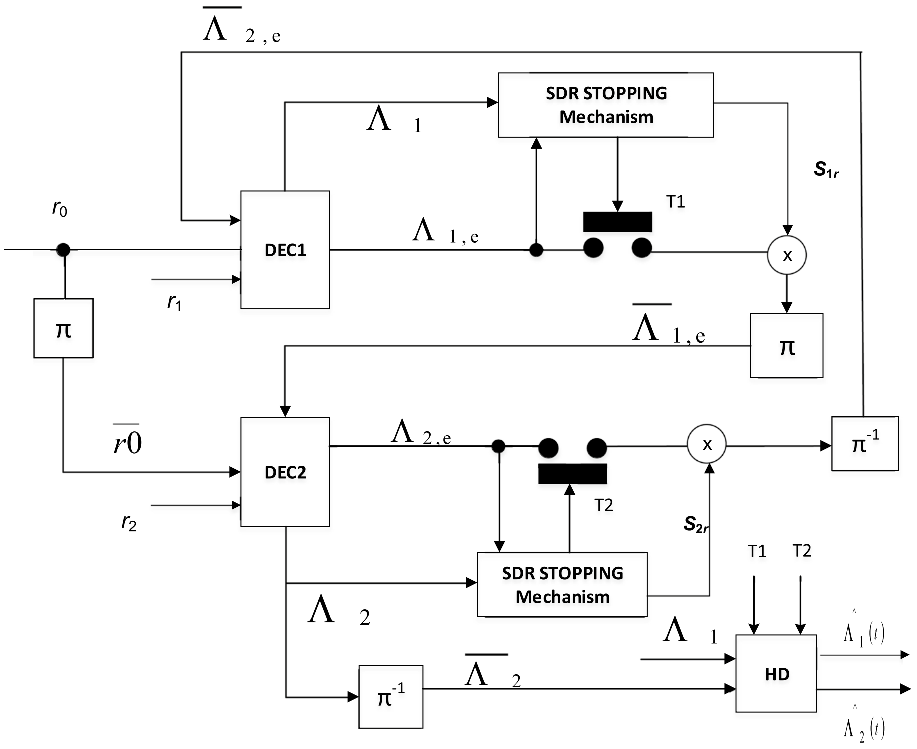

Fig. 6 Turbo decoding structure with SDR scaling and stopping mechanism. Source: [28]

4.2.1 Logical Complexity with Decoding Method 1

The computational complexity breakdown for each Turbo decoder using Method 1 with packet length of N is explained in [27] and presented in Table 1.

The equations for the number of computations per mathematical operation for Method 1 are as follows [27]:

$$C_{M1_{log}}^{binary} = 16\mathrm{N} \tag{10}$$

$$C_{M1_{exp}}^{binary} = 3\mathrm{N} \tag{11}$$

$$C_{M1_{max}}^{binary} = (8 + 8 + 2)\mathrm{N} \tag{12}$$

$$C_{M1_{add}}^{binary} = (16 + 16 + 16 + 32 + 2)\mathrm{N} \tag{13}$$

$$C_{M1_{sub}}^{binary} = (48 + 1 + 2)\mathrm{N} \tag{14}$$

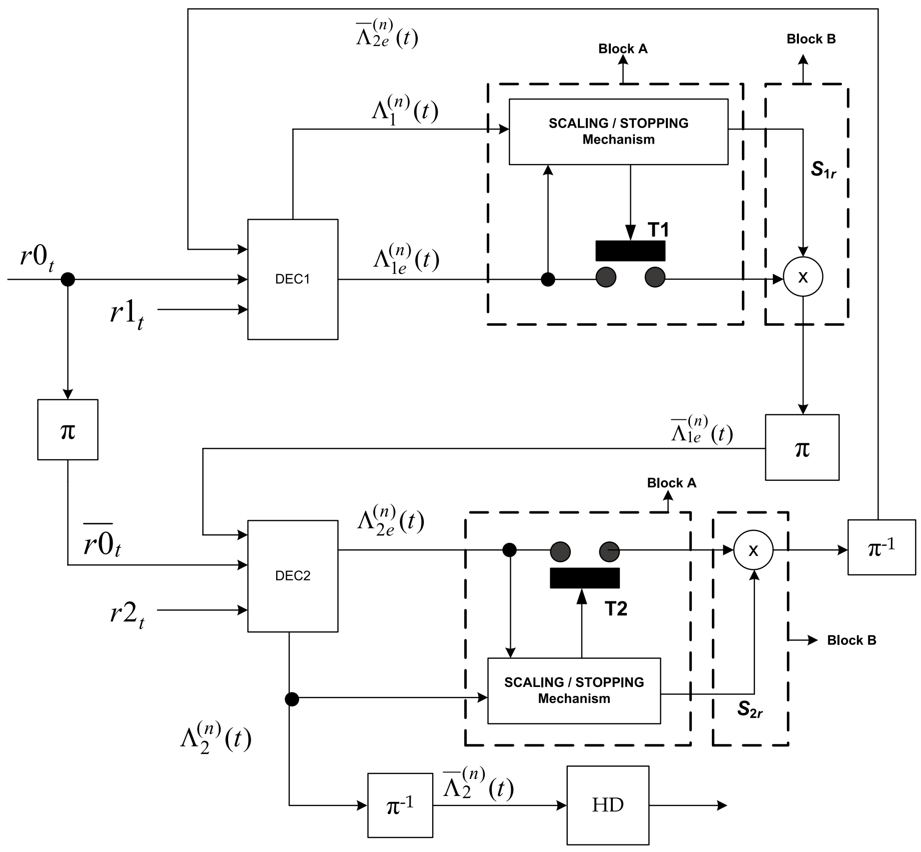

Fig. 7 Turbo decoding structure with Regression-based scaling and stopping mechanism. Source: [29]

Table 1 Computational breakdown for each decoder with Method 1

	Log()	Exp()	Max()	Add()	Sub()	Mult()	Div()
Branch transition metric	1×16×N			1×16×N	3×16×N	2×16×N	
Forward metric			1×8×N	2×8×N			
Backward metric			1×8×N	2×8×N			
A-posteriori LLR			2×N	32×N	1×N		
Extrinsic LLR					2×N		
A-posteriori Probabilities		3×N		2×N			2×N
Total	16 N	3 N	18 N	82 N	51 N	32 N	2 N

$$C_{M1_{mult}}^{binary} = 32\text{N} \tag{15}$$

$$C_{M1_{div}}^{binary} = 2\text{N} \tag{16}$$

where,

$C^{binary}_{M1_{log}}$ is the number of computations required for logarithm operations for Method 1,
$C^{binary}_{M1_{exp}}$ is the amount of computations required for exponential operations for Method 1,
$C^{binary}_{M1_{max}}$ is the number of computations required for Maximum operations for Method 1,
$C^{binary}_{M1_{add}}$ is the amount of computations required for addition operations for Method 1,
$C^{binary}_{M1_{sub}}$ is the number of computations required for subtraction operations for Method 1,
$C^{binary}_{M1_{mult}}$ is the number of computations required for multiplication operations for Method 1,
$C^{binary}_{M1_{div}}$ is the number of computations required for division operations for Method 1.

The equations for the computational complexities of the decoding Method 1 for binary LTE Turbo codes in terms of binary logical operations are as follows:

$$C^{binary}_{M1_{log}} = 16\mathrm{N} = 16\mathrm{N}(11\mathrm{K} - 6) \tag{17}$$

$$C^{binary}_{M1_{exp}} = 3\mathrm{N}\left(2^{\mathrm{K}} - 1\right)\left(10\mathrm{K}^2 + 22\mathrm{K} + 8\right) \tag{18}$$

$$C^{binary}_{M1_{max}} = 18\mathrm{N}\left(M_s - 1\right)\left(2\mathrm{K}^2 + 10\mathrm{K} + \mathrm{K}log_2(\mathrm{K}) - 6\right) \tag{19}$$

$$C^{binary}_{M1_{add}} = 82\mathrm{N}(5\mathrm{K} - 3) \tag{20}$$

$$C^{binary}_{M1_{sub}} = 51\mathrm{N}(11\mathrm{K} - 6) \tag{21}$$

$$C^{binary}_{M1_{mult}} = 32\mathrm{N}\left(10\mathrm{K}^2 + 22\mathrm{K} + 8\right) \tag{22}$$

$$C^{binary}_{M1_{div}} = 2\mathrm{N}\left(\mathrm{K} + \left(17 \times \left\lfloor \frac{\mathrm{K}}{2} \right\rfloor\right)\right) \tag{23}$$

$$\begin{aligned} C^{binary}_{M1_{total}} = {} & 16\mathrm{N}(11\mathrm{K} - 6) + 3\mathrm{N}\left(\left(2^{\mathrm{K}}-1\right)\left(10\mathrm{K}^2 + 22\mathrm{K} + 8\right)\right) \\ & + 18\mathrm{N}\left(\left(2\mathrm{K}^2 + 10\mathrm{K} + \mathrm{K}log_2(\mathrm{K})-6\right)\right)\left(M_s - 1\right) + 82\mathrm{N}(5\mathrm{K} - 3) \\ & + 51\mathrm{N}(11\mathrm{K} - 6) + 32\mathrm{N}\left(10\mathrm{K}^2 + 22\mathrm{K} + 8\right) + 2\mathrm{N}\left(\mathrm{K} + \left(17 \times \left\lfloor \frac{\mathrm{K}}{2} \right\rfloor\right)\right) \end{aligned} \tag{24}$$

4.2.2 Logical Complexity with Decoding Method 2

The computational complexity breakdown for each Turbo decoder using Method 2 with packet length of N is explained in [27] and presented in Table 2.

The equations for the number of computations per mathematical operation for Method 2 is as follows [27]:

Table 2 Ccomputational breakdown for each decoder with Method 2

	Max()	Add()	Sub()
Branch transition metric		1×8×N	
Forward metric	1×8×N	4×8×N	
Backward metric	1×8×N	4×8×N	
Un-coded extrinsic log confidences		2×8×N	
Extrinsic LLR	2×N		1×N
Total	18 N	88 N	1 N

$$C_{M2_{max}}^{binary} = (8 + 8 + 2)\text{N} \tag{25}$$

$$C_{M2_{add}}^{binary} = (8 + 32 + 32 + 16)\text{N} \tag{26}$$

$$C_{M2_{sub}}^{binary} = 1\text{N} \tag{27}$$

where,

$C_{M2_{max}}^{binary}$ is the number of computations required for Maximum operations for Method 2,
$C_{M2_{add}}^{binary}$ is the number of computations required for addition operations for Method 2,
$C_{M2_{sub}}^{binary}$ is the number of computations required for subtraction operations for Method 2,

The equations for the computational complexities of the decoding Method 2 for binary LTE Turbo codes in terms of binary logical operations are as follows:

$$C_{M2_{max}}^{binary} = 18\text{N}\left(M_s - 1\right)\left(2\text{K}^2 + 10\text{K} + \text{K}log_2(\text{K}) - 6\right) \tag{28}$$

$$C_{M2_{add}}^{binary} = 88\text{N}(5\text{K} - 3) \tag{29}$$

$$C_{M2_{sub}}^{binary} = 1\text{N}(11\text{K} - 6) \tag{30}$$

$$C_{M2_{total}}^{binary} = 18\text{N}\left(\left(2\text{K}^2 + 10\text{K} + \text{K}log_2(\text{K}) - 6\right)\right)\left(M_s - 1\right) + 88\text{N}(5\text{K} - 3) + \text{N}(11\text{K} - 6) \tag{31}$$

4.2.3 Logical Complexity with Decoding Method 3

The computational complexity breakdown for each Turbo decoder using Method 3 with packet length of N is explained in [27] and presented in Table 3.

The equations for the number of computations per mathematical operation for Method 3 is as follows:

$$C_{M3_{exp}}^{binary} = 3\text{N} \tag{32}$$

Table 3 Computational breakdown for each decoder with Method 3

	Exp()	Max()	Add()	Sub()	Mult()
Branch transition metric			2×16×N		3×16×N
Forward metric		1×8×N	2×8×N		
Backward metric		1×8×N	2×8×N		
A-posteriori LLR		2×N	32×N	1×N	
Extrinsic LLR				2×N	
A-posteriori probabilities	3×N		7×N		
Total	3 N	18 N	103 N	3 N	48 N

$$C^{binary}_{M3_{max}} = (8 + 8 + 2)\mathrm{N} \tag{33}$$

$$C^{binary}_{M3_{add}} = (32 + 16 + 16 + 32 + 7)\mathrm{N} \tag{34}$$

$$C^{binary}_{M3_{sub}} = (1 + 2)\mathrm{N} \tag{35}$$

$$C^{binary}_{M3_{mult}} = 48\mathrm{N} \tag{36}$$

where,

$C^{binary}_{M3_{exp}}$ is the number of computations needed for exponential operations for Method 3,
$C^{binary}_{M3_{max}}$ is the number of computations needed for Maximum operations for Method 3,
$C^{binary}_{M3_{add}}$ is the number of computations needed for addition operations for Method 3,
$C^{binary}_{M3_{sub}}$ is the number of computations needed for subtraction operations for Method 3,
$C^{binary}_{M3_{mult}}$ is the number of computations needed for multiplication operations for Method 3.

The equations for the computational complexities of the decoding Method 3 for binary LTE Turbo codes in terms of binary logical operations are as follows:

$$C^{binary}_{M3_{exp}} = 3\mathrm{N}\left(2^{\mathrm{K}} - 1\right)\left(10\mathrm{K}^2 + 22\mathrm{K} + 8\right) \tag{37}$$

$$C^{binary}_{M3_{max}} = 18\mathrm{N}\left(M_s - 1\right)\left(2\mathrm{K}^2 + 10\mathrm{K} + \mathrm{K}log_2(\mathrm{K}) - 6\right) \tag{38}$$

$$C^{binary}_{M3_{add}} = 103\mathrm{N}(5\mathrm{K} - 3) \tag{39}$$

$$C^{binary}_{M3_{sub}} = 3\mathrm{N}(11\mathrm{K} - 6) \tag{40}$$

$$C^{binary}_{M3_{mult}} = 48\mathrm{N}\left(10\mathrm{K}^2 + 22\mathrm{K} + 8\right) \tag{41}$$

$$\begin{aligned} C^{binary}_{M3_{total}} &= 3\mathrm{N}\left(\left(2^{\mathrm{K}}-1\right)\left(10\mathrm{K}^2 + 22\mathrm{K} + 8\right)\right) \\ &+ 18\mathrm{N}\left(\left(2\mathrm{K}^2 + 10\mathrm{K} + \mathrm{K}log_2(\mathrm{K})-6\right)\right)\left(M_s - 1\right) \\ &+ 103\mathrm{N}(5\mathrm{K} - 3) + 3\mathrm{N}(11\mathrm{K} - 6) + 48\mathrm{N}\left(10\mathrm{K}^2 + 22\mathrm{K} + 8\right) \end{aligned} \tag{42}$$

4.3 Logical Complexity with SDR Scaling and Stopping

The scaling parameter at iteration *n* for each decoder *d* is calculated as:

$$S_{dn} = \frac{1}{\mathrm{N}} \sum_{t=1}^{\mathrm{N}} f\left(\wedge^{(n)}_{de}, \wedge^{(n)}_{d}\right) \tag{43}$$

where,

$f\left(\wedge^{(n)}_{de}, \wedge^{(n)}_{d}\right) = 1$ if $\wedge^{(n)}_{de}$, and $\wedge^{(n)}_{d}$ have the same sign, otherwise $f\left(\wedge^{(n)}_{de}, \wedge^{(n)}_{d}\right) = 0$,
N represents the size of the frame in bits.

Table 4 presents the breakdown related to the SDR based scaling parameter at each half-iteration, the details of which can be obtained in the work of [27].

The equations for the number of computations per mathematical operation for SDR scaling and stopping in terms of binary logical operations are as follows:

$$C^{binary}_{SDR_{comp}} = (\mathrm{N} + 1)(11\mathrm{K} - 6) \tag{44}$$

$$C^{binary}_{SDR_{add}} = (\mathrm{N} - 1)(5\mathrm{K} - 3) \tag{45}$$

$$C^{binary}_{SDR_{div}} = \mathrm{K} + \left(17\mathrm{x}\frac{\mathrm{K}}{2}\right) \tag{46}$$

$$C^{binary}_{SDR_{mult}} = 10\mathrm{K}^2 + 23\mathrm{K} + 7 \tag{47}$$

where,

$C^{binary}_{SDR_{comp}}$ is the number of computations required for comparison operations for SDR scaling and stopping,

Table 4 Complexity breakdown for one SDR-based scale factor

	Comp()	Add()	Div()	Mult()	Total
SDR scaling and stopping mechanism	N + 1	N − 1	1		2 N + 1

Where, Comp() represents Comparisons, Add() represents Additions, and Div() represents Divisions.

$C_{SDR_{add}}^{binary}$ is the number of computations required for addition operations for SDR scaling and stopping,
$C_{SDR_{div}}^{binary}$ is the number of computations required for division operations for SDR scaling and stopping,
$C_{SDR_{mult}}^{binary}$ is the number of computations required for multiplication operations for SDR scaling and stopping.

4.4 Logical Complexity with Regression Scaling and Stopping

The scaling parameter based on regression, $r_d^{2(n)}$, is presented in (47).

$$r_d^{2(n)} = \left(\frac{\sum_{t=1}^{N} \left(\wedge_d^{(n)}(t) - \widehat{\wedge_d^{(n)}} \right) \mathrm{x} \left(\wedge_{de}^{(n)}(t) - \widehat{\wedge_{de}^{(n)}} \right)}{\sqrt{\sum_{t=1}^{N} \left(\wedge_d^{(n)}(t) - \widehat{\wedge_d^{(n)}} \right)^2 \mathrm{x} \sum_{t=1}^{N} \left(\wedge_{de}^{(n)}(t) - \widehat{\wedge_{de}^{(n)}} \right)^2}} \right)^2 \tag{47}$$

where,

$d = \{1, 2\}$ which is the decoder number,
N is the length of the packet and is set to 6144 in this simulation,
$r_d^{2(n)}$ is the scaling parameter at iteration n for decoder d,
$\wedge_d^{(n)}(t)$ is the tth a-posteriori LLR of decoder d at iteration n and time t,
$\widehat{\wedge_d^{(n)}}$ is the mean a-posteriori LLR at iteration n for decoder d,
$\wedge_{de}^{(n)}(t)$ is the tth extrinsic LLR at iteration n for decoder d,
$\widehat{\wedge_{de}^{(n)}}$ is the mean extrinsic LLR at iteration n for decoder d,
n takes values ½, 1, … I (maximum number of iterations).

Intuitively, a correlation value of 1.0 between the a-posteriori LLR and extrinsic values yields a stopping criterion. However, based on simulations carried out and as explained in [29], a threshold of 0.98 can be used for the stopping mechanism.

The breakdown for the Regression based scaling and stopping at every half-iteration is presented in Table 5.

The equations for the number of computations per mathematical operation for Regression scaling and stopping in terms of binary logical operations are as follows:

$$C_{Reg_{sub}}^{binary} = 4\mathrm{N}(11\mathrm{K} - 6) \tag{48}$$

$$C_{Reg_{mult}}^{binary} = (3\mathrm{N} + 2)\left(10\mathrm{K}^2 + 23\mathrm{K} + 7\right) \tag{49}$$

Table 5 Complexity breakdown: one Regression-based scaling and stopping

	Sub()	Mult()	Add()	Div()	Comp()	Total
Regression-based stopping and scaling	4 N	3 N+2	3 N−3	1	1	10 N+1

$$C_{Reg_{add}}^{binary} = (3\mathrm{N} - 3)(5\mathrm{K} - 3) \tag{50}$$

$$C_{Reg_{div}}^{binary} = \mathrm{K} + \left(17 \times \left\lfloor \frac{\mathrm{K}}{2} \right\rfloor\right) \tag{51}$$

$$C_{Reg_{comp}}^{binary} = (11\mathrm{K} - 6) \tag{52}$$

where,

$C_{Reg_{sub}}^{binary}$ is the number of calculations necessary for subtraction operations for Regression scaling and stopping,
$C_{Reg_{mult}}^{binary}$ is the number of calculations necessary for multiplication operations for Regression scaling and stopping,
$C_{Reg_{add}}^{binary}$ is the number of calculations necessary for addition operations for Regression scaling and stopping,
$C_{Reg_{div}}^{binary}$ is the number of calculations necessary for division operations for Regression scaling and stopping,
$C_{Reg_{comp}}^{binary}$ is the number of calculations necessary for comparison operations for Regression scaling and stopping.

5 Performance and Complexity Analysis

The performance and complexity analysis of the different Binary LTE Turbo decoding algorithms has been performed in this section.

5.1 Performance Analysis

In this sub-section, the graphs pertaining to the error performances and iteration profiles are plotted and analysed. For each of the Turbo decoding method, simulations have been performed using the following system criteria:

- Packet size: 6144 bits
- Number of packets: 200
- Code-rate: 1/3
- Maximum number of iterations: 12
- Modulation: Binary Phase Shift Keying (BPSK)
- Channel Noise: Additive white Gaussian Noise (AWGN)

Simulations have been performed for the following 3 schemes for each of the Turbo decoding method:

- Decoding without scaling or stopping
- Decoding with SDR-based scaling and stopping
- Decoding with Regression-based scaling and stopping

The BER performance graph is presented in Fig. 8.

When comparing the conventional decoding methods without any scaling or stopping mechanisms from the above figure, it can be observed that Method 2 performs better than both Methods 1 and 3 with gains of 0.2 dB and 0.1 dB respectively at a BER of 10^{-7}. When applying SDR scaling and stopping to the 3 decoding methods, there is visually no significant improvement in terms of the error performance as compared to the conventional decoding mechanisms. Regression-based scaling and stopping mechanism improves significantly the error performances of decoding Methods 1 and 3. Methods 1 and 3 provide a gain of 0.35 and 0.3 dB respectively compared to the conventional decoding methods at a BER of 10^{-3} itself. Decoding Method 2 does not perform well after reaching the water-fall region. The performance degrades at BER 10^{-3} and an error-floor is observed at a BER of around 10^{-4}. The iterations profile is demonstrated in Fig. 9.

The number of iterations for the conventional decoding methods without scaling and stopping mechanisms remains constant at 12 which is the specified value for maximum number of iterations. The decoding methods with SDR-based scaling and stopping mechanisms demonstrate a continuous reduction in the number of iterations which is a significant improvement to compensate for the almost no BER performance gain compared to the conventional decoding methods. The 3 decoding methods with SDR scaling and stopping provide gains of 3 and 4.5 iterations at E_b/N_0 values of 0.6 and 0.7 dB respectively compared to the conventional decoding schemes. With the incorporation of regression-based stopping and scaling, the gains obtained in terms of the iterations profile is even more significant compared to those of the conventional decoding methods and the ones using SDR-based scaling and stopping mechanisms. Methods 1 and 3 with regression-based stopping and scaling have already reached a BER of 10^{-7} at E_b/N_0 of 0.6 dB and at this same E_b/N_0 they provide a gain of nearly 6 and 3 iterations compared to the conventional decoding methods and those using SDR-scaling and stopping respectively. Method 2 with regression-based scaling and stopping outperforms the conventional and SDR-based schemes throughout the E_b/N_0 range. It also uses fewer iterations as compared to regression-based Methods 1 and 3 above E_b/N_0 of 0.75 dB. This improved iterations profile is not significant enough when

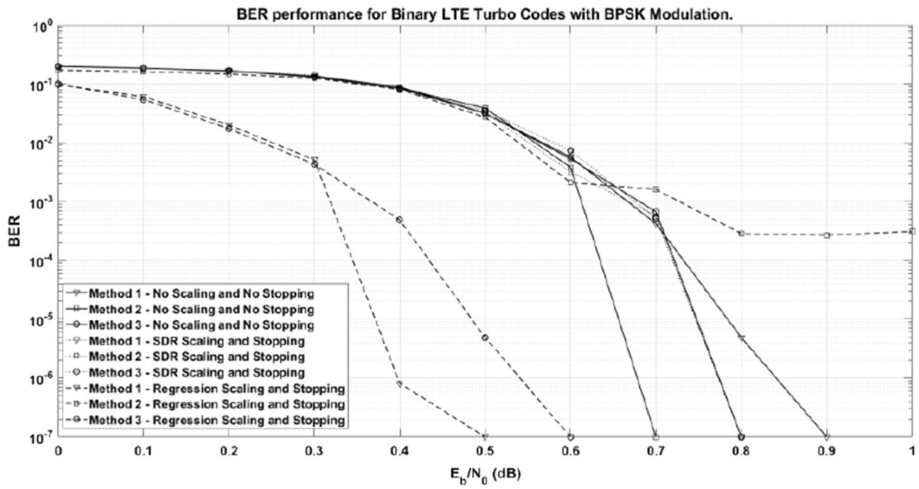

Fig. 8 BER performance graph for Binary LTE Turbo Decoding methods

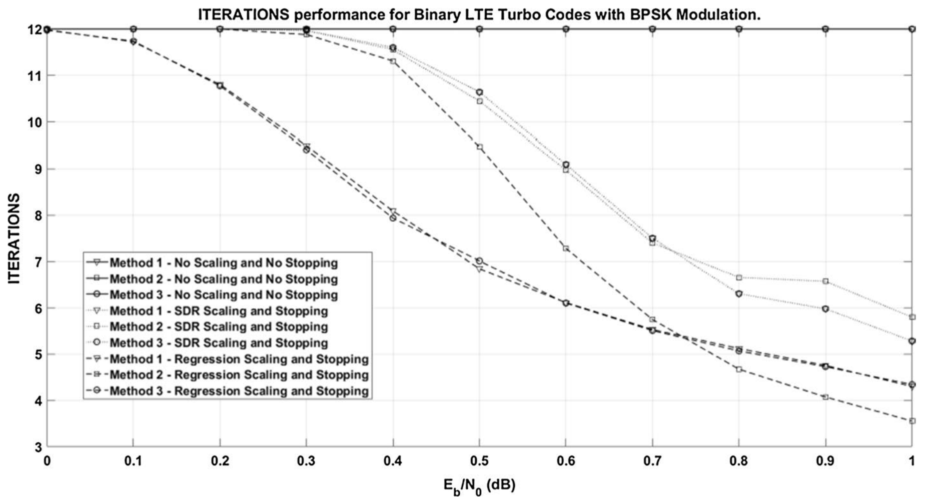

Fig. 9 Iterations profile for Binary LTE Turbo Decoding methods

considering the BER performance. The error-floor which is observed in the BER performance of Method 2 using regression-based scaling and stopping does not suggest a good trade-off between performance and complexity reduction in this case.

5.2 Total Computational Complexity without Scaling/Stopping

The total computational complexity analysis for binary LTE Turbo codes has been performed in this sub-section for different values of K. Tables 6, 7, and 8 depict the total number of binary logical operations for values of K = 16, 32, and 64 respectively.

The bar charts for Logarithm, Exponential, Maximum, Addition, Subtraction, Multiplication, and Division operations in the Binary LTE Turbo codes are shown in Figs. 10, 11, 12, 13, 14, 15 and 16 respectively. The bar chart for the total number of binary operations is shown in Fig. 10.

Figure 11 is broken down into Figs. 11a–c for more clarity.

Figures 11a–c show that exponential operations are used only in Methods 1 and 3.

Figure 12 shows that all 3 techniques use the same extent of Maximum processes.

Table 6 Total number of binary logical operations with K = 16

	Method 1	Method 2	Method 3
Log()	2720 N	0	0
Exp()	574,086,600 N	0	574,086,600 N
Max()	13,140 N	13,140 N	13,140 N
Add()	6314 N	6776 N	7931 N
Sub()	8670 N	170 N	510 N
Mult()	93,440 N	0	140,160 N
Div()	304 N	0	0
TOTAL	574,211,224 N	20,086 N	574,248,341 N

Table 7 Total number of binary logical operations with K = 32

	Method 1	Method 2	Method 3
Log()	5536 N	0	0
Exp()	1.4112e14 N	0	1.4112e14 N
Max()	45,396 N	45,396 N	45,396 N
Add()	12,874 N	13,816 N	16,171 N
Sub()	17,646 N	346 N	1038 N
Mult()	350,464 N	0	525,696 N
Div()	608 N	0	0
TOTAL	1.4112e14 N	59,558 N	574,248,341 N

Table 8 Total number of binary logical operations with K = 64

	Method 1	Method 2	Method 3
Log()	11,168 N	0	0
Exp()	2.3451e24 N	0	574,086,600 N
Max()	13,140 N	165,780 N	13,140 N
Add()	6314 N	27,896 N	7931 N
Sub()	8670 N	698 N	510 N
Mult()	93,440 N	0	140,160 N
Div()	1216 N	0	0
TOTAL	2.3451e24 N	194,374 N	574,248,341 N

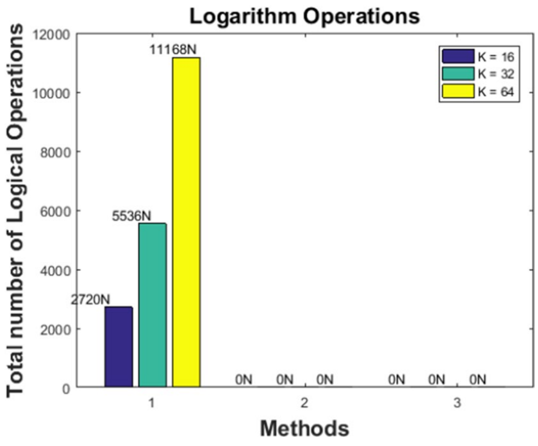

Fig. 10 Bar chart for Logarithm operations in Binary LTE Turbo codes

Figure 13 shows that Method 3 uses more addition operations than Method 2 which in turn uses more addition operations than Method 1.

Figure 14 shows that Method 1 uses more subtraction operations than Method 3 which in turn uses more addition operations than Method 2.

Figure 15 shows that only Methods 1 and 3 use multiplication operations. Method 3 uses more multiplication operations than Method 1.

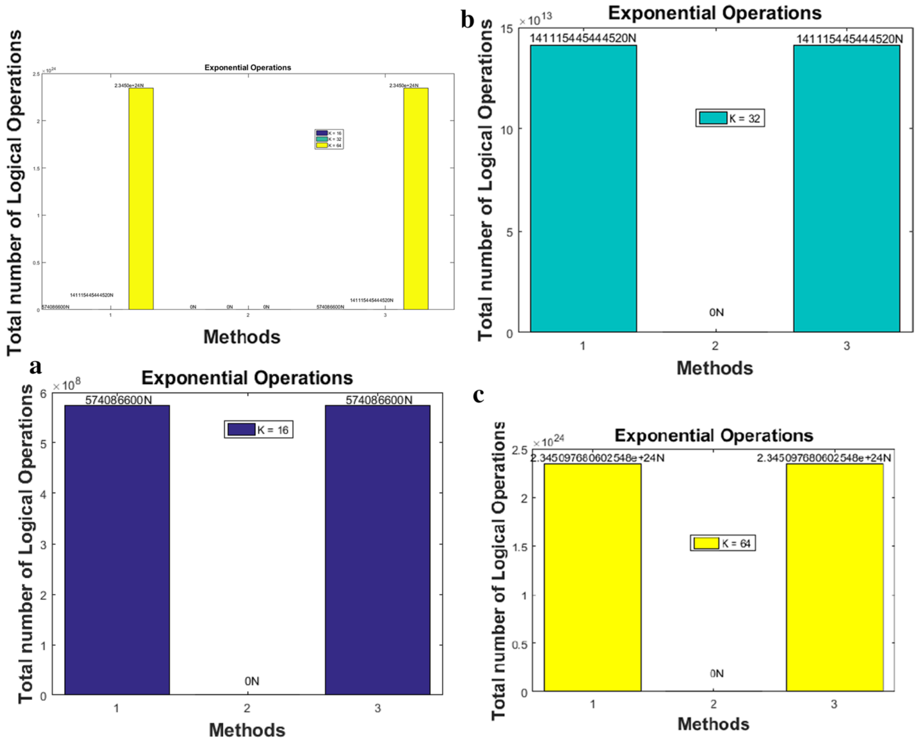

Fig. 11 Bar chart for Exponential operations in Binary LTE Turbo codes **a** K = 16, **b** K = 32, **c** K = 64

Fig. 12 Bar chart for Maximum operations in Binary LTE Turbo codes

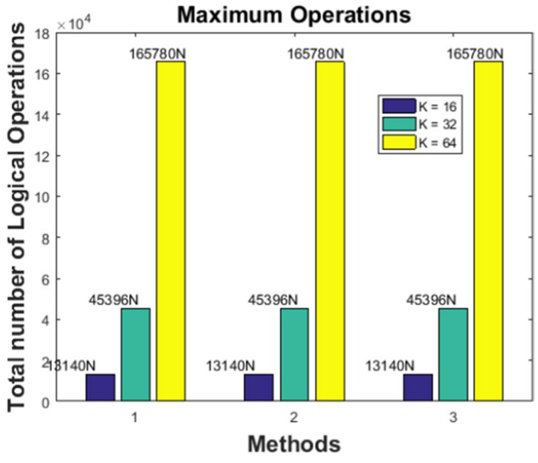

Figure 17 is broken down into Fig. 17a–c for more clarity. All 3 figures show that Methods 1 and 3 use more operations than Method 2. The total number of binary logical operations used by Method 2 is significantly lower than Methods 1 and 3 in the context of binary Turbo codes. Considering K = 16, Methods 1 and 3 use approximately 25,590 times more binary logical operations in total than Method 2 at each half-iteration. Therefore using

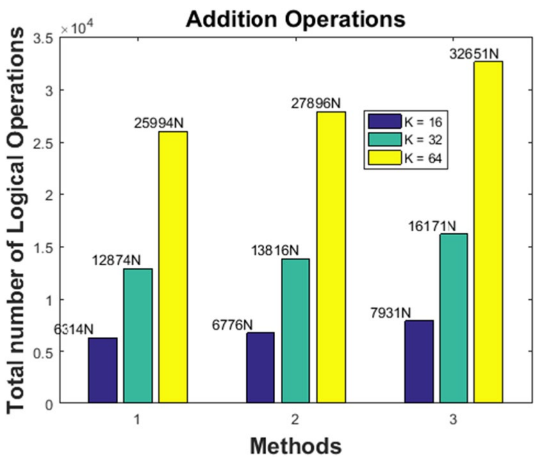

Fig. 13 Bar chart for Addition operations in Binary LTE Turbo codes

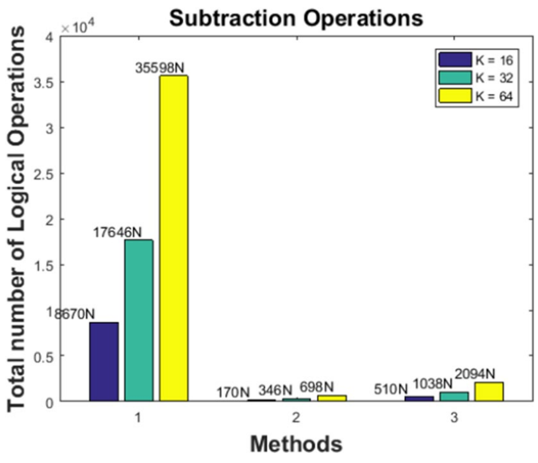

Fig. 14 Bar chart for Subtraction operations in Binary LTE Turbo codes

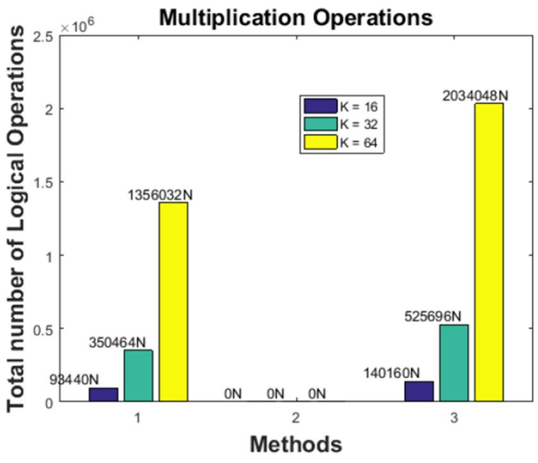

Fig. 15 Bar chart for Multiplication operations in Binary LTE Turbo codes

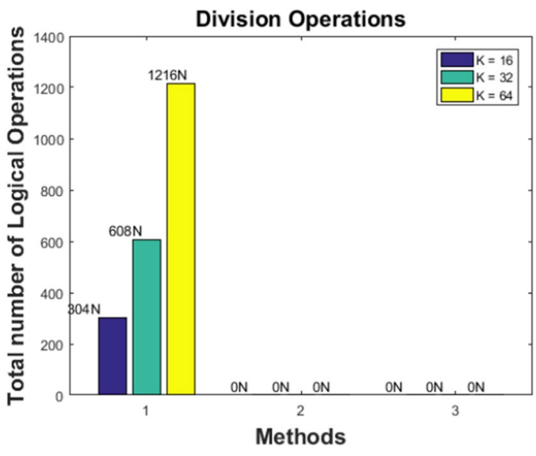

Fig. 16 Bar chart for Division operations in Binary LTE Turbo codes

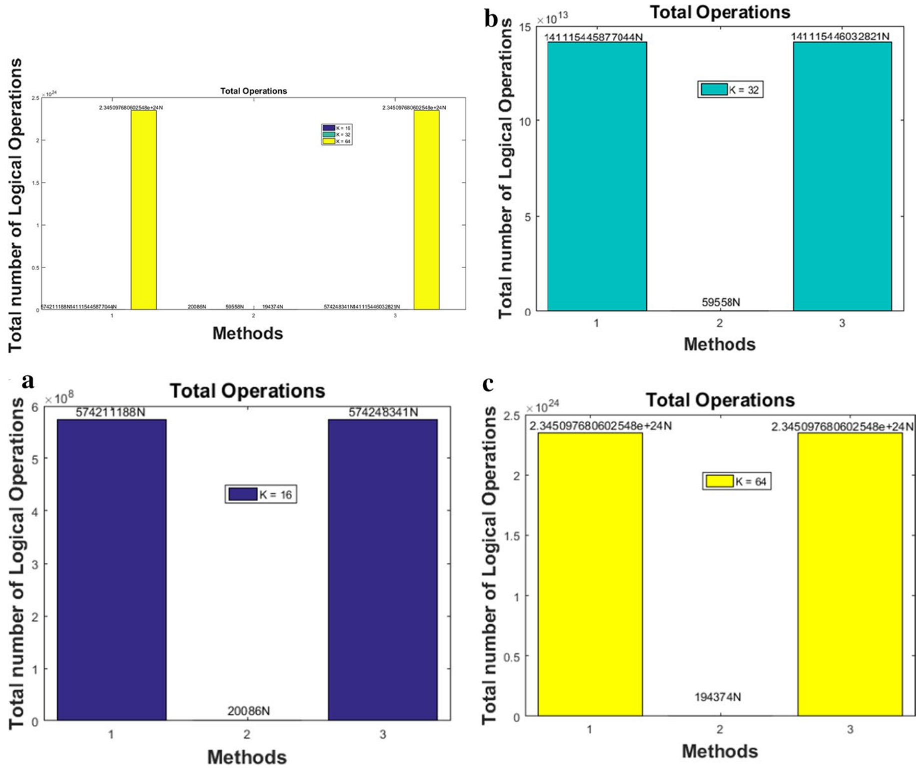

Fig. 17 Bar chart for Total operations in Binary LTE Turbo codes **a** K = 16, **b** K = 32, **c** K = 64

Method 2 with binary Turbo codes significantly reduces the complexity which in turn leads to a more energy efficient/power saving implementation. The energy efficiency of Turbo decoding Method 2 is supported by the error performance profiles of the 3 conventional Max-Log MAP decoding methods without the incorporation of any extrinsic information

Table 9 Total number of binary logical operations for binary LTE Turbo codes with SDR-based stopping and scaling with K = 16 and N = 6144

	Method 1	Method 2	Method 3
Log()	2720 N	0	0
Exp()	574,086,600 N	0	574,086,600 N
Max()	13,140 N	13,140 N	13,140 N
Add()	6631 N − 317	7093 N − 317	8248 N − 317
Sub()	9368 N + 698	868 N + 698	1208 N + 698
Mult()	93,440 N + 42,439	42,439	140,160 N + 42,439
Div()	304 N + 608	608	608
TOTAL	574,212,203 N + 43,428	21,101 N + 43,428	574,249,356 N + 43,428
Total	3.5280e+12	129,687,972	3.5282e+12

Table 10 Total number of binary logical operations for binary LTE Turbo codes with Regression-based scaling and stopping with K = 16 and N = 6144

	Method 1	Method 2	Method 3
Log()	5536 N	0	0
Exp()	1.4112e14 N	0	1.4112e14 N
Max()	45,396 N	45,396 N	45,396 N
Add()	13,825 N − 951	14767 N − 951	17122 N − 951
Sub()	20,438 N + 698	3138 N + 698	3830 N + 698
Mult()	477,781 N + 84,878	127,317 N + 84,878	653,013 N + 84,878
Div()	608 N + 608	608	608
TOTAL	1.4112e+14 N + 85,233		
Total	8.6704e+17	1.1712e+09	8.6704e+17

scaling and stopping technique. The BER graphs for the 3 decoding methods almost overlap over the whole E_b/N_0 region.

5.3 Total Computational Complexity with Scaling/Stopping

The total computational complexity analysis for binary LTE Turbo codes with SDR and Regression-based scaling and stopping mechanisms has been performed in this sub-section for K = 64 and N = 6144.

Table 9 shows the total number of binary logical operations for the 3 decoding approaches with SDR-based scaling and stopping. The same is shown in Table 10 for the 3 decoding methods with Regression-based stopping and scaling incorporated.

5.4 Comparative Analysis of Computational Complexity

The total number of logical computations with K = 16 and N = 6144 is given in Table 11.

The bar chart representation of the values in the above table is shown in Fig. 18.

Table 11 Total number of binary logical operations for binary LTE Turbo codes

	Conventional decoding	Decoding with SDR-based stopping and scaling	Decoding with regression-based stopping and scaling
Method 1	3.5280e+12	3.5280e+12	8.6704e+17
Method 2	123,408,384	129,687,972	1.1712e+09
Method 3	3.5282e+12	3.5282e+12	8.6704e+17

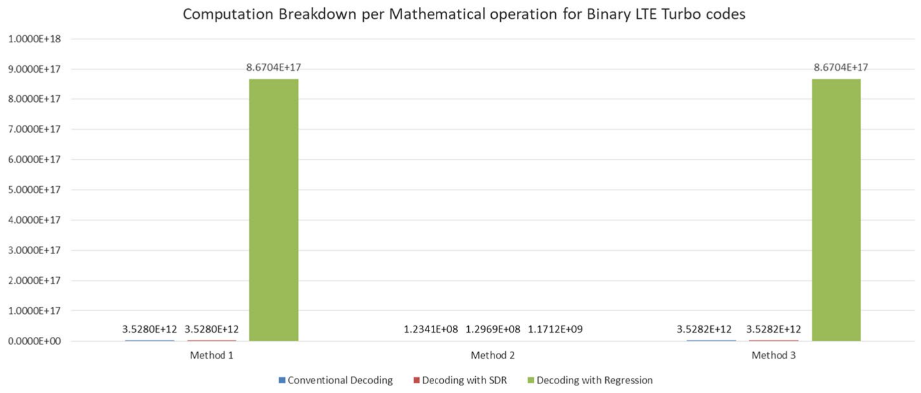

Fig. 18 Bar chart for total number of binary logical operations for binary LTE Turbo codes

The bars pertaining to Methods 1 and 3 reveal that SDR scaling and stopping does not increase the number of computations significantly as compared to the Regression-based stopping and scaling. The order in terms of the number of computations remains at 10^{12} when incorporating SDR-based stopping and scaling while the order goes up to 10^{17} when using Regression-based stopping and scaling. This rise in the number of computations is compensated by the significant improvement in error performance and reduced number of iterations over the E_b/N_0 range. Figure 19 is given below to better analyze the trend with Method 2.

The same trend is observed with Method 2. The order in terms of the number of computations remains at 10^8 when incorporating SDR-based stopping and scaling while when using Regression-based stopping and scaling, the order goes up 10^9. Despite the order of the total number of computations being lower with Method 2, the error performance graph reveals that the incorporation of Regression-based extrinsic information scaling and stopping with Method 2 causes an error-floor to occur at a BER of around 10^{-4}. SDR-scaling and stopping on the contrary is a good candidate to be used with Method 2 when considering the error performance, iterations profile and the trade-off pertaining to the increase in total number of computations.

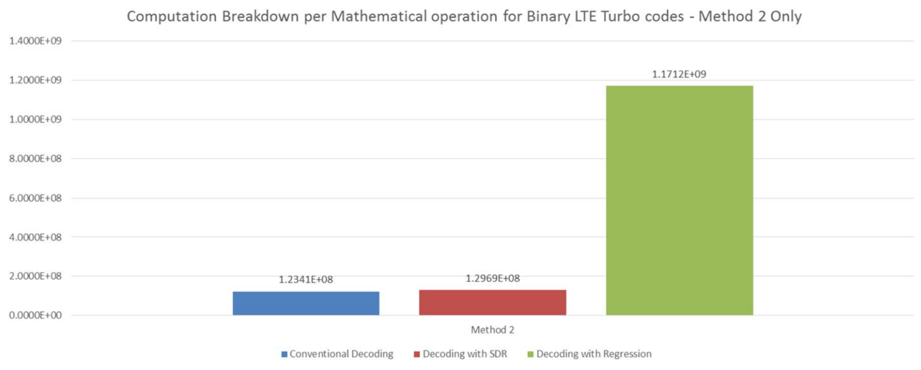

Fig. 19 Bar chart for total number of binary logical operations for binary LTE Turbo codes—Method 2 Only

6 Conclusion

The work presented in this paper is essentially the derivation and analysis of the computational complexity of different decoding algorithms for binary LTE Turbo codes. The evaluation of the computational complexity is performed in terms of binary logical operations for generalisation. Results demonstrate the variation in computational complexities when using different algorithms for Turbo decoding. When considering streams of 16 bits, Method 3 uses 0.0065% more operations in total as compared to Method 1. Furthermore, Method 2 uses only 0.0035% of the total logical complexity needed with Method 1. The incorporation of SDR and Regression-based scaling and stopping with the Turbo decoding Methods have also been simulated and analysed. When considering Methods 1 and 3, the order of the computational complexity remains at 10^{12} when using SDR with the provision of an average gain of 0.1 dB in error performance and a gain of 3 iterations at an E_b/N_0 value of 0.6 dB. The average gain in error performance is 0.3 dB when using the regression-based stopping and scaling. In terms of the iterations profile, gains of above 6 iterations are obtained at an E_b/N_0 value of 0.7 dB over the conventional decoding methods.

This work is important since the use of a decoding method with fewer number of binary logical operations significantly reduces the computational complexity which in turn leads to a more energy efficient/power saving hardware implementation. Also, evaluating the computational complexity in terms of the number of binary logical operations provides a more generalised mechanism to compare different decoding algorithms for error control codes in contrast to being limited to only complexities based on the types of hardware used.

Several future works can be foreseen from this work. A straightforward future work would be to perform a comparative analysis of the computational complexity in terms of the number of binary logical operations between Turbo codes and other error control codes for example non-binary Turbo codes and Low Density Parity Check Codes (LDPC). Another future work would be to formulate the exact mathematical expression for the Logarithm operation instead of assuming the maximum bound.

Acknowledgements The authors would like to thank the University of Mauritius for providing the necessary facilities in conducting this work.

Compliance with Ethical Standards

Conflicts of Interest The authors declare that they have no conflict of interest.

References

1. Hajiyat, A. R. M., Sali, A., Mokhtar, M., & Hashim, F. (2019). Channel coding scheme for 5G mobile communication system for short length message transmission. *Wireless Personal Communications, 2019*(106), 377–400.
2. Arora, K., Singh, J., & Randhawa, Y. S. (2020). A survey on channel coding techniques for 5G wireless networks. *Telecommunication Systems, 73,* 637–663.
3. Foukalas, F., & Tsiftsis, T. A. (2018). Energy efficient power allocation for carrier aggregation in heterogeneous networks: partial feedback and circuit power consumption. *IEEE Transactions on Green Communications and Networking, 2*(1), 1–1.
4. Mowla, M. M., Ahmad, I., Habibi, D., & Phung, Q. V. (2017). A green communication model for 5G systems. *IEEE Transactions on Green Communications and Networking, 1*(3), 264–280.
5. Ibañez, R., Abisset-Chavanne, E., Aguado, J. V., Gonzalez, D., Cueto, E., & Chinesta, F. (2018). A manifold learning approach to data-driven computational elasticity and inelasticity. *Archives of Computational Methods in Engineering, 25*(1), 47–57.
6. Pinto, R. N., Afzal, A., D'Souza, L. V., & Ansari, Z. (2017). Computational fluid dynamics in turbomachinery: A review of state of the art. *Archives of Computational Methods in Engineering, 24*(3), 467–479.
7. Miñano, M., & Montáns, F. J. (2018). WYPiWYG damage mechanics for soft materials: A data-driven approach. *Archives of Computational Methods in Engineering, 25*(1), 165–193.
8. Neggers, J., Allix, O., Hild, F., & Roux, S. (2018). Big data in experimental mechanics and model order reduction: Today's challenges and tomorrow's opportunities. *Archives of Computational Methods in Engineering, 25*(1), 143–164.
9. Mishra, P., & Ul Amin, M. (2020). *Performance evaluation for low complexity cascaded Sphere Decoders using best detection algorithm. ICT Express,* vol. In Press.
10. Nguyen, T. T. B., & Lee, H. (2019). Low-complexity multi-mode multi-way split-row layered LDPC decoder for gigabit wireless communications. *Integration, 65,* 189–200.
11. Kizil, C. H., Diou, C., Rabiai, M., & Tanougast, C. (2020) FPGA implementation of UWB-IR impulse generator and its corresponding decoder based on discrete wavelet packet. *AEU—-International Journal of Electronics and Communications, 114.*
12. Roberts, M. K. (2019). Simulation and implementation design of multi-mode decoder for WiMAX and WLAN applications. *Measurement, 131,* 28–34.
13. Mageswari, N., Mahadevan, K., & Kumar, M. K. (2019). An α-factor architecture for RS decoder implemented on 90 nm CMOS technology for computer computing applications devices. *Microprocessors and Microsystems, 71.*
14. Suaganthy, M., Karthikeyan, A., & Kuppusamy, P. G. (2019). Investigation of turbo decoding techniques based on lottery arbiter in 3D network on chip. *Microprocessors and Microsystems, 71.*
15. Son, J., Cheun, K., & Yang, K. (2017). Low-complexity decoding of block turbo codes based on the chase algorithm. *IEEE Communications Letters, 21*(4), 706–709.
16. Fayyaz, U. U., & Barry, J. R. (2013). A low-complexity soft-output decoder for polar codes. In *IEEE global communications conference (GLOBECOM)* , Atlanta, GA, USA.
17. Fayyaz, U. U., & Barry, J. R. (2014). Low-complexity soft-output decoding of polar codes. *EEE Journal on Selected Areas in Communications, 32*(5), 958–966.
18. Li, J., Wang, X., He, J., Su, C., & Shan, L. (2019). Turbo decoder design based on an LUT-normalized Log-MAP algorithm. *Entropy, 21*(8), 1–13.
19. Wu, P.H.-Y. (2001). On the complexity of turbo decoding algorithms. In *53rd IEEE vehicular technology conference*, Rhodes, Greece.
20. Yoon, S., Ahn, B., & Heo, J. (2020). An advanced low-complexity decoding algorithm for turbo product codes based on the syndrome. *EURASIP Journal on Wireless Communications and Networking, 2020*(126), 1–21.
21. Hassan, A., Dessouky, M. I., Abouelazm, A. E., & Shokair, M. (2012). Evaluation of complexity versus performance for turbo code and LDPC under different code rates, in *The fourth international conference on advances in satellite and space communications (SPACOMM)*, Chamonix/Mont Blanc, France.

22. Wu, Z., Gong, C., & Liu, D. (2017). Computational complexity analysis of FEC decoding on SDR platforms. *Journal of Signal Processing Systems, 89*(2), 209–224.
23. Mano, M. M. (2002). *Digital design* (3rd ed.). India: Pearson Prentice Hall.
24. Balasubramonian, R. (2015). Lecture 8: Binary Multiplication & Division. Available: https://www.cs.utah.edu/~rajeev/cs3810/slides/3810-08.pdf. [Accessed 14 April 2018].
25. Topsoe, F. (2004). Some bounds for the logarithmic function. *Research Report Collection, 7*(2), 1–20.
26. Berrou, C., Glavieux, A., & Thitimajshima, P. (1993). Near Shannon limit error-correcting coding and decoding: Turbo codes, In *IEEE Trans.*, 1993.
27. Beeharry, Y., Fowdur, T. P., & Soyjaudah, K. M. S. (2017). Performance Analysis of bit-level decoding algorithms for binary LTE turbo codes with early stopping. *Istanbul University—Journal of Electrical and Electronics Engineering (IU-JEEE), 17*(2), 3399–3415.
28. Lin, Y., Hung, W., Lin, W., Chen, T., & Lu, E. (2006) An efficient soft-input scaling scheme for turbo decoding" in *IEEE international conference on sensor networks, ubiquitous, and trustworthy computing*, Taichung, Taiwan.
29. Fowdur, T. P., Beeharry, Y., & Soyjaudah, K. M. S. (2016). A novel scaling and early stopping mechanism for LTE turbo code based on regression analysis. *Annals of Telecommunications, 71,* 369–388.

Dr. Yogesh Beeharry holds a BEng (Hons) Electronics and Communication engineering with First class honours from the University of Mauritius. He was also the recipient of the Mrs. L. F. Lim Fat Engineering Gold medal. He obtained a full-time MPhil/PhD scholarship from the Tertiary Education Commission and was awarded his PhD degree in Telecommunications Engineering in 2019. He is presently Lecturer in the Department of Electrical and Electronic Engineering at the University of Mauritius. His main research interests are: error control coding, Turbo codes, source and channel coding, big data real-time analytics, machine learning and deep learning algorithms.

Dr. T. P. Fowdur received his BEng (Hons) degree in Electronic and Communication Engineering with first class honours from the University of Mauritius in 2004. He was also the recipient of a Gold medal for having produced the best degree project at the Faculty of Engineering in 2004. In 2005, he obtained a full-time PhD scholarship from the Tertiary Education Commission of Mauritius and was awarded his PhD degree in Electrical and Electronic Engineering in 2010 by the University of Mauritius. He joined the University of Mauritius as an academic in June 2009 and is presently an Associate Professor at the Department of Electrical and Electronic Engineering of the University of Mauritius. His research interests include communications theory, multimedia communications, mobile and wireless communications, networking, security and internet of things. He has published over 60 papers in these areas and is actively involved in research supervision, reviewing of papers and also organising on international conferences.

Prof. Dr. K. M. S. Soyjaudah had a long and successful career at the University of Mauritius where he contributed significantly in academic research, namely in the fields of Communications Engineering and Computer Science and Engineering. He has about 220 research publications. He successfully supervised to completion two post-doctoral research fellows and 16 PhDs. His expertise in research is also regularly solicited by overseas universities for the examination of PhD theses. Professor Soyjaudah was the recipient of the UK Commonwealth Scholarship on two occasions, namely to read for a BSc(Hons) in Physics at Queen Mary College and for a MSc in Digital Electronics at Kings College, University of London. He holds a PhD in Digital Communication Engineering. He further read for a LLB (Hons) from London University and a PGD in legal studies at the bar from Manchester law school. He is a Barrister Member of the Middle Temple, London, UK. Professor Soyjaudah was the Chairman of Mauritius Qualifications Authority from 2002 to 2005. He was a Board Director of Multicarrier (Mauritius) Ltd from 2001 to May 2016 and a Board Member and Technical Expert in the Energy Efficient Management Office. Professor Soyjaudah was the Executive Director of the ICTA. He was also the Executive Director of the Tertiary Education Commission. Professor Soyjaudah is also a Senior Member of the Institute of Electrical and Electronics Engineers.