Received August 26, 2019, accepted September 8, 2019, date of publication September 18, 2019, date of current version October 3, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2942152

Performance Analysis of High Throughput MAP Decoder for Turbo Codes and Self Concatenated Convolutional Codes

FARZANA SHAHEEN[1], MUHAMMAD FASIH UDDIN BUTT[1], SHAHRUKH AGHA[1], SOON XIN NG[2], AND ROBERT G. MAUNDER[2]

[1]Department of Electrical and Computer Engineering, COMSATS University Islamabad, Islamabad 45550, Pakistan
[2]School of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, U.K.

Corresponding author: Muhammad Fasih Uddin Butt (fasih@comsats.edu.pk)

ABSTRACT The effect of parallelism on Bit Error Rate (BER) performance of Turbo Code (TC) and Self Concatenated Convolutional Code (SECCC) with different levels of parallelism and frame sizes is investigated. Next Iteration Initialization (NII) method is employed for mitigating the BER degradation resulting from increased parallelism. In order to analyze and compare the architectural performance of both schemes, this paper presents the Very High Speed Integrated Circuit Hardware Description Language (VHDL) design of Maximum Aposteriori Probability (MAP) decoder for TC and SECCC, both employing the same constituent code. The simulation results show that for BER of 10^{-4}, without parallelism, TC is 0.4 dB superior to SECCC, whereas, with parallelism of 64, the difference in performance between both schemes reduces to 0.25 dB. It is found that SECCC outperforms TC for frame sizes less than or equal to 2048 bits, when invoking a parallelism of 16, 32 and 64. The BER performance of both schemes shows that SECCC outperforms TC at parallelism of 256 by 0.3 dB at BER of 10^{-4}. Hence, for high throughput architectures employing higher parallelism (beyond 64 and 128) without significant degradation in BER performance, SECCC performs better than TC. The synthesis results of VHDL design of the MAP decoder obtained using Xilinx ISE verify that both schemes have equal clock frequency and resource consumption. It is demonstrated that the MAP decoder achieves the clock frequency of 86.3 MHz which is capable of producing a throughput of 691 Mbps using parallelism of 64.

INDEX TERMS MAP algorithm, high throughput, bit error rate, parallelism, SECCC, turbo codes, VHDL.

I. INTRODUCTION

Turbo Codes (TCs) were introduced in 1993 by Berrou [1]. They are Parallel Concatenated Convolutional Codes (PCCC) [2] which belong to Forward Error Correcting (FEC) codes. They are able to operate near Shannon's capacity limit [3] and hence, employed to support a variety of communication standards such as 3^{rd} Generation Partnership Project Long Term Evolution (3GPP LTE), IEEE Standard P802.16 also known as Worldwide interoperability for Microwave Access (WiMAX), Global System for Mobile communications (GSM), Universal Mobile Telecommunication System (UMTS), and Digital Video Broadcasting-satellite Services to Handheld (DVB-SH) [4].

The associate editor coordinating the review of this manuscript and approving it for publication was Jafar A. Alzubi.

For achieving a near capacity performance, complex decoding algorithms e.g., the Bahl-Cocke-Jelinek-Raviv (BCJR) algorithm were adopted [5]. The algorithmic complexity and the iterative nature of turbo decoder put a great challenge to the hardware designers for achieving their desired design goals, e.g., high-throughput, minimum latency, low complexity, low Bit Error Ratio (BER) and reduced power and energy consumption. However, there always exists a trade-off between these design goals. Hence, in the case of real-time communication system, if an optimal trade-off between these parameters does not exist, the decoder will exhibit undesirable performance.

Future communication standards demand for several Gbps data rates [6], in which multiple channel decoders will be operated in parallel in order to achieve high throughput targets. In addition, multiple code blocks in each transport block will be processed in parallel to meet the demands. The high throughput achieved by utilising parallel decoder architectures depends on several factors i.e., clock frequency, number of iterations, number of decoding units based on Maximum A posteriori Probability (MAP) algorithm and the technology used. Additionally, throughput increases linearly with an increase in the number of MAP decoding units, which results in increasing the resource utilization and chip area [7]. Several researchers have exploited the concept of parallelism to achieve high-throughput e.g., [8]–[11], some are focused on resource optimization [12], [13] and power reduction [14], [15] rather than throughput. A 3GPP-LTE advanced turbo decoder presented in [16] has used state-metric-initialization technique to reduce the latency of SISO decoder for achieving high throughput. A fully parallel decoding algorithm for TC was proposed in [17], which is a novel alternative to Log-BCJR algorithm. This algorithm is compatible with all TCs. It tends to increase throughput and reduce latency. Based on this algorithm, the implementation of a fully parallel turbo decoder was presented in [18]. High performance iterative algorithms have also been developed in [19] and [20] for Multiple Input Multiple Output Orthogonal Frequency Division Multiplexing (MIMO-OFDM) systems and 5G recievers, respectively. Recently, an arbitrary turbo decoder was presented in [21] to achieve higher processing throughputs and low latency, that uses rescheduling to avoid contention and enable parallelism of 128 and higher. Turbo decoder for achieving throughput of 100 *Gbps* is presented in [22] for higher code rates. A parallel turbo decoder architecture, which covers full range of code rates and provides higher throughput gains and better hardware efficiency, is presented in [23]. On the other hand, BER performance of an error correction code is affected by higher parallelism. The aspect that the BER performance of error correction codes decreases at higher parallelism is an important consideration for design implementation in high throughput scenarios. At higher parallelism levels, the input data block is subdivided into smaller blocks to be processed by the MAP decoders in parallelism levels. By dividing the data block into smaller sub-blocks, the size of sub-trellises become very small and hence, results in producing low BER performance. This BER degradation can be mitigated by performing more iterations. However, in order to reduce the performance loss with higher parallelism, two well-known techniques are adopted for parallel decoder design. One is based on Acquisition (ACQ) and the other is Next Iteration Initialization (NII) technique [24]. In ACQ, the state-metrics are initialized at the window boundaries [25] whereas, the NII method implicitly initializes the state-metrics over several decoding iterations. NII method is considered more preferable because of its less sensitivity to high code rates. However, in [26], the strengths of both methods are combined to obtain a high throughput and hardware efficient turbo decoder architecture.

In this work, the aim is to observe the performance of SECCC with short frame sizes and parallelism and compare it with TC. It is found that SECCC performs better than TC for short frame sizes and higher parallelism. The better performance of SECCC for short frame lengths and higher parallelism concludes the fact that in case of SECCC, single trellis is longer than each of the two trellises of TC, therefore SECCC performs better for smaller sized frames at higher parallelism. This work demonstrates the importance of code design and implementation for systems demanding high throughput. Besides BER performance, it is equally important to see the architectural performance of both schemes. Hence for the sake of enabling a complete comparison, the VHDL design of MAP decoder is also presented in this paper. The design is configured and synthesized for TC and SECCC to see the resource utilization and throughput for both schemes. Moreover, parallelism is important for producing high throughput. Hence, while comparing the BER performance of both schemes with parallelism, it is also important to see the architectural performance of parallel SECCC and TC decoder. However, the performance of architecture presented in this paper can be further optimized in future for specific applications.

	Acronyms
ACQ	Acquisition
AWGN	Additive White Gaussian Noise
BER	Bit Error Ratio
BPSK	Binary Phase Shift Key
CMI	Code Matched Interleaver
EXIT	EXtrinsic Information Transfer
FEC	Forward Error Correction
FER	Frame Error Rate
FPGA	Field Programmable Gate Array
FPTD	Fully Parallel Turbo Decoder
FSM	Finite State Machine
HSDPA	High Speed Downlink Packet Access
LLRs	Log Likelihood Ratios
LUT	Look Up Table
MAP	Maximum A posteriori Probability
MDU	MAP Decoding Unit
MIMO	Multiple Input Multiple Output
NII	Next Iteration Initialization
OFDM	Orthogonal Frequency Division Multiplexing
PCCC	Parallel Concatenated Convolutional Code
PWEP	Pair-Wise Error Probability
ROM	Read Only Memory
RSC	Recursive Systematic Convolutional
SECCC	Self Concatenated Convolutional Code
SISO	Soft Input Soft Output
SNR	Signal to Noise Ratio
SOVA	Soft Output Viterbi Algorithm
TC	Turbo Code
UXMAP	Iteration Unrolled XMAP
WEF	Weight Enumerating Function

SECCC belongs to PCCC and like some irregular TC [27], it is constructed by a single constituent code and exhibits a single MAP decoder [28]. The schematic diagrams of encoding and decoding process of SECCC scheme is shown in Fig. 3. Unlike the TC, a single component decoder in SECCC scheme exchanges extrinsic information iteratively with itself to achieve a desired performance. The SECCC scheme employing BPSK modulation was presented by [29], [30]. The SECCC scheme was further investigated for non-binary higher modulation in [31] and [32] to achieve

Symbols	
a	Fading amplitude of channel
α	Forward state Matrix
β	Backward state Matrix
c	Codeword
dB	Decibel
E_b	Bit energy
N_0	Noise power
f_{clk}	Clock frequency
γ	State transition matrix
Gbps	Giga bits per second
I	Iteration index
k	Size of message
K	Constraint length
L_c	Reliability value of channel
L_e	Extrinsic LLR
max	Maximum
Mbps	Mega bits per second
N	Frame/Block size
p	Parallelism level
R_1	Mother code rate
R_2	Puncturing rate
R	Code rate
u_k	Information bit at k instant
ν	memory of encoder
y_{kl}	Received symbols
π/π^{-1}	Interleaver/De-interleaver

bandwidth efficiency while iterative decoding is invoked to achieve power efficiency. The SECCC was further analyzed by [33] for its applications in power efficient cooperative communication schemes. However, after doing a thorough literature survey, it was found that the implementation of SECCC and its BER performance characteristics with parallelism has not yet been performed. Moreover, the performance comparison between TC and SECCC exhibiting the same convolutional code has not been done in the literature. The novel findings from the results obtained through Matlab simulations and FPGA synthesis of MAP decoder developed in VHDL for TC and SECCC are summarised below:

- We present BER characteristics of parallel SECCC decoder, which is reported for the first time in literature.
- On the basis of this parallel SECCC decoding scheme, we analyze the performance of bigger sized frame (6144-bits) as well as smaller sized frames (2048, 512 bits) with different levels of parallelism.
- As the SECCC scheme with parallelism does not already exist in literature, so for comparing its performance we develop the TC scheme with the same RSC code and code rate (as used for SECCC scheme) and show that SECCC scheme has a significant improvement in BER performance compared to TC scheme, for shorter frames ($\leq$ 2048-bits) with higher parallelism ($\geq$ 16).
- We also analyze the performance of bigger frame of 6144-bits at higher parallelism of 256, where SECCC shows significant improvement in BER performance.
- For the sake of enabling a complete comparison of both schemes, we also introduce the VHDL design of MAP decoder and configure it for SECCC and TC. Additionally, through synthesis of this configurable MAP decoder, it is shown that TC and SECCC schemes exhibit the same architectural performance.

The paper is organized as follows. The operating structures for encoding and decoding of TC and SECCC schemes are presented in Section II-A and II-B, respectively. Section II-C elaborates the mathematical model for Max-log-MAP algorithm. The design for MAP decoder and its parallel architecture is presented in Section III-A and III-B, respectively. Section IV-B presents the EXtrinsic Information Transfer (EXIT) chart for SECCC scheme and expressions for its union bound analysis whereas the simulation results of TC and SECCC with different frame sizes and parallelism are discussed in Section IV-A. The synthesis results are provided in Section IV-C. Finally, the conclusions and future work are presented in Section V.

II. ENCODING AND DECODING OF TC AND SECCC

The process of encoding is presented in Section II-A, where the construction of TC and SECCC is discussed employing the same generator polynomial and code rate. The decoding process is explained in Section II-B and the Max-Log-MAP algorithm which has been used to implement the decoder is presented in Section II-C.

A. ENCODING

The construction of TC and SECCC encoders is discussed in this section. Fig. 1(a) shows a turbo encoder, constructed by the parallel concatenation of at least two RSC codes, which are same and connected through an interleaver [34]. The interleaver counteracts the effects of bursts errors and thus enhances the error correcting capabilities of the FEC code. The interleaver scrambles or re-arranges the encoded symbols over multiple code blocks with no repetition and spreads out long noise burst sequences. The scrambled information is provided to the second component decoder and the uncorrelated information exchange between the two component decoders is facilitated. The interleavers may be periodic

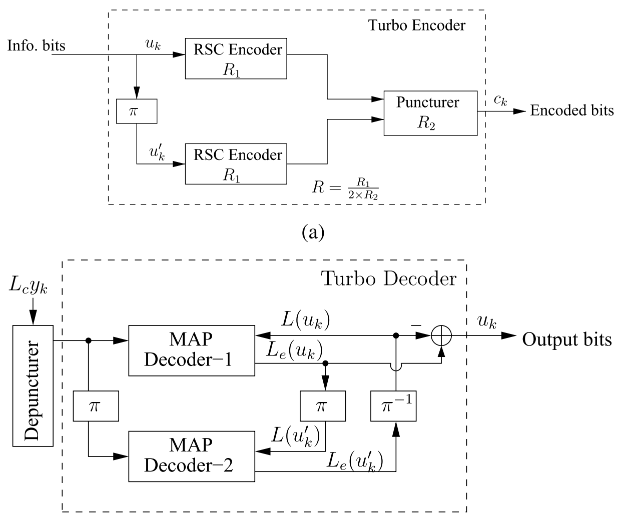


FIGURE 1. Turbo encoder and decoder [1]. (a) Turbo encoder. (b) Turbo decoder.

or pseudo-random. Periodic interleavers are classified into block and convolutional interleavers. Block interleaver has good performance for non-puncturing small code lengths but for large code lengths random interleavers perform better both for puncturing and non-puncturing codes [35]. Since, we are using puncturing in our example component codes, we have a pseudo-random interleaver. The more "scrambled" the interleaver is, the more "uncorrelated" the information exchange is. The main role of interleaver is to eliminate low weight input patterns which contribute significantly to the error probability. The Code Matched Interleaver (CMI) [36] is an optimum interleaver, which breaks several low weight input sequences depending on the component codes. This CMI can effectively eliminate several spectral lines of the original distance spectrum and increase the overall Hamming distance of the code. Consequently, the code performance at high Signal to Noise Ratio (SNR) is improved and the error floor is lowered [37]. However in this paper, we have not focused on the interleaver design and we employ a pseudo-random interleaver, which would give an average performance.

Fig. 1(a) depicts the block diagram of a turbo encoder. The Recursive Systematic Convolutional (RSC) codes are mostly used as the constituent component codes. At any time k, the input to the encoder is a bit u_k, which is converted to the corresponding code bit c_k based on the generator polynomials. The structure and function of the encoder is determined by the generator polynomials and the constraint length which affect the distance properties and the error correcting capability of a convolutional code. Hence, the combination of generator polynomials should be optimum to maximize the minimum free distance of the code for achieving a good error correction performance of the code [38], [39]. In this paper, an RSC component encoder having constraint length K $= 4$, memory $\nu = 3$, with $(13)_8 \rightarrow (\mathbf{1011})$ as feedback generator polynomial and $(15,\ 17)_8 \rightarrow (\mathbf{1101},\ \mathbf{1111})$ as feedforward generator polynomial is considered as an example, as shown in Fig. 2(a). However, the RSC code with $\nu = 4$ is also a good option to obtain better performance [40]. The state-transition diagram and 8-state trellis with a minimum free distance for RSC component encoder are shown in Fig. 2(b) and Fig. 2(c), respectively. As given in [41], the codeword can be generated from:

$$c_k = \sum_{\substack{i=0\\ l=1}}^{\substack{l=n\\ K-1}} g_{li} u_{k-i} \ \mathbf{modulo}\ 2, \tag{1}$$

where g_{li} represents the i^{th} bit from binary representation of the l^{th} feedforward generator polynomial, K is a constraint length and u_k is an information bit at any time k, may be a 0 or 1. In some examples, the encoded bits are punctured at rate $R_2 = \frac{1}{2}$ to produce a rate $R = \frac{R_1}{2 \times R_2} = \frac{1}{3}$ [42]. The coding rate of the RSC encoder is $R_1 = \frac{1}{3}$, Hence the coding rates of both the TC and SECCC encoders are $\frac{1}{6}$ before puncturing. To achieve a final coding rate of $R = \frac{1}{3}$

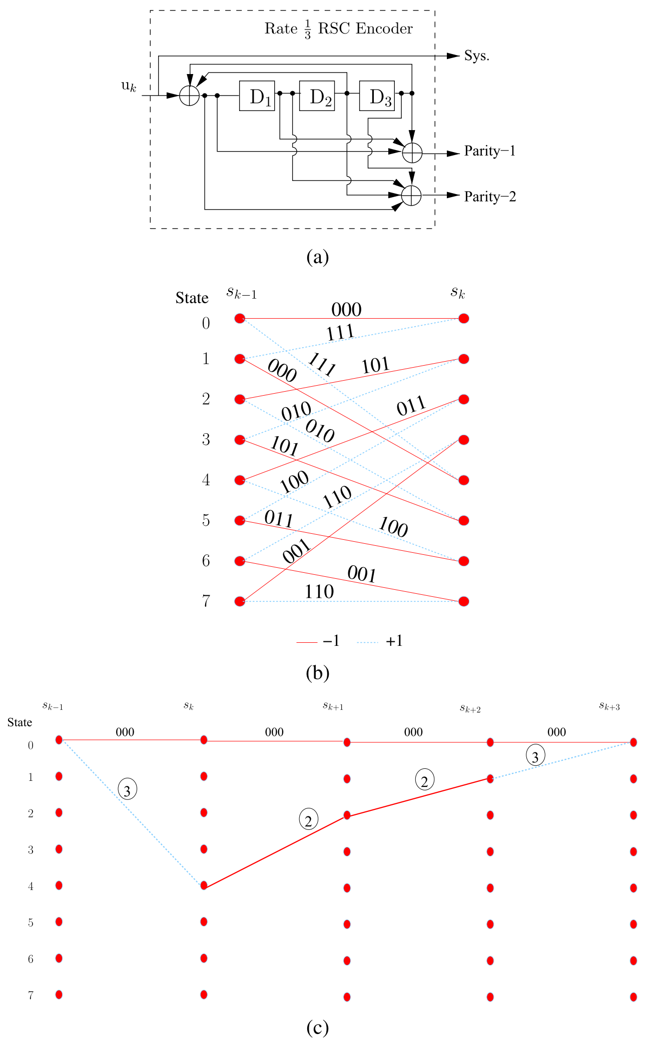


FIGURE 2. **Example of component encoder with $(13)_8$ and $(15,\ 17)_8$ feedback and feedforward generator polynomials [33]. (a) RSC Encoder. (b) State transition diagram for 8-state RSC code. (c) Minimum free distance of 8-state trellis for RSC encoder.**

a puncturing rate of $R_2 = \frac{1}{2}$ was invoked, where half of the coded bits are punctured. The SECCC encoder is shown in Fig. 3(a), we used the same code rate and puncturing for SECCC. The information bits u_k and their interleaved version u'_k are converted to a serial stream, which is now fed to the RSC encoder. The encoded bits are then punctured at rate $\frac{1}{2}$ to produce a code rate of $\frac{1}{3}$.

B. DECODING

The decoder for TC is shown in Fig. 1(b). It comprises of two SISO decoding units connected in parallel to each other through interleaver and deinterleaver. The un-interleaved version of the channel Log Likelihood Ratios (LLRs), produced by the first encoder in Fig. 1(a), is received by first MAP decoder. Since, the bits are punctured at $R_2 = \frac{1}{2}$ to achieve a code rate $R = \frac{1}{3}$, the depuncturer inserts zeros in the places of punctured bits. The extrinsic LLRs produced in the first half iteration by the first MAP decoder is appropriately

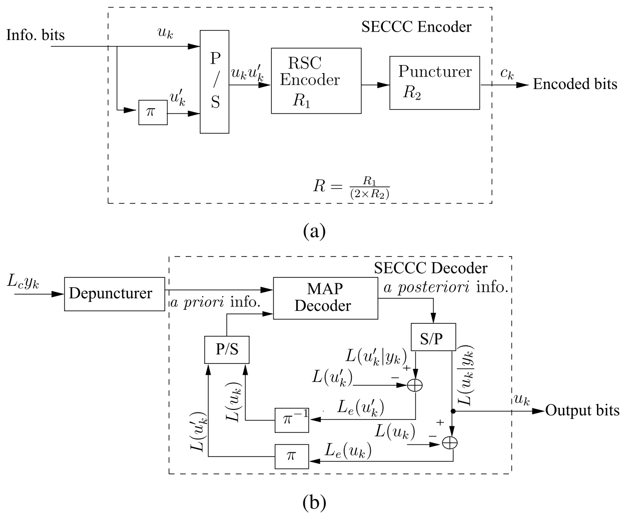


FIGURE 3. **SECCC encoder and decoder. (a) SECCC encoder. (b) SECCC decoder.**

interleaved and along with the interleaved version of channel LLRs, which are produced by the second encoder in Fig. 1(a), are processed by the second MAP decoder to produce *a posteriori* information as a result of the iteration. For the second iteration, the extrinsic information produced by the second MAP decoder is properly de-interleaved and along with the channel LLRs is fed to the first MAP decoder. This iterative process continues for certain number of iterations to achieve the desired BER performance. SECCC scheme comprises of a single RSC encoder and a single MAP decoder, as shown in Fig. 4. Unlike TC, in SECCC scheme, the component decoder produces the extrinsic information and exchanges it with itself for a specific number of iterations to achieve a desired BER performance. SECCC is near in performance to TC. Fig. 3(b) elaborates the concept of SECCC decoding in which the output of the MAP decoder is converted to two parallel streams which need to be appropriately interleaved/de-interleaved. These two parallel streams are again merged as one serial sequence and fed back to the MAP decoder. The same becomes *a priori* information for the next iteration. For understanding the decoding process, it would be helpful to define and understand the following important terms:

- ***a priori* information**: This is also called an intrinsic information which is denoted by $L(u_k)$. It is the known information about a bit before commencing the decoding process.
- ***a posteriori* LLR**: This is the output of component decoder, which it generates with all available information about the concerned bit u_k. It is denoted by $L(u_k|y)$.
- **extrinsic information**: This information is denoted by $L_e(u_k)$ and obtained by excluding the $L(u_k)$ (*a priori* information) from the $L(u_k|y)$ (*a posteriori* output) as depicted in Fig. 1(b) and Fig. 3(b). After being interleaved or de-interleaved $L_e(u_k)$ is sent as an *a priori* information to the other component decoder to generate more refined LLR for bit u_k in the next half iteration.
- **Iteration**: One iteration for turbo decoder completes when first component decoder produces an extrinsic information (for the original information sequence of the channel LLRs alone) and provides it to the second component decoder, where it is utilized as *a priori* information along with the channel LLRs to produce *a posteriori* information. In this paper, an iteration for the MAP decoder is defined as the minimum number of MAP algorithm operations that are repeated. According to this definition, iteration means invoking the MAP-algorithmic unit only once, and this is equal to one SECCC iteration, as presented in Fig. 4(a). The iteration for TC is depicted in Fig. 4(b), which is equivalent to two SECCC iterations. However, throughout this paper, an iteration is defined as one SECCC iteration.

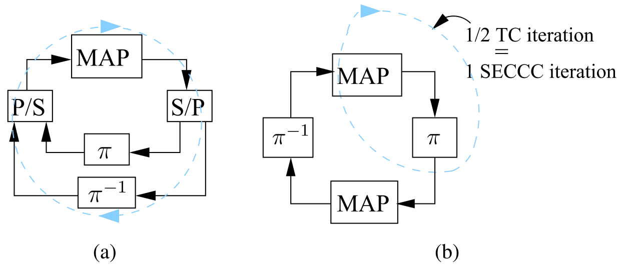


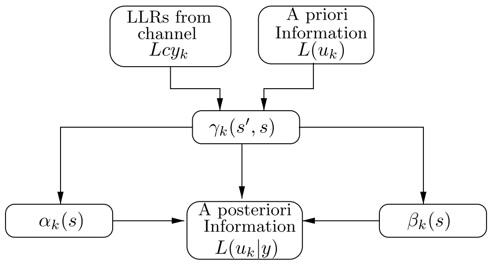
FIGURE 4. **Iteration for TC and SECCC. (a) Iteration for SECCC. (b) Iteration for TC.**

C. MAX-LOG-MAP ALGORITHM

Soft Output Viterbi Algorithm (SOVA) [43] and MAP algorithm [5] are the two well-known algorithms used for decoding TCs. SOVA was considered to be the preferred decoding algorithm for decoding block and convolutional codes. Unlike SOVA, MAP algorithm is considered to be optimal but more complex. However, in Log-MAP algorithm, the multiplications and additions of the MAP algorithm are replaced with additions and max* operations (or with max operations in the Max-log-MAP algorithm). In comparison to SOVA, Log-MAP algorithm is three times more complex whereas the complexity of Max-log-MAP algorithm is twice as that of SOVA [37]. Both these algorithms, at any time step k and for all the trellis paths which enter each state at that time step, calculate the measure of similarity or distance between the transmitted and received symbols, along the transmission length. The SOVA selects only the maximum likelihood paths at any time step, also known as the surviving paths and discards the least likely paths.

On the other hand, the MAP algorithm inspects every possible path at any time step along the whole trellis, divides these paths into two groups (one for bit ‘0’ and the other for bit ‘1’) and then calculates the log-likelihood values for all of these two sets. However, the MAP algorithm minimizes the probability of an incorrect path through the trellis and provides the estimated bit sequence as well as the probabilities for each correctly decoded bit [44]. Although the original MAP algorithm is optimal but it is computationally more intensive as it involves multiplications, additions and exponentials (non-linear functions). The flow of operations followed in the MAP algorithm is shown in Fig. 5. However, for the purpose of reducing computational complexity, the original MAP algorithm was first simplified to Log-MAP algorithm and then to Max-log-MAP algorithm [45] with a little sacrifice on BER performance. Since, Log-MAP algorithm is as optimal as MAP algorithm, but is less complex than MAP algorithm due to its operations in log domain. However, the Max-log-MAP algorithm is the sub-optimal version of MAP algorithm and it reduces the complexity of Log-MAP algorithm further by using the maximization approximation. Max-log-MAP algorithm because of its less complexity, is widely used in turbo decoder implementations, but with a degradation in BER performance. However, this performance degradation can be compensated by using extrinsic scaling factor [46]. We have used a scaling factor of 0.75 in our BER simulations which has improved the BER performance by 0.3 dB over the standard Max-log-MAP algorithm at 10^{-4} [47], [48]. This algorithm calculates the *a posteriori* LLR $L(u_k|y)$ of each bit u_k by considering only the two best transitions. No trellis termination is used here. The state transition metric is calculated, as given in [44]:

$$\gamma_k(s', s) = \frac{1}{2}u_k.L(u_k) + \frac{L_c}{2}y_{ks}u_k + \chi_k(s', s). \tag{2}$$



LLRs from channel Lcy_k
A priori Information $L(u_k)$
$\gamma_k(s', s)$
$\alpha_k(s)$
A posteriori Information $L(u_k|y)$
$\beta_k(s)$

FIGURE 5. **The MAP algorithm [44].**

The first term in (2) is the *a priori* probability, whereas y_{ks} in the second term represents the received systematic channel LLRs. The third term $\chi_k(s', s)$ is equal to $\frac{L_c}{2}\sum_{l=2}^{n} y_{kl}x_{kl}$ and is used to calculate the state transition metrics for the two parity bits. Similarly, the recursive calculation of forward state metric denoted by $\alpha_k(s)$ and the backward state metric denoted by $\beta_{k-1}(s)$ is performed by (3) and (4), respectively [44]. $\gamma_k(s', s)$ in (3) and (4) is the corresponding state transition metric.

$$\alpha_k(s) = \max_{all\ s'}\{\alpha_{k-1}(s') + \gamma_k(s', s)\}. \tag{3}$$

$$\beta_{k-1}(s') = \max_{all\ s}\{\beta_k(s) + \gamma_k(s', s)\}. \tag{4}$$

The initial conditions for (3) and (4) are given below:

$$\begin{cases} \alpha(0) = 1 \\ \alpha(s) = 0 \quad \text{for } s \neq 0. \end{cases} \tag{5}$$

$$\beta(s) = \frac{1}{2^{\nu}} \quad \text{for all } s. \tag{6}$$

The *a posteriori* LLRs $L(u_k|y)$ for each bit u_k is calculated by using (7)

$$L(u_k|y) = \max_{(s,s')\Rightarrow u_k=+1}\{\alpha_{k-1}(s') + \gamma_k(s', s) + \beta_k(s)\} - \max_{(s,s')\Rightarrow u_k=-1}\{\alpha_{k-1}(s') + \gamma_k(s', s) + \beta_k(s)\}. \tag{7}$$

The extrinsic information is calculated by (8)

$$L_e(u_k) = L(u_k|y) - L(u_k) - L_c y_{ks}, \tag{8}$$

where $L(u_k|y)$ (*a posteriori* LLR) is produced by the component decoder, $L(u_k)$ is the *a priori* LLR which is initially zero in the logarithmic domain, whereas in iterative decoding process, an estimate of $L(u_k)$ is provided by each component decoder to the other component decoder. $L_c y_{ks}$ represents the received soft LLR for the systematic bit u_k from the channel demodulator. The calculation of extrinsic LLR $L_e(u_k)$ depends on the constraints imposed by the component code used. $L_c y_{ks}$ and $L(u_k)$ are subtracted from $L(u_k|y)$ to produce the extrinsic LLR $L_e(u_k)$ for the systematic bit u_k and not for the parity bit. $L_e(u_k)$ is then appropriately interleaved/deinterleaved and become as *a priori* for the next iteration.

III. DECODER ARCHITECTURE

The Max-Log-MAP algorithm has been used to develop the VHDL design of MAP decoder which can be configured for both TC and SECCC decoder. Multiple MAP decoder units are capable of operating with different parallelism to obtain the parallel MAP decoder architecture. The MAP decoder and parallel MAP decoder architectures are explained in Section III-A and III-B, respectively.

A. ARCHITECTURE OF THE MAP DECODER

Fig. 6(a) depicts the block diagram of a general architecture of a turbo decoder with some additional units added to that shown in [49]. The architecture contains the program interface, control unit, input buffer, address generator, MAP Decoding Unit (MDU) and output buffer. MDU comprises computational units for α, β, γ and LLRs, as is shown in Fig. 6(b). As already explained above, turbo decoder has at least two Soft-In-Soft-Out (SISO) algorithmic units which are connected to each other in parallel through interleaver and deinterleaver. However, due to hardware re-usability, same MDU can be configured through a control mechanism to be used alternatively as an inner and outer decoding component. In first half iteration, the encoded bits (which are now channel LLRs) produced by the upper encoder of Fig. 1(a) are processed by the MDU and the extrinsic LLRs produced from these bits are stored in intermediate memory. Now, in the second half iteration, the MDU starts processing the second set of LLRs produced by the lower encoder, using the de-interleaved version of previously generated extrinsic LLRs as *a priori* information. Max-log-MAP algorithm has been used as a SISO decoding algorithm for most of the implementations in the literature. The *max* operation in this algorithm

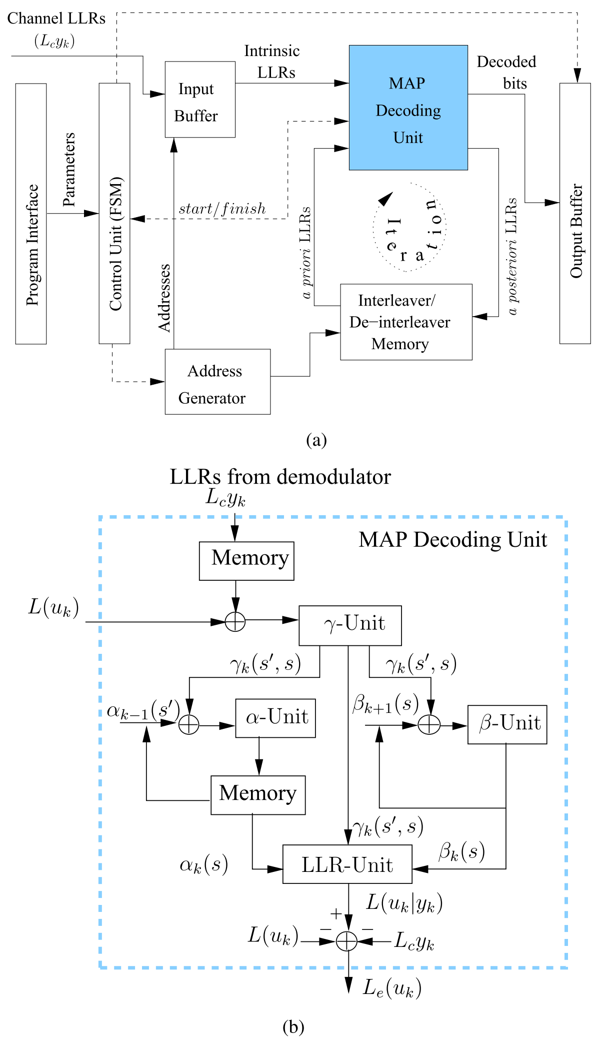


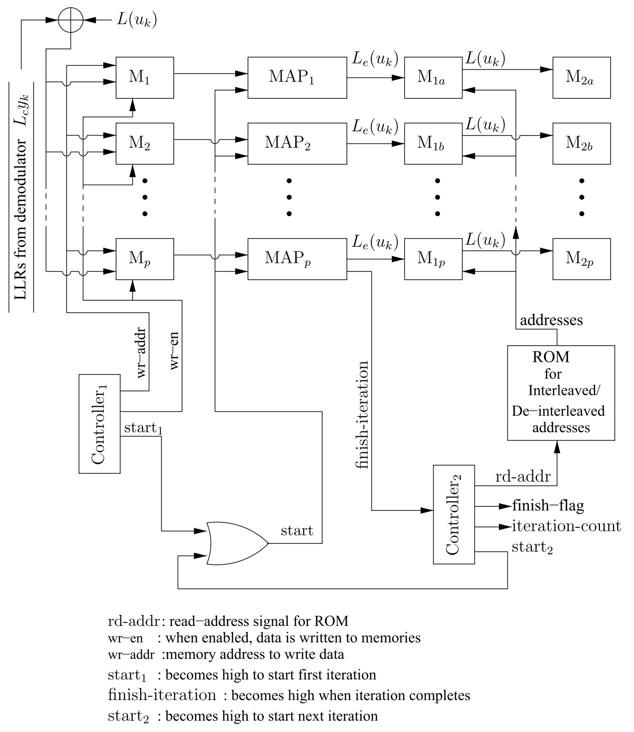
FIGURE 6. Block diagram of MAP decoder and architecture of MAP decoding unit (MDU) [49]. (a) Block diagram of MAP decoder. (b) Architecture of MDU.

is the approximation of the logarithm of exponential terms i.e., $ln(e^{\lambda_1} + e^{\lambda_2}) \approx \max(\lambda_1, \lambda_2)$. The *max* operation is performed using compare and select sequence of steps. The interleaver/deinterleaver uses the same permutation pattern as used at the encoder side. The system parameters e.g., frame size, number of iterations, code and puncture rates are passed to the control unit through the program interface. The control mechanism is implemented in the form of a Finite State Machine (FSM) for coordination among computational and storage units based on the received parameters. These control signals are represented by dash-lines in Fig. 6(a). The LLRs of the encoded bits from the channel are stored in the input buffer, which are then fed to the MDU. The control unit sends a *start* signal to MDU to initiate the decoding process. After completing first half iteration, a *finish* signal is generated by the control FSM for the address generator, which reads the interleaver/de-interleaver memory block to send the correct extrinsic information to the MDU for starting the next half iteration. Meantime, while the decoding unit is processing one block, the second input block is received by the input buffer. The MDU performs the decoding process as already stated above. The extrinsic LLRs produced in the first half iteration are written in memory in interleaved manner and read by the decoding unit in de-interleaved manner. We have used a pseudo-random interleaver in our design. The permutation indexes random pattern are generated in Matlab and with these permutation indexes, ROM is defined in VHDL [50]. Finally, after a fixed number of iterations, the control FSM generates a signal to load the decoded bits in the output buffer.

The architecture shown in Fig. 6 can be configured to build TC and SECCC decoders. The main difference between TC and SECCC is that in case of TC, the single physical MAP decoder can be separated into two virtual decoders, but in SECCC, the single decoder is not separable. However, the architecture presented in Fig. 6 offers the same algorithmic and architectural complexity to both TC and SECCC schemes for the same number of iterations.

B. ARCHITECTURE OF PARALLEL MAP DECODER

The parallel MAP decoder architecture is presented in this section and its block diagram is shown in Fig. 7. A number of parallel MAP decoder architectures with different parallelism levels for TCs have been presented in the literature [8], [51]–[59]. The parallel-MAP decoder shown in Fig. 7 can be configured for any level of parallelism $p \in \{2, 4, 8, 16, 32, 64\}$ for decoding frames of sizes divisible by the level of parallelism. A sub-block with size $\frac{N}{p}$ - bits is received



$L(u_k)$
LLRs from demodulator $L_c y_k$
M_1
M_2
M_p
MAP_1
MAP_2
MAP_p
$L_e(u_k)$
M_{1a}
M_{1b}
M_{1p}
$L(u_k)$
M_{2a}
M_{2b}
M_{2p}
addresses
ROM for Interleaved/ De−interleaved addresses
wr−addr
wr−en
Controller$_1$
start$_1$
finish−iteration
start
Controller$_2$
rd-addr
finish−flag
iteration−count
start$_2$
rd-addr: read−address signal for ROM
wr−en : when enabled, data is written to memories
wr−addr :memory address to write data
start$_1$: becomes high to start first iteration
finish-iteration : becomes high when iteration completes
start$_2$: becomes high to start next iteration

FIGURE 7. Parallel MAP decoder architecture for TC and SECCC.

and decoded by each MAP decoder, where, N is the size of frame and p shows the level of parallelism, thus reducing the decoding delay for each iteration [60]. The parallel decoder architecture shown in Fig. 7 contains a stack of p MAP decoders, having three storage blocks or memories where each memory contains p sub-memories. For example, the first memory contains sub-memories from M_1 to M_p which store the input *a priori* LLRs, the second has sub-memories from M_{1a} to M_{1p} which store extrinsic information $L_e(u_k)$, and the third has sub-memories from M_{2a} to M_{2p} which store the interleaved and de-interleaved information. For synchronizing and routing data between MAP units and memories, there are two controllers named Controller$_1$ and Controller$_2$. The random permutation pattern of addresses are saved in ROM in order to address the memory collision problem [50]. To start the decoding process, the p number of start$_1$ signals generated by Controller$_1$ are received by the corresponding MAP decoders. When all the MAP decoders produce extrinsic information L_e after the first iteration, the MAP$_p$ decoder sends a finish-iteration signal to Controller$_2$. A *rd-addr* signal is generated by Controller$_2$ for reading randomly permuted interleaved and de-interleaved addresses from the ROM. These addresses are then given to sub-memories M_{1a} to M_{1p} for fetching the relevant extrinsic information from them. The fetched information is stored in sub-memories M_{2a} to M_{2p} and is then utilized as *a priori* information for the next iteration. Controller$_2$ generates the *start*$_2$ signal for all the MAP units to initiate the next iteration. A similar procedure is adopted to run each iteration, and after a desired number of iterations, Controller$_2$ sets the finish-flag high.

IV. RESULTS AND DISCUSSIONS

The error correction performance of channel codes can be analyzed by EXIT charts, union bounds or by plotting BER against $\frac{E_b}{N_0}$ to find the minimum value of $\frac{E_b}{N_0}$ suitable for reliable communication, where E_b is the information bit energy and N_0 is the noise variance. The EXIT charts and the expression for calculation of union bounds of SECCC scheme with BPSK modulation and AWGN channel are given in Section IV-A. The simulations are carried out in Matlab to evaluate the BER performance of TC and SECCC decoder. The simulation results with different frame sizes and parallelism are presented in Section IV-B. In order to measure the architectural performance of TC and SECCC decoder, the VHDL design of the MAP decoder discussed in Section III is configured for TC and SECCC and synthesized by using Xilinx ISE for Virtex-6 FPGA (XCVLX240T). Finally, IV-C presents the synthesis results.

A. EXTRINSIC INFORMATION TRANSFER (EXIT) CHARACTERISTIC AND UNION BOUNDS FOR SECCC SCHEME

EXIT charts proposed by ten Brink [61] are helpful to predict the convergence behavior of iterative decoder. EXIT charts plot the resulting extrinsic information characteristics of constituent decoders into a single diagram where both curves are the mirror images of each other. Convergence is only possible if the transfer characteristics do not intersect. The convergence estimates the average number of required decoding steps or iterations.

EXIT chart analysis for different code rates and modulation schemes of High Speed Downlink Packet Access (HSDPA) turbo decoder is presented in [62], whereas for various SECCC schemes, the EXIT chart analysis is presented in [33] and [63].

The EXIT charts for SECCC scheme based on Log-MAP decoder are shown in Figs. 8 and 9 at 1 dB and 1.7 dB, respectively, with code rate R $= \frac{1}{3}$, $\nu = 3$ and AWGN channel for interleaver size of 20000-bits. Fig. 8 also shows the trajectories (snapshot number 2 and 18) of SECCC iterative decoding at $\frac{E_b}{N_0} = 1$ dB. The trajectories at this value of $\frac{E_b}{N_0}$ can pass through the tunnel and reach the (1,1) convergence point by increasing the number of iterations to 40. Since, high number of iterations result in increasing hardware complexity and decrease the decoding speed, therefore system is configured to operate at 8 decoding iterations. Here, I_A and I_E represent *a priori* and extrinsic mutual information of the bit stream, respectively. Furthermore, Fig. 9 also shows that the trajectories (snapshot number 2 and 18) in EXIT chart reach the (1,1) convergence point with 8 decoding iterations at relatively higher $\frac{E_b}{N_0}$ value. In order to reduce the decoding complexity, we have considered the Max-log-MAP algorithm (with a scaling of 0.75) in our simulation results shown Fig. 10. More specifically, the BER of the SECCC has started to converge to a low value at $\frac{E_b}{N_0} = 1.8$ dB (instead of 1.7 dB, as predicted by its EXIT chart in Fig. 9) when the Max-log-MAP algorithm is employed in the decoder. This small performance loss of 0.1 dB is due to the employment of

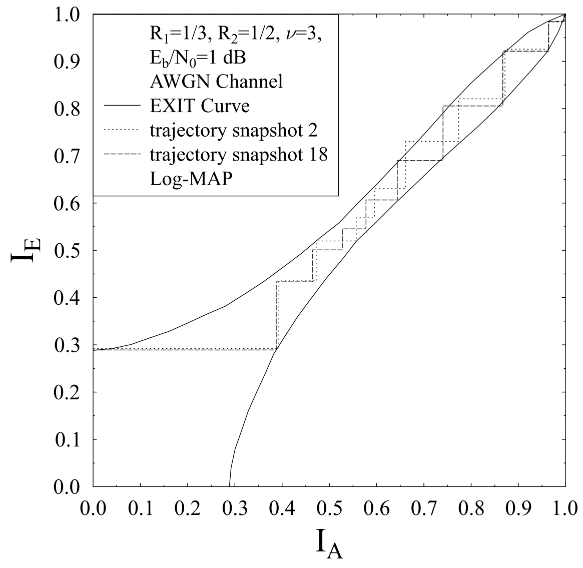


FIGURE 8. EXIT chart for $\frac{1}{3}$ SECCC code at $\frac{E_b}{N_0} = 1$ dB and 40 iterations.

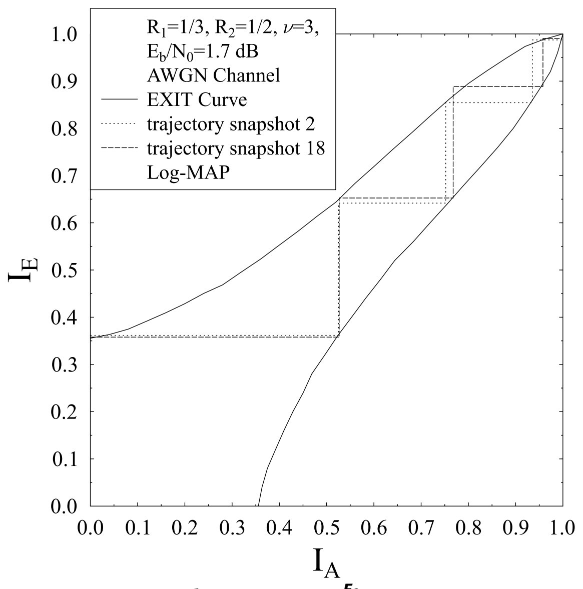


FIGURE 9. EXIT chart for $\frac{1}{3}$ SECCC code at $\frac{E_b}{N_0}$ = 1.7 dB and 8 iterations.

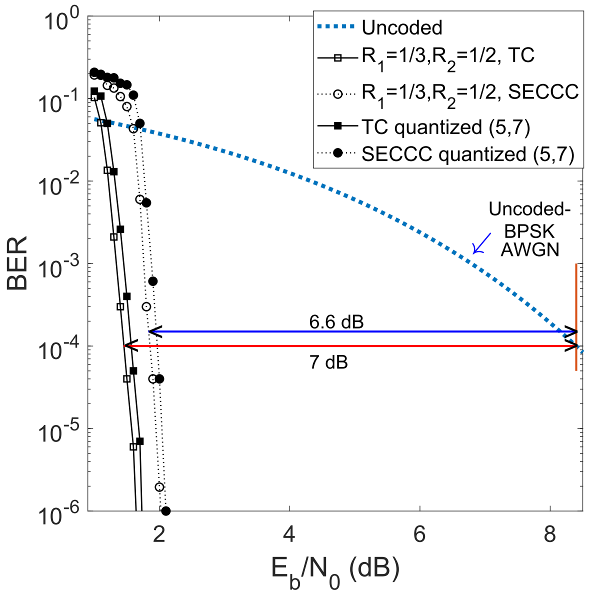


FIGURE 10. BER performance of rate $\frac{1}{3}$ Uncoded, TC and SECCC at 8 iterations with 100 frames of 20,000 bits, Max-log-MAP algorithm with scaling factor 0.75, without parallelism when communicating over AWGN channel.

the Max-log-MAP instead of the Log-MAP algorithm in the BER simulation [47].

EXIT charts can estimate the BER floors for considerably large interleaver sizes. Another technique to determine the BER floor is the truncated union bound analysis, which can be employed for arbitrary interleaver sizes. The truncated union bound analysis for SECCCs with uniform interleaver has been presented in [28]. It facilitates to design various SECCCs for desired BER floor. The following relation expresses the union bound for the average BER of a channel code, as given in [64]:

$$P_b \leq \frac{1}{k} \sum_{\Delta_H} B_{\Delta_H} P(\mathbf{x} \longrightarrow \hat{\mathbf{x}}), \tag{9}$$

where, $P(\mathbf{x} \longrightarrow \hat{\mathbf{x}})$ denotes the Pair-Wise Error Probability (PWEP) and for AWGN channel: $P(\mathbf{x} \longrightarrow \hat{\mathbf{x}}) = Q(\sqrt{2\frac{E_b}{N_0}\Delta_H})$. Here, $\mathbf{x}$ is the encoded sequence without errors whereas, $\hat{\mathbf{x}}$ is the erroneous encoded sequence. B_{Δ_H} is the distance spectrum of the code and $B_{\Delta_H} = \sum_w \frac{w}{N}.A_{w,\delta}$, where $A_{w,\delta}$ is the Weight Enumerating Function (WEF) and represents the average number of error events in the sequence with w and δ showing the number of erroneous systematic and erroneous parity bits, respectively. Δ_H is the effective hamming distance.

As we know that in case of SECCC scheme, to complete one turbo decoding step, one MAP decoder has to iterate with itself twice, so we consider two hypothetical MAP decoders. Hence, the WEF for SECCC is defined as follows [28]

$$A_{w,\delta} = A^{(1)}_{2w,\delta^{(1)}}.A^{(2)}_{2w,\delta^{(2)}}.P^{N,w}_{\pi}. \tag{10}$$

The third term in the above equation specifies the probability of occurance of all erroneous events for an interleaver size of N-bits. Eqs. (9) and (10) can be combined to give a union bound for SECCC scheme with BPSK modulation for transmission over AWGN channel. The detailed derivation of union bound for SECCC scheme can be found in [28].

$$P_b \leq \sum_{\Delta_H} \sum_{w} \frac{A^{(1)}_{2w,\delta^{(1)}}.A^{(2)}_{2w,\delta^{(2)}}}{\binom{N}{w}.\binom{N}{w}}.\frac{w.Q(\sqrt{2\frac{E_b}{N_0}\Delta_H})}{kN}. \tag{11}$$

B. ERROR RATE PERFORMANCE

In this section, based on the schematic of Fig. 1(a) and Fig. 1(b), the BER performance of TC and SECCC, without parallelism and with different parallelism levels (p) is presented. The Frame Error Rate (FER) is also calculated at higher parallelism of 128 and 256 for TC and SECCC and compared in literature. We have used a pseudo-random interleaver in our Matlab simulations. Indeed, the S-random interleaver would give a better performance, while the CMI would be optimum. However, in this paper we did not focus on the interleaver design. The performance of these schemes without parallelism is shown in Fig. 11 for different frame sizes. Like other channel codes [65], TC have a feature to perform better for longer frame sizes. However, the purpose of the research presented in this paper is to observe the behavior of SECCC scheme for different frame sizes and with different levels of parallelism. Since, there is no work reported in literature regarding the parallelism of SECCC, we compared this behavior of SECCC with TC employing the same code rate, frame sizes and parallelism. The results obtained from the analysis of both schemes showed that SECCC performs better than TC for frame sizes less than or equal to 2048 with parallelism higher than 16. Fig. 11 shows the minimum attainable BER for both TC and SECCC schemes with different frame sizes while performing 8 iterations and using a scaling factor of 0.75 with Max-log-MAP algorithm [46]. TC and SECCC show the same performance for frame size of 40 bits, however TC outperforms SECCC for larger frames.

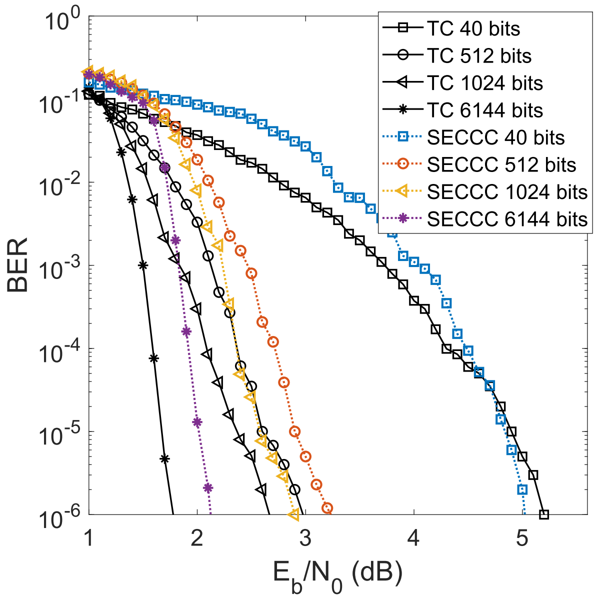


FIGURE 11. BER performance of rate $\frac{1}{3}$ TC and SECCC, with 1000 frames, different frame sizes, 8 decoding iterations, Max-log-MAP with scaling factor 0.75, without parallelism, when communicating over AWGN channel.

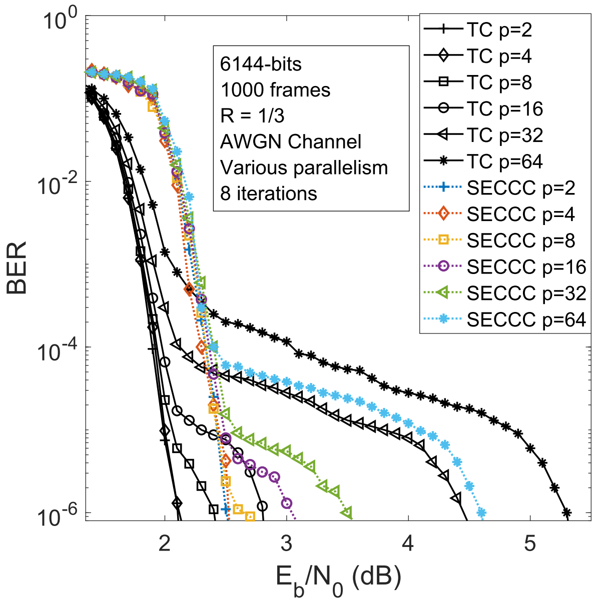


FIGURE 12. BER performance of rate $\frac{1}{3}$ TC and SECCC with 1000 frames of 6144 bits, Max-log-MAP algorithm and different levels of parallelism based on non-initialised method, when communicating over AWGN channel.

The floating and quantized (using quantization of (5,7), where 5 bits are for integer and 7 bits for fractional part) performance of 20,000 bit frame is shown in Fig. 10. The BER performance of TC at $\frac{E_b}{N_0}$ = 1.4 dB and SECCC at $\frac{E_b}{N_0}$ = 1.8 dB, exhibiting the same decoding iterations are compared with uncoded BPSK scheme at $\frac{E_b}{N_0}$ = 8.4 dB. The coding gain of 7 dB and 6.6 dB is achieved for TC and SECCC, respectively, in comparison to uncoded BPSK in order to achieve a BER of 10^{-4}, when transmitting over AWGN channel.

The effect of parallelism on TC scheme has been presented in the literature [8], [56], [57]. However, the effect of parallelism on SECCC scheme is investigated in this paper and compared with TC. Fig. 12 shows the BER plot for rate $\frac{1}{3}$ TC and SECCC with 100 frames of 6144 bits and different parallelism levels $p \in \{2, 4, 8, 16, 32, 64\}$. This is based on non-initilised method and with standard Max-log-MAP algorithm, where αs for all windows in every iteration are initialized with 0 and βs for all windows in every iteration are initialised with 1. However, the BER plot shown in Fig. 13 is obtained by combining the NII method [24] with parallelism and also using the scaling factor [46] for performance improvement. NII-method improved the BER performance by 0.2 dB whereas, further improvement of 0.3 dB for BER of 10^{-4} is achieved by multiplying the scaling factor of 0.75 with the extrinsic information in each iteration [47]. This NII-method initialises the αs at the left end of one window with the αs obtained at the right end of the left-neighbouring window in the previous iteration. Likewise, it initialises the βs at the right end of one window with the βs obtained at the left end of the right-neighbouring window in the previous iteration. To elaborate this method, consider that there are parallelism

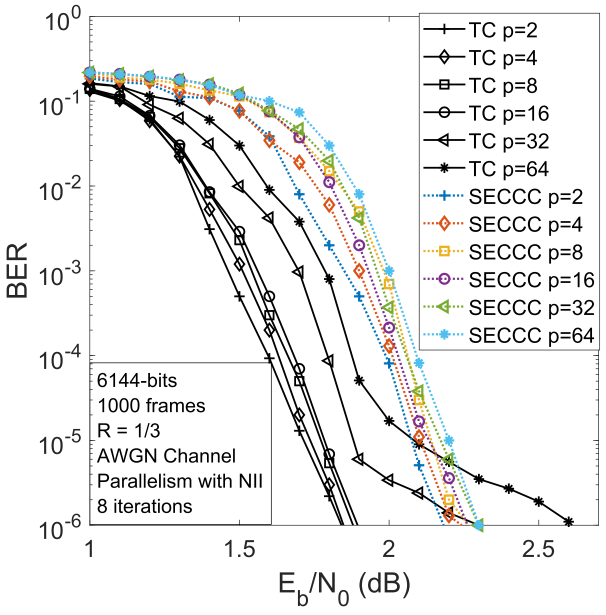


FIGURE 13. BER performance of rate $\frac{1}{3}$ TC and SECCC with 1000 frames of 6144 bits, Max-log-MAP algorithm with scaling factor 0.75 and different levels of parallelism based on NII, when communicating over AWGN channel.

levels denoted by p which are in the following order {2, 4, 8, 16, 32, 64, 128, 256} and I denotes iteration number. According to NII method, at any parallelism level k_p and iteration I, the backward state metrics are initialized with the ones computed at the level (k_{p+1}) in the previous iteration $(I-1)$, i.e., $\beta(I, k_p) = \beta(I-1, k_{p+1})$. Similarly, the forward state metrics at any parallelism level k_p are initialized with the ones computed at the level (k_{p-1}) in the previous iteration $(I-1)$ i.e., $\alpha(I, k_p) = \alpha(I-1, k_{p-1})$. However, the first iteration is performed with the uninitialized forward and backward state metrics with use of Eqs. (5) and (6), respectively.

It can be observed in Fig. 12 and Fig. 13 that the difference in performance for BER of 10^{-4} between $p = 2$ to $p = 64$ for TC is 0.3 dB whereas for SECCC, it is 0.15 dB. Moreover, the BER performance of both schemes for higher parallelism of $p = 128$ and $p = 256$ is also analyzed with both non-initialized method and NII-method, as shown in Fig. 14 and Fig. 15, respectively. The BER curve obtained with non-initialized method at higher parallelism shows an unusual behavior which is due to the fact that the bits near the two ends of each window do not benefit from both αs and βs and they only benefit from either one of them. Hence, some bits have better error correction than others and the floor in BER plot occurs when the SNR is high enough to recover the bits with better error correction, but is not high enough to recover those bits which are near the ends of the windows. By employing the NII method, our simulation results in Fig. 15 show that the error floor disappears completely for SECCC whereas, a small error floor still exists in case of TC for BER lower than 10^{-5}. Fig. 15 shows that TC with NII method shows better performance than SECCC for BER of 10^{-4} at $p = 128$ and for 10^{-2} at $p = 256$. However, for achieving BER lower than 10^{-4} and 10^{-2} at $p = 128$ and 256, respectively, SECCC shows better performance than TC.

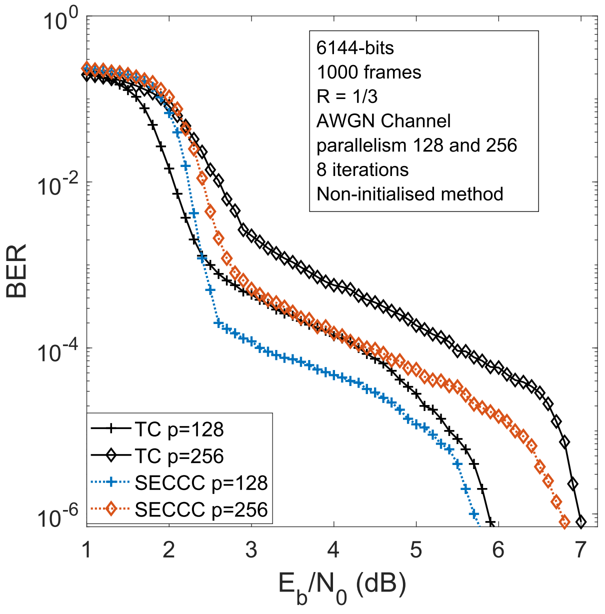


FIGURE 14. BER performance of rate $\frac{1}{3}$ TC and SECCC with 1000 frames of 6144 bits, 8 decoding iterations, Max-log-MAP, non-initialised method, parallelism of 128 and 256, when communicating over AWGN channel.

Fig. 16 shows the effect of parallelism on TC and SECCC, with varying frame sizes to achieve a BER of 10^{-4} at minimum required $\frac{E_b}{N_0}$ and 8 decoding iterations, while transmitting over AWGN channel. Fig. 16 depicts that the small-sized frames with higher parallelism require higher $\frac{E_b}{N_0}$ values to achieve the desired BER as compared to the

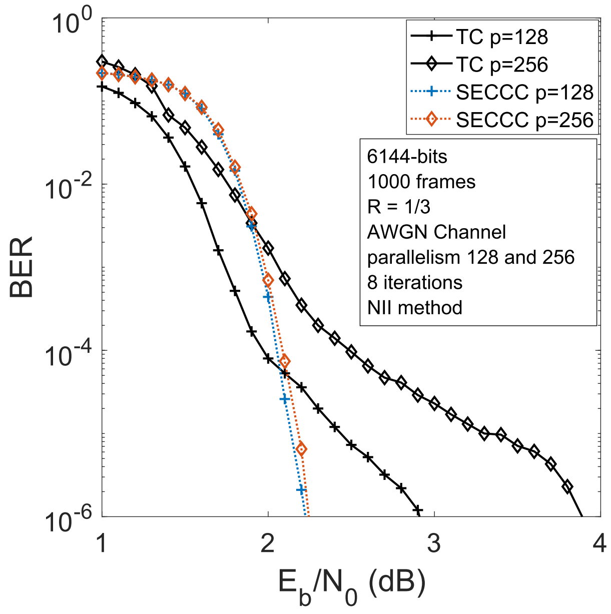


FIGURE 15. BER performance of rate $\frac{1}{3}$ TC and SECCC with 1000 frames of 6144 bits, 8 decoding iterations, Max-log-MAP with scaling factor 0.75, NII method, parallelism of 128 and 256, when communicating over AWGN channel.

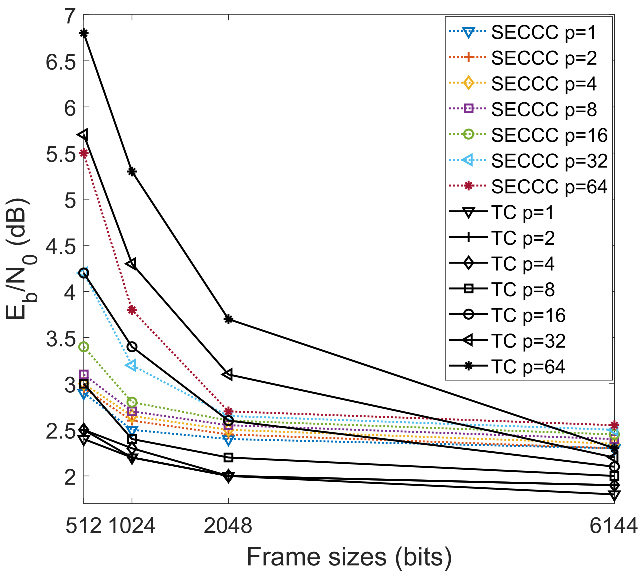


FIGURE 16. Parallelism of TC and SECCC with various frame-sizes to achieve BER of 10^{-4} at minimum required $\frac{E_b}{N_0}$ and 8 iterations, when communicating over AWGN channel.

large sized frames. SECCC outperforms TC for frame sizes less than or equal to 2048 bits at parallelism of 16, 32 and 64, as shown in Fig. 16. For example, for a frame size of 2048 bits, the performance of SECCC is superior than TC by 0.2 dB, 0.55 dB and 1 dB at parallelism of 16, 32 and 64, respectively. This is due to the fact that at higher parallelism levels a 2048 bits frame size is subdivided at smaller-sized subframes and the length of a single trellis of SECCC is twice the length of each of the two trellises of TC. Moreover, the degradation in BER performance between parallelism levels for different frame sizes in case of SECCC is lower than TC. Hence, there are certain frame sizes and parallelism levels where SECCC performs better than TC.

Fig. 17 expresses the error performance of TC and SECCC in terms of FER. It can be observed from Fig. 17 that both schemes have equal FER of 10^{-2} at $\frac{E_b}{N_0}$ of 2.5 dB for $p = 128$. However, SECCC provides FER much lower than 10^{-3} beyond 2.5 dB. At parallelism level of 256, SECCC curve decays faster for FER of 10^{-3} than TC. The FER results shown in Fig. 17 are comparable to those shown for LTE-turbo decoder of rate $\frac{1}{3}$ [22] where, the Fully Parallel Turbo Decoder (FPTD) and Iteration Unrolled XMAP (UXMAP) provide the FER of 10^{-3} at $\frac{E_b}{N_0}$ of 2.5 dB while performing 17 and 4 iterations, respectively. The SECCC at parallelism of 128 and 256 shows the same FER performance with 8 decoding iterations (which are equivalent to 4 turbo iterations).

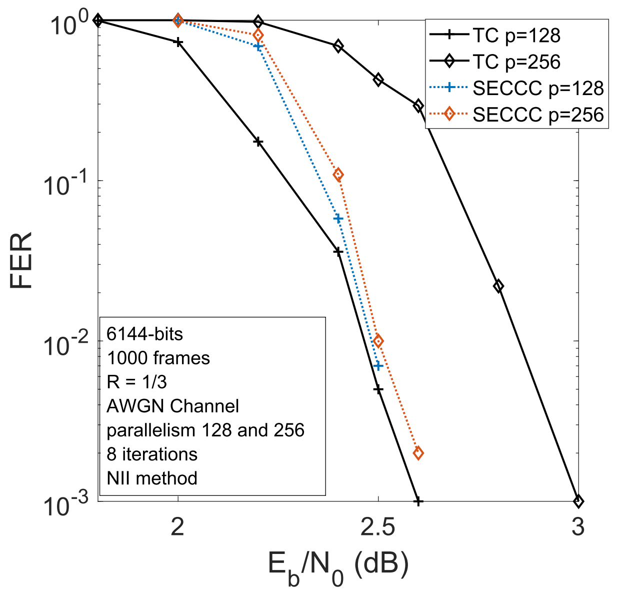


FIGURE 17. FER performance of rate $\frac{1}{3}$ TC and SECCC, frame size of 6144 bits, 8 decoding iterations, 1000 frames, Max-log-MAP, NII-method, parallelism of 128 and 256, when communicating over AWGN channel.

C. SYNTHESIS RESULTS

The VHDL design of MAP decoder architecture shown in Fig. 6(b) is configured for SECCC and TC decoders and synthesized using Xilinx ISE on Virtex-6 FPGA (XCVLX240T) for different frame sizes and parallelism levels. Synthesis results are expressed in the form of clock frequency in MHz and resource consumption in Look Up Tables (LUTs). The clock frequency for each of SECCC and TC decoder are the same, which is 86.3 MHz. The resource consumption in terms of LUTs for different frame sizes without parallelism is given in Table 1. The throughput achieved without parallelism is 10.7 Mbps with 8 decoding iterations, as shown in Table 1. It can be seen that throughput increases linearly with parallelism. The estimate of throughput with different parallelism for frame size of 6144 bits is given in Table 2. The sub-frame length decreases with increasing parallelism levels. The resource consumption depends on length of sub-frames and the number of decoders operating in parallelism levels. The parallel architecture shown in Fig. 7 is configured for 2, 4, 8, 16, 32, 64, 128 and 256 parallelism levels and the resource consumption using frame size of 6144 bits for each parallelism level is presented in Table 2. The throughput is calculated by $\frac{f_{clk} \times p}{I}$ [7], where f_{clk} is the clock frequency, p is level of parallelism and I represents the number of iterations.

TABLE 1. Resource utilization of TC and SECCC decoder for different frame sizes without parallelism.

Frame length (bits)	Resources (LUTs)	Throughput (Mbps) at 8 iterations
40	6890	10.7
512	45603	10.7
1024	54723	10.7
6144	114246	10.7

TABLE 2. Resource utilization for TC and SECCC for frame length 6144 bits with parallelism.

Parallelism (p)	Sub-frame length (bits)	Resources (LUTs)	Throughput (Mbps) at 8 iterations
2	3072	176072	21.575
4	1536	235311	43.15
8	768	393856	86.3
16	384	634112	172.6
32	192	1020920	345.2
64	96	1643642	691.4
128	48	2301056	1380.8
256	24	3221504	2761.6

Another important aspect is complexity which refers to the number of operations per bit. For a MAP decoder, the complexity can be quantified in terms of the number of trellis states per bit [44]. Complexity of TCs can be expressed as $C(TC) = 2 \times 2^{\nu} \times I$, where 2^{ν} is the number of states of the encoder with memory ν and I is the number of activations of decoding algorithm [66]. For SECCC, $C(SECCC) = 2^{\nu} \times I = 0.5C(TC)$. Therefore, the complexity of SECCC with 8 iterations is equal to the complexity of TC with 4 iterations [33].

In case of parallel decoder architecture with parallelism levels p, the frame of size N-bits is divided among p MAP decoders such that each MAP decoder processes a $\frac{N}{p}$-bit sub-frame. Hence, parallel MAP decoder with parallelism p is p times faster in decoding speed but it is p times higher in hardware complexity. However, according to the above definition of complexity, the complexity in terms of total processed trellis states stays the same for the sequential as well as parallel MAP decoder.

V. CONCLUSION AND FUTURE WORK

In this paper, the BER performance of SECCC is investigated with/without parallelism and with different frame sizes and also compared with the TC, where both schemes employ the same RSC code and code rate. In order to invoke a complete comparison, the VHDL design of MAP decoder for both schemes is synthesized using Xilinx ISE. The synthesis results show that both schemes produce equal throughput and exhibit equal resource utilization for the same number of iterations, frame sizes and parallelism. Based on the simulation results presented in Section IV-B it can be concluded that for BER of 10^{-4}, SECCC outperforms TC for frame sizes less than or equal to 2048 bits with parallelism of 16, 32 and 64 as well as for frame sizes greater than or equal to 6144 bits with parallelism of 256. However, Fig. 13 and Fig. 15 show that for achieving BER of 10^{-6} at parallelism of 32, 64, 128 and 256 with NII-method, TC still exhibits a small error floor. At higher parallelism the frame size is divided into smaller sized sub-frames and in case of SECCC, the single trellis is longer than each of the two trellises of TC, therefore SECCC performs better for smaller sized frames at higher parallelism. From the synthesis results tabulated in Table 2, parallel MAP decoder can achieve a throughput of 691.4 Mbps, 1.38 Gbps and 2.76 Gbps at parallelism of 64, 128 and 256, respectively. The FER performance of SECCC at higher parallelism of 128 and 256 is comparable to FPTD and UXMAP of [22].

The future communication standards will demand high throughput and low latency which is possible by employing high parallelism but with a degradation in BER performance. Until 2018, most of the implementations in literature have achieved a maximum parallelism of 64, while the recent work proposed in [21] employs parallelism of 128 to serve the throughput and latency demands of 5G. However, the increasing demands of throughput for future communication standards may require parallelism greater than 128. It is shown in this paper that SECCC employing the same constituent code and code rate as TC gives better performance at $p = 128$ for achieving BER lower than 10^{-4} and outperforms it significantly for parallelism greater than 128. Hence, by using certain frame sizes and parallelism, both TC or SECCC decoder architectures can independently provide the desired performance. Hence, this analysis is beneficial in terms of proposing a reconfigurable architecture as a future work, capable of operating in either TC or in SECCC mode and can support any frame size and parallelism without significant degradation in the BER performance. This paper is focused on the successful demonstration of the concept of SECCC with parallelism. However, for improved throughput and BER performance, a state-of-the-art technique ACQ combined with NII [26] will be considered in our future work for the design of LTE-SECCC scheme.

REFERENCES

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Geneva, Switzerland, vol. 2, May 1993, pp. 1064–1070.

[2] S. Benedetto and G. Montorsi, "Unveiling turbo codes: Some results on parallel concatenated coding schemes," *IEEE Trans. Inf. Theory*, vol. 42, no. 2, pp. 409–428, Mar. 1996.

[3] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, no. 4, pp. 623–656, Oct. 1948.

[4] *Digital Video Broadcasting (DVB); Implementation guidelines for Satellite Services to Handheld devices (SH) below 3GHz*, Standard ETSI EN 302 583 V1.1.0 (2008-01) 2011.

[5] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inf. Theory*, vol. IT-20, no. 2, pp. 284–287, Mar. 1974.

[6] D. López-Pérez, A. Garcia-Rodriguez, L. Galati-Giordano, M. Kasslin, and K. Doppler, "IEEE 802.11be—Extremely high throughput: The next generation of Wi-Fi technology beyond 802.11ax," Feb. 2019, *arXiv:1902.04320*. [Online]. Available: https://arxiv.org/abs/1902.04320

[7] Y. Sun and J. R. Cavallaro, "Efficient hardware implementation of a highly-parallel 3GPP LTE/LTE-advance turbo decoder," *Integration*, vol. 44, no. 4, pp. 305–315, Sep. 2011.

[8] L. F. Gonzalez-Perez, L. C. Yllescas-Calderon, and R. Parra-Michel, "Parallel and configurable turbo decoder implementation for 3GPP-LTE," in *Proc. Int. Conf. Reconfigurable Comput. FPGAs (ReConFig)*, Dec. 2013, pp. 1–6.

[9] R. Shrestha and R. P. Paily, "High-throughput turbo decoder with parallel architecture for LTE wireless communication standards," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 9, pp. 2699–2710, Sep. 2014.

[10] Z. Yan, G. He, W. He, S. Wang, and Z. Mao, "High performance parallel turbo decoder with configurable interleaving network for LTE application," *Integration*, vol. 52, pp. 77–90, Jan. 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167926015000553

[11] H. Luo, Y. Zhang, W. Li, L.-K. Huang, J. Cosmas, D. Li, C. Maple, and X. Zhang, "Low latency parallel Turbo decoding implementation for future terrestrial broadcasting systems," *IEEE Trans. Broadcast.*, vol. 64, no. 1, pp. 96–104, Mar. 2018.

[12] V. Belov and S. Mosin, "FPGA implementation of LTE turbo decoder using MAX-log MAP algorithm," in *Proc. 6th Medit. Conf. Embedded Comput. (MECO)*, Jun. 2017, pp. 1–4.

[13] M. Zhan, J. Wu, and H. Wen, "An optimized resource efficient approximation of max* operator for recalculation in turbo code decoder," in *Proc. 5th Int. Conf. Enterprise Syst. (ES)*, Sep. 2017, pp. 168–172.

[14] D. K. Halim, T. C. Ming, N. M. Song, and D. Hartono, "Software-based turbo decoder implementation on low power multi-processor system-on-chip for Internet of Things," in *Proc. 4th Int. Conf. New Media Stud. (CONMEDIA)*, Nov. 2017, pp. 137–141.

[15] K. Eluri, Y. B, K. Balasubramanian, and D. Mishra, "Low power and area efficient max-log-MAP decoder," in *Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI)*, Sep. 2018, pp. 431–435.

[16] S. Belfanti, C. Roth, M. Gautschi, C. Benkeser, and Q. Huang, "A 1Gbps LTE-advanced turbo-decoder ASIC in 65nm CMOS," in *Proc. Symp. VLSI Circuits*, Jun. 2013, pp. C284–C285.

[17] R. G. Maunder, "A fully-parallel turbo decoding algorithm," *IEEE Trans. Commun.*, vol. 63, no. 8, pp. 2762–2775, Aug. 2015.

[18] A. Li, P. Hailes, R. G. Maunder, B. M. Al-Hashimi, and L. Hanzo, "1.5 Gbit/s FPGA implementation of a fully-parallel turbo decoder designed for mission-critical machine-type communication applications," *IEEE Access*, vol. 4, pp. 5452–5473, 2016.

[19] T. Cui, F. Gao, A. Nallanathan, H. Lin, and C. Tellambura, "Iterative demodulation and decoding algorithm for 3GPP/LTE-A MIMO-OFDM using distribution approximation," *IEEE Trans. Wireless Commun.*, vol. 17, no. 2, pp. 1331–1342, Feb. 2018.

[20] A. Ghaffari, M. Léonardon, A. Cassagne, C. Leroux, and Y. Savaria, "Toward high-performance implementation of 5G SCMA algorithms," *IEEE Access*, vol. 7, pp. 10402–10414, 2019.

[21] L. Xiang, M. F. Brejza, R. G. Maunder, B. M. Al-Hashimi, and L. Hanzo, "Arbitrarily parallel turbo decoding for ultra-reliable low latency communication in 3GPP LTE," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 4, pp. 826–838, Apr. 2019.

[22] S. Weithoffer, C. A. Nour, N. Wehn, C. Douillard, and C. Berrou, "25 years of turbo codes: From Mb/s to beyond 100 Gb/s," in *Proc. IEEE 10th Int. Symp. Turbo Codes Iterative Inf. Process. (ISTC)*, Dec. 2018, pp. 1–6.

[23] S. Weithoffer, F. Pohl, and N. Wehn, "On the applicability of trellis compression to Turbo-Code decoder hardware architectures," in *Proc. 9th Int. Symp. Turbo Codes Iterative Inf. Process. (ISTC)*, Sep. 2016, pp. 61–65.

[24] J. Dielissen and J. Huisken, "State vector reduction for initialization of sliding windows MAP," in *Proc. IEEE 2nd Int. Symp. Turbo Codes Related Topics*, Sep. 2000, pp. 387–390.

[25] C. Benkeser, A. Burg, T. Cupaiuolo, and Q. Huang, "Design and optimization of an HSDPA Turbo decoder ASIC," *IEEE J. Solid-State Circuits*, vol. 44, no. 1, pp. 98–106, Jan. 2009.

[26] C. Roth, S. Belfanti, C. Benkeser, and Q. Huang, "Efficient parallel turbo-decoding for high-throughput wireless systems," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 6, pp. 1824–1835, Jun. 2014.

[27] A. Sholiyi, J. A. Alzubi, O. A. Alzubi, O. Almomani, and T. O'Farrell, "Near capacity irregular turbo code," Feb. 2016, *arXiv:1604.01358*. [Online]. Available: https://arxiv.org/abs/1604.01358

[28] S. X. Ng, M. F. U. Butt, and L. Hanzo, "On the union bounds of self-concatenated convolutional codes," *IEEE Signal Process. Lett.*, vol. 16, no. 9, pp. 754–757, Sep. 2009.

[29] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Serial concatenated trellis coded modulation with iterative decoding," in *Proc. IEEE Int. Symp. Inf. Theory*, Ulm, Germany, Jun./Jul. 1997, p. 8.

[30] D. Divsalar and F. Pollara, "Hybrid concatenated codes and iterative decoding," in *Proc. IEEE Int. Symp. Inf. Theory*, Jun./Jul. 1997, pp. 10.

[31] H.-A. Loeliger, "New turbo-like codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Ulm, Germany, Jun./Jul. 1997, p. 109.

[32] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Self-concatenated trellis coded modulation with self-iterative decoding," in *Proc. IEEE Global Telecommun. Conf.*, Sydney, NSW, Australia, vol. 1, Nov. 1998, pp. 585–591.

[33] M. F. U. Butt, S. X. Ng, and L. Hanzo, "Self-concatenated code design and its application in power-efficient cooperative communications," *IEEE Commun. Surveys Tuts.*, vol. 14, no. 3, pp. 858–883, 3rd Quart., 2012.

[34] J. A. Alzubi, O. A. Alzubi, and T. M. Chen, *Forward Error Correction Based On Algebraic-Geometric Theory*. Cham, Switzerland: Springer, 2014.

[35] M. Synthia and M. S. Ali, "Performance study of turbo code with interleaver design," *Int. J. Sci. Eng. Res.*, vol. 2, no. 7, Jul. 2011.

[36] J. Yuan, B. Vucetic, and W. Feng, "Combined turbo codes and interleaver design," *IEEE Trans. Commun.*, vol. 47, no. 4, pp. 484–487, Apr. 1999.

[37] B. Vucetic and J. Yuan, *Turbo Codes: Principles and Applications*. Norwell, MA, USA: Kluwer, 2001.

[38] M. S. C. Ho, S. S. Pietrobon, and T. Giles, "Improving the constituent codes of turbo encoders," in *Proc. IEEE GLOBECOM*, vol. 6, Nov. 1998, pp. 3525–3529.

[39] D. Divsalar and F. Pollara, "On the design of Turbo codes," Jet Propuls. Lab., Pasadena, CA, USA, Tech. Rep. 42-123, Nov. 1995.

[40] C. Douillard and C. Berrou, "Turbo codes with rate-m/(m+1) constituent convolutional codes," *IEEE Trans. Commun.*, vol. 53, no. 10, pp. 1630–1638, Oct. 2005.

[41] B. Sklar, *Digital Communications: Fundamentals and Applications*. Upper Saddle River, NJ, USA: Prentice-Hall, 2001. [Online]. Available: https://books.google.com.pk/books?id=HFwbnQAACAAJ

[42] J. Hagenauer, "Rate-compatible punctured convolutional codes (RCPC codes) and their applications," *IEEE Trans. Commun.*, vol. 36, no. 4, pp. 389–400, Apr. 1988.

[43] J. Hagenauer and P. Hoeher, "A Viterbi algorithm with soft-decision outputs and its applications," in *Proc. IEEE Global Telecommun. Conf. Exhib. 'Commun. Technol. Beyond'*, Nov. 1989, pp. 1680–1686.

[44] L. Hanzo, T. H. Liew, B. L. Yeap, R. Y. S. Tee, and S. X. Ng, *Turbo Coding, Turbo Equalisation and Space-Time Coding*, 2nd ed. Hoboken, NJ, USA: Wiley, 2002.

[45] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal MAP decoding algorithms operating in the Log domain," in *Proc. Int. Conf. Commun.*, Seattle, WA, USA, Jun. 1995, pp. 1009–1013.

[46] A. Worm, P. Hoeher, and N. Wehn, "Turbo-decoding without SNR estimation," *IEEE Commun. Lett.*, vol. 4, no. 6, pp. 193–195, Jun. 2000.

[47] J. Vogt and A. Finger, "Improving the max-log-MAP turbo decoder," *Electron. Lett.*, vol. 36, no. 23, pp. 1937–1939, Nov. 2000.

[48] M. Taskaldiran, R. C. S. Morling, and I. Kale, "A comparative study on the modified Max-Log-MAP turbo decoding by extrinsic information scaling," in *Proc. Wireless Telecommun. Symp.*, Apr. 2007, pp. 1–5.

[49] A. Giulietti, B. Bougard, and L. Van Der Perre, *Turbo codes: Desirable and designable*. Norwell, MA, USA: Kluwer, 2004.

[50] A. Sghaier, S. Areibi, and B. Dony, "A pipelined implementation of OFDM transmission on reconfigurable platforms," in *Proc. Can. Conf. Electr. Comput. Eng.*, May 2008, pp. 801–804.

[51] G. Wang, A. Vosoughi, H. Shen, J. R. Cavallaro, and Y. Guo, "Parallel interleaver architecture with new scheduling scheme for high throughput configurable turbo decoder," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2013, pp. 1340–1343.

[52] T. Ilnseher, M. May, and N. Wehn, "A multi-mode 3GPP-LTE/HSDPA turbo decoder," in *Proc. IEEE Int. Conf. Commun. Syst.*, Nov. 2010, pp. 336–340.

[53] R. Al-Dujaily, A. Li, R. G. Maunder, T. Mak, B. M. Al-Hashimi, and L. Hanzo, "A scalable Turbo decoding algorithm for high-throughput network-on-chip implementation," *IEEE Access*, vol. 4, pp. 9880–9894, 2016.

[54] M. J. Thul, F. Gilbert, T. Vogt, G. Kreiselmaier, and N. Wehn, "A scalable system architecture for high-throughput turbo-decoders," in *Proc. IEEE Workshop Signal Process. Syst.*, Oct. 2002, pp. 152–158.

[55] M. May, T. Ilnseher, N. Wehn, and W. Raab, "A 150Mbit/s 3GPP LTE Turbo code decoder," in *Proc. Conf. Design, Automat. Test Eur. (DATE)*, Mar. 2010, pp. 1420–1425.

[56] C.-C. Wong, M.-W. Lai, C.-C. Lin, H.-C. Chang, and C.-Y. Lee, "Turbo decoder using contention-free interleaver and parallel architecture," *IEEE J. Solid-State Circuits*, vol. 45, no. 2, pp. 422–432, Feb. 2010.

[57] G. Wang, H. Shen, Y. Sun, J. R. Cavallaro, A. Vosoughi, and Y. Guo, "Parallel interleaver design for a high throughput HSPA+/LTE multi-standard turbo decoder," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 5, pp. 1376–1389, May 2014.

[58] P. Murugappa, A. Baghdadi, and M. Jezequel, "Parameterized area-efficient multi-standard turbo decoder," in *Proc. Design, Autom. Test Eur. Conf. Exhibit. (DATE)*, Mar. 2013, pp. 109–114.

[59] O. Muller, A. Baghdadi, and M. Jezequel, "From parallelism levels to a multi-ASIP architecture for turbo decoding," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 17, no. 1, pp. 92–102, Jan. 2009.

[60] R. Dobkin, M. Peleg, and R. Ginosar, "Parallel VLSI architecture for MAP turbo decoder," in *Proc. 13th IEEE Int. Symp. Pers., Indoor Mobile Radio Commun.*, vol. 1, Sep. 2002, pp. 384–388.

[61] S. ten Brink, "Convergence behavior of iteratively decoded parallel concatenated codes," *IEEE Trans. Commun.*, vol. 49, no. 10, pp. 1727–1737, Oct. 2001.

[62] A. Sholiyi and T. O. , "Farrell, O. A. Alzubi, and J. A. Alzubi, "Performance evaluation of turbo codes in high speed downlink packet access using EXIT charts," *Int. J. Future Gener. Commun. Netw.*, vol. 10, no. 8, pp. 1–14, Aug. 2017.

[63] M. F. U. Butt, R. A. Riaz, S. X. Ng, and L. Hanzo, "Near-capacity iteratively decoded binary self-concatenated code design using EXIT charts," in *Proc. IEEE GLOBECOM IEEE Global Telecommun. Conf.*, Nov./Dec. 2008, pp. 1–5.

[64] C. Schlegel and L. Perez, *Trellis Turbo Coding*. Hoboken, NJ, USA: Wiley, 2003.

[65] O. A. Alzubi, T. M. Chen, J. A. Alzubi, H. Rashaideh, and N. Al-Najdawi, "Secure channel coding schemes based on algebraic-geometric codes over hermitian curves," *J. UCS*, vol. 22, no. 4, pp. 552–566, 2016.

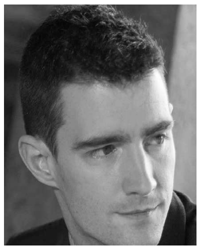
[66] H. Chen, R. G. Maunder, and L. Hanzo, "A survey and tutorial on low-complexity turbo coding techniques and a holistic hybrid ARQ design example," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 1546–1566, 2013.

FARZANA SHAHEEN received the M.Sc. degree in electronics from Quaid-e-Azam University, in June 2006, and the M.S. degree in electrical engineering from COMSATS University, Islamabad, Pakistan, in 2012, where she is currently pursuing the Ph.D. degree in electrical engineering and is also a Lecturer with the Department of Physics, Faculty of Electronics. Her Ph.D. research is based on the implementation aspects of channel codes. Her research interests include channel coding, FPGA-based architecture designs, and control systems.

MUHAMMAD FASIH UDDIN BUTT received the B.E. degree from the National University of Science and Technology (NUST), Pakistan, in 1999, the M.E. degree from the Center for Advanced Studies in Engineering, UET Taxila, Pakistan, with specialization in digital communication/computer networks, in 2003, and the Ph.D. degree from the Communications Research Group, School of Electronics and Computer Science, University of Southampton, U.K., in June 2010. He is currently a Tenured Associate Professor with the Department of Electrical and Computer Engineering, COMSATS University Islamabad, Islamabad, Pakistan where he has been serving as an Academician, since 2002. He has published more than 35 research articles in various reputed journals and conference proceedings. His research interests include channel coding, iterative detection, cooperative cognitive radio networks, mmWave radio-over-fiber technologies, energy harvesting and physical layer security.

SHAHRUKH AGHA received the Ph.D. degree in software and hardware techniques for accelerating MPEG motion estimation from Loughborough University, U.K., in 2006. He is currently an Assistant Professor with the Department of Electrical and Computer Engineering, COMSATS University Islamabad, Islamabad, Pakistan. His research interests include low power real time VLSI implementations of digital video and signal processing algorithms.

SOON XIN NG received the B.Eng. degree (Hons.) in electronic engineering and the Ph.D. degree in telecommunications from the University of Southampton, Southampton, U.K., in 1999 and 2002, respectively, where he has been a member of academic staff with the School of Electronics and Computer Science, since August 2006. From 2003 to 2006, he was a Postdoctoral Research Fellow working on collaborative European research projects known as SCOUT, NEWCOM, and PHOENIX. He is currently an Associate Professor in telecommunications with the University of Southampton. He has published over 250 articles. He has coauthored two John Wiley/IEEE Press books in his research field. He was involved in OPTIMIX and CONCERTO European projects as well as IU-ATC and UC4G projects. He was the Principal Investigator of an EPSRC project on cooperative classical and quantum communications systems. His research interests include adaptive coded modulation, coded modulation, channel coding, space-time coding, joint source and channel coding, iterative detection, OFDM, MIMO, cooperative communications, distributed coding, quantum communications, quantum error correction codes, and joint wireless-and-optical-fiber communications. He is a Chartered Engineer and a Fellow of the Higher Education Academy, U.K.



ROBERT G. MAUNDER received the B.Eng. degree (Hons.) in electronic engineering and the Ph.D. degree in telecommunications from the School of Electronics and Computer Science, University of Southampton, U.K., in 2003 and 2007, respectively. He was a Lecturer with the NGW Group, School of Electronics and Computer Science, University of Southampton, U.K., in 2007, an Associate Professor, in 2013, and a Professor, in 2017. He is currently the Founder and CTO of AccelerComm Ltd., which is commercializing his research as soft-IP. His research interests include joint source/channel coding and the holistic design of algorithms and hardware implementations for wireless communications. He has published more than 100 IEEE articles in these areas. He was a Chartered Engineer of the IET, in 2013, and a Fellow of the IET, in 2017.