
Mr. T. Narasimha Rao, Rekha Surendar

1 Assistant Professor, ECE, Madhira Institute of Technology & science, Chilkur, Nalgonda, India
2 PG Scholar, ECE, Madhira Institute of Technology & science, Chilkur, Nalgonda, India

narasimharao@gmail.com, rekkhasurendar@gmail.com

Abstract: Turbo codes have recently been considered for energy-constrained wireless communication applications, since they facilitate a lower transmission energy consumption. However, in order to reduce the overall energy consumption, lookup table-log-BCJR (LUT-Log-BCJR) architectures having a low processing energy consumption are required. In this paper, we decompose the LUT-Log-BCJR architecture into its most fundamental add compare select (ACS) operations and perform them using a novel low-complexity ACS unit. We demonstrate that our architecture employs an order of magnitude fewer gates than the most recent LUT-Log-BCJR architectures, facilitating a 71% energy consumption reduction. Compared to state-of-the-art maximum logarithmic Bahl-Cocke-Jelinek-Raviv implementations, our approach facilitates a 10% reduction in the overall energy consumption at ranges above 58 m.

INTRODUCTION: With rapid growth of multimedia services, the convolutional turbo code (CTC) has been widely adopted as one of forward error correcting (FEC) schemes of wireless standards to have a reliable transmission over noisy channels. Single-binary (SB) CTC, proposed in 1993 [1], has been the well-known FEC code that can achieve high data rates and coding gains close to the Shannon limit. The SB CTC code has been adopted in the FEC schemes of wideband code division multiple access (WCDMA) [2], high-speed downlink packet access (HSDPA) [2], and long term evolution (LTE) [2]. In 1999, non-binary CTC [3] was introduced to achieve superior performance than the SB CTC. In recent years, double-binary (DB) CTC has been adopted in advanced wireless communication standards, such as worldwide interoperability for microwave access (WiMAX) [4]. Table I lists the detailed specifications and the CTC schemes of the prevalent wireless standards for wide area networks (WANs). Some CTC decoders have been implemented as application-specific integrated circuits (ASICs), such as the HSDPA CTC decoder [5] and WiMAX CTC decoder [6]. Recently, high-end portable/mobile devices become prevalent in wireless markets. There are large growing emergence and demands for an inexpensive solution to access the ubiquitous wireless services. Meanwhile, these wireless standards, such as 3GPP and WiMAX standards, adopted CTC schemes with different coding parameters and different throughput rates [8], [9]. To deal with the accelerated evolution of these standards, the multi standard platforms which can seamlessly work across the multiple standards were proposed in [7]. Hence, a CTC decoding accelerator which works in the multi standard platforms is desired to achieve the smooth migration for the multiple wireless applications. In this paper, we propose area-efficient scalable processor designs of the maximum a posteriori algorithm (MAP) [10] for multi standard CTC schemes. In Figs. 1(a)–(c), the proposed MAP processor can be the deterministic or reconfigurable FEC components. In Fig. 1(d), several proposed MAP processors can be used in multiple processors for system-on-chip (MPSoC) technology for a multi standard platform. The proposed scalable MAP processor design for high-throughput multi standard CTC schemes has two features stated as follows. Reliable data transmission in wireless communication systems requires sophisticated channel coding schemes and corresponding high-throughput, low-area, and energy-efficient decoder implementations. Convolutional codes (CCs), used in stand-alone form [3] or as part of turbo codes [4], are among the most popular codes used in current and next-generation wireless communication standards, such as HSDPA [5], 3GPP-LTE [6], LTE-Advanced [7], or IEEE 802.11n [8]. CCs and turbo codes are of significant practical interest due to the fact that they offer excellent error-correction performance and can be implemented to achieve high throughput, while being efficient in terms of silicon area and power consumption [9]–[16]. Since the advent of turbo codes [4], iterative decoding algorithms relying on CCs, became a key enabler for wireless communication systems operating close to the Shannon limit. CCs have also been considered for wireless communication systems employing iterative detection and decoding, i.e., where reliability information is exchanged iteratively between a detector and the channel decoder. In particular, iterative detection and decoding in multiple-input
multiple-output (MIMO) wireless systems [17] or systems exhibiting inter-symbol interference [18] is becoming an integral part of future transceiver designs because it is an efficient means to substantially improve the throughput and quality-of-service (i.e., link reliability, coverage, and range) compared to non-iterative decoding schemes.

II. LITERATURE SURVEY:

Cooperative and relay communication has received much attention recently [2]–[5] due to the potential performance improvement. At the network level, relays can extend wireless network coverage if a direct source-to-destination link does not exist [6]. At the physical level, the use of relays can achieve the diversity gains offered by multiple antenna space-time systems through using several relay nodes, possibly with only one antenna per physical node [4]. The latter is the primary focus of this paper. This paper considers a three node (source (S), relay (R) and destination (D)) wireless cooperative communication system, which can be considered as a building block for larger wireless networks. The basic cooperative schemes that have been previously considered include Amplify-and-Forward (AF) and Decode-and-Forward (DF) [2]. Here we consider a form of DF. It has been shown in [7] that DF schemes, in which the relay nodes always forward the decoded results without indicating their reliability, tend not to achieve the desired diversity gain due to erroneous decoding at the relay. Simple selective DF was proposed in [7] to solve this problem, where the relay keeps silent when errors are detected in its decoded data. Note this usually requires the use of a cyclic redundancy check (CRC) code embedded in the source-to-relay (S-R) link. Among the coded DF schemes, those using distributed Turbo codes (DTCs) have been shown to offer good performance [8], [9]. In [8], a DTC was proposed using DF, where the two component codes are separately generated at the source and relay. It, however, assumes an ideal S-R link. The DTC schemes proposed in [10], improved significantly without using a CRC, by appropriately weighting the relayed information which may contain errors. To address this problem, a simple non-selective DF scheme was proposed for DTCs in [1]. In [1], the relay decodes the received signals without performing a CRC check.

III. TURBODECODER ALGORITHM

The MAP algorithm, which provides the probability of the decoded bit $u_k$ being either $+1$ or $-1$ for the received symbol sequence $y$ by calculation of the LLR values as

$$L(u_k | y) = \log \left( \frac{p(u_k = +1 | y)}{p(u_k = -1 | y)} \right)$$  \hspace{1cm} (1)

where $p(u_k = +1 | y)$ and $p(u_k = -1 | y)$ denote the probabilities of bit $u_k$ being $+1$ and $-1$, respectively.

The turbo decoder specified in LTE consists of two recursive convolutional encoders, i.e., an interleaver and a feedthrough path, as shown in Fig. 1(a). The feed-through passes one block of $K$ information bits, which are called systematic bits $x_k$, where $k = 0, 1, ..., K - 1$. The parity generated by the convolutional encoder is denoted by $x_k$. By permuting the systematic bits via the interleaver, the second sequence of parity is generated by passing through the second convolutional encoder, which is denoted by $x_k$. On the receiver side, the reliability of bits is computed iteratively by exchanging the extrinsic LLRs between two SISO decoders based on (1), as depicted in Fig. 1(b). Another representation of a convolutional encoder is by using a trellis diagram, as shown in Fig. 1(c), depicting two steps of the LTE turbo encoder. Applying few mathematical manipulations on (1) leads to

$$LLR(u_k) = \log \left( \frac{\sum_{u_k=+1} \delta_{k-1}(s') \beta_k(s) \gamma_k(s', s)}{\sum_{u_k=-1} \delta_{k-1}(s') \beta_k(s) \gamma_k(s', s)} \right)$$  \hspace{1cm} (2)
IV. FORMULATION FOR RADIX-4 RECURSION COMPUTATION

The encoder and the trellis diagram of the LTE standard, consisting of eight states, are shown in Fig. 1(a) and (c), respectively. The radix-4 trellis diagram is partially shown in Fig. 1(d). According to (15) and the fact that \( u_k \) and \( c_k \) can take one of the two values of +1 or −1, the \( \gamma \) values of each stage in the trellis diagram have 32 possible values, with half of them being negative, which are summarized in Table I. According to the \( \gamma \) values in Table I and (14), the backward state values in the \( k \)-th stage can be written as

\[
\beta_k(0) = \max \{ \beta_k(3) + \gamma_k(5), \beta_k(7) + \gamma_k(8), \beta_k(1) - \gamma_k(7), \beta_k(5) - \gamma_k(6) \} \\
\beta_k(5) = \max \{ \beta_k(3) - \gamma_k(8), \beta_k(7) - \gamma_k(5), \beta_k(1) + \gamma_k(6), \beta_k(5) + \gamma_k(6) \} \\
\beta_k(6) = \max \{ \beta_k(7) - \gamma_k(4), \beta_k(3) - \gamma_k(1), \beta_k(5) + \gamma_k(2), \beta_k(1) + \gamma_k(3) \} \\
\beta_k(7) = \max \{ \beta_k(7) + \gamma_k(1), \beta_k(3) + \gamma_k(4), \beta_k(5) - \gamma_k(3), \beta_k(1) - \gamma_k(2) \}.
\]

Similar equations hold true for \( \alpha \) values. In this brief, a smart relation among these formulas is introduced, leading to an optimized hardware implementation, which is described in the following.

V. ACS ARCHITECTURE

The common approach to implement the recursion unit is by using the ACS architecture. In this case, the radix-2 recursion unit is implemented by using an adder, a comparator unit, and a selector unit dictated by (12), as shown in Fig. 2(a), where common approximations of the logarithmic term in \( \log(1 + e^{-|z-t|}) \) are used for implementing the LUT. Few designs such as the one in [10] have been proposed to reduce the latency of this architecture, all in radix-2. However, in recent wireless communication systems with a clear demand for a high-throughput framework, a radix-4 architecture is a common approach [11] and should be efficiently designed.

Fig. 2(b) shows the architecture of a radix-4 ACS unit consisting of three radix-2 ACS units. The main advantage of using a radix-4 architecture is its concurrent computation of two-bit recursion metrics leading to a higher throughput. However, compared with its radix-2 counterpart, it has a higher latency and silicon area overhead. Therefore, due to the nature of the recursive computation, which highly restricts the clock frequency, achieving a high throughput is by far a more challenging task in a radix-4 framework. Although several designs have been proposed so far to shorten the latency of the radix-4 architectures, a large silicon area overhead is their unwanted byproduct [10]. Therefore, the goal of this brief is to alleviate the area overhead of radix-4 architectures.

WIRELESS SENSOR NETWORKs (WSNs) can be considered to be energy constrained wireless scenarios, since the sensors are operated for extended periods of time, while relying on batteries that are small, lightweight and inexpensive. In environmental monitoring WSNs for example, despite employing low transmission duty cycles and low average throughputs of less than 1 Mbit/s [1], [2], the sensors’ energy consumption is dominated by the transmission energy (measured in J/bit), since they may be separated by up to 1 km. For this reason, turbo codes have recently found application in these scenarios [3], [4], since their near-capacity coding gain facilitates reliable communication when using a reduced transmission energy. Note however that this
reduction in is offset by the turbo decoder’s energy consumption, as well as the (typically negligible) energy consumption of the turbo encoder [4]. Therefore, turbo codes designed for energy constrained scenarios have to minimize the overall energy consumption. Recent application-specific integrated circuit (ASIC)-based turbo decoder architectures [5]–[7] have been designed for achieving a high transmission throughput, rather than for a low transmission energy. For example, turbo codes have facilitated transmission throughputs in excess of 50 Mbit/s in cellular standards, such as the 3rd Generation Partnership Project 3GPP Long Term Evolution (LTE) and recent ASIC turbo decoder architectures have been designed for throughputs that are in excess of 100 Mbit/s [5], [6]. This has been achieved by employing the Max-Log-BCJR turbo decoding algorithm, which is a low-complexity approximation of the optimal Logarithmic Bahl-Cocke-Jelinek-Raviv (Log-BCJR) algorithm [8]. The Max-Log-BCJR algorithm appears to lend itself to both high-throughput scenarios, as well as to the above-mentioned energy-constrained scenarios. This is because a low turbo decoder energy consumption is implied by Max-Log-BCJR algorithm’s low complexity. However, this is achieved at the cost of degrading the coding gain by 0.5 dB compared to the optimal Log-BCJR algorithm [9], increasing the required transmission energy by 10%. As we shall demonstrate in Section IV, this disadvantage of the Max-Log-BCJR outweighs its attractively low complexity, when optimizing the overall energy consumption of sensor nodes that are separated by dozens of meters. This motivates the employment of the lookup-table-logBCJR (LUT-Log-BCJR) algorithm [8] in energy-constrained scenarios, since it approximates the optimal Log-BCJR more closely than the Max-Log-BCJR and therefore does not suffer from the associated coding gain degradation. However, to the best of our knowledge, no LUT-Log-BCJR ASICs have been specifically designed for energy-constrained scenarios.

Fig. 3. Turbo encoder and decoder scheme.

Fig. 4. (a) Conventional LUT-Log-BCJR architecture. (b) Timing of the sliding-window technique

two pipelined steps using the corresponding dedicated hardware components of Fig. 2(a). 1) First, the transition metrics [20, (2)], that correspond to the current window are generated. Here, each transition metric is set either equal to the corresponding a priori LLR or to zero, depending on the particular pair of states and that the transition is between and on the Generator Polynomials (GPs) of the encoder. 2) Next, the state metrics A [20, (3)] that correspond to the current window are generated. Here, each state metric is given by (1) where represents the set of all states that can transition into the state , depending on the GPs of the encoder. Note that the forward recursion for the first window is initialized independently. By contrast, the forward recursion for the other windows is initialized using state metrics that were obtained during the forward recursion of the preceding window. It is for this reason that the windows must be processed in their natural order, as shown in Fig. 2. The operation is used to represent the Jacobian logarithm detailed in [21], which may be approximated using a LUT [17] for the parameters and according to and can be extended to three or more parameters using associativity. Here, we assume the employment of a twos complement fixed-point LLR representation, which includes a 5-bit integer part and a 3-bit fraction part. As a result, there are entries in the LUT, each of which has values that are multiples of . As we will show in Fig. 7, this arrangement yields a near-ideal BER performance.
[22], provided that the integer parts of the LLR values are clipped to the range that can be represented using three bits. During the forward recursion, one set of state metrics is written to Mem 2 of Fig. 2(a) per clock cycle in the ascending order of the bit index. When the backward recursion is performed for a particular window, one pair of its corresponding a priori LLRs is read from Mem 1 of Fig. 2(a) and processed per clock cycle, in the descending order of the bit index. Simultaneously, the corresponding set of state metrics are read from Mem 2 and processed per clock cycle. As a result, a particular window's backward recursion cannot be performed until after its forward recursion has been completed, as shown in Fig. 2(b). The backward recursion of the LUT-Log-BCJR algorithm can be performed in four pipelined steps using the corresponding dedicated hardware components of Fig. 2(a). 1) First, the transition metrics that correspond to the current window are regenerated, as described above. 2) Next, the state metrics [20, (4)] that correspond to the current window are generated. Here, each state metric is given by LLR per clock cycle, as shown in Fig. 2. Therefore, it achieves a high throughput, provided that it can be operated at a high clock frequency. However, the recursions involve calculations that must be performed in series. Therefore, conventional architectures typically employ additional hardware during synthesis to achieve a short critical path, a high clock frequency and a high throughput [24]. A number of variants of the LUT-Log-BCJR architecture of Fig. 2 have been proposed for further increasing the decoding throughput. For example, [25] employs parallel repetitions of the blocks shown in Fig. 2(a) to “parallel-process” the schedule of Fig. 2(b). Alternatively, [12] employs a radix-4 variant, which processes two sets of state metrics at a time. In summary, conventional LUT-Log-BCJR architectures achieve high throughputs by employing substantial hardware, which imposes a high chip area and consequently a high energy consumption, as quantified later in Section IV. Note that the energy consumption of the conventional LUT-Log-BCJR architecture cannot be significantly reduced by simply reducing the clock frequency, in order to meet the lower throughput demands of energy-constrained scenarios.

VI. PROPOSED SCHEME

PROPOSED LUT-LOG-BCJR ARCHITECTURE In this section, we propose a novel LUT-Log-BCJR architecture for energy-constrained scenarios, which avoids the wastage of energy that is inherent in the conventional architecture of Section II. Our philosophy is to redesign the timing of the conventional architecture in a manner that allows its components to be efficiently merged. This produces an architecture comprising only a low number of inherently low-complexity functional units, which are collectively capable of performing the entire LUT-Log-BCJR algorithm. Further wastage is avoided, since the critical paths of our functional units are naturally short and equally-lengthened, eliminating the requirement for additional hardware to manage them. Furthermore, our approach naturally results in a low area and a high clock frequency, which implies a low static energy consumption. As we will show in Section III-A, the LUT-Log-BCJR algorithm is naturally suited to this philosophy, since it can be decomposed into classic ACS operations. In Section III-B we tackle the challenge of devising an architecture that is sufficiently flexible for performing the entire LUT-Log-BCJR algorithm, using only a small number of functional units. Furthermore, Section III-C proposes a functional unit that is capable of performing ACS operations, while maintaining a short critical path and a low complexity. Finally, in Section III-D, we will design a controller for our architecture, using the LUT-Log-BCJR decoder of the 3GPP LTE turbo decoder as an application example.

For example, this allows the four ACS operations equivalent to a calculation to be performed in four consecutive clock cycles using a single ACS unit, as
Fig. 5. Energy-efficient LUT-Log-BCJR architecture.

detailed in Section III-C. The second register level comprises REG bank 1 and REG bank 2 of Fig. 3, which are used to temporarily store the LUT-Log-BCJR variables between consecutive values of the bit index during the recursions decoding processes. The REG bank 1 comprises registers for the a priori LLRs and dummy registers for the required LUT constants of (2). Meanwhile, the sets of or metrics are stored in REG bank 2 of Fig. 3. The main memory stores all the required a priori LLR sequences and extrinsic LLR sequences during the decoding process and the state metrics from the previous window, which facilitates the processing of the entire LUT-Log-BCJR algorithm. Since the proposed architecture supports a fully parallel arrangement of an arbitrary number of ACS units of Fig. 3, it may be readily applied to any LUTLog-BCJR decoder, regardless of the specific convolutional encoder parameters2 employed. Note that in contrast to the different-length data paths of Fig. 2(a), the identical parallel data paths shown in Fig. 3 have equal lengths, which avoids energy wastage, as described above.

Novel ACS Unit In this section we propose the novel low-gate-count ACS unit of Fig. 4, which performs one ACS operation per clock cycle. The control signals of the ACS unit are provided by the operation code, which can be used to perform the functions listed in Table II. Note that the operation code approximates the absolute difference between two operands, as required by (2). Its result is equivalent to . However, for , the result is given by . In the two’s complement operand representation employing fraction bits, this is equivalent to decrementing the binary representation of , which is equivalent to subtracting . Note that a simpler ACS unit implementation is facilitated by this deliberately introduced inaccuracy, which can be trivially canceled out during the calculation. More specifically, a calculation can be performed.

![Fig. 5. Energy-efficient LUT-Log-BCJR architecture.](image1)

![Fig. 6. ACS unit.](image2)

**TABLE II**

<table>
<thead>
<tr>
<th>Function</th>
<th>Operation Code</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>00000002</td>
</tr>
<tr>
<td></td>
<td>10000002</td>
</tr>
<tr>
<td></td>
<td>11000102</td>
</tr>
<tr>
<td></td>
<td>01001002</td>
</tr>
</tbody>
</table>

Example Controller Design

\[
\hat{r} = \max(R_1, R_2) + \begin{cases} 
0.75 & \text{if } C_1 = 0_2, C_2 = 0_2 \\
0.5 & \text{if } C_1 = 0_2, C_2 = 1_2 \\
0.25 & \text{if } C_1 = 1_2, C_2 = 0_2 \\
0 & \text{if } C_1 = 1_2, C_2 = 1_2 
\end{cases}
\]

Example Controller Design As described in Section III-B, the proposed architecture can be readily applied to any LUT-Log-BCJR decoder, regardless of the corresponding convolutional encoder parameters employed. This is achieved by specifically designing a controller for the LUT-Log-BCJR decoder. To exemplify this, we designed a controller for a sliding-window implementation of the LTE turbo code’s
LUT-Log-BCJR decoder, which corresponds to an encoder having memory elements. Since the proposed architecture employs parallel ACS units, it facilitates the parallel processing of state metrics at a time. As a result, “just-in-time” processing of the forward and backward recursions may be achieved, dispensing with the need for additional registers. This facilitates a reasonable throughput and a low energy consumption, as shown later in Section IV. Our controller meets the timing diagram of Fig. 5, which was designed to implement the sliding-window based LUT-Log-BCJR algorithm. To reduce the memory required for storing the state metrics of (1), the sliding-window implementation performs the forward and backward recursions of the LUT-Log-BCJR algorithm for windows of just bit indices. For the pre-backward recursion, windows of 24 bit indices are employed, as advocated in [27]. As shown in the columns of Fig. 5, both the forward and pre-backward recursions require 7 clock cycles per bit index, while the backward recursion requires 24 clock cycles. Observe that a total of clock cycles are required for processing a window of LLRs, which gives an average of 32.31 clock cycles per LLL. The activities of the ACS units and the two register banks are shown in the rows of Fig. 5, where both additions and subtractions require a single clock cycle, while the calculations require four clock cycles. The hardware inactivity during the extrinsic LLR calculation is caused by the data dependencies that are implied by (5), requiring an implementation using a binary tree structure of operations. As shown in Fig. 5, the proposed architecture performs the pre-backward recursion for just 24 of the 128 bit-indices in each window. By contrast, the conventional architectures typically perform the pre-backward recursion for all bit-indices in each window, as shown in Fig. 2(b). This therefore represents wastage, which is eliminated in the proposed architecture, giving an energy saving as discussed above. Moreover, the proposed architecture can be readily scaled to include either more or less ACS units, as well as reconfigured by adjusting the controller design. It can therefore be readily applied to other turbo code designs or decoding algorithms, such as the Viterbi algorithm or other variations of the Log-BCJR algorithm. For example, for a turbo code employing convolutional encoders having an input bit sequence and an output bit sequence, but a different number of memory elements, the optimal number of ACS units to include in the architecture is given by . Regardless of , the calculation of the state metrics or will still require the same seven clock cycles, as in Fig. 5, since the ACS units are capable of computing these in parallel, each employing one and two addition operations. Similarly, the calculation of the transition metrics will still require the same four clock cycles, as shown in Fig. 5, since each of the ACS units is capable of calculating a pair of transition metrics using three addition operations. Finally, the LLR calculation of Fig. 5 requires clock cycles, which is the duration required for carrying out operations and one subtraction. Since the specific choice of has little effect on the timing diagram of Fig. 5, it may be readily employed as the basis of the controller design for a wide variety of turbo code configurations.

VII. TURBO DECODER COMPLEXITY AND ENERGY ANALYSIS

To analyze the complexity and the energy efficiency of the proposed LUT-Log-BCJR architecture, we implemented an LTE turbo decoder using Taiwan Semiconductor Manufacturing Company (TSMC) 90 nm technology. The turbo decoder comprises four parts, namely a LUT-Log-BCJR decoder, an interleaver, a controller and the memory. The interleaver was implemented according to the latest low-complexity LTE interleaver designs [28], [29]. The memory employs one (128 64)-bit on-chip single-port SRAM module for storing the state metrics. Similarly, it employs five (6144 6)-bit on-chip single-port SRAM modules for storing the two sets of a priori LLRs, the two sets of extrinsic LLRs and the single set of systematic LLRs. The layout of the decoder is provided Fig. 6. Chip layout of the turbo decoder in Fig. 6. As shown in Fig. 6, the hardware complexity of the proposed architecture is so low that the chip area is actually dominated by the memory module, which consumes 40% of the overall energy consumption according to our post-layout simulation results. By contrast, the chip area of conventional LUT-Log-BCJR architectures is typically dominated by the decoder, despite employing similar amounts of memory. In Table III, we compare the proposed architecture to the latest LUT-Log-BCJR and Max-Log-BCJR decoder architectures [5], [6], [10], [11], [13]. The area and energy consumptions are estimated based on post-layout simulations. The implementation results arising from different technologies are also scaled3 to give a fair comparison. As shown in Table III, the energy consumption of the proposed architecture is significantly lower than that of the conventional LUT-Log-BCJR architectures. Furthermore, our proposed architecture has a similar energy
consumption to that of the recent Max-Log-BCJR decoders, but facilitates a 10% lower transmission energy, as discussed in Section I. To analyze the overall energy consumption of the LUT-Log-BCJR and the Max-Log-BCJR decoders, the BER performance of the proposed architecture and the ideal performance of the two types of the decoders are quantified in Fig. 7.4 Here, BPSK modulation is assumed, since it is widely adopted in the existing wireless sensor networks [30]. Furthermore, we assumed transmissions over a non-dispersive uncorrelated worst-case Rayleigh fading channel. As shown in Fig. 7, the BER performance of the proposed LUT-Log-BCJR architecture is within a tiny fraction of a decibel from that achieved by the ideal Log-BCJR algorithm. Furthermore, as discussed in Section I, the low complexity of the Max-Log-BCJR is achieved at the cost of requiring a 0.5 dB higher transmission energy per bit to achieve a BER of , as shown in Fig. 7. As a result, the LUT-Log-BCJR algorithm facilitates an overall energy consumption—including the energy consumed during both transmission and decoding—that is 10% lower than that of the Max-Log-BCJR at long transmission ranges, where the energy consumption of the turbo decoder is negligible compared.

**VIII. CONCLUSION**

In this paper, we demonstrated that upon aiming for a high throughput, conventional LUT-Log-BCJR architectures may have wasteful designs requiring high chip areas and hence high energy consumptions. However, in energy-constrained applications, achieving a low energy consumption has a higher priority than having a high throughput. This motivated our low-complexity energy-efficient architecture, which achieves a low area and hence a low energy consumption by decomposing the LUT-Log-BCJR algorithm into its most fundamental ACS operations. In addition, the proposed architecture may be readily reconfigured for different turbo codes or decoding algorithms. We validated the architecture by implementing an LTE turbo decoder, which was found, in Table III, to have an order-of-magnitude lower area than conventional LUT-Log-BCJR decoder implementations and an approximately 71% lower energy consumption of 0.4 nJ/bit/iteration. Compared to state of the art Max-Log-BCJR implementations, our approach facilitates a 10% reduction in the overall energy consumption at transmission ranges above 58 m. Furthermore, we demonstrated that our implementation has a throughput of 1.03 Mb/s, which is appropriate for energy-constrained applications, such as in environmental monitoring WSNs [2], [32].

**REFERENCES**


Mr. T NARASIMHA RAO received the Master of Technology degree in ECE from the SANA ENGINEERING COLLEGE-JNTUH, he received the Bachelor of Technology degree from SV ENGINEERING COLLEGE-JNTUH. He is currently working as assistant professor in the Department of ECE with Madhira Institute of Technology And Sciences, kodad. His interest subjects are VLSI, Digital Electronics, Digital Signal Processing and etc. Currently he is pursuing het M.Tech at Madhira institute of technology and sciences, Kodad, Nalgonda, Telangana. He interests are VLSI and video processing.