A Low-Complexity Turbo Decoder Architecture for Energy-Efficient Wireless Sensor Networks

Liang Li, Robert G. Maunder, Bashir M. Al-Hashimi and Lajos Hanzo
School of ECS, University of Southampton, SO17 1BJ, UK
Email: {ll08r, rm, bmah, lh}@ecs.soton.ac.uk

***Abstract*—Turbo codes have recently been considered for energy-constrained wireless communication applications, since they facilitate a low *transmission energy* consumption. However, in order to reduce the *overall* energy consumption, Look-Up-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 (Max-Log-BCJR) implementations, our approach facilitates a 10% reduction in the overall energy consumption at ranges above 58 m.**

***Index Terms*—energy-efficient, error-correcting code, turbo code, Log-BCJR algorithm**

I. INTRODUCTION

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 $1\ \mathrm{Mbit/s}$ [1], [2], the sensors' energy consumption is dominated by the transmission energy $E_{\mathrm{b}}^{\mathrm{tx}}$ (measured in $\mathrm{J/bit}$), since they may be separated by up to $1\ \mathrm{km}$. 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 $E_{\mathrm{b}}^{\mathrm{tx}}$. Note however that this reduction in $E_{\mathrm{b}}^{\mathrm{tx}}$ is offset by the turbo decoder's energy consumption $E_{\mathrm{b}}^{\mathrm{pr}}$, 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 $(E_{\mathrm{b}}^{\mathrm{tx}} + E_{\mathrm{b}}^{\mathrm{pr}})$.

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 $50\ \mathrm{Mbit/s}$ 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 $100\ \mathrm{Mbit/s}$ [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 $E_{\mathrm{b}}^{\mathrm{pr}}$ 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 $E_{\mathrm{b}}^{\mathrm{tx}}$ 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 $E_{\mathrm{b}}^{\mathrm{tx}} + E_{\mathrm{b}}^{\mathrm{pr}}$ of sensor nodes that are separated by dozens of meters.

This motivates the employment of the Look-Up-Table-Log-BCJR (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. Previous LUT-Log-BCJR turbo decoder designs [10]–[13] were developed as a part of the on-going drive for higher and higher processing throughputs, although their throughputs have since been eclipsed by the Max-Log-BCJR architectures. This opens the door for a new generation of LUT-Log-BCJR ASICs that exchange processing throughput for energy efficiency.

As we shall discuss in Section II, the energy consumption $E_{\mathrm{b}}^{\mathrm{pr}}$ of conventional LUT-Log-BCJR architectures cannot be significantly reduced by simply reducing their clock frequency and throughput. This motivates our novel architecture of Section III, which is specifically designed to have a minimal hardware complexity and hence a low energy consumption $E_{\mathrm{b}}^{\mathrm{pr}}$. In Section IV, we validate our architecture in the context of an LTE turbo decoder and demonstrate that it has an order of magnitude lower chip area, hence reducing the energy consumption $E_{\mathrm{b}}^{\mathrm{pr}}$ of the state-of-the-art LUT-Log-BCJR implementation by 71%. Compared to state-of-the-art Max-Log-BCJR implementations, our approach facilitates a 10% reduction in the overall energy consumption of $(E_{\mathrm{b}}^{\mathrm{tx}} + E_{\mathrm{b}}^{\mathrm{pr}})$ at transmission ranges above 58 m. Finally, Section V concludes the paper.

II. Conventional LUT-Log-BCJR architecture

As shown in Figure 1, a turbo encoder [14] comprises a parallel concatenation of two convolutional encoders, each of which has a structure comprising m number of memory elements, where $m = 3$ is used in the LTE encoders, for example. Each encoder converts an uncoded bit sequence $\mathbf{b}_1 = \{b_{1,j}\}_{j=1}^{N}$ into the corresponding encoded bit sequence $\mathbf{b}_2 = \{b_{2,j}\}_{j=1}^{N}$, where N is the length of the input bit sequences. Correspondingly, Figure 1 depicts a turbo decoder [15], [16], which comprises a parallel concatenation of two decoders, that employ the LUT-Log-BCJR algorithm. Rather than operating on bits, each LUT-Log-BCJR decoder processes Logarithmic Likelihood Ratios (LLRs) [14], where each LLR $\tilde{b} = \ln \frac{P(b=0)}{P(b=1)}$ quantifies the decoder's confidence concerning its estimate of a bit b from the bit sequences $\mathbf{b}_1$ and $\mathbf{b}_2$. Each LUT-Log-BCJR decoder processes two *a priori* LLR sequences, namely $\tilde{\mathbf{b}}_1^{\mathrm{a}} = \{\tilde{b}_{1,j}^{\mathrm{a}}\}_{j=1}^{N}$ and $\tilde{\mathbf{b}}_2^{\mathrm{a}} = \{\tilde{b}_{2,j}^{\mathrm{a}}\}_{j=1}^{N}$, which are converted into the extrinsic LLR sequence $\tilde{\mathbf{b}}_1^{\mathrm{e}} = \{\tilde{b}_{1,j}^{\mathrm{e}}\}_{j=1}^{N}$. This extrinsic LLR sequence is iteratively exchanged with that generated by the other LUT-Log-BCJR decoder, which is used as the *a priori* LLR sequence $\tilde{\mathbf{b}}_1^{\mathrm{a}}$ in the next iteration [17].

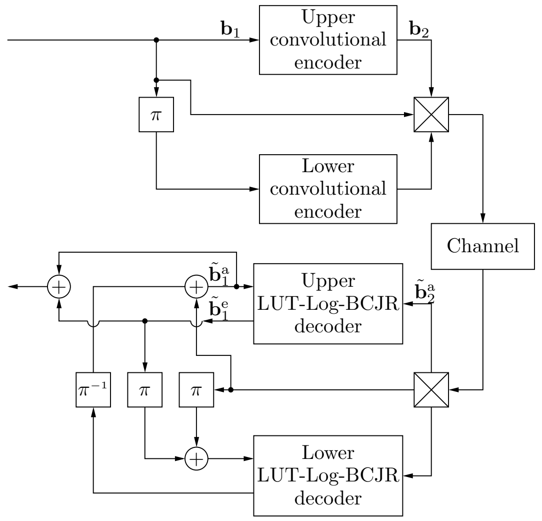
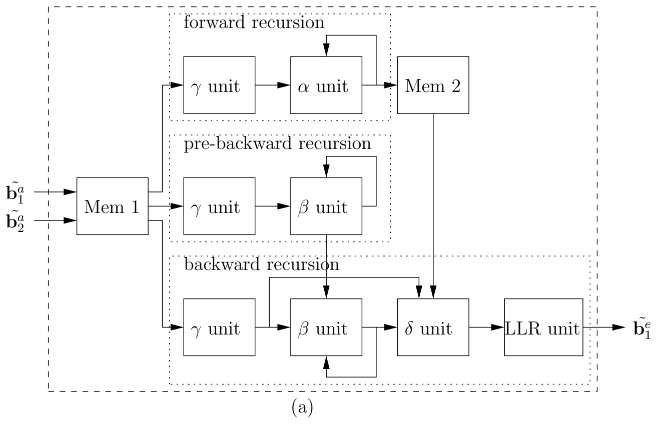

$\mathbf{b}_1$
Upper convolutional encoder
$\mathbf{b}_2$
π
Lower convolutional encoder
Channel
$\tilde{\mathbf{b}}_1^{\mathrm{a}}$
$\tilde{\mathbf{b}}_1^{\mathrm{e}}$
Upper LUT-Log-BCJR decoder
$\tilde{\mathbf{b}}_2^{\mathrm{a}}$
π^{-1}
π
π
Lower LUT-Log-BCJR decoder

Fig. 1. A turbo encoder and decoder scheme.

Figure 2 (a) depicts the conventional LUT-Log-BCJR architecture, which employs the sliding-window technique [18], [19] to generate the LLR sequence $\tilde{\mathbf{b}}_1^{\mathrm{e}}$ as the concatenation of W equal-length sub-sequences. Each of these windows is generated separately, using a forward, a pre-backward and a backward recursion, as shown in Figure 2. These three different recursions are performed concurrently for three different windows, as exemplified in Figure 2 (b) for $W = 5$. This schedule results in the completion of the windows in their natural order, starting with that containing the first LLR $\tilde{b}_{1,1}^{\mathrm{e}}$ and ending with the one containing the last LLR $\tilde{b}_{1,N}^{\mathrm{e}}$.

When the forward recursion is performed for a particular window, one pair of its corresponding *a priori* LLRs $\tilde{b}_{1,j}^{\mathrm{a}}$ and $\tilde{b}_{2,j}^{\mathrm{a}}$ is read from Mem 1 of Figure 2 (a) and processed per clock cycle, in the ascending order of the bit index j. The forward recursion of the LUT-Log-BCJR algorithm can



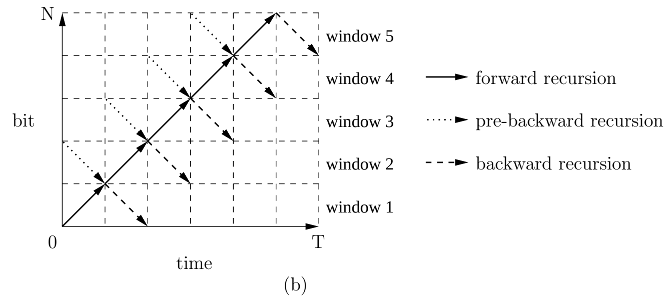


Fig. 2. (a) Conventional LUT-Log-BCJR architecture. (b) Timing of the sliding-window technique.

be performed in two pipelined steps using the corresponding dedicated hardware components of Figure 2 (a):

1) Firstly, the transition metrics $\mathbf{\Gamma}$ [20, Equation (2)], that correspond to the current window are generated. Here, each γ transition metric $\gamma_{i,j}(s, s')$ is set either equal to the corresponding *a priori* LLR $\tilde{b}_{i,j}^{\mathrm{a}}$ or to zero, depending on the particular pair of states s and s' that the transition is between and on the Generator Polynomials (GPs) of the encoder.
2) Next, the state metrics $\mathbf{A}$ [20, Equation (3)] that correspond to the current window are generated. Here, each α state metric is given by

$$\alpha_{j+1}(s') = \max_{s \to s'}{}^{*} \left(\alpha_j(s) + \sum_{i=1}^{2} \gamma_{i,j}(s, s') \right), \quad (1)$$

where, $s \to s'$ represents the set of all states s that can transition into the state s', 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 α 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 Figure 2. The $\max^*$ operation is used to represent the Jacobian logarithm detailed in [21], which may be approximated using a Look-Up Table (LUT) [17] for the parameters p and q according to

$$\max^*(\tilde{p},\tilde{q}) \approx \max(\tilde{p},\tilde{q}) + \begin{cases} 0.75 & \text{if } |\tilde{p}-\tilde{q}| = 0 \\ 0.5 & \text{if } |\tilde{p}-\tilde{q}| \in \{0.25, 0.5, 0.75\} \\ 0.25 & \text{if } |\tilde{p}-\tilde{q}| \in \{1, 1.25, 1.5, 1.75, 2\} \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

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 $z = 2$-bit fraction part. As a result, there are $2^z = 4$ entries in the LUT, each of which has values that are multiples of $2^{-z} = 0.25$. As we will show in Figure 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 α state metrics is written to Mem 2 of Figure 2 (a) per clock cycle in the ascending order of the bit index j.

When the backward recursion is performed for a particular window, one pair of its corresponding *a priori* LLRs $\tilde{b}^{\mathrm{a}}_{1,j}$ is read from Mem 1 of Figure 2 (a) and processed per clock cycle, in the descending order of the bit index j. Simultaneously, the corresponding set of α 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 Figure 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 Figure 2 (a):

1) Firstly, the transition metrics $\mathbf{\Gamma}$ that correspond to the current window are re-generated, as described above.
2) Next, the state metrics $\mathbf{B}$ [20, Equation (4)] that correspond to the current window are generated. Here, each β state metric is given by

$$\beta_{j-1}(s) = \max^*_{s \to s'} \left(\beta_j(s') + \sum_{i=1}^{2} \gamma_{i,j}(s,s') \right). \quad (3)$$

Note that the backward recursion for the last window is initialized independently. By contrast, the backward recursion for the other windows is initialized using β state metrics that were previously obtained during the pre-backward recursion of the next window. This is achieved using step 1 and 2 of the backward recursion and initializing the latter independently. It is for this reason that the pre-backward recursions of Figure 2 (b) are performed before the backward recursions of the preceding windows.
3) Next, the transition metrics $\mathbf{\Delta}$ [20, Equation (5)] that correspond to the current window are generated, according to

$$\delta_j(s,s') = \alpha_j(s) + \gamma_{2,j}(s,s') + \beta_j(s'). \quad (4)$$

4) Finally, the value of each extrinsic LLR in the current window of the sequence $\tilde{\mathbf{b}}^{\mathrm{e}}_1$ is generated according to

$$\tilde{b}^{\mathrm{e}}_{1,j} = \max^*_{s \overset{0}{\to} s'} [\delta_j(s,s')] - \max^*_{s \overset{1}{\to} s'} [\delta_j(s,s')], \quad (5)$$

where $s \overset{x}{\to} s'$ is the set of transitions that imply $b_{i,j}$ has a binary value of x.

As shown in Figure 2 (b), one extrinsic LLR $\tilde{b}^{\mathrm{e}}_{1,j}$ is output per clock cycle in descending order of the bit index j. By pipelining the forward, pre-backward and backward recursions using separate dedicated hardware for implementing the operations of Equations (1), (3), (4) and (5), the conventional architecture generates one extrinsic LLR per clock cycle, as shown in Figure 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[1] 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 Figure 2 have been proposed for further increasing the decoding throughput. For example, [25] employs parallel repetitions of the blocks shown in Figure 2 (a) to 'parallel-process' the schedule of Figure 2 (b). Alternatively, [12] employs a radix-4 variant, which processes two sets of α or β 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. While this would allow voltage scaling and a corresponding reduction of energy consumption, this approach would waste energy by powering the additional hardware that was introduced to manage the critical path. On the other hand, if voltage scaling is not employed, the limit on the critical path is relaxed, allowing the removal of the additional hardware that was introduced to manage it. While this facilitates a corresponding reduction in the *dynamic* energy consumption, the reduced throughput implies an increased *static* energy consumption, particularly in the case of high-density technologies. Furthermore, the lengthening of the critical path implies a greater variety of path lengths, particularly since the backward recursion path of Figure 2 (a) is significantly longer than those of the other recursions. This in turn implies that a greater fraction of the static energy consumption can be considered to be wasted, by giving short data paths more time to settle than necessary. In summary, efforts to slow down the conventional LUT-Log-BCJR architecture result in energy wastage, which cannot be avoided without completely redesigning the architecture.

III. Proposed LUT-Log-BCJR architecture

In this section, we propose a novel LUT-Log-BCJR architecture for energy-constrained scenarios, which avoids the

[1]This approach is analogous to using the faster but more complicated lookahead adder [23], instead of the slow but simple ripple carry adder.

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-lengthed, 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.

A. Decomposition of the LUT-Log-BCJR algorithm

Observe that Equations (1), (3), (4) and (5) of the LUT-Log-BCJR algorithm comprise only additions, subtractions and the $\max^*$ calculation of Equation (2). While each addition and subtraction constitutes a single ACS operation, each $\max^*$ calculation can be considered equivalent to four ACS operations, as shown in Table I. In the general case,

TABLE I
DECOMPOSITION OF $\max^*$ OPERATION.

Op 1	simultaneously calculate $\max(\tilde{p}, \tilde{q})$ and $\lvert\tilde{p}-\tilde{q}\rvert$
Op 2	determine if $\lvert\tilde{p}-\tilde{q}\rvert > 0.75$
Op 3	determine if $\lvert\tilde{p}-\tilde{q}\rvert > 0$ or $\lvert\tilde{p}-\tilde{q}\rvert > 2$, depending on the outcome of Operation 2
Op 4	add $\max(\tilde{p}, \tilde{q})$ to the value selected from the set {0.75, 0.5, 0.25, 0}

where $z > 0$ fraction bits are employed in the twos complement fixed-point LLR representation, a total of $(z + 2)$ ACS operations are required to carry out the $\max^*$ calculation. By contrast, only a single ACS operation is required when $z = 0$ or when employing the Max-Log-BCJR algorithm, which approximates the $\max^*$ by the $\max$ operation. Similarly, fewer ACS operation are required, when employing the Constant-Log-BCJR [26] algorithm. These alternative algorithms reduce the hardware complexity and increase the throughput, therefore reducing the energy consumption E_{b}^{pr}. However, this is achieved at the cost of requiring a higher transmission energy E_{b}^{tx} to achieve the same BER performance. As a result, these transformations are typically detrimental to the overall energy consumption of $(E_{\text{b}}^{\text{tx}} + E_{\text{b}}^{\text{pr}})$, as discussed in Section I.

B. Proposed energy-efficient LUT-Log-BCJR architecture

Inspired by the analysis of Section III-A, the proposed energy-efficient LUT-Log-BCJR architecture is shown in Figure 3. Unlike conventional architectures, it does not use separate dedicated hardware for the three recursions shown in Figure 2. Instead, our architecture implements the entire algorithm using 2^m ACS units in parallel, each of which performs one ACS operation per clock cycle. Furthermore, the

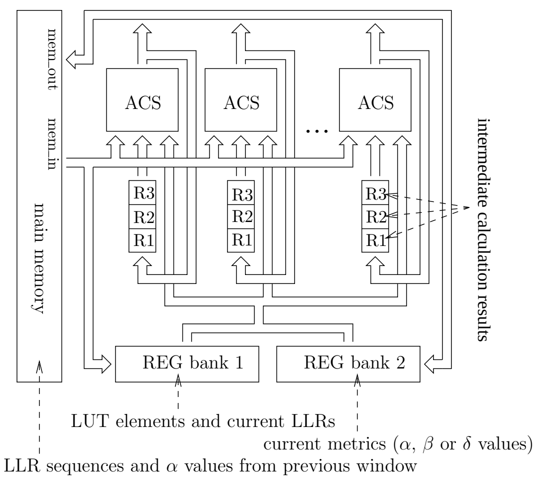

mem_out
mem_in
main memory
ACS
ACS
...
ACS
R3
R2
R1
R3
R2
R1
R3
R2
R1
intermediate calculation results
REG bank 1
REG bank 2
LUT elements and current LLRs
current metrics (α, β or δ values)
LLR sequences and α values from previous window

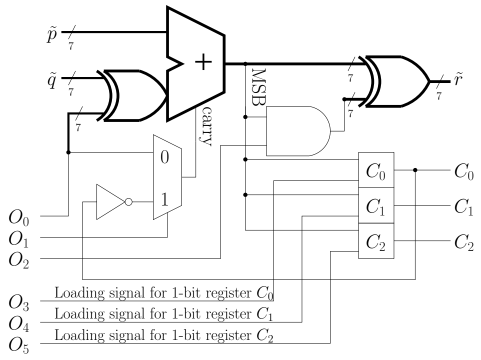
Fig. 3. Energy-efficient LUT-Log-BCJR architecture.

proposed architecture employs a twin-level register structure to minimize the highly energy-consuming main-memory access operations. At the first register level, each ACS unit is paired with a set of general purpose registers R1, R2 and R3. These are used to store intermediate results that are required by the same ACS unit in consecutive clock cycles. For example, this allows the four ACS operations equivalent to a $\max^*$ calculation to be performed in four consecutive clock cycles using a single ACS unit, as detailed in Section III-C. The second register level comprises REG bank 1 and REG bank 2 of Figure 3, which are used to temporarily store the LUT-Log-BCJR variables between consecutive values of the bit index j during the recursions decoding processes. The REG bank 1 comprises registers for the *a priori* LLRs $\tilde{b}_{1,j}^{\text{a}}$ and $\tilde{b}_{2,j}^{\text{a}}$ and dummy registers for the required LUT constants of Equation (2). Meanwhile, the sets of α, β or δ metrics are stored in REG bank 2 of Figure 3. The main memory stores all the required *a priori* LLR sequences and extrinsic LLR sequences during the decoding process and the α 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 Figure 3, it may be readily applied to any LUT-Log-BCJR decoder, regardless of the specific convolutional encoder parameters[2] employed. Note that in contrast to the different-length data paths of Figure 2 (a), the 2^m identical parallel data paths shown in Figure 3 have equal lengths, which avoids energy wastage, as described above.

[2]These parameters include the number of uncoded and encoded bit sequences, the constraint length and the GPs.

C. Novel ACS unit

In this section we propose the novel low-gate-count ACS unit of Figure 4, which performs one ACS operation per clock cycle. The control signals of the ACS unit are provided by the



$\tilde{p}$
$\tilde{q}$
7
+
MSB
carry
0
1
C_0
C_1
C_2
$\tilde{r}$
O_0
O_1
O_2
O_3
O_4
O_5
Loading signal for 1-bit register C_0
Loading signal for 1-bit register C_1
Loading signal for 1-bit register C_2

Fig. 4. ACS unit.

operation code $O = \{O_0, O_1, O_2, O_3, O_4, O_5\}$, which can be used to perform the functions listed in Table II. Note that

TABLE II
OPERATIONS OF THE ACS UNIT.

O	function
000000_2	$\tilde{r} = \tilde{p} + \tilde{q}$
100000_2	$\tilde{r} = \tilde{p} - \tilde{q}$
101100_2	$\tilde{r} = \begin{cases} \tilde{p} - \tilde{q} & \text{if } \tilde{p} \geq \tilde{q} \\ (\tilde{q} - \tilde{p}) - 0.25 & \text{if } \tilde{p} < \tilde{q} \end{cases}$ $C_0 = \begin{cases} 0_2 & \text{if } \tilde{p} \geq \tilde{q} \\ 1_2 & \text{if } \tilde{p} < \tilde{q} \end{cases}$
110010_2	$\tilde{r} = \begin{cases} \tilde{p} - \tilde{q} & \text{if } C_0 = 0_2 \\ \tilde{p} - \tilde{q} - 0.25 & \text{if } C_0 = 1_2 \end{cases}$ $C_1 = \begin{cases} 0_2 & \text{if } \tilde{r} \geq 0 \\ 1_2 & \text{if } \tilde{r} < 0 \end{cases}$
110001_2	$\tilde{r} = \begin{cases} \tilde{p} - \tilde{q} & \text{if } C_0 = 0_2 \\ \tilde{p} - \tilde{q} - 0.25 & \text{if } C_0 = 1_2 \end{cases}$ $C_2 = \begin{cases} 0_2 & \text{if } \tilde{r} \geq 0 \\ 1_2 & \text{if } \tilde{r} < 0 \end{cases}$

the operation code $O = 101100_2$ approximates the absolute difference between two operands, as required by Equation (2). Its result is equivalent to $\tilde{r} = |\tilde{p} - \tilde{q}|$ for $\tilde{p} \geq \tilde{q}$. However, for $\tilde{p} < \tilde{q}$, the result is given by $\overline{\tilde{p} - \tilde{q}}$. In the two's complement operand representation employing $z = 2$ fraction bits, this is equivalent to decrementing the binary representation of $(\tilde{q} - \tilde{p})$, which is equivalent to subtracting $2^{-z} = 0.25$. Note that a simpler ACS unit implementation is facilitated by this deliberately introduced inaccuracy, which can be trivially canceled out during the $\max^*$ calculation. More specifically, a $\max^*$ calculation can be performed with the following four operations, which store intermediate results in the registers R_1, R_2 and R_3, of Figure 3.

Op 1 In this clock cycle the $\max^*$ calculation is activated by using the operation code $O = 101100_2$ of Table II and loading operands $\tilde{p}$ and $\tilde{q}$ from the registers R_1 and R_2 of Figure 3, respectively. The result $\tilde{r}$ is then stored in register R_3, which is the approximated as $|R_1 - R_2|$. The result C_0 determines $\max(R_1, R_2)$.

Op 2 The LUT comparison performed during the second ACS operation is activated by the operation code $O = 110010_2$ of Table II. Operand $\tilde{p}$ uses the constant decimal value 0.75, which is provided by the register bank 1 in the architecture of Figure 3. Operand $\tilde{q}$ takes value from R_3, which is the approximated $|R_1 - R_2|$ that was obtained in the previous clock cycle. In this clock cycle, the result $\tilde{r}$ is not stored, while the result stored in C_1 provides the outcome of the test $|R_1 - R_2| > 0.75$, as required by the second ACS operation described in Section III-A.

Op 3 Similarly to the previous clock cycle, the result of the test $|R_1 - R_2| > 0$ or of the test $|R_1 - R_2| > 2$ is determined depending on whether it was previously decided that $|R_1 - R_2| > 0.75$. More specifically, we employ the operation code $O = 110001_2$ of Table II, use the value stored in R_3 for the ACS unit's operand $\tilde{q}$ and substitute the constant value of 0 or 2 for $\tilde{p}$, as appropriate. As shown in Equation (2), these constant values are the first and third entries of the LUT.

Op 4 The $\max^*$ calculation of Equation (2) is completed in the fourth clock cycle by using the operation code $O = 000000_2$ of Table II. Here the operand $\tilde{p}$ is provided by the maximum of R_1 and R_2, as identified by C_0 of Figure 4. Meanwhile, a value for the operand $\tilde{q}$ is selected from the set $\{0.75, 0.5, 0.25, 0\}$, depending on the contents of C_1 and C_2 of Figure 4. As a result, we have

$$\tilde{r} = \max(R_1, R_2) + \begin{cases} 0.75 & \text{if } C_1 = 0_2, C_2 = 0_2 \\ 0.5 & \text{if } C_1 = 0_2, C_2 = 1_2 \\ 0.25 & \text{if } C_1 = 1_2, C_2 = 0_2 \\ 0 & \text{if } C_1 = 1_2, C_2 = 1_2 \end{cases}, \tag{6}$$

as required by Equation (2).

D. 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 $m = 3$ memory elements. Since the proposed architecture employs $2^m = 8$ parallel ACS units, it facilitates the parallel processing of $2^m = 8$ α or β 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 Figure 5, which was designed to implement the sliding-window based LUT-Log-BCJR algorithm. To reduce the memory required for storing the α state metrics of Equation (1), the sliding-window implementation performs the forward and backward recursions of the LUT-Log-BCJR algorithm for windows of just $N = 128$ bit indices j. For the pre-backward recursion, windows of 24 bit indices j are employed, as advocated in [27]. As shown in

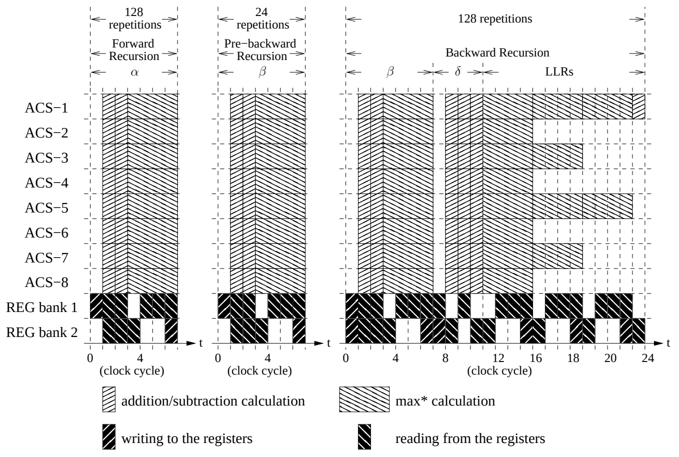


Fig. 5. Hardware activation schedule.

the columns of Figure 5, both the forward and pre-backward recursions require 7 clock cycles per bit index j, while the backward recursion requires 24 clock cycles. Observe that a total of $(7 \times 128 + 7 \times 24 + 24 \times 128)$=4136 clock cycles are required for processing a window of $N = 128$ LLRs, which gives an average of 32.31 clock cycles per LLR. The activities of the ACS units and the two register banks are shown in the rows of Figure 5, where both additions and subtractions require a single clock cycle, while the $\max^*$ calculations require four clock cycles. The hardware inactivity during the extrinsic LLR calculation is caused by the data dependencies that are implied by Equation (5), requiring an implementation using a binary tree structure of $\max^*$ operations.

As shown in Figure 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 Figure 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 $\mathbf{b}_1$ and an output bit sequence $\mathbf{b}_2$, but a different number of memory elements m, the optimal number of ACS units to include in the architecture is given by 2^m. Regardless of m, the calculation of the 2^m state metrics α or β will still require the same seven clock cycles, as in Figure 5, since the 2^m ACS units are capable of computing these in parallel, each employing one $\max^*$ and two addition operations. Similarly, the calculation of the 2^{m+1} δ transition metrics will still require the same four clock cycles, as shown in Figure 5, since each of the 2^m ACS units is capable of calculating a pair of δ transition metrics using three addition operations. Finally, the LLR calculation of Figure 5 requires $(4m + 1)$ clock cycles, which is the duration required for carrying out m $\max^*$ operations and one subtraction. Since the specific choice of m has little effect on the timing diagram of Figure 5, it may be readily employed as the basis of the controller design for a wide variety of turbo code configurations.

IV. 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 π, a controller and the memory. The interleaver π 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 α 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 in Figure 6. As shown in Figure 6, the hardware complexity of

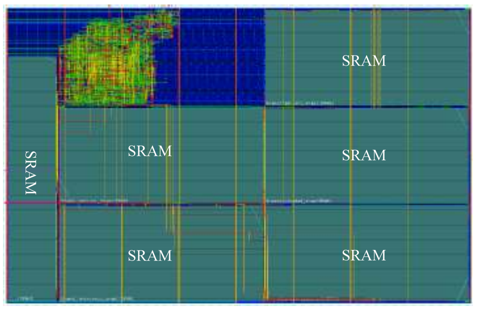


Fig. 6. Chip layout of the turbo decoder.

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 scaled[3] to give a fair comparison. As shown in Table III, the energy consumption E_b^{pr} 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 E_b^{pr} to that of the recent Max-Log-BCJR decoders, but facilitates a 10% lower transmission energy E_b^{tx}, as discussed in Section I.

To analyze the overall energy consumption $(E_b^{tx} + E_b^{pr})$ of the LUT-Log-BCJR and the Max-Log-BCJR decoders, the

[3]The energy consumption and area are adjusted using scaling factors of $1/s^3$ and $1/s^2$ respectively, where s is the ratio of the old technology scale to the new one [6].

TABLE III
COMPARISON OF THE IMPLEMENTED TURBO DECODER.

Publication	Proposed	[10]	[11]	[13]	[5]	[6]
Algorithm	LUT-Log	LUT-Log	LUT-Log	LUT-Log	Max-Log	Max-Log
Block size (bit)	6144	5114	5114	5114	6144	6144
Technology (nm)	90	180	180	180	65	120
Supply voltage (V)	1.0	1.8	1.8	1.8	-	1.2
Area A (mm^2) (Scaled for 90 nm)	0.35	9 (2.25)	14.5 (3.63)	8.2 (2.05)	2.1 (4.0)	3.57 (2.0)
Gate count (exclusive of memory)	7.5k	85k	410k	65k	-	553k
Memory required (kbit)	188	239	450	161	-	129
Clock frequency F (MHz)	333	111	145	100	300	390.6
Decoding iterations	5	10	8	6.5	6	5.5
Throughput T (Mb/s)	1.03	2	10.8	4.17	150	390.6
Power consumption (mW) (Scaled for 90 nm)	4.17	292 (36.5)	956 (119.4)	320 (40)	300 (796.4)	788.9 (332.8)
Energy consumption (nJ/bit/iteration) (Scaled for 90 nm)	0.4	14.6 (1.8)	11.1 (1.4)	12.7 (1.59)	0.31 (0.81)	0.37 (0.16)
$E_b^{tx} + E_b^{pr}$ (nJ/bit) when transmitting over 39 m (5 iterations)	10.16	17.16	15.16	16.06	13.42	10.17
$E_b^{tx} + E_b^{pr}$ (nJ/bit) when transmitting over 58 m (5 iterations)	41.92	48.92	46.92	47.82	49.88	46.63

BER performance of the proposed architecture and the ideal performance of the two types of the decoders are quantified in Figure 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 Figure 7, the BER performance of the proposed

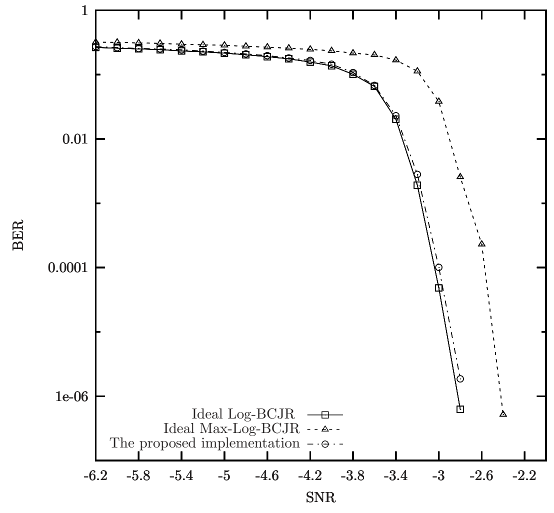


Fig. 7. BER performance of various decoding algorithms, in the case where 5 iterations are employed to decode a 6144-bit LTE block, which was transmitted over an uncorrelated Rayleigh fading channel.

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 10^{-4}, as shown in Figure 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 E_b^{pr} of the turbo decoder is negligible compared to the transmission energy E_b^{tx} required. Indeed, the analysis[5] of [3], [31] reveals that a small difference in BER performance has a significant effect on the overall energy consumption $(E_b^{tx} + E_b^{pr})$. As a result, the proposed architecture offers the lowest overall energy consumption when the transmission distance is beyond 39 m, as shown in Table III. Compared to the most energy efficient Max-Log-BCJR design [6] in Table III, which has an energy consumption of 0.16 nJ/bit/iteration, the proposed LUT-Log-BCJR decoder achieves more than 10% overall energy savings when the transmission distance reaches 58 m, as shown in Table III.

Indeed, Figure 8 shows the overall energy consumption difference between the Max-Log-BCJR of [6] and the proposed architecture, which is formulated as $f(d) = (E_b^{tx} + E_b^{pr})_{\text{Max-Log-BCJR}} - (E_b^{tx} + E_b^{pr})_{\text{LUT-Log-BCJR}}$. As indicated by negative values of $f(d)$ in Figure 8, the Max-Log-BCJR decoder of [6] has a (slightly) lower overall energy consumption than the proposed decoder when transmitting across short ranges of less than 39 m. By contrast, the proposed architecture offers a significant overall energy saving that increases exponentially beyond a range of 39 m, relative to the state-of-the-art Max-Log-BCJR decoder [6].

As discussed in Section III, the proposed architecture achieves an energy saving, because it efficiently employs a novel low-complexity ACS unit having a short critical path, which avoids the energy wastage that occurs in conventional architectures. As discussed in Section III-D, this principle may be generally applied to any arbitrary turbo code configuration, for achieving similar energy savings to those demonstrated for our example of the topical LTE LUT-Log-BCJR turbo decoder.

[4]Since different simulation parameters and channel models are used in previous publications, we compare the BER performance of our proposed architecture with the idealized upper-bound performance of the various algorithms, which was obtained using floating-point simulation.

[5]This analysis assumes a receiver noise figure of 5 dB, a power amplifier efficiency of 33%, a carrier frequency of 5.8 GHz and a worst-case path-loss exponent of 4.

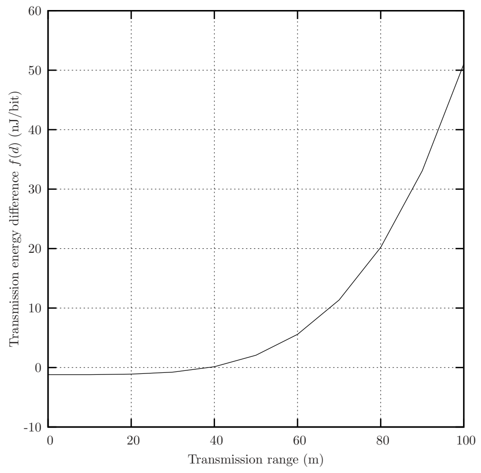


Fig. 8. The energy consumption difference between the Max-Log-BCJR decoder of [6] and the proposed architecture at BER = 10^{-4}.

V. Conclusions

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

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks: The International Journal of Computer and Telecommunications Networking*, vol. 52, pp. 292–422, 2008.

[2] P. Corke, T. Wark, R. Jurdak, H. Wen, P. Valencia, and D. Moore, "Environmental Wireless Sensor Networks," *Proceedings of the IEEE*, vol. 98, no. 11, pp. 1903–1917, 2010.

[3] S. L. Howard, C. Schlegel, and K. Iniewski, "Error Control Coding in Low-Power Wireless Sensor Networks: When is ECC Energy-Efficient?" *EURASIP Journal of Wireless Communications and Networking, Special Issue: CMOS RF Circuits for Wireless Applications*, vol. 2006, Arti, pp. 1–14, 2006.

[4] L. Li, R. G. Maunder, B. M. Al-Hashimi, and L. Hanzo, "An energy-efficient error correction scheme for IEEE 802.15.4 wireless sensor networks," *Transactions on Circuits and Systems II*, vol. 57, no. 3, pp. 233–237, 2010.

[5] M. May, T. Ilnseher, N. Wehn, and W. Raab, "A 150Mbit/s 3GPP LTE Turbo Code Decoder," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Dresden, Germany, 2010, pp. 1420–1425.

[6] C. Studer, C. Benkeser, S. Belfanti, and Q. Huang, "Design and Implementation of a Parallel Turbo-Decoder ASIC for 3GPP-LTE," *IEEE Jouranal of Solid-State Circuits*, vol. 46, pp. 8–17, 2011.

[7] C. Wong, Y. Lee, and H. Chang, "A 188-size 2.1mmˆ2 Reconfigurable Turbo Decoder Chip with Parallel Architecture for 3GPP LTE System," in *2009 Symposium on VLSI Circuits*, Kyoto, Japan, 2009, pp. 288–289.

[8] P. Robertson, P. Hoeher, and E. Villebrun, "Optimal and Sub-Optimal Maximum A Posteriori Algorithms Suitable for Turbo Decoding," *European Transactions on Telecommunications*, vol. 8, no. 2, pp. 119–125, 1997.

[9] W.-P. Ang and H. K. Garg, "A new iterative channel estimator for the log-MAP & max-log-MAP turbo decoder in Rayleigh fading channel," in *Global Telecommunications Conference*, vol. 6, San Antonio, TX , USA, 2001, pp. 3252–3256.

[10] M. A. Bickerstaff, D. Garrett, T. Prokop, C. Thomas, B. Widdup, G. Zhou, L. M. Davis, G. Woodward, C. Nicol, and R.-H. Yan, "A unified turbo/Viterbi channel decoder for 3GPP mobile wireless in 0.18-μm CMOS," *IEEE Journal of Solid-State Circuits*, vol. 37, no. 11, pp. 1555–1564, 2002.

[11] M. Bickerstaff, L. Davis, C. Thomas, D. Garrett, and C. Nicol, "A 24Mb/s radix-4 Log-MAP turbo decoder for 3GPP-HSDPA mobile wireless," in *IEEE International Solid-State Circuits Conference*, vol. 1, 2003, pp. 150–484.

[12] Z. Wang, "High-Speed Recursion Architectures for MAP-Based Turbo Decoders," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 15, no. 4, pp. 470–474, 2007.

[13] F.-M. Li, C.-H. Lin, and A.-Y. Wu, "Unified Convolutional/Turbo Decoder Design Using Tile-Based Timing Analysis of VA/MAP Kernel," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 16, no. 10, pp. 1063–8210, 2008.

[14] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon Limit Error Correcting Coding and Decoding: Turbo Codes," in *Proceedings of the IEEE International Conference on Communications*, vol. 2, Geneva, Switzerland, 1993, pp. 1064–1070.

[15] L. Hanzo, T. H. Liew, B. L. Yeap, R. Tee, and S. X. Ng, *Turbo Coding, Turbo Equalisation and Space-Time Coding*. John Wiley & Sons Inc, 2011.

[16] L. Hanzo, J. P. Woodard, and P. Robertson, "Turbo decoding and detection for wireless applications," in *Proceedings of the IEEE*, vol. 95, no. 6, 2007, pp. 1178–1200.

[17] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and Sub-optimal MAP decoding algorithms operating in the log domain," in *Proceedings of IEEE International Conference of Communication*, vol. 2, Seattle, WA, USA, 1995, pp. 1009–1013.

[18] C. Schurgers, F. Catthoor, and M. Engels, "Memory Optimization of MAP Turbo Decoder Algorithms," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 9, no. 2, pp. 305–312, 2001.

[19] G. Masera, M. Mazza, G. Piccinini, F. Viglione, and M. Zamboni, "Architectural Strategies for Low-Power VLSI Turbo Decoders," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 10, no. 3, pp. 279–285, 2002.

[20] C. M. Wu, M. D. Shieh, C. H. Wu, Y. T. Hwang, and J. H. Chen, "VLSI Architectural Design Tradeoffs for Sliding-Window Log-MAP Decoders," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 13, no. 4, pp. 439–447, 2005.

[21] A. J. Viterbi, "An Intuitive Justification and a Simplified Implementation of the MAP Decoder for Convolutional Codes," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 2, pp. 162–264, 1998.

[22] L. Li, R. G. Maunder, B. M. Al-Hashimi, and L. Hanzo, "Design of Fixed-Point Processing Based Turbo Codes Using Extrinsic Information Transfer Charts," in *Proceeding of IEEE Vehicular Technology Conference*, Ottawa, Canada, 2010, pp. 1–5.

[23] E. Boutillon, C. Douillard, and G. Montorsi, "Iterative Decoding of Concatenated Convolutional Codes: Implementation Issues," in *Proceedings of the IEEE*, vol. 95, no. 6, 2007, pp. 1201–1227.

[24] Y. Zhang and K. K. Parhi, "High-throughput radix-4 logMAP turbo decoder architecture," in *Asilomar conference on Signals, System and Computers*, Pacific Grove, CA, USA, 2006, pp. 1711–1715.

[25] Z. He, P. Fortier, and S. Roy, "Highly-Parallel Decoding Architectures for Convolutional Turbo Codes," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 14, no. 10, pp. 1063–8210, 2006.

[26] M. C. Valenti and J. Sun, “The UMTS turbo Code and an Efficient Decoder Implementation Suitable for Software-Defined Radios,” *International Journal of Wireless Information Networks*, vol. 8, no. 4, pp. 203–215, 2001.

[27] C. Benkeser, A. Burg, T. Cupaiuolo, and Q. Huang, “Design and Optimization of an HSDPA Turbo Decoder ASIC,” *IEEE Journal of Solid-State Circuits*, vol. 44, no. 1, pp. 98–106, 2009.

[28] S.-G. Lee, C.-H. Wang, and W.-H. Sheen, “Architecture Design of QPP Interleaver for Parallel Turbo Decoding,” in *IEEE Vehicular Technology Conference*, Taipei, Taiwan, 2010, pp. 1–5.

[29] Y. Sun and J. R. Cavallaro, “Efficient Hardware Implementation of A Highly-Parallel 3GPP LTE, LTE-Advance Turbo Decoder,” *Integration, the VLSI Journal*, vol. 44, no. 1, pp. 1–11, 2010.

[30] G. J. Pottie and W. J. Kaiser, “Wireless integrated network sensors,” *Communications of the ACM*, vol. 43, no. 5, pp. 51–58, May 2000.

[31] N. Sadeghi, S. Howard, S. Kasnavi, K. I. V. C. Gaudet, and C. Schlegel, “Analysis of error control code use in ultra-low-power wireless sensor networks,” in *Proceedings of International Symposium on Circuits and Systems*, Island of Kos, 2006, pp. 3558–3561.

[32] G. Barrenetxea, F. Ingelres, G. Schaefer, and M. Vetterli, “Wireless Sensor Networks for Environmental Monitoring: The SensorScope Experience,” in *IEEE International Zurich Seminar on Communications*, Zurich, 2008, pp. 98–101.

Liang Li received his B.S. degree in Mircoelectronics from Peking University, Beijing, China, in 2006 and his M.Sc. degree from University of Southampton, Uk, in 2008. He is currently working toward the Ph.D. degree with the Communications Research Group, School of Electronics and Computer Science, University of Southampton, Southampton, UK. His research interests include energy-efficient hardware architectures for turbo or LDPC codes and their applications in energy-constrained scenarios, such as wireless sensor networks.

Robert G. Maunder has studied with the School of Electronics and Computer Science, University of Southampton, UK, since October 2000. He was awarded a first class honors BEng in Electronic Engineering in July 2003, as well as a PhD in Wireless Communications and a lectureship in December 2007. Rob’s research interests include the implementation of joint source/channel coding, iterative decoding, irregular coding and modulation schemes. He has published a number of IEEE papers in these areas.

Bashir M. Al-Hashimi (M99-SM01-F09) received the B.Sc. degree (with 1st-class classification) in Electrical and Electronics Engineering from the University of Bath, UK, in 1984 and the Ph.D. degree from York University, UK, in 1989. Following this he worked in the microelectronics design industry and in 1999, he joined the School of Electronics and Computer Science, Southampton University, UK, where he holds the Endowment ARM Chair in Computer Engineering. He has authored one book on SPICE simulation, (CRC Press, 1995), and coauthored two books, Power Constrained Testing of VLSI circuits (Springer, 2002), and System-Level Design Techniques for Energy-Efficient Embedded Systems (Springer, 2004). He edited the book, System-on-Chip: Next Generation Electronics (IEE Press, 2006). He has published over 260 refereed papers in journals conference proceedings. He is a Fellow of the IEE and a Fellow of the British Computer Society. He is the Editor-in-Chief of the IEE Proceedings: Computers and Digital Techniques, and a member of the editorial board of the Journal of Electronic Testing: Theory and Applications (JETTA), and Journal of Low Power Electronics. He was the General Chair of the 11th IEEE European Test Symposium (UK 2006), and the General Chair of DATE 2011. He is the coauthor of two Best Paper Awards: the James Beausang at the ITC 2000, relating to low power BIST for RTL data paths, and at the CODES-ISSS Symposium 2009, relating to low-energy fault-tolerance techniques. He is a co-author of a paper on test data compression which has been selected for a Springer book featuring the most influential work over the ten years of the DATE conference.

Lajos Hanzo (http://www-mobile.ecs.soton.ac.uk) FREng, FIEEE, FIET, Fellow of EURASIP, DSc received his degree in electronics in 1976 and his doctorate in 1983. In 2009 he was awarded the honorary doctorate “Doctor Honaris Causa” by the Technical University of Budapest. During his 35-year career in telecommunications he has held various research and academic posts in Hungary, Germany and the UK. Since 1986 he has been with the School of Electronics and Computer Science, University of Southampton, UK, where he holds the chair in telecommunications. He has successfully supervised in excess of 70 PhD students, co-authored 20 John Wiley/IEEE Press books on mobile radio communications totalling in excess of 10 000 pages, published 1200+ research entries at IEEE Xplore, acted both as TPC and General Chair of IEEE conferences, presented keynote lectures and has been awarded a number of distinctions. Currently he is directing an academic research team, working on a range of research projects in the field of wireless multimedia communications sponsored by industry, the Engineering and Physical Sciences Research Council (EPSRC) UK, the European IST Programme and the Mobile Virtual Centre of Excellence (VCE), UK. He is an enthusiastic supporter of industrial and academic liaison and he offers a range of industrial courses. He is also a Governor of the IEEE VTS. Since 2008 he has been the Editor-in-Chief of the IEEE Press and since 2009 a Chaired Professor also at Tsinghua University, Beijing. In 2012 he became one of four EURASIP Fellows. For further information on research in progress and associated publications please refer to http://www-mobile.ecs.soton.ac.uk