IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 5, 2016, accepted January 26, 2016. Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2016.2523063

Stochastic Computing Improves the Timing-Error Tolerance and Latency of Turbo Decoders: Design Guidelines and Tradeoffs

ISAAC PEREZ-ANDRADE, SHIDA ZHONG, ROBERT G. MAUNDER, BASHIR M. AL-HASHIMI, (Fellow, IEEE), AND LAJOS HANZO, (Fellow, IEEE)
Electronics and Computer Science, University of Southampton, Southampton, SO17 1BJ, U.K.

Corresponding author: L. Hanzo (lh@ecs.soton.ac.uk)

This work was supported in part by the European Research Council within the Beam-Me-Up Project, in part by the Engineering and Physical Sciences Research Council, Swindon, U.K., under Grant EP/J015520/1 and Grant EP/L010550/1, and in part by Technology Strategy Board, Swindon, U.K., under Grant TS/L009390/1. The work of I. Perez-Andrade was supported by the Consejo Nacional De Ciencia Y Tecnología, Mexico, within the Secretaría de Educación Pública y del Gobierno Mexicano.
The research data for this paper is available at http://dx.doi.org/10.5258/SOTON/375728

ABSTRACT Stochastic computing has been recently proposed for the hardware implementation of both low-density parity-check (LDPC) decoders and turbo decoders, which facilitate near-optimal error correction capabilities in wireless communication applications. Previous contributions have demonstrated that stochastic LDPC decoders offer an attractive tradeoff between their error correction capabilities, hardware performance, and timing-error tolerance. Motivated by this, we propose a pair of stochastic turbo decoder (STD) designs having significantly enhanced timing-error tolerance and significantly reduced processing latency. Moreover, we characterize the tradeoffs between chip area, energy efficiency, latency, throughput, and error correction capabilities of both the timing-error-tolerant STD and of the reduced-latency STD. We demonstrate that our proposed timing-error-tolerant STD operated at 1.20 V, with a clock period of 2.2 ns and in the presence of a three-standard deviation power supply variation of 7%, exhibits an unimpaired performance, compared with the state-of-the-art STD, operated at 1.20 V and 4 ns and with no power supply variations. This corresponds processing throughput improvement by a factor of 2.42 and energy consumption reduction by a factor of 4. Finally, we demonstrate that our proposed reduced-latency STD has a processing latency that is an order of magnitude lower than that of the state-of-the-art STD. This is despite reducing the chip area by a factor of 4, increasing the processing throughput by a factor of 65, while consuming only 0.005 times the energy of the state-of-the-art STD, when using binary phase-shift keying for communication over an additive white Gaussian noise channel having $E_b/N_0 = 3$ dB.

INDEX TERMS ASIC, fault tolerance, low-latency, power supply variation, stochastic computing, timing errors, turbo decoder.

NOMENCLATURE

APP	A Posteriori Probability
ASIC	Application Specific Integrated Circuit
AWGN	Additive White Gaussian Noise
BER	Bit Error Ratio
BCJR	Bahl-Cocke-Jelinek-Raviv
BPSK	Binary Phase Shift Keying
BS	Bernoulli Sequence
CG	Clock Gating
CRC	Cyclic Redundancy Check
DC	Decoding Cycle
DFF	D-type Flip Flop
EM	Edge Memory
FPGA	Field-Programmable Gate Array
JKFF	JK-type Flip Flop
LDPC	Low-Density Parity-Check
LSB	Least Significant Bit
LLR	Logarithmic Likelihood Ratio
LogBCJR	Logarithmic BCJR
LTE	Long-Term Evolution
MCMTC	Mission-Critical Machine-Type Communication
MSB	Most Significant Bit
MUX	Multiplexer
NDS	Noise-Dependent Scaling
RCTFM	Reduced-Complexity Tracking Forecast Memory
RLSTD	Reduced-Latency Stochastic Turbo Decoder

SNR	Signal-to-Noise Ratio
SR	Shift Register
STD	Stochastic Turbo Decoder
TFM	Tracking Forecast Memory
VLSI	Very-Large-Scale Integration.

I. INTRODUCTION

Both Low-Density Parity-Check (LDPC) [1] codes and turbo codes [2] are widely used for error correction in wireless communications standards such as WiMAX [3], WiFi [4], DVB-S2 [5], CDMA2000 [6], UMTS [7] and LTE [8]. Traditional hardware implementations of LDPC decoders and turbo decoders rely on fixed-point binary arithmetic for representing and processing the probabilities of the received bits having the value 0 or 1. However, practical fixed-point LDPC and turbo decoder impose a high complexity owing to the interconnection and routing problems of fully-parallel designs [9], [10], or due to the large amount of memory of both partially-parallel and of fully-serial designs [11], [12]. Stochastic computing [13] has been proposed as a low-complexity design alternative to fixed-point binary arithmetic, where the probabilities are represented by streams of bits, known as Bernoulli Sequences (BSs) [13]. Only a single bit of each BS is processed per clock cycle and the specific fraction of bits having the logical value 1 determines the value of the probability represented. As a benefit of this, arithmetic operations in stochastic computing can be implemented using low-complexity digital circuits.

Timing errors in synchronous systems occur when the delay in a circuit path exceeds the clock period, owing to variations in the operating conditions of the system, such as the supply voltage or clock period. Furthermore, timing errors are more likely to occur when the supply voltage is reduced and the clock period is not adjusted accordingly, owing to the quadratic dependency of propagation delays on the supply voltage. As a result of this, the error correction capabilities of fixed-point LDPC decoders and turbo decoders in the presence of timing errors can be severely affected, unless sophisticated fault-tolerant design techniques are considered. More specifically, fixed-point implementations are particularly vulnerable to timing errors affecting the Most Significant Bit (MSB) of the fixed-point numbers, since this radically alters the represented bit value probabilities. By contrast, stochastic computing exhibits an inherent tolerance to processing errors. Since every bit of a BS has an identical significance, a processing error causing a single bit-flip will only result in a small change to the overall value of the probability represented. This represents a key advantage of the stochastic implementations of error correction decoders, particularly as Very-Large-Scale Integration (VLSI) technology scales reduce and the prevalence of timing errors increases, owing to the effects of IR-drop and $L \cdot d_i/d_t$, manufacturing imperfections and external factors such as crosstalk, electrostatic discharges and electromagnetic interference [14]–[18], among others. However, stochastic decoders require a large number of clock cycles in order to achieve the same near-optimal error correction capability as their fixed-point implementation counterparts. Regretfully, this degrades their latency, throughput and energy efficiency. Owing to these impediments, stochastic decoders have been deemed unsuitable for practical low-latency next-generation Mission-Critical Machine-Type Communication (MCMTC) systems, such as those required by vehicular traffic safety and control, as well as industrial process automation and manufacturing [19]. In these applications, short emergency and control messages constituted by a low number of bits must be reliably transmitted with ultra-low latency, hence motivating the employment of error correction decoders having ultra-low processing latencies on the order of microseconds [19]. Motivated by this background, this paper proposes a pair of stochastic turbo decoder designs having significantly enhanced timing error tolerance and significantly reduced processing latency, as discussed in the following subsections.

A. RELATED WORK

The following literature review is based on the fields of fault-tolerant design in iterative decoders and hardware implementation of stochastic iterative decoders, where Figure 1 summarizes some of the seminal contributions in these fields.

Stochastic computing has been recently proposed for the fully-parallel decoding of LDPC codes [20]–[32], Bose-Chaudhuri-Hocquenghem codes [33], Reed-Solomon codes [33], cortex codes [34], convolutional codes [35] and turbo codes [36]–[39]. Most of these contributions have focused on the employment of stochastic computing in traditional iterative decoding algorithms, where the main design objective has been to maximize the attainable error correction capabilities. In parallel to this, fault-tolerance in iterative decoders has mostly been explored from the algorithmic point of view in [40]–[48], with only few contributions [49]–[53] having considered the practical implications of fault-tolerant hardware design. Within these contributions, different approaches have been explored in order to improve the decoder's fault-tolerance. For example, hardware redundancy and voting units are employed in [49] for error detection and correction. Similarly, processing errors are corrected on the basis of probabilistic analyses in [50] and [51]. By contrast, our previous contribution [52] was the first one in the open literature to explore the inherent tolerance of stochastic LDPC decoders to timing errors, both in the presence of clock- and voltage-scaling.

Substantial research efforts have been invested in reducing the number of clock cycles required by stochastic decoders for achieving a near-optimal error correction performance. In particular, the authors of [24], [25], [27], and [30] considered the employment of Noise-Dependent Scaling (NDS) and Edge Memories (EMs) in stochastic LDPC decoders, reducing the number of clock cycles required to achieve near-optimal error correction performance from several thousands

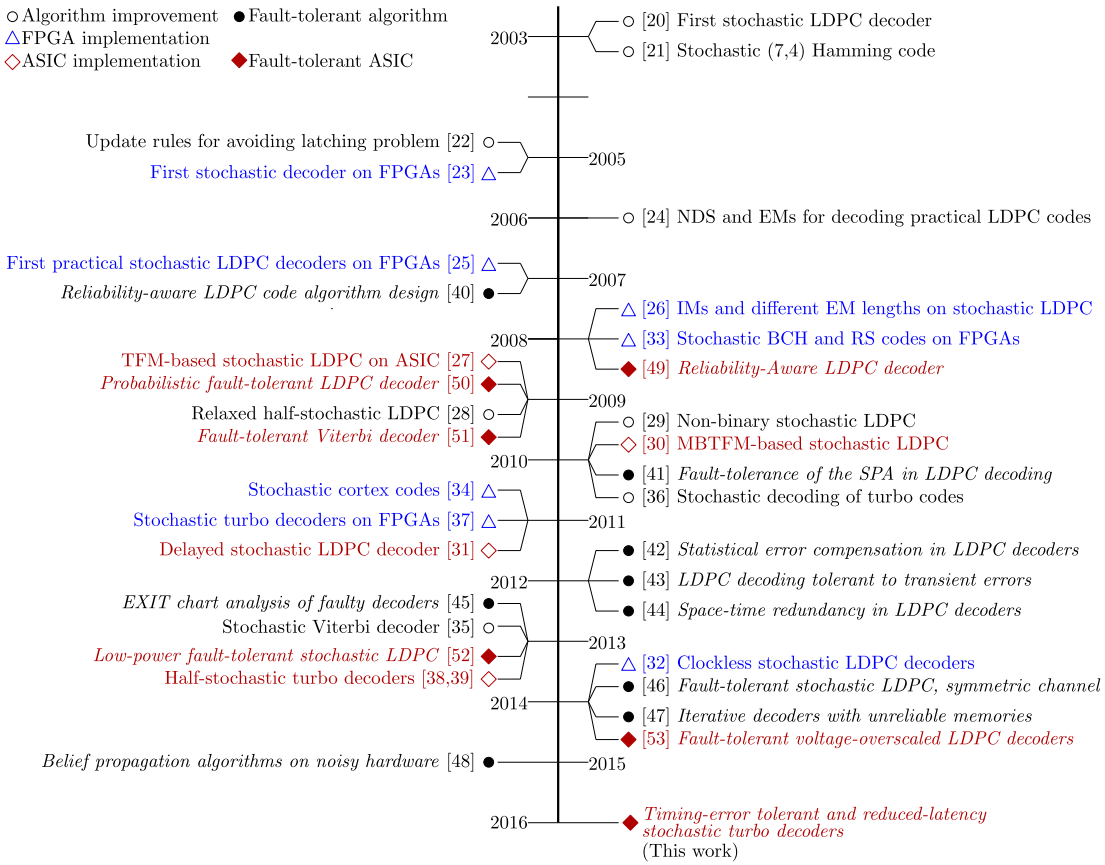


FIGURE 1. Selected previous contributions in the field of hardware implementation of stochastic iterative decoders and fault-tolerant design for iterative decoders.

to as low as a few hundreds. By contrast, the fully-parallel Stochastic Turbo Decoder (STD) of [36] requires several thousands of processing cycles, despite the employment of NDS and EMs. This problem is partially overcome in [36] and [37] by further increasing the grade of processing parallelism. However, this significantly increases the hardware complexity of the STD, as it will be detailed is Section VIII. The authors of [38] and [39] proposed modified BS representations for the implementation of STDs. These contributions significantly reduced the number of clock cycles required without significantly increasing the hardware complexity of the design. However, these designs may be considered to represent partially-stochastic turbo decoders, since their operation is based on fixed-point arithmetic circuits, which do not benefit from the inherent tolerance of stochastic decoders to timing errors.

B. CONTRIBUTIONS AND PAPER STRUCTURE

The previous work mentioned in Section I-A has mainly focused on enhancing the error correction capabilities of stochastic decoders and separately of fault-tolerant iterative decoders. However, fault-tolerance in stochastic decoders has only been addressed in our previous work [52] for the case of stochastic LDPC decoders. Moreover, the above-mentioned contributions in the field of fault-tolerant iterative decoders

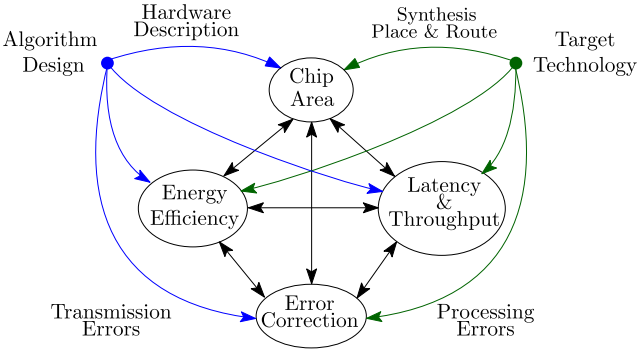


FIGURE 2. Relationship between different design trade-offs.

have failed to provide a comprehensive analysis of the various design trade-offs involved in the hardware implementation of error tolerant techniques, which is a common approach in performance-oriented design methodologies [54]. To elaborate further, the main design objective of these contributions is to enhance both the robustness and reliability of the decoders against processing errors. However, this is typically reported without considering the impact of fault-tolerant techniques on the associated hardware design trade-offs, such as the chip area, energy efficiency, latency, throughput and error correction capabilities, despite the strong dependence they have upon each other, as depicted in Figure 2. In this context,

a compelling Pareto-optimal design has a set of characteristics, where none of them can be further improved without degrading at least one of the others. In addition to this, the techniques introduced in [36] and [37] for reducing the number of processing cycles required by the STD impose a potentially excessive hardware complexity increase. As a result, the proposed STDs may not be suitable for practical next-generation low-latency MCMTC systems.

Against this background, we propose enhancements of the STD of [36] for improving its hardware characteristics. We first improve its tolerance to timing errors in the presence of power supply variations. More specifically, the first enhanced STD design proposes:

1.1) *The employment of synchronizers for preventing the catastrophic cascading of metastability owing to timing errors.*
1.2) *The simultaneous decoding of two received frames for improving the processing throughput.*
1.3) *The employment of Tracking Forecast Memories (TFMs) [27] in STDs for the first time, in order to enhance their hardware implementation and decoding capabilities.*
1.4) *The inclusion of a pipelining stage for enhancing the decoding capabilities of the STD in the presence of power supply variations.*

In parallel to this, we present a second improved STD design, which reduces the number of clock cycles required for achieving near-optimal error correction performance by an order of magnitude, without increasing the chip area. This is achieved by employing:

2.1) *OR gates for performing approximate stochastic additions.*
2.2) *A reduced-complexity TFM design for overcoming the latching problem.*
2.3) *A single D-type Flip Flop (DFF) for estimating each decoded bit.*

Moreover, we analyze the different trade-offs presented in Figure 2 for both of the improved STD designs.

The amalgam of those two STD designs yields significant performance enhancements, which substantially improve the appeal of STDs in practical next-generation timing-error-tolerant and low-latency MCMTC systems.

The remainder of this paper is structured as depicted in Figure 3. Section II reviews the operation of turbo decoders. Section III reviews the concept of stochastic computing in iterative decoding. Section IV details the hardware implementation of STDs. Note that experts of turbo decoders, stochastic computing and stochastic turbo decoding may wish to skip the comprehensive tutorials of Sections II to IV, respectively. Sections V to VII present the timing-error tolerant STD design. More specifically, Section V describes how to improve the STD's tolerance to timing errors, Section VI details its hardware implementation trade-offs, while Section VII characterizes its error correction performance in the presence of timing errors. Following this, Sections VIII to X portray our reduced-latency STD design.

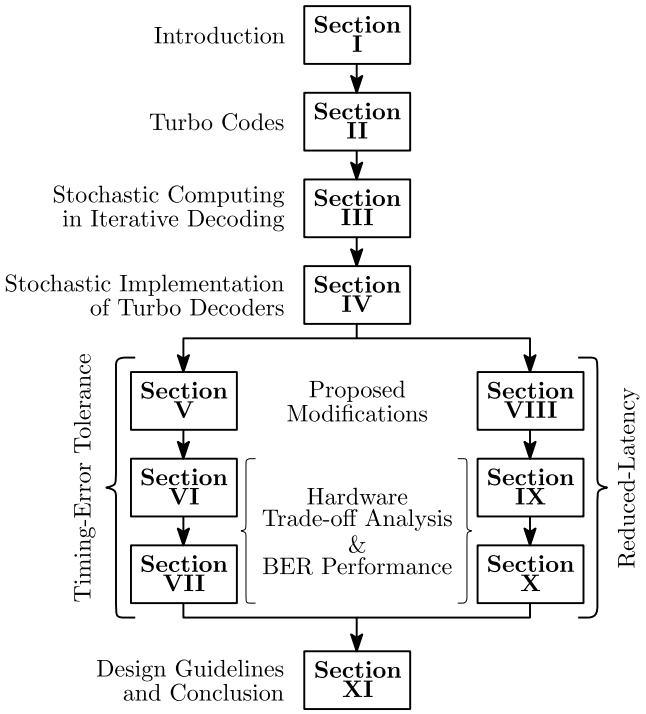


FIGURE 3. Summary of paper structure.

More specifically, Section VIII details how to reduce the number of clock cycles required by the decoder, Section IX quantifies its error correction performance and Section X characterizes its hardware efficiency. Finally, Section XI presents our concluding remarks and offers design guidelines for STDs.

II. TURBO CODES

In this section, we review the concepts of turbo encoding and turbo decoding [2] presented in Figure 4. A turbo code comprises the parallel concatenation of two convolutional codes. As a result of this, a message frame comprising N bits $\mathbf{b}_1^{\mathrm{u}} = [b_{1,k}^{\mathrm{u}}]_{k=1}^{N}$ can be turbo encoded with the aid of the two parallel concatenated convolutional encoders, as shown in Figure 4a. Each convolutional encoder operates in the same manner, with the upper convolutional decoder having the bits $\mathbf{b}_1^{\mathrm{u}}$ for its input. However, these bits are reordered by the interleaver Π to provide $\mathbf{b}_1^{\mathrm{l}}$, which is then input into the lower encoder. Here, the superscripts 'u' and 'l' indicate relevance to the upper and lower convolutional encoders, respectively. However, these superscripts will be omitted, when the discussion applies equally to both convolutional encoders.

Each convolutional encoder operates on the basis of a trellis such as the 8-state Long-Term Evolution (LTE) trellis of Figure 4c. Here, each bit of $\mathbf{b}_1$ triggers one of 16 different transitions among these states and the encoding of the first bit $b_{1,1}$ commences from a particular previous state s'. If tailbiting [55] is employed, then this initial state is carefully selected such that it is equal to the final state s reached after encoding the final bit $b_{1,N}$. The simplified turbo encoder of

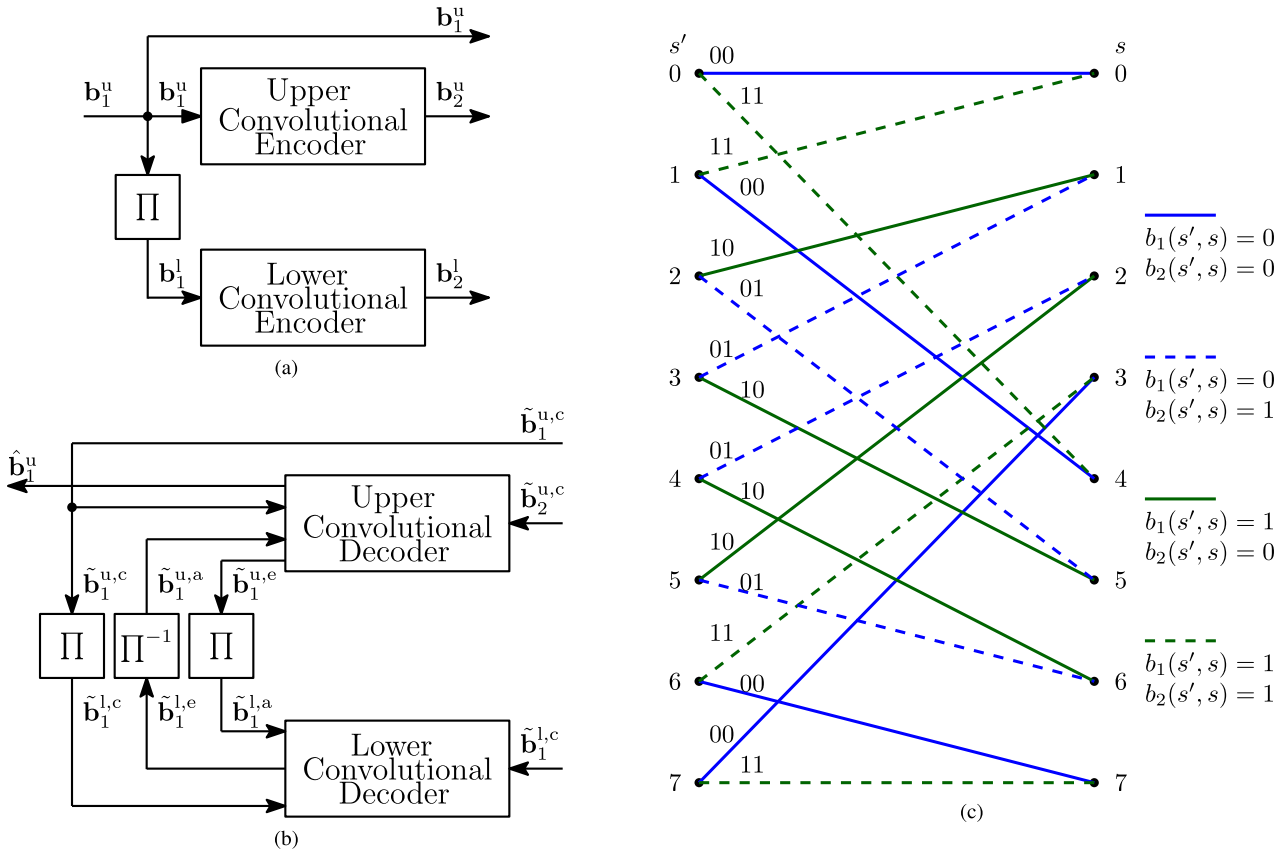


FIGURE 4. **(a) Simplified turbo encoder. (b) Conventional structure of a turbo decoder. (c) State transition diagram of the LTE turbo code.**

Figure 4a outputs a frame of systematic bits $\mathbf{b}_1^{\mathrm{u}} = [b_{1,k}^{\mathrm{u}}]_{k=1}^{N}$ and two frames of parity bits $\mathbf{b}_2^{\mathrm{u}} = [b_{2,k}^{\mathrm{u}}]_{k=1}^{N}$ and $\mathbf{b}_2^{\mathrm{l}} = [b_{2,k}^{\mathrm{l}}]_{k=1}^{N}$, which are provided by the upper and lower convolutional encoder, respectively. After their transmission over a wireless channel, the received frames $\tilde{\mathbf{b}}_1^{\mathrm{u,c}} = [\tilde{b}_{1,k}^{\mathrm{u,c}}]_{k=1}^{N}$, $\tilde{\mathbf{b}}_2^{\mathrm{u,c}} = [\tilde{b}_{2,k}^{\mathrm{u,c}}]_{k=1}^{N}$ and $\tilde{\mathbf{b}}_2^{\mathrm{l,c}} = [\tilde{b}_{2,k}^{\mathrm{l,c}}]_{k=1}^{N}$ are entered into the turbo decoder, which is comprised of two convolutional decoders, as shown in Figure 4b. The upper convolutional decoder of Figure 4b employs the received frames $\tilde{\mathbf{b}}_1^{\mathrm{u,c}}$, $\tilde{\mathbf{b}}_2^{\mathrm{u,c}}$ and the frame of *a priori* soft bits $\tilde{\mathbf{b}}_1^{\mathrm{u,a}}$ provided by the lower convolutional decoder, in order to provide the frame of *extrinsic* soft bits $\tilde{\mathbf{b}}_1^{\mathrm{u,e}}$. Following this, the *extrinsic* soft bits $\tilde{\mathbf{b}}_1^{\mathrm{u,e}}$ are interleaved and passed to the lower convolutional decoder as the frame of *a priori* soft bits $\tilde{\mathbf{b}}_1^{\mathrm{l,a}}$. The lower convolutional decoder employs the received frame $\tilde{\mathbf{b}}_2^{\mathrm{l,c}}$ and the interleaved received frame $\tilde{\mathbf{b}}_1^{\mathrm{l,c}}$ to provide the *extrinsic* soft bits $\tilde{\mathbf{b}}_1^{\mathrm{l,e}}$, which are de-interleaved in the block $\prod^{-1}$ and passed as the frame of *a priori* soft bits $\tilde{\mathbf{b}}_1^{\mathrm{u,a}}$ to the upper convolutional decoder. These soft bits express not only *what* the most likely value of the corresponding bits are, but also *how* likely these bit values are. More specifically, each soft bit $\tilde{b}_{1,k}^{\mathrm{u,c}}$ expresses the two probabilities $P^{\mathrm{c}}(b_{1,k}^{\mathrm{u,c}} = 0)$ and $P^{\mathrm{c}}(b_{1,k}^{\mathrm{u,c}} = 1)$, where the subscripts and superscripts may be replaced for the case of the other soft bits in Figure 4b.

The convolutional decoders are iteratively operated on the basis of the Bahl-Cocke-Jelinek-Raviv (BCJR) algorithm [56], which comprises Equations (1) to (6), as shown at the top of the next page. The BCJR algorithm employs (1) for calculating a branch metric $\gamma_k(s', s)$ for each transition of Figure 4c from a previous state s' into the next state s. Here, $P^{\mathrm{a}}(b_{1,k} = b_1(s', s))$ is the probabilities that are expressed by the *a priori* soft bit $\tilde{b}_{1,k}^{\mathrm{a}}$ provided by the other convolutional decoder. Note that at the start of the iterative decoding process, $P^{\mathrm{a}}(b_{1,k} = b_1(s', s)) = 0.5$ is assumed. The *extrinsic* branch metrics $\gamma_k^{\mathrm{e}}(s', s)$ of (2) correspond to the received probability of the parity bit $b_{2,k}$ provided by the other convolutional encoder. Following this, the state metrics $\alpha_k(s)$ of (3) and $\beta_k(s')$ of (4) are calculated for quantifying the probabilities associated with each of the 8 possible previous and next states of Figure 4c. Note that $\alpha_{k-1}(s') = 1/8$ and $\beta_k(s) = 1/8$ is assumed for all s' and s at the start of the iterative decoding process. If tailbiting [55] is employed, then $\alpha_0(s') = \alpha_N(s')$ and $\beta_N(s) = \beta_0(s)$ may be employed. Furthermore, (5) is employed for determining the probabilities $P^{\mathrm{e}}(b_{1,k} = 0)$ and $P^{\mathrm{e}}(b_{1,k} = 1)$, which are expressed by the *extrinsic* soft bit $\tilde{b}_{1,k}^{\mathrm{e}}$. Finally, (6) is employed for determining the *a posteriori* hard decision $\hat{b}_{1,k}$, pertaining to the bit $b_{1,k}$. This iterative process is repeated until an accurate estimation of the decoded frame $\hat{\mathbf{b}}_1$ can be obtained or until the maximum affordable number of iterations has been reached.

$$\gamma_k(s', s) = P^{\mathrm{a}}(b_{1,k} = b_1(s', s))P^{\mathrm{c}}(b_{1,k} = b_1(s', s))P^{\mathrm{c}}(b_{2,k} = b_2(s', s)) \quad (1)$$

$$\gamma_k^{\mathrm{e}}(s', s) = P^{\mathrm{c}}(b_{2,k} = b_2(s', s)) \quad (2)$$

$$\alpha_k(s) = \frac{\sum_{\text{all } s'} \gamma(s', s)\alpha_{k-1}(s')}{\sum_{\text{all } (s',s)} \gamma(s', s)\alpha_{k-1}(s')} \quad (3)$$

$$\beta_{k-1}(s') = \frac{\sum_{\text{all } s} \gamma(s', s)\beta_k(s)}{\sum_{\text{all } (s',s)} \gamma(s', s)\beta_k(s)} \quad (4)$$

$$P^{\mathrm{e}}(b_{1,k} = j) = \frac{\sum_{\text{all}(s',s)\to(b_1(s',s)=j)} \gamma_k^{\mathrm{e}}(s', s)\alpha_{k-1}(s')\beta_k(s)}{\sum_{\text{all } (s',s)} \gamma_k^{\mathrm{e}}(s', s)\alpha_{k-1}(s')\beta_k(s)} \quad (5)$$

$$\hat{b}_{1,k} = \arg\max_{j\in\{0,1\}} \sum_{\text{all } (s',s)\to b_1(s',s)=j} \gamma_k(s', s)\alpha_{k-1}(s')\beta_k(s) \quad (6)$$

III. STOCHASTIC COMPUTING IN ITERATIVE DECODING

In stochastic computing, probabilities are represented by means of streams of bits known as Bernoulli Sequences (BSs) [13], which are generated by statistically independent processes and with only one bit of each Bernoulli Sequence (BS) being processed in each clock cycle that is referred to as a Decoding Cycle (DC) of stochastic decoders. The probability P represented by a BS is determined by the fraction of its bits having the value of 1. Owing to this, the same probability can be represented by different BSs having the same fraction of bits with the value 1, but in different positions. This is exemplified in Figure 5a, where the probability $P = 0.75$ is represented by different BSs, each having 12 out 16 bits with value 1. As an explicit benefit of this, stochastic computing offers an inherent tolerance to processing errors. To elaborate further, a single bit-flip caused by a processing error will only change the overall value of the represented probability by a fraction proportional to the length of the BS. In this way, if the BS is sufficiently long, the represented probability will not be significantly affected, if some of the bits are corrupted. As an example of this, if any of the 12 bits having the value of logical 1 of the BSs of Figure 5a is flipped to logical 0, the resultant BSs will represent the probability of $P = 11/16 = 0.6875$, which corresponds to an absolute error of $1/16$.

A. STOCHASTIC ARITHMETIC

In stochastic computing, arithmetic combinations of the probabilities represented by two or more BSs can be implemented using low-complexity digital circuits. In particular, the binary complement $\bar{P} = (1 - P)$ of a probability P can be obtained using a NOT gate, as exemplified in Figure 5b.

The multiplication $P_{mult} = P_A \cdot P_B$ of two probabilities P_A and P_B can be performed using a bitwise logical AND of the two corresponding BSs S_A and S_B, as exemplified in Figure 5c. Similarly, the multiplication $P_{mult} = \prod_{i=1}^{M} P_i$ of the probabilities represented by M BSs can be performed with the aid of an M-input AND gate.

The weighted mean $P_{add} = P_A \cdot (1 - P_{sel}) + P_B \cdot P_{sel}$ of two probabilities P_A and P_B can be obtained by using a two-input Multiplexer (MUX) to randomly select bits from the corresponding BSs S_A and S_B, with the aid of the BS S_{sel}, which represents the probability $P_{sel} = 0.5$ for the case of a non-weighted mean, as exemplified in Figure 5d. The outgoing bit of the MUX will have the value of 1 only if the bit of the input selected by S_{sel} has the value of 1. More specifically, we have $S_{add} = 1$ only if $S_A = 1$ and $S_{sel} = 0$ or $S_B = 1$ and $S_{sel} = 1$. Similarly, the mean $P_{add} = [\sum_{i=1}^{M} P_i]/M$ of M probabilities can be obtained using an M-input MUX to randomly select bits from the corresponding M BSs.

The division of the probability P_J by the probability P_K can be approximated by entering the corresponding BSs S_J and S_K into the J and K inputs of a JK-type Flip Flop (JKFF), respectively, as exemplified in Figure 5e. Here, the output Q of the JKFF adopts the value of the input J if J and K disagree. By contrast, if $J = 0$ and $K = 0$, then Q retains the same value that it had in the previous clock cycle. If $J = 1$ and $K = 1$, the output Q is toggled relative to its value in the previous clock cycle. The bits in the BS S_Q obtained at the output Q of the JKFF will adopt the value 1 with the probability $P_Q = P_J/[P_J + P_K]$, which approximates P_J/P_K if $P_J \ll P_K$. In the example presented in Figure 5e, the outgoing BS S_Q represents the probability $P_Q = 0.2$, whereas the resulting normalization gives $0.2/(0.2 + 0.7) \approx 0.22$ and the approximated division gives $0.2/0.7 \approx 0.28$. Accurate stochastic computation realizations of division, integration, square and square operations can be obtained by combining the above-described basic stochastic arithmetic circuits, as detailed in [13], [57], and [58].

B. LATCHING PROBLEM

In the stochastic decoding of Low-Density Parity-Check (LDPC) codes, JKFFs are susceptible to the

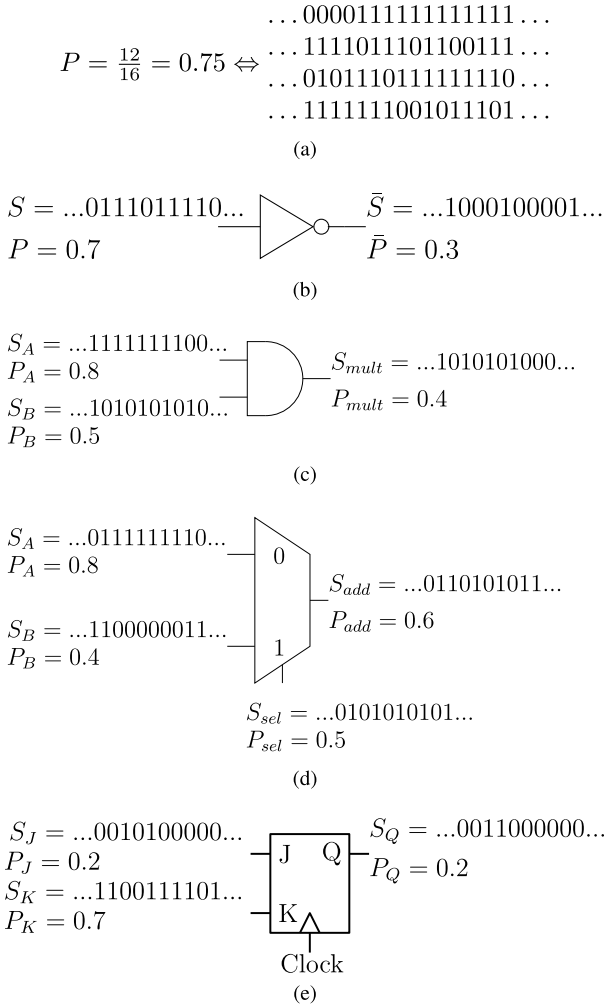


FIGURE 5. **Stochastic circuits for arithmetic operations: (a) Different BSs for $P = 0.75$. (b) Complement. (c) Multiplication. (d) Scaled addition. (e) Approximate division and normalization.**

latching problem, as detailed in [21]. This occurs when the bits of the BSs become stuck at 0 or 1 for several DCs, which severely affects the attainable error correction capability of the stochastic LDPC decoder [24]. This occurs when there is a high correlation among the bits of the BSs, owing to the short cycles in the factor graph of the LDPC code [21], [59]. A number of approaches have been proposed for overcoming the latching problem. For example, the employment of Noise-Dependent Scaling (NDS) was firstly proposed in [26]. This technique induces switching activity in order to help stochastic LDPC decoders become unstuck when they encounter the latching problem. This is particularly useful at high channel Signal-to-Noise Ratios (SNRs), where the probabilities represented by the soft bits received from the channel are very close to 0 or 1, causing the BSs to become stuck at 0 or 1, respectively. In this method, the probabilities received from the channel are scaled depending on the channel's noise power spectral density N_0. Assuming a Binary Phase Shift Keying (BPSK) transmission over an Additive White Gaussian Noise (AWGN) channel, a bit value probability $P(b = 0)$ is converted into a scaled bit value probability $P'(b = 0)$ according to

$$P'(b=0) = \frac{1}{1 + \exp\left(\frac{\eta N_0}{\psi} \cdot \log \frac{1-P(b=0)}{P(b=0)}\right)}, \tag{7}$$

where η and ψ are parameters than can be chosen to optimize the Bit Error Ratio (BER) performance of the stochastic LDPC decoder.

Another method of assisting stochastic LDPC decoders to become unstuck, when encountering the latching problem is replacing the JKFFs used for divisions by re-randomization units known as Edge Memories (EMs), which were firstly proposed in [24]. The EMs introduced in [24] consist of m-bit Shift Registers (SRs), which can be considered to behave as an m-length JKFF having a selectable output, as shown in the blue box printed using dashed lines in Figure 6.

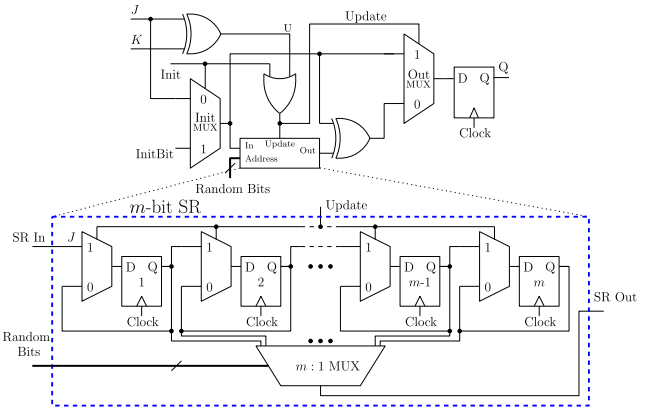


FIGURE 6. **EM employed for the re-randomization of BSs in stochastic decoding.**

Just like a JKFF, the values stored by an SR-based EM are updated if the input bits J and K differ from each other $J \neq K$, whereupon the regenerative bit J is stored in the first D-type Flip Flop (DFF) of the SR and passed to the output DFFQ of the EM in analogy to the behavior of a JKFF, with the aid of the OutMUX. In this event, the signals U and Update of Figure 6 are asserted and the contents of the SR will be shifted by one position, with the oldest bit in the SR being discarded in order to ensure that only the m-most recent regenerative bits are stored. By contrast, when the J and K input bits are equal, one of the previous regenerative bits is randomly chosen from the SR and passed to the output DFFQ of the EM, in analogy to the behavior of a JKFF. This is achieved with the aid of an $m : 1$ MUX with pseudo-random selector bits that change in every DC, as shown in the lower part of Figure 6. Additionally, when $J = K = 1$, the outgoing bit of the SR is inverted with the aid of an XOR gate, before being provided to the output DFFQ, in analogy to the behavior of a JKFF. The SR-based EM of Figure 6 can be initialized prior the beginning of the decoding process by means of the Init signal. During the initialization phase, the assertion of the signal Init causes the InitMUX to feed the initialization bits InitBits into the SR. The values of

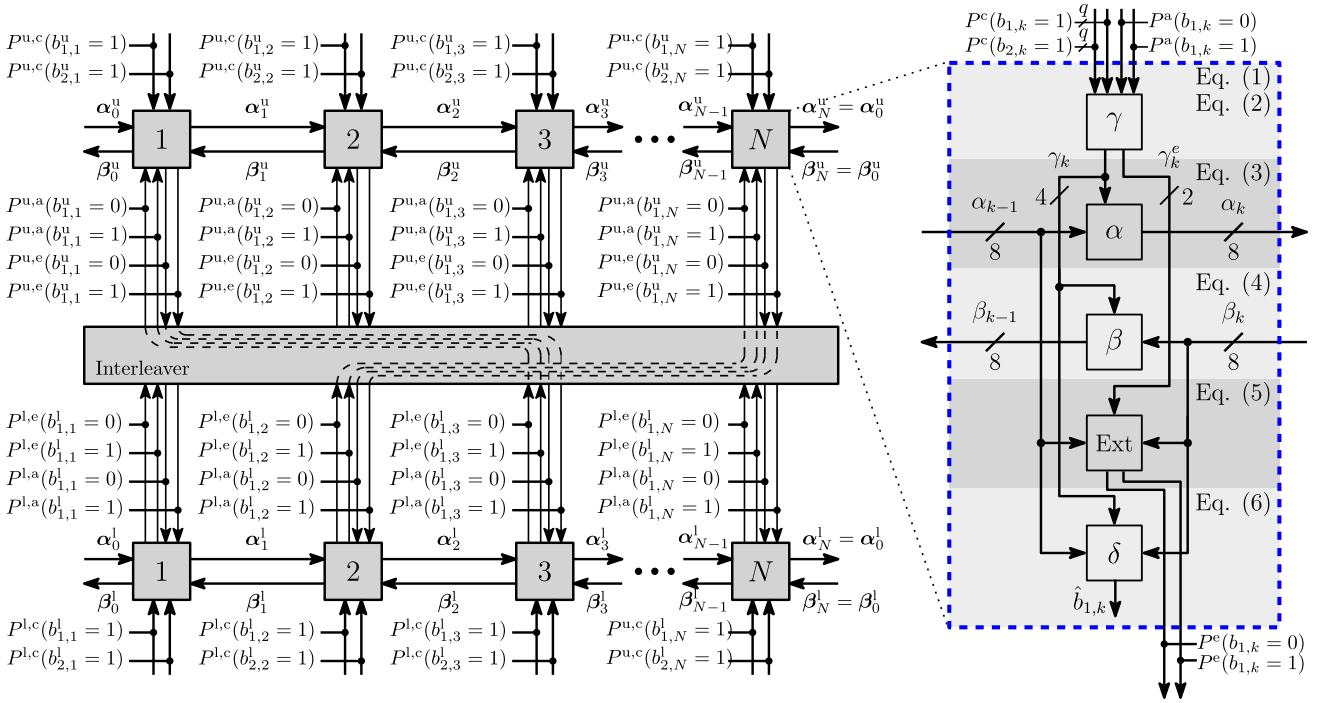


FIGURE 7. Block diagram of the fully-parallel STD.

InitBit can be chosen to optimize the BER performance of the decoder.

IV. STOCHASTIC IMPLEMENTATION OF TURBO DECODERS

This section reviews the hardware implementation requirements of the Stochastic Turbo Decoder (STD), which is briefly summarized in [36] and is detailed in [60], although the latter is written in French. Therefore, this paper offers the first detailed treatment of the STD in English. In the fully-parallel stochastic decoding of turbo codes [36], [60], the block diagram in the blue box of Figure 7 is replicated for each bit decoded by each convolutional decoder of Figure 4b, with the two convolutional decoders being separated by a hard-wired interleaver, as shown in the left part of Figure 7. The STD of [36] and [60] adopts tailbiting [55] as described in Section II. This guarantees that the initial and final states of the trellis of each convolutional decoder are identical. Owing to this, the state metrics $\alpha_N(s)$ output by the N^{th} block are provided as the inputs $\alpha_0(s')$ of the first block in each convolutional decoder, as shown in Figure 7. Likewise, the state metrics $\beta_0(s')$ output by the first block are provided as the inputs $\beta_N(s)$ of the N^{th} block. The incoming edges α_{k-1}, β_k, $P^{\mathrm{a}}(b_{1,k}=0)$ and $P^{\mathrm{a}}(b_{1,k}=1)$ of Figure 7 corresponds to one bit of a BS from a neighboring block. In a similar manner, the outgoing edges α_k, β_{k-1}, $P^{\mathrm{e}}(b_{1,k}=0)$ and $P^{\mathrm{e}}(b_{1,k}=1)$ corresponds to an outgoing bit of a BS. By contrast, the incoming edges $P^{\mathrm{c}}(b_{1,k}=1)$ and $P^{\mathrm{c}}(b_{2,k}=1)$ correspond to q-bit fixed-point probabilities provided by the channel. The block diagram of Figure 7 processes and exchanges one bit of each BS in each DC. Furthermore, each block of Figure 7 corresponds to the stochastic implementation of Equations (1) to (6). Building on this, the following sections present the stochastic hardware implementation requirements of Equations (1) to (6).

A. BRANCH METRICS

The module γ of Figure 7 performs the conversion of the q-bit fixed-point representations of the received channel probabilities $P^c(b_{1,k}=1)$ and $P^c(b_{2,k}=1)$ into BSs. In addition to this, this module generates BSs representing the branch metrics $\gamma_k(s',s)$ and the *extrinsic* branch metrics $\gamma_k^e(s',s)$ of (1) and (2), respectively. The conversion of the received channel probabilities into BSs is achieved with the aid of two q-bit fixed-point comparators and two q-bit pseudo-random numbers, as shown in the blue box of Figure 8. Here, q represents the number of quantization bits employed for representing the received channel probabilities and its value can be chosen based on the best trade-off between the advisable BER performance and the hardware requirements imposed, where $q=7$ in the STD of [36] and [60], for example. In this structure, $P^c(b_{1,k}=1)$ and $P^c(b_{2,k}=1)$ remain constant throughout the decoding process and the pseudo-random numbers change in every DC. The outgoing bit of each comparator is set to 1, if the corresponding probability is greater than the pseudo-random number and 0 otherwise. Following their conversion to BSs, the received channel probabilities are employed for obtaining the branch metrics $\gamma_k(s',s)$ of (1). This is achieved with the aid of four 3-input AND gates, as shown in the green box of Figure 8,

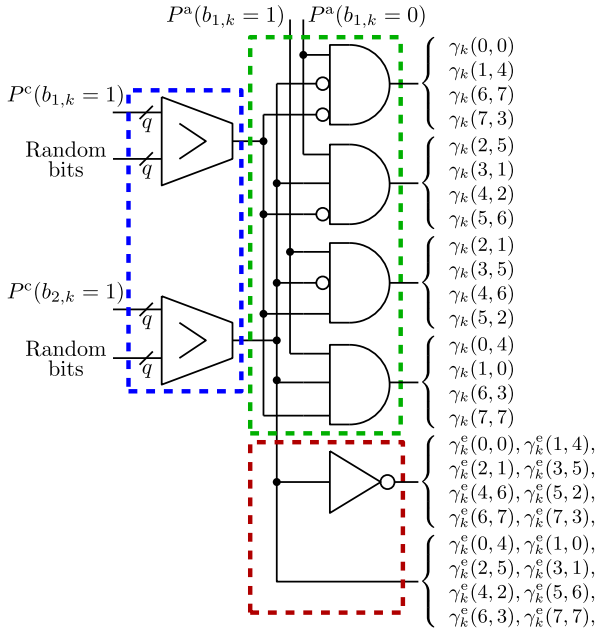


FIGURE 8. Stochastic realization of $\gamma_k(s', s)$ and γ^e_k [60, Fig. 2.5].

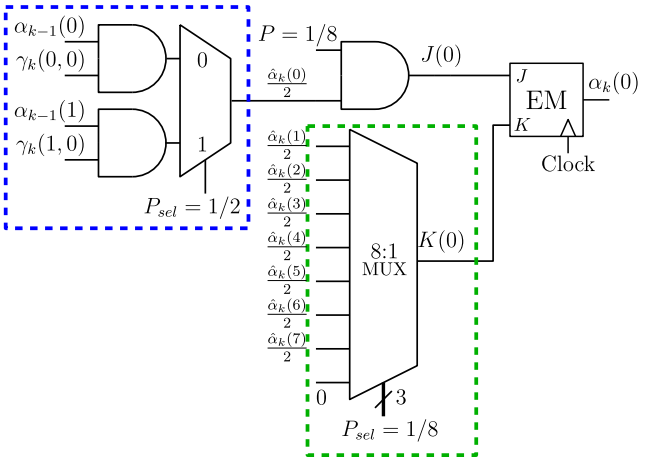


FIGURE 9. Stochastic realization of $\alpha_k(0)$ [60, Fig. 2.9].

corresponding to the four possible combinations of the systematic and parity bits $b_1(s', s)$ and $b_2(s', s)$ shown in Figure 4c. In addition to this, the *extrinsic* branch metrics $\gamma^e_k(s', s)$ of (2) express the probabilities $P^c(b_{2,k} = 0)$ and $P^c(b_{2,k} = 1)$, pertaining to the parity bit $b_{2,k}$. As a result of this, the stochastic realization of $\gamma^e_k(s', s)$ can be implemented with two BSs for representing $P^c(b_{2,k} = 0)$ and $P^c(b_{2,k} = 1)$, as shown in the red box of Figure 8. The BSs representing the *a priori* probabilities $P^a(b_{1,k} = 0)$ and $P^a(b_{1,k} = 1)$ are provided by the other convolutional decoder, but this will not yet have generated any output during the first DC. Therefore, the first bits of the BSs can be initialized with random binary values during the first DC.

B. STATE METRICS

The modules α and β of Figure 7 compute the forward recursion of (3) and the backward recursion of (4), respectively. This is achieved with the aid of AND gates, MUXs and EMs for the multiplication, addition and normalization of BSs, respectively, as shown in Figure 9. In the following discussion, the stochastic implementation of the state metrics is described for the forward recursion state metrics $\alpha_k(s)$ of (3), for the case where $s = 0$. The implementation of (3) for all other states $s \in [1, 7]$ and of the backward recursion of (4) can be performed following the same principles. In the Long-Term Evolution (LTE) turbo decoder of [36] and [60], (3) can be modified as

$$\alpha_k(s) = \frac{\sum\limits_{\text{all } s'} \gamma(s', s)\alpha_{k-1}(s')}{\sum\limits_{\text{all } (s',s)} \gamma(s', s)\alpha_{k-1}(s')} = \frac{\hat{\alpha}_k(s)}{\sum\limits_{s=0}^{7} \hat{\alpha}_k(s)}, \qquad (8)$$

where $\hat{\alpha}_k(s) = [\sum_{s'=0}^{7} \gamma(s', s)\alpha_{k-1}(s')]/2$ represents the non-normalized forward recursion. According to the state transition diagram of Figure 4c, the state $s = 0$ can only be reached from the previous states $s' = 0$ and $s' = 1$. Owing to this, the term $\hat{\alpha}_k(0)$ of (8), is simply $\hat{\alpha}_k(0) = [\gamma(0, 0)\alpha_{k-1}(0) + \gamma(1, 0)\alpha_{k-1}(1)]/2$. Therefore, the stochastic implementation of $\hat{\alpha}_k(0)$ can be performed by the circuit presented in the blue box of Figure 9, using a pair of 2-input AND gates and a 2-input MUX, as described in Section III-A. JKFFs may be employed for performing the division required for normalizing probabilities, although they are susceptible to the latching problem, as described in Section III-B. Owing to this, the STD of [36] and [60] employs the 32-bit SR-based EMs of Figure 6, as shown in Figure 9. Hence, Equation (8) can be expressed as

$$\alpha_k(s) = \frac{\hat{\alpha}_k(s)}{\hat{\alpha}_k(s) + \sum\limits_{\bar{s}\in[0,7]\setminus\bar{s}=s} \hat{\alpha}_k(\bar{s})} = \frac{J(s)}{J(s) + K(s)}, \qquad (9)$$

where $J(s) = [\hat{\alpha}_k(s)]/8$, $K(s) = [\sum_{\bar{s}\in[0,7]\setminus\bar{s}=s} \hat{\alpha}_k(\bar{s})]/8$ and $\bar{s} \in [0, 7] \setminus \bar{s} = s$ denotes the exclusion of the state $\bar{s} = s$ from the set of states $\bar{s} \in [0, 7]$. Here, $K(s)$ can be implemented with the aid of stochastic computing using an 8-input MUX with 3 pseudo-random selector bits representing a probability of $P_{sel} = 1/8$ and with one of the MUX inputs connected to logic 0, as shown in the green box of Figure 9. In addition to this, the factor of 8 division in the computation of $J(s)$ can be performed using an AND gate to multiply $\hat{\alpha}(s)$ with a BS representing the probability of 1/8. In the first DC, the inputs pertaining the BSs of $\alpha_{k-1}(s')$ can be initialized based on the best trade-off between BER performance and latency. To elaborate further, our simulations of Section VII suggest that the decoding latency can be reduced when the bits of the BSs of the state metrics $\alpha_{k-1}(s')$ and $\beta_k(s)$ are initialized with random bit values. Moreover, the contents of the EM can be initialized as described in Section III-B during the first 32 DCs with a BS representing the probability $P = 0.5$. The resulting stochastic implementation of (3), for the case

of $s = 0$ is shown in Figure 9. Here, the inputs of the 8-input MUX are provided by structures similar to that shown in the blue box but corresponding to the other states, $s \in [1, 7]$.

C. EXTRINSIC PROBABILITIES

The calculation of the *extrinsic* probabilities of (5) is performed by the module Ext of Figure 7. The stochastic implementation of (5) can be implemented using the circuit of Figure 10.

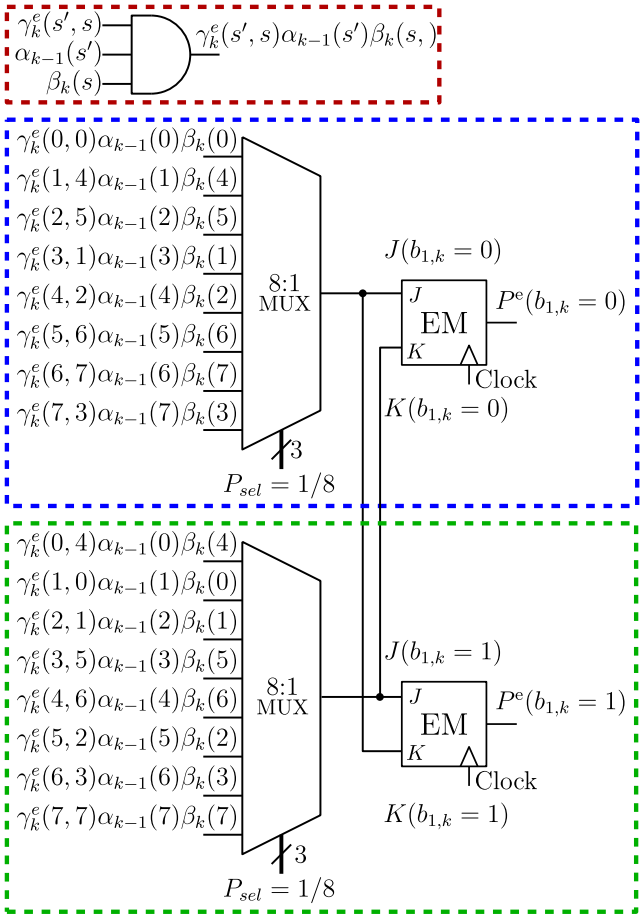


FIGURE 10. Stochastic realization of the calculation of the *extrinsic* probabilities in (5) [60, Fig. 2.11].

In the following discussion, the stochastic implementation of the *extrinsic* probabilities of (5) is described for the specific case where $b_{1,k}(s', s) = 0$, as represented by the blue box in Figure 10. The implementation of (5) for the case where $b_{1,k}(s', s) = 1$ is represented by the green box in Figure 10 and can be performed following the same principles. In analogy to the discussion presented in Section IV-B, (5) can be modified for the case where $b_{1,k}(s', s) = 0$ as

$$P^{\mathrm{e}}(b_{1,k} = 0) = \frac{J(b_{1,k} = 0)}{J(b_{1,k} = 0) + J(b_{1,k} = 1)}, \qquad (10)$$

where

$$J(b_{1,k} = 0) = \frac{\sum_{\mathrm{all}(s',s)\rightarrow(b_1(s',s)=0)} \gamma_k^{\mathrm{e}}(s', s)\alpha_{k-1}(s')\beta_k(s)}{8}$$

and $J(b_{1,k} = 1) = K(b_{1,k} = 0)$ correspond to the set of state transitions engendered by the input bits $b_{1,k} = 0$ and $b_{1,k} = 1$, respectively. According to the state transition diagram of Figure 4c, the input bit $b_{1,k} = 0$ triggers the set of transitions $(s', s) = \{(0, 0), (1, 4), (2, 5), (3, 1), (4, 2), (5, 6), (6, 7), (7, 3)\}$. As a result of this, $J(b_{1,k} = 0)$ can be expressed as

$$\begin{aligned} J(b_{1,k} = 0) = [&\gamma_k^{\mathrm{e}}(0, 0)\alpha_{k-1}(0)\beta_k(0) \\ &+ \gamma_k^{\mathrm{e}}(1, 4)\alpha_{k-1}(1)\beta_k(4) \\ &+ \gamma_k^{\mathrm{e}}(2, 5)\alpha_{k-1}(2)\beta_k(5) \\ &+ \gamma_k^{\mathrm{e}}(3, 1)\alpha_{k-1}(3)\beta_k(1) \\ &+ \gamma_k^{\mathrm{e}}(4, 2)\alpha_{k-1}(4)\beta_k(2) \\ &+ \gamma_k^{\mathrm{e}}(5, 6)\alpha_{k-1}(5)\beta_k(6) \\ &+ \gamma_k^{\mathrm{e}}(6, 7)\alpha_{k-1}(6)\beta_k(7) \\ &+ \gamma_k^{\mathrm{e}}(7, 3)\alpha_{k-1}(7)\beta_k(3)]/8. \end{aligned} \qquad (11)$$

Here, $J(b_{1,k} = 0)$ can be implemented with the aid of stochastic computing using a set of 8 3-input AND gates, one for each of the 8 multiplications of (11), and one 8-input MUX for the averaging of the 8 product terms of (11), as shown in the red and the blue box of Figure 10, respectively. In addition to this, the normalization of (11) can be performed using the EM structure of Figure 6 with the input bits $J = J(b_{1,k} = 0)$ and $K = J(b_{1,k} = 1)$. Similarly to the initialization of the EMs of the state metrics, the contents of the EM in this module can be initialized during the first 32 DCs with a BS representing the probability $P = 0.5$.

D. A POSTERIORI PROBABILITY

The estimation of the decoded bit $\hat{b}_{1,k}$ of (6) is performed by the module δ of Figure 7. This is achieved in stochastic computing with the aid of AND gates and 8-input MUXs, as shown in Figure 11. The circuit of Figure 11 calculates the term $\hat{P}(b_{1,k} = 0)$, which is proportional to the probability of the bit having the value 0, as shown in the blue box of Figure 11. This can be obtained by analyzing (6) for the case where $b_{1,k}(s', s) = 0$ and with the aid of the state transition diagram of Figure 4c. In this way, $\hat{P}(b_{1,k} = 0)$ can be expressed as

$$\begin{aligned} \hat{P}(b_{1,k} = 0) = [&\gamma_k(0, 0)\alpha_{k-1}(0)\beta_k(0) \\ &+ \gamma_k(1, 4)\alpha_{k-1}(1)\beta_k(4) \\ &+ \gamma_k(2, 5)\alpha_{k-1}(2)\beta_k(5) \\ &+ \gamma_k(3, 1)\alpha_{k-1}(3)\beta_k(1) \\ &+ \gamma_k(4, 2)\alpha_{k-1}(4)\beta_k(2) \\ &+ \gamma_k(5, 6)\alpha_{k-1}(5)\beta_k(6) \\ &+ \gamma_k(6, 7)\alpha_{k-1}(6)\beta_k(7) \\ &+ \gamma_k(7, 3)\alpha_{k-1}(7)\beta_k(3)]/8, \end{aligned} \qquad (12)$$

The stochastic implementation of (12) can be performed with the aid of 8 3-input AND gates, one for each of the 8 multiplications of (12), and one 8-input MUX for the addition of the 8 product terms of (12), as shown in the red and the

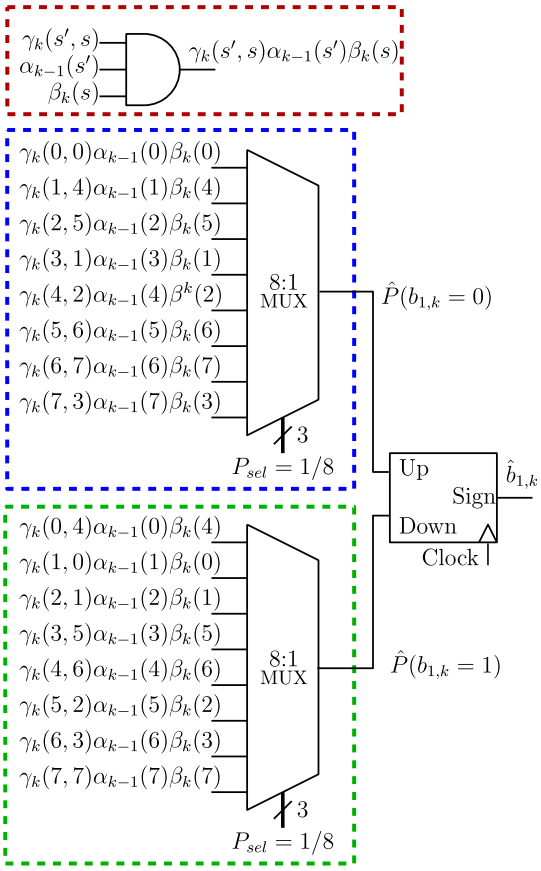


FIGURE 11. Stochastic realization of the calculation of the APP [60, Fig. 2.12].

blue box of Figure 11, respectively. The same principle can be applied for the stochastic implementation of $\hat{P}(b_{1,k} = 1)$, which is proportional to the probability of the decoded bit being 1, as shown in the green box of Figure 11. Additionally, the estimation of the decoded bit $\hat{b}_{1,k}$ can be performed with a signed up/down saturated counter, as shown in Figure 11. The up/down counter increments its value if the BS representing $\hat{P}(b_{1,k} = 0)$ takes the value of 1 and decreases its value if the BS representing $\hat{P}(b_{1,k} = 1)$ takes the value of 1. The counter will not change its value if $\hat{P}(b_{1,k} = 0) = \hat{P}(b_{1,k} = 1)$. Lastly, the counter saturates its value if either the maximum or the minimum count value has been reached. In the STD of [36] and [60], a 4-bit up/down counter with a maximum value of $+7$ and a minimum value of -8 is employed. The estimation of the decoded bit $\hat{b}_{1,k}$ is performed in each DC by considering the sign bit of the saturated counter. In this way, if the value of the counter is ≥ 0, the estimated decoded bit is $\hat{b}_{1,k} = 0$ and 1 otherwise.

V. TIMING-ERROR TOLERANT STOCHASTIC TURBO DECODER

Timing errors in synchronous systems occur when the clock period is not sufficiently long for all the signals to propagate from the output of DFFs, through the combinational logic and to the input of other DFFs. This occurs when techniques such as voltage-scaling or clock-scaling are employed for reducing the energy consumption or increasing the throughput of the system. Furthermore, a reduction in the power supply, owing to voltage scaling or to power supply noise, increases the likelihood of occurrence of processing errors, as a result of the quadratic dependency of propagation delays on the power supply. Whenever a timing error occurs, there is a chance that the affected DFF might enter into a metastable state, in which the digital signals have an indeterminate value that does not correspond to either a logic 0 or 1. However, given sufficient time, the metastable state will randomly evolve to a stable but unpredictable logic value of 0 or 1 [61]. This effectively imposes an additional propagation delay on the affected signal, hence increasing the likelihood of timing errors and metastability occurring at the next DFF. In this way, a single metastable event can trigger subsequent metastability occurrences in successive DFFs, causing catastrophic propagation of metastability that destroys the operation of the entire circuit. Moreover, metastability might cause undesired glitches, logic inconsistency and late transitions [61], which may result in the corruption of the bits stored in the EMs in the context of stochastic decoders, hence severely degrading the error correction capability of the decoder, as our previous work on stochastic LDPC decoders demonstrated [52]. This motivates modifications to the STD of [36] and [60] in order to enhance its tolerance to timing errors by preventing the catastrophic propagation of metastabiliy. In the following sections, we present several novel enhancements to the STD of [36] and [60], which not only improve its tolerance to timing errors, but also significantly improves its latency, throughput, energy efficiency and error correction capabilities in the presence of timing errors caused by power supply variations. In addition to this, we describe how each of these enhancements affects each of the design trade-offs presented in Figure 2. Each of these enhancements is detailed in the following subsections, which will show that they may be implemented by replacing Figure 6 with Figure 12 and Figure 11 with Figure 14.

A. OUTPUT SYNCHRONIZERS FOR MITIGATING THE CATASTROPHIC PROPAGATION OF METASTABILITY

As mentioned above, variations in the power supply increase the likelihood of timing errors and metastabiliy, which may catastrophically propagate through the system and destroy the entire operation of the STD. In order to reduce the probability of metastability cascading through the STD of [36] and [60] owing to timing errors caused by power supply variations, we propose the replacement of the single output DFFs of the STD's α, β and Ext blocks shown in Figures 7, 9 and 10, so that they employ the synchronizer circuits of Figure 12, which are comprised of two DFFs, labeled DFF1 and DFF2. Wherever combinational logic is present between two DFFs, the occurrence of metastability in the first DFF increases the likelihood of metastability occurring at the second DFF, potentially causing the catastrophic propagation

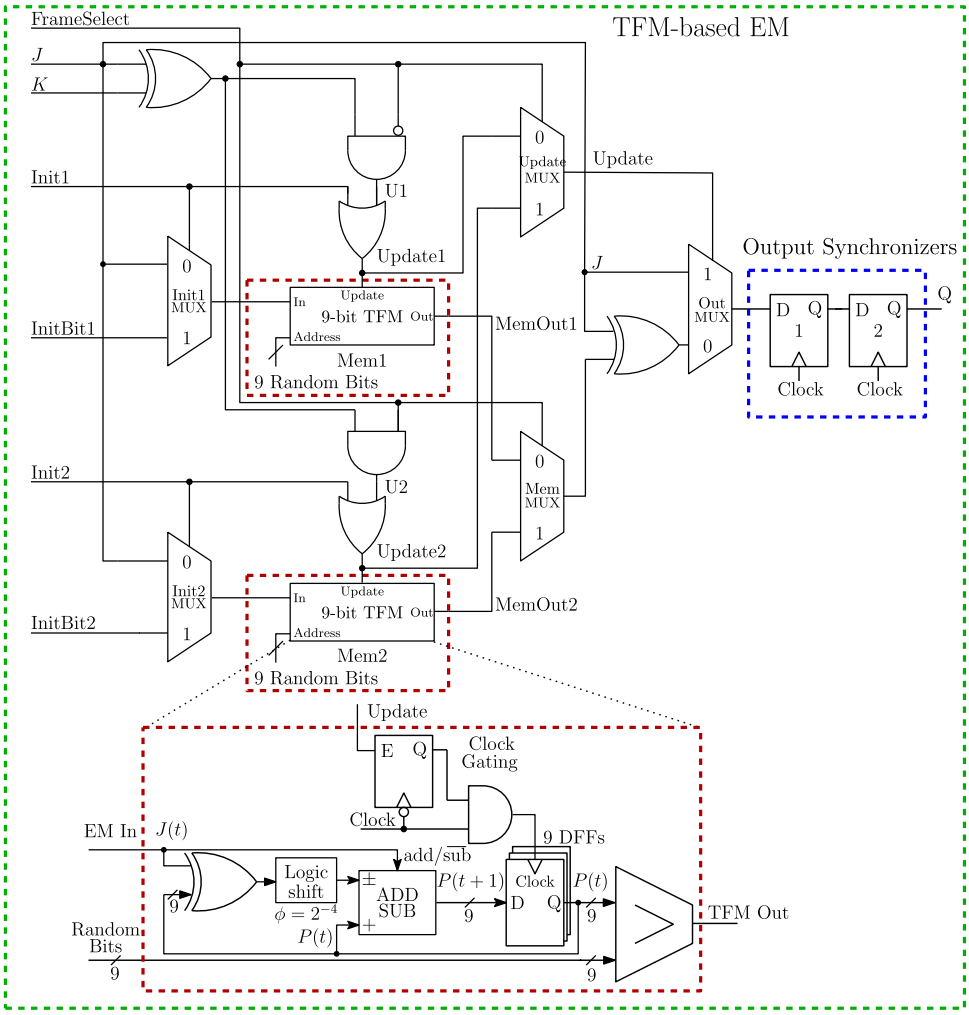


FIGURE 12. Modified TFM-based EM with output synchronizers, relying on two TFMs for concurrently decoding two frames and clock gating.

of metastability, as described above. However, since there is no combinational logic in the path between DFF1 and DFF2 of the synchronizers of Figure 12, the propagation delay associated with this path is negligible, when compared to the clock period. As a benefit of this, if DFF1 enters into a metastable state due to a timing error, the time available for its metastability to be resolved is maximized and the probability of metastability cascading to DFF2 is significantly reduced [62].

The employment of synchronizer circuits is a commonly used technique for preventing the propagation of metastability during the transfer of data between different clock domains in asynchronous systems [62]. However, in our timing analysis presented in Section VII-A, we demonstrate that timing errors will frequently occur if the nominal operating conditions of the STD are reduced below the recommended safety margins for the sake of improving either the throughput or the energy consumption. Timing errors may also occur due to power supply variations. In addition to this, in the stochastic decoding of turbo codes presented in [36] and [60], up to 250×10^3 DCs are needed before a reliable final decision can be obtained. More specifically, our analysis suggests that thousands of metastability events occur during the decoding of each frame, when the STD of [36] and [60] is operated continuously in the presence of power supply variations. This results in a high likelihood of timing errors and metastability occurring and cascading through the circuit, destroying the decoding process and severely affecting the error correction capabilities of the STD, unless synchronizers circuits are employed.

The introduction of the synchronizer circuits enhances the decoding capabilities of the STD in the presence of timing errors caused by power supply variations, as we will demonstrate in Section VII. However, this modification increases the chip area, latency and energy consumption of the STD and reduces its throughput, owing to the increased number of clock cycles required for exchanging each bit of the BSs, as we will show in Section VI. In order to mitigate this throughput reduction, Section V-B describes how the STD can be modified to decode two frames concurrently.

B. DECODING OF TWO FRAMES CONCURRENTLY

The employment of the synchronizers described in Section V-A mitigates the probability of a single metastability event destroying the entire operation of the STD. However, this modification increases the chip area and increases the number of clock cycles required for exchanging each bit of the BSs, hence increasing both the latency as well as the energy consumption and reducing the throughput of the STD. Nonetheless, these additional clock cycles can be exploited for the concurrent decoding of a second received frame, in order to eliminate the throughput reduction. With this objective in mind, the STD can be modified to process information pertaining to alternate received frames in alternate clock cycles. This is achieved by modifying the EMs in the α, β and Ext blocks shown in Figures 7, 9 and 10 for simultaneously storing information pertaining to each of the two received frames. Note that while this eliminates the above-mentioned throughput reduction that is introduced by the synchronizers, this does not eliminate the above-mentioned latency, chip area and energy consumption increases, as we will show in Section VI. Instead, these disadvantages of introducing synchronizers will be eliminated using the techniques described in the following subsections.

The green box of Figure 12 shows how the EM structures of Figures 6, 9 and 10 can be modified for the simultaneous decoding of two frames, by including two independent sets of memories, labeled Mem1 and Mem2, as shown in the red boxes. In this configuration, each set of memories will store information related to a particular decoded frame. Along with the additional memory, we introduce a FrameSelect signal and the logic required for controlling which specific set of DFFs to use in each clock cycle. The FrameSelect signal alternately adopts values of 0 and 1 in alternate clock cycles, alternating the activation of Mem1 and Mem2 for their update and read operations, according to their control signals. More specifically, the contents of Mem1 and Mem2 can be updated or read according to the value of the control signals U1 and Update1, for the case of Mem1, and U2 and Update2 for the case of Mem2. When the FrameSelect signal has a logic value of 0 and $J \neq K$, the signals U1 and Update1 adopt the value of 1, whereupon the contents of Mem1 are updated according to the regenerative bit J. In this scenario, the Update signal adopts the value of Update1=1, owing to the UpdateMUX, whereupon the regenerative bit J is passed to the first DFF of the synchronizers through OutMUX. By contrast, when the FrameSelect signal has a logic value of 0 and $J = K$, the U1 and Update1 signals adopt the value of 0. In this case, the Update signal adopts the value of 0 and a randomly selected bit from Mem1 is xored with J and passed to the first DFF of the synchronizers through MemMUX and OutMUX. The operation of Mem2, for the case when FrameSelect=1, follows the same principle described above. Note that the initialization of each memory can be performed independently from each other and from the value adopted by FrameSelect. This is achieved using the signals Init1 and Init2, which directly dictate the value of Update1 and Update2 and control the initialization multiplexers Init1MUX and Init2MUX, corresponding to Mem1 and Mem2, respectively. Moreover, a memory can be initialized whilst the other memory is being employed during a decoding process. This allows one of the memories to be reset so that it can begin decoding a new frame, even if the other memory is still being used to decode a different frame. As described in Sections IV-B and IV-C, each memory is initialized before the beginning of each decoding process with a BS corresponding to a probability of 0.5.

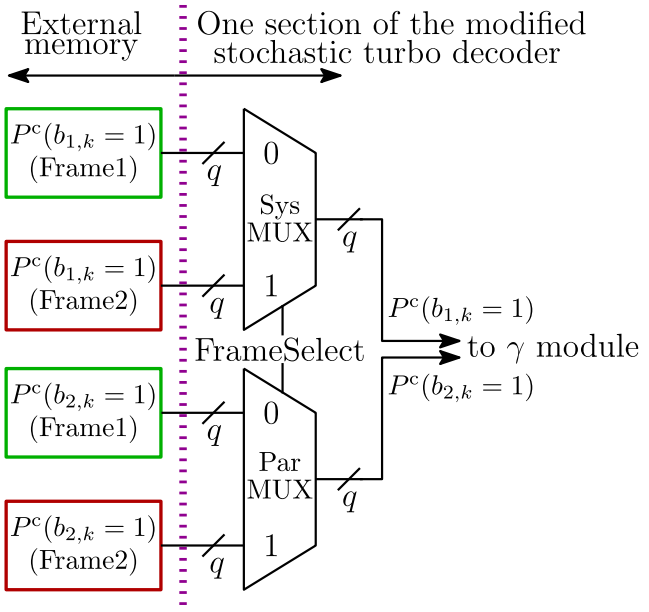


FIGURE 13. External set of systematic and probability probabilities for concurrently decoding two frames.

Along with the inclusion of the additional memories and control logic presented in Figure 12, the STD alternates between providing the corresponding systematic and the parity probabilities related to each of the two received frames. For the purpose of our investigation, we consider these sets of probabilities to be stored in a memory external to the decoder, which are selected according to the value of the FrameSelect signal, as shown in Figure 13. In this configuration, the set of MUXs labeled SysMUX in Figure 13 selects the between the systematic probabilities $P^c(b_{1,k} = 1)$ of the received frames Frame1 and Frame2, when FrameSelect adopts the value of 0 and 1, respectively. Similarly, the received frame of parity probabilities $P^c(b_{2,k} = 1)$ of Frame1 and Frame2 is selected with the aid of the ParMUX, when FrameSelect adopts the value of 0 and 1, respectively. In each clock cycle, the selected probabilities $P^c(b_{1,k} = 1)$ and $P^c(b_{2,k} = 1)$ are provided to the γ module of the corresponding section of the STD.

As described above, the synchronizers and additional EMs increase not only the chip area, but also the energy consumption of the STD. To overcome these problems, we recommend the employment of low-power design techniques. In our investigation, we employ Clock Gating (CG) for reducing the dynamic energy consumption of the Application

Specific Integrated Circuit (ASIC) implementation of the STD, as we will demonstrate in Section VI. More specifically, the dynamic energy consumption of the STD may be reduced by only enabling the clock signal of those specific DFFs, whose contents have to be updated in a particular clock cycle, as shown in Figure 12. More specifically, the clock signal is AND gated with the output of an active-low latch, which is driven by the Update signal, as shown in Figure 12. Moreover, the area used by the clock-gated SR-based EMs will be significantly reduced, as we will show in Section VI, since the feedback MUX used for each DFF in the SR of Figure 6 will no longer be required. Instead, the DFFs will update their contents only when their clock signal is enabled, maintaining their current value otherwise. Furthermore, the associated propagation delays of the EMs are reduced, owing to the reduced fanout load of the Update signals [52]. More specifically, the Update signal in the EM of Figure 6 has a fanout load of 32 Multiplexers, compared to a fanout load of a single latch in the clock-gated EMs of Figure 12.

Note however that CG does not mitigate the additional static energy consumption that is associated with the additional set of EMs. In order to overcome this static energy increase, the STD may employ other low-power design techniques, such as power gating or multiple- and variable-threshold transistor design, although this is achieved at the cost of increasing the complexity of the design. As an example of this, power gating may be employed for switching off specific blocks of the STD, when they are not being operated. However, state retention registers are required for restoring the state of the blocks upon power-up. Owing to this overhead, power gating is only effective, when specific blocks may be switched off for a significant amount of time. Motivated by this, our future work will conceive techniques for applying early-stopping to different blocks at different stages in the iterative decoding process. However, in the meantime, the low-power design techniques presented in this paper focus only on the reduction of the STD's dynamic energy consumption.

When decoding two frames concurrently, the STD is required to estimate two sets of decoded bits $b_{1,k}$. Owing to this, an additional up/down counter is introduced in the δ module of Figures 7 and 11 for providing two independent decoded bits pertaining the two independent decoding process of Frame1 and Frame2, as shown in the blue boxes of Figure 14. In this configuration, two clock-gating latches are employed for updating the counters labeled Counter1 and Counter2. More specifically, the contents of Counter1 determines the decoded bit $b_{1,k}$ of the decoded frame Frame1 and is only updated when FrameSelect=1. By contrast, the decoded bit $b_{1,k}$ for the case of the decoded frame Frame2 is determined when FrameSelect=0. As part of the trade-off analysis of the hardware implementation of the STD presented in Section VI, we will demonstrate that the chip area, latency, throughput and energy consumption of the STD is not significantly affected by the introduction of the additional counter.

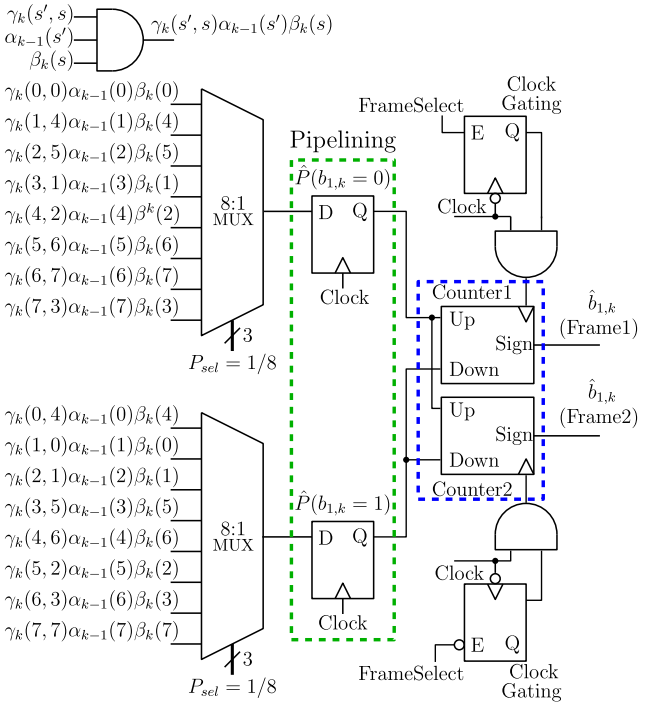


FIGURE 14. Modified stochastic realization of the calculation of the APP for concurrently decoding two frames.

C. TRACKING FORECAST MEMORY-BASED EDGE MEMORIES IN STOCHASTIC TURBO DECODERS

As described in Section V-B, the introduction of the additional EMs for the concurrent decoding of two received frames increases the chip area, latency and energy consumption of the STD. These problems can be overcome by the employment of Tracking Forecast Memory (TFM)-based EMs, whilst enhancing the error correction capability of the STD in the presence of timing errors, as we will demonstrate in Sections VI and VII. The employment of SR-based [24], [26] and TFM-based [27], [30] EMs has been proposed in order to overcome the latching problem in stochastic LDPC decoders. However, in the STD of [36] and [60], only SR-based EMs have been previously employed. In this work, we demonstrate for the first time that the employment of TFM-based EMs, as shown in the lower part of Figure 12, is also beneficial in stochastic turbo decoding.

The TFM-based EM of Figure 15 was proposed in [27]. Here, the m-bit SR of Figure 6 is replaced by an n-bit TFM, which stores a fixed-point binary number to quantify a moving average probability of the regenerative bit J assuming the value 1. This is achieved by considering the previous bits of BSs, but placing special emphasis on the most recent bits, as detailed in [27]. More specifically, TFMs employ a decaying mechanism to ensure that only the most recent regenerative bits are considered for the calculation of the stored probability $P(t+1)$, which is calculated with the aid of the relaxation parameter ϕ for determining the significance given to the most recent regenerative bit J.

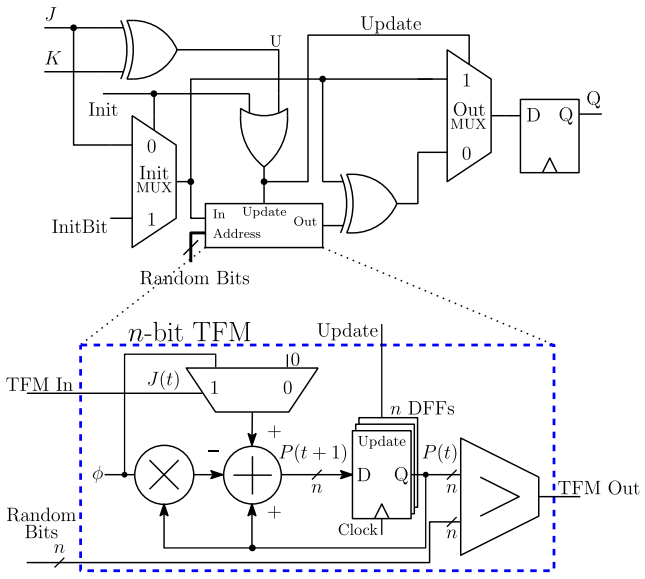


FIGURE 15. Architecture of TFM-based EMs, as proposed in [27].

As in the SR-based EM, when $J \neq K$ in Figure 15, the fixed-point probability $P(t+1) \in [0, 1]$ of the TFM in time $t+1$ is updated according to:

$$P(t+1) = (1-\phi)P(t) + \phi J(t), \tag{13}$$

where $\phi \in (0, 1)$ is the relaxation parameter, which can be chosen to optimize the BER performance of the stochastic LDPC decoder, and $J(t) \in \{0, 1\}$ is the regenerative bit. By contrast, when $J = K$, the output of the TFM-based EM is determined by comparing the n-bit probability $P(t)$ to an n-bit pseudo-random number. If the probability stored by the TFM is larger than or equal to the random number, the outgoing bit is set to 1, otherwise it is set to 0.

Considering that J can only take the value of 0 or 1 at any given time, (13) can be modified as follows:

$$\begin{aligned} P(t+1) &= (1-\phi)P(t) + \phi J(t) \\ &= P(t) - \phi P(t) + \phi J(t) \\ &= P(t) + \phi[J(t) - P(t)], \end{aligned}$$

which can be further simplified as

$$P(t+1) = \begin{cases} P(t) - \phi P(t) & J(t) = 0 \\ P(t) + \phi \bar{P}(t) & J(t) = 1, \end{cases} \tag{14}$$

where $\bar{P}(t) = 1 - P(t)$ is the complementary probability of $P(t)$. As a benefit of this, the two adders of the TFM of Figure 15 are substituted by a single adder/subtractor, where the complementary probability $\bar{P}(t)$ can be obtained using XOR gates, if $P(t)$ is stored as an unsigned fixed-point number. Meanwhile, the multiplication $\phi \cdot P(t)$ can be readily implemented using a hard-wired logical shift, if the relaxation parameter ϕ is chosen as a negative power of 2, as shown in the lower red box of Figure 12. According to (14), the complementary probability $\bar{P}(t)$ is only necessary if $J(t) = 1$. This functionality can be implemented with the aid MUXs and NOT gates, with the selector bits of the MUXs adopting the value of J, as described in [30]. Alternatively, the same functionality can be obtained with 2-input XOR gates, with one of the inputs adopting the value of J and the other input connected to the individual bits of $P(t)$, as shown in the lower part of Figure 12. Additionally, the add/$\overline{\text{sub}}$ signal determines, whether an addition or a subtraction will be performed by the ADD/SUB block, when $J = 1$ and $J = 0$, respectively. The n-bit fixed-point comparator is employed for determining the outgoing bit when $J \neq K$, with TFMOut adopting the value of 1 if the probability stored in the TFM is larger than or equal to the n-bit pseudo-random number. Finally, at the start of the STD decoding process, each TFM is initialized to store the probability $P = 0.5$, which can be achieved in a single clock cycle by setting the Most Significant Bit (MSB) of each TFM to logic 1 and the rest of the bits to logic 0. This single clock cycle compares favorably to the 32 clock cycles required to initialize the SR Figure 6 with a given BS. The employment of TFMs in EMs effectively reduces the chip area requirements and energy consumption of the STDs, when compared to the employment of SRs in EMs in the STD of [36] and [60]. More specifically, for the proposed modifications, we recommend the employment of 9-bit TFMs, with a relaxation parameter $\phi = 2^{-4}$. As a result of this, the proposed TFM-based EMs can be realized by using only 9 DFFs, compared to the 32 DFFs required in the SR-based EMs of [36] and [60]. Moreover, the SR's 32 to 1 MUX of Figure 6 can be replaced with 9-bit fixed-point comparators.

In addition to the advantages described above, TFMs are capable of tracking changes in the regenerative bit's probability more accurately than SRs, as detailed in [30]. This effectively reduces the number of DCs required for successfully decoding a frame when TFMs are employed, hence eliminating the potential latency increase resulting from the employment of the synchronizers of Section V-A Similarly, TFM-based EMs facilitate the use of lower clock periods than SR-based EMs, as Section VII will show. This may be attributed to the relatively low complexity of the TFMs, as well as to the relatively low fanout loads imposed on their Update signal. Owing to this, the proposed TFMs-based EMs offer a desirable trade-off between chip area, energy efficiency, latency, throughput and decoding performance, as Sections VI and VII will show. Figure 12 illustrates the resultant TFM-based EM circuit, relying on synchronizers, two sets of EMs and CG.

D. PIPELINING

The role of the δ module of Figure 7 is to compute the APP and to make the final decision for the decoded bit $\hat{b}_k$ output by the STD. Owing to this, the occurrence of a timing error within this module, caused by power supply variations, might lead to an incorrect decision for a decoded bit, hence severely affecting the error correction capability of the STD. Furthermore, our timing analysis presented in Section VII-A

revealed that timing errors are more likely to occur in the Ext and δ modules of Figure 7, owing to their long critical paths. In order to overcome this problem, we propose the employment of a pipelining stage consisting of two DFFs placed in parallel, as shown in the green box of Figure 14. The additional DFFs break the combinational path that ends at the saturated counter. This reduces the time required for signals to propagate, hence reducing the occurrence of timing errors within the computation of the APP and improving the STD's error correction capability, as we will show in Section VII-B. Moreover, the employment of a pipeline stage does not impose a significant hardware overhead, as we will show in Section VI, since only two DFFs per decoded bit are employed. Similarly, the throughput and energy consumption of the STD are only slightly degraded, since the decision for the decoded bit $\hat{b}_{1,k}$ is only delayed by one DC.

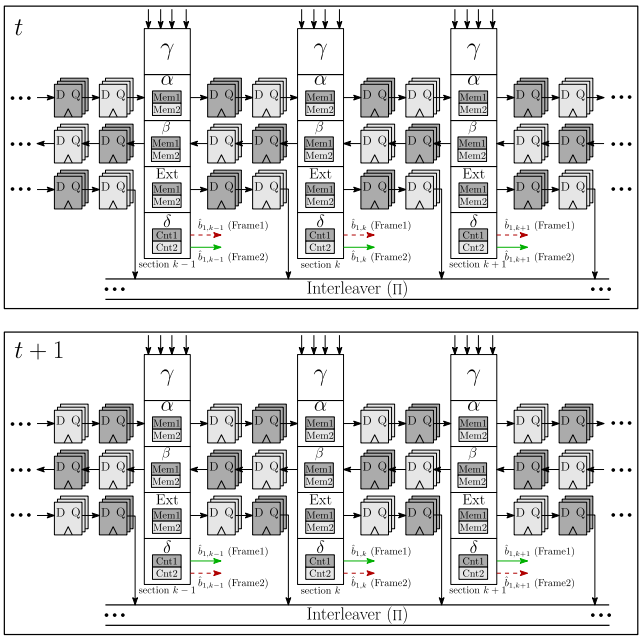


FIGURE 16. Data scheduling of the modified STD.

The data flow of the modified STD is illustrated in Figure 16, where each set of light-shaded and dark-shaded blocks and DFFs store information pertaining the decoding frames Frame1 and Frame2, respectively. In the upper part of Figure 16, the first set of memories Mem1 can represent either SR-based or TFM-based EMs. These EMs and the first DFF of the synchronizers will update their contents according to the bit of the BS resulting from the decoding process of Frame1 at time t. In time $t+1$, the contents of the second DFFs of the synchronizers will contain the stochastic bit generated at time t and pertaining Frame1, owing to the synchronizers shifting their contents to the following DFF, as shown in the dark-shaded DFF of the lower part of Figure 16. In this same manner, the contents of Mem2 and the first DFFs of the synchronizers will update their value according to the decoding process of Frame2 in time $t+1$, as shown in the light-shaded DFFs of the lower part of Figure 16. Note that Counter1 is enabled one time instant after Mem1 is enabled, owing to the introduction of the pipelining stage. This is represented with a green solid line in the lower part of Figure 16. Alternatively, a red dashed line in Figure 16 represents either Counter1 or Counter2 in idle mode, in which their contents are not updated and no new estimation of $\hat{b}_{1,k}$ is made for that particular frame.

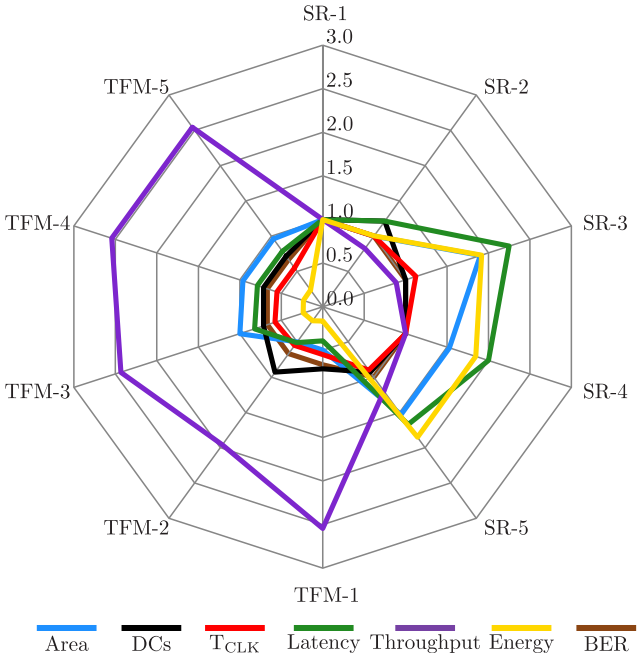


FIGURE 17. Hardware implementation results of the modified STD schemes. The presented results are normalized relative to SR-1, when $V_{DD} = 1.2$ V and $E_b/N_0 = 3.0$ dB.

VI. TRADE-OFF ANALYSIS OF THE HARDWARE IMPLEMENTATIONS OF THE TIMING-ERROR TOLERANT STOCHASIC TURBO DECODER

This section presents the different hardware trade-offs of various STD implementations. Table 1 characterizes the STD improvements of Section V in terms of diverse characteristics, including: the chip area per decoded bit, number of equivalent NAND gates, average number of DCs required for successfully decoding a frame, when employing early stopping, minimum clock period, latency, throughput and processing energy consumption per decoded bit, when using TSMC 90 nm technology. These results are obtained for the implementation of an 8-state, 50-bit, rate 1/3, tailbiting STD using an S-Random interleaver and the state-transition diagram of Figure 4c, as presented in [36]. Additionally, we employ NDS with $\eta = 1$ and $\psi = 2$, as described in Section III-B. Each scheme presented in Table 1 and in Figure 17 corresponds to the successive inclusion of each of the improvements described in Section V, with SR-1 being the state-of-the-art STD of [36]. More specifically, SR-2 corresponds to the introduction of output synchronizers

TABLE 1. Hardware implementation results of the STD when operated at $E_b/N_0 = 3.0$ dB.

Scheme	Improvement	Chip area per decoded bit	Average DCs	V_{DD} (V)	T_{CLK} (ns)	Latency (μs)	Throughput (Kbps)	Energy (nJ/bit)
SR-1	Benchmark	0.0260 mm^2 8617 gates	14×10^3	1.20	4.0	56	892	335
				0.84	8.0	112	446	164
SR-2	Synchronizers	0.0260 mm^2 8744 gates	17×10^3	1.20	4.0	68	735	337
				0.84	8.0	136	367	165
SR-3	Extra EMs	0.0495 mm^2 15505 gates	14×10^3	1.20	4.5	126	793	643
				0.84	8.6	240	415	315
SR-4	Clock Gating	0.0397 mm^2 12853 gates	14×10^3	1.20	4.0	112	892	620
				0.84	7.6	212	469	303
SR-5	Pipeline	0.0397 mm^2 12933 gates	13×10^3	1.20	3.6	93	1068	621
				0.84	6.7	174	574	304
TFM-1	TFM-based	0.0129 mm^2 4596 gates	10×10^3	1.20	2.2	22	2272	55
				0.84	4.1	41	1219	27
TFM-2	Synchronizers	0.0129 mm^2 4724 gates	13×10^3	1.20	2.2	28	1748	66
				0.84	4.1	53	938	32
TFM-3	Extra EMs	0.0250 mm^2 8817 gates	10×10^3	1.20	2.3	46	2173	88
				0.84	4.2	84	1190	45
TFM-4	Clock Gating	0.0250 mm^2 8516 gates	10×10^3	1.20	2.2	44	2272	77
				0.84	4.1	82	1219	37
TFM-5	Pipeline	0.0250 mm^2 8531 gates	10×10^3	1.20	2.2	44	2272	77
				0.84	4.1	82	1219	37

into SR-1; SR-3 refers to the introduction of the additional set of EMs into SR-2 to enable the concurrent decoding of two frames; SR-4 corresponds to the introduction of CG into SR-3 and finally, SR-5 refers to the introduction of the pipelining stage into SR-4. The schemes TFM-1 to TFM-5 present the TFM-based STDs counterparts of SR-1 to SR-5, with TFM-5 being the modified STD employing all enhancements proposed in Section V.

The results of Table 1 were obtained from the physical layout generated by the automatic place and route of the above mentioned STDs using Cadence SoC Encounter. The results of chip area per decoded bit were obtained from the layout of the γ, α, β, Ext and δ modules of Figure 7, where the γ module includes q=7-bit fixed point comparators for converting the probabilities provided by the channel into BSs. Additionally, the results of Table 1 were obtained for the cases, where the supply voltage of the STDs is set to either 1.20 V or 0.84 V, in the absence of power supply variations. Both the critical clock period and the energy consumption of the STDs are obtained from Synopsys PrimeTime. The average number of DCs, latency, throughput and energy efficiency results were obtained from post-layout gate-level simulations, with extracted parasitics and annotated delays without timing errors, when allowing a maximum of 10^5 DCs and using early stopping, as described in Section II. We assume that a fully-parallel Cyclic Redundancy Check (CRC) [63] is employed in each DC for determining whether the frame of estimated decoded bits provided by the STDs contains any errors. Although the corresponding hardware characteristics are not considered in Table 1, they may be considered to have only a negligible effect, as detailed in [64]. We also assume that the different STDs operate at their critical clock period, for the case of BPSK communication over an AWGN channel and an SNR per bit of $E_b/N_0 = 3.0$ dB. The notation (Scheme, V_{DD}, T_{CLK}) will be used in the following sections in order to simplify the discussion of the results. In addition to Table 1, Figure 17 presents the hardware implementation trade-offs associated with each scheme of Table 1, relative to the benchmark scheme SR-1, when the various different STDs operate at $V_{DD} = 1.2$ V. Note that similar trends were found for the case when $V_{DD} = 0.84$ V.

We begin our discussion by analyzing the impact of each enhancement in the hardware implementation of the STD. This is followed by a comparison of the different hardware implementation trade-offs of SR-based and TFM-based STDs.

As discussed in Section V-A as well as confirmed in Table 1 and Figure 17, the employment of synchronizers increases the average number of DCs, latency and energy consumption and reduces the throughput of the STDs. This can be observed when comparing the SR-2 and TFM-2 schemes to the SR-1 and TFM-1 arrangements of Table 1 and Figure 17, respectively. More specifically, observe in the fourth column of Table 1 that SR-2 requires an average of 17×10^3 DCs for successfully decoding a frame, compared to the 14×10^3 required in SR-1, when these schemes are operated at $E_b/N_0 = 3.0$ dB and achieve a BER of 4.6×10^{-4}, as we will show in Section VII-B. Likewise, TFM-2 requires an average

of 13×10^3 DCs, compared to 10×10^3 DCs required by TFM-1 to achieve a BER of 2.8×10^{-4} at $E_b/N_0 = 3.0$ dB, which is the same BER that is achieved by the ideal floating-point Logarithmic BCJR (LogBCJR) implementation employing 8 iterations. These differences are explicitly visualized in Figure 17.

The inclusion of the additional EMs for concurrently decoding two frames increases the chip area, extends the latency of the STDs, increases the energy consumption, but facilitates an increased throughput, as observed for the schemes SR-3 and TFM-3 of Table 1 and Figure 17, when compared to SR-2 and TFM-2, respectively. As seen in Table 1, the average numbers of DCs required by schemes SR-3 and TFM-3 are identical to those of SR-1 and TFM-1, respectively. However, the latency of SR-3 and TFM-3 is increased, when compared to SR-2 and TFM-2, respectively. This may be attributed to the decoding process of SR-3 and TFM-3 is completed in alternate clock cycles. Despite the increased latency, SR-3 and TFM-3 exhibit an increased throughput, when compared to SR-2 and TFM-2, respectively, owing to the simultaneous decoding of two frames, as detailed in Section V-B. Similarly, SR-3 and TFM-3 exhibit an increased energy consumption owing to the additional EMs, when compared to SR-2 and TFM-2, respectively.

As detailed in Section V-B as well as observed in Table 1 and Figure 17, the employment of clock gating in SR-4 and TFM-4, reduces the chip area, the latency and the energy consumption and increases the throughput of the STDs, when compared to SR-3 and TFM-3, respectively. Particularly, the chip area reduction of SR-4 may be attributed to the elimination of the 32 MUXs needed in the SRs, as described in Section V-B. By contrast, only the equivalent number of NAND gates of TFM-4 is slightly reduced, when compared to TFM-3. Additionally, as explained in Section V-D, the inclusion of the pipelining stage does not significantly increase the area requirements of the STDs and does not extend the latency. As a result of this, the employment of clock gating and the pipelining stage facilitates lower clock periods, which is reflected in the improved latency, throughput and energy consumption of SR-4 and SR-5, when compared to SR-3, as well as of TFM-4 and TFM-5, when compared to TFM-3.

Let us now compare the hardware implementation trade-offs of SR-based and TFM-based STDs. When comparing the chip area requirements of SR-based and TFM-based STDs, Table 1 and Figure 17 show that TFM-based schemes exhibit lower area requirements than their SR-based counterparts. More specifically, TFM-1, TFM-2 and TFM-3 present area requirements that are about 0.50× the area of SR-1, SR-2 and SR-3, respectively. Additionally, the implementation of TFM-4 and TFM-5 requires 0.62× the chip area of SR-4 and SR-5, respectively. Furthermore, TFM-5, which presents all the proposed enhancements, requires 0.96× the area of the benchmark SR-1. This lower area requirement of TFM-based schemes may be attributed to the TFM's employment of only 9 DFFs, instead of the 32 DFFs needed in a SR, as explained in Section V-B. As explained in Section V-C, TFM-based STDs require fewer DCs for achieving iterative decoding convergence, when compared to SR-based STDs. This, in addition to the reduced clock periods offered by TFM-based STDs, facilitate reduced latencies, reduced energy consumptions and increased throughputs, when compared to their SR-based STDs counterparts. As an example of this, the latency, throughput an energy consumption of (TFM-5, 1.20 V, 2.2 ns) are 0.78×, 2.55× and 0.23× those of (SR-1, 1.20 V, 4.0 ns), respectively. Furthermore, when the STDs operate at a supply voltage of 0.84 V, (TFM-5, 0.84 V, 4.1 ns) presents a latency of 82 μs, an energy consumption of 37 nJ/bit and a throughput of 1219 Kbps. These results suggest that (TFM-5, 0.84 V, 4.1 ns) may increase the throughput by a factor of 1.36, while consuming only 0.11× the energy of that of (SR-1, 1.20 V, 4.0 ns), without increasing the chip area, albeit at the cost of increasing the latency by a factor of 1.46.

The significantly improved dynamic energy efficiency of TFM-5 relative to SR-1 may be mainly attributed to the use of TFM-based EMs, instead of SR-based EMs. More specifically, the energy consumption per TFM per DC in active- and standby-mode is only 6 nJ and 0.2 nJ, respectively, compared to 88 nJ and 4.37 nJ for an SR, when the supply voltage is set to 1.20 V. As detailed in Section V-B, the static energy consumption may be expected to be higher in the SR-3 to SR-5 and TFM-3 to TFM-5 schemes than in the SR-1 to SR-2 and TFM-1 to TFM-2 arrangements, respectively. However, the static energy consumption of TFM-5 may be expected to be similar to that of SR-1, since these designs have similar chip areas AND gate counts.

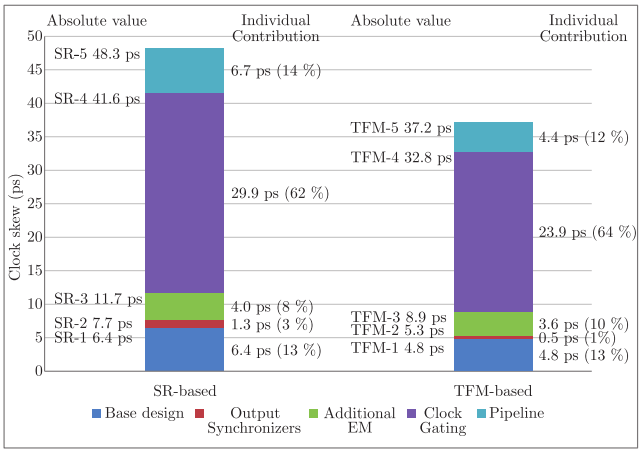


FIGURE 18. Clock skew of the different STD implementations.

Figure 18 quantifies the effect of each proposed enhancement of Section V on the clock network of the STD. Each successive enhancement increases the clock skew of the STD, since each enhancement is achieved at the cost of increasing the number of DFFs in the STD, with the main source of

clock skew in the proposed designs being the insertion of clock gating for the sake of reducing the dynamic energy consumption of the STDs. The increased clock skew of the proposed designs increases the likelihood of timing errors occurring owing to power supply variations. As part of our design methodology, these clock skew results are considered, when performing the timing analysis of the different STDs implementations in Section VII-A for determining the critical clock period of each design.

VII. ERROR CORRECTION CAPABILITIES OF THE TIMING-ERROR TOLERANT STOCHASTIC TURBO DECODERS

In this section, we characterize the decoding performance of the STD in the presence of timing errors, when BPSK modulation is used for communication over an AWGN channel.

A. TIMING ERROR MODEL

Supply voltage variations in an ASIC may be caused by effects such as IR-drop, $L \cdot d_i/d_t$ noise, crosstalk, electrostatic discharges, particle strikes, switching noise and fabrication process variations [14]–[18], among other causes. In accordance with [65], we model these effects by representing the supply voltage of the ASIC with a Gaussian distribution having a mean of μ, which represents the nominal supply voltage, and a standard deviation of σ, which represents the power supply variations. In our analysis, the clock period T_{CLK}, the nominal supply voltage μ and the degree of power supply variations σ are fixed for the whole operation of the decoder [65]. However, a different random value of V_{DD} is selected from the Gaussian distribution for each clock cycle, which is then used for all gates in the STD during the current clock cycle. The selected values of μ are 1.20 and 0.84, corresponding to the nominal supply voltage of TSMC 90 nm technology and this value scaled down 30%, respectively. The value of σ is selected to obtain particular values of $3\sigma/\mu$, namely 0.03, 0.05 and 0.07 which correspond to the three standard deviation variation fraction of the selected supply voltage, as detailed in [66]. Similarly, the selected values of T_{CLK} correspond to the critical clock periods of each scheme presented in Section VI. More specifically, we considered the clock skews of Section VI and a short-path timing analysis for creating feasible timing budgets and for determining the corresponding critical clock period of each design.

In order to characterize the presence of timing errors in the STD, we performed a post-layout timing analysis with extracted parasitics and annotated delays of the different TSMC 90 nm implementations of the STDs for a range of supply voltages. Figure 19 presents only the largest propagation delays of the various STDs under different values of the supply voltage. However, for the purpose of our timing error model, we consider the critical path delay of every DFF in each of the different STDs separately. In each clock cycle, the chosen value of V_{DD}, T_{CLK} and Figure 19 are used for determining the delay encountered by each signal propagating to each DFF.

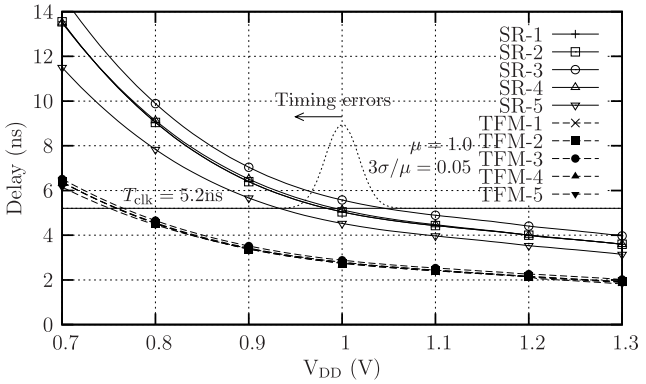


FIGURE 19. Delays of critical paths of different implementations of the STD.

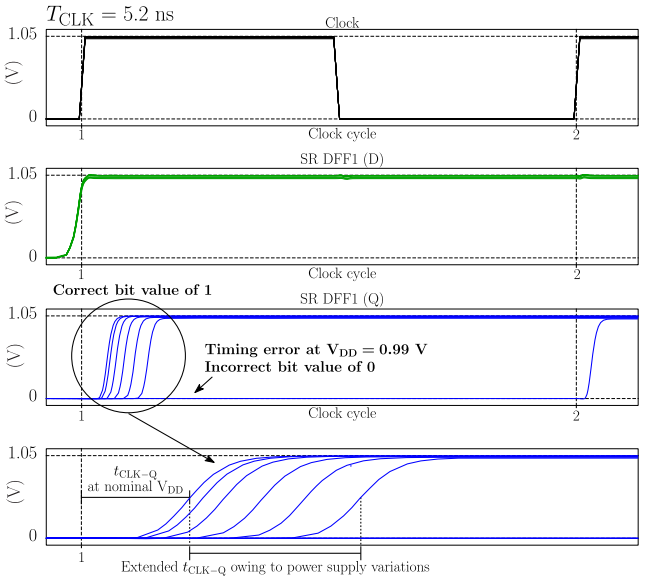


FIGURE 20. SPICE simulation of a critical path under different supply voltages of $V_{DD} \in \{0.99, 1.00, 1.01, 1.02, 1.03, 1.04, 1.05\}$ V.

Since the STDs operate at their critical clock period, timing errors will be encountered if the selected value V_{DD} is smaller than the nominal supply voltage μ. Similarly, we assume that timing errors occur, whenever the delays are larger than the fixed T_{CLK}. As a result of this, some paths will experience timing errors in some clock cycles, but not in others. Similarly, a large overall number of timing errors will occur in some clock cycles and only a small number of timing errors will be encountered in others. If a timing error is indeed encountered, our error model assumes that a random bit value will be clocked into the affected DFFs and the same value will be propagated through the circuit in the subsequent clock cycles. This random value responds to unpredictable glitches and late transitions caused by timing errors, as shown in Figure 20, as well as resolved metastable states owing to the use of synchronizers [61]. For illustrative purposes, Figure 19 includes a Gaussian distribution associated with $\mu = 1.0$, $3\sigma/\mu = 0.05$ (nominal V_{DD} of 1.0 V and

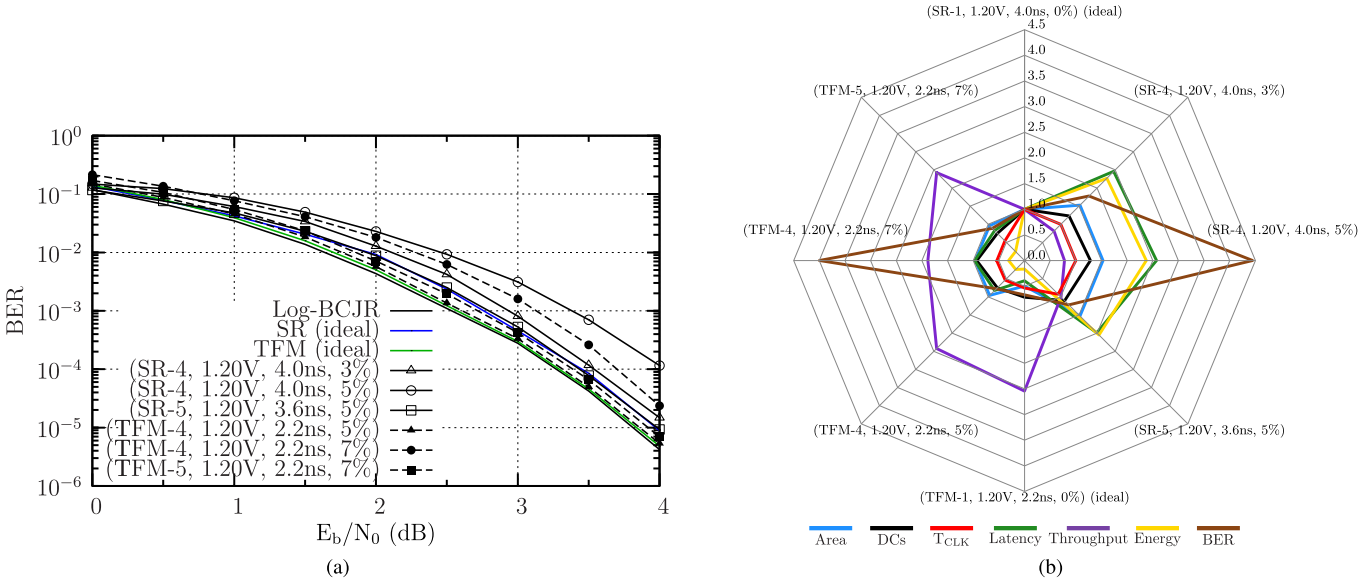


FIGURE 21. BER and hardware and performance of the modified STDs in the presence of timing errors and power supply variations when $V_{DD} = 1.20$ V: (a) BER performance of the modified STDs operated at $V_{DD} = 1.20$ V. (b) Hardware performance of the modified STD operated at $V_{DD} = 1.20$ V and $E_b/N_0 = 3.0$ dB.

three-sigma standard deviation power supply variation of 5%) and a clock period of $T_{CLK} = 5.2$ ns. In this particular example, SR-3 will be subject to timing errors if the selected value of V_{DD} drops below 1.05 V, since the critical path delay exceeds the critical clock period in this condition, as shown in Figure 20. More specifically, the waveforms of Figure 20 were obtained using SPICE simulations of the critical path in the SR-3 scheme, when using ST90 nm technology operated at different supply voltages and at a fixed critical clock period of $T_{CLK} = 5.2$ ns. Owing to this, the signal output by the critical path arrives at the D input of the first DFF in the SR-based EM at the same time, as the first clock edge shown in Figure 20. Here, power supply voltages of 1.00 V $\leq V_{DD} \leq$ 1.05 V are sufficiently high to avoid generating a timing error in the DFF. However, if the supply voltage is reduced to $V_{DD} = 0.99$ V, a timing error is observed. In this scenario, a bit value of 0 is erroneously stored in the SR DFF and presented at its Q output after the first clock edge of Figure 20. In addition to this, Figure 20 demonstrates that the propagation delay t_{CLK-Q} of the DFF is increased, when the supply voltage is reduced, increasing the likelihood of timing errors and metastability occurring in the subsequent DFFs, as detailed in Section V. By contrast, the TFM-based STDs will not experience any timing errors in this configuration, since their critical paths have lower critical clock periods.

B. DECODING PERFORMANCE IN THE PRESENCE OF TIMING ERRORS

The notation (Scheme, V_{DD}, T_{CLK}, %) will be used in this section to refer to a specific STD implementation operated at the given supply voltage, clock period and three standard deviation percentage of power supply noise $3\sigma/\mu$, respectively. Figure 21 presents the BER and hardware performance of the SR- and TFM-based STD, when operated at $V_{DD} = 1.20$ V. Figure 21a presents the BER performance for a floating point implementation of the conventional LogBCJR turbo decoder using 8 decoding iterations as a benchmark. We also present BER plots of the SR- and TFM-based STD, when allowing a maximum of 10^5 DCs in the absence of timing errors, which confirm that STDs are capable of achieving similar BER performances as the ideal floating point turbo decoder. Note that SR-1 to SR-5 offer identical BER performance in the absence of timing errors, since the proposed modifications only impact the BER in the presence of timing errors. Likewise, TFM-1 to TFM-5 offer identical BER performance in the absence of timing errors. As mentioned in Section V-A, our timing analysis of Section VII-A suggests that thousands of metastability events occur when the different STDs are operated continuously in the presence of power supply variations. Since SR-1 and TFM-1 do not consider the employment of synchronizers for preventing the catastrophic propagation of metastability, we do not plot their BER in the presence of timing errors, since it would be very poor. Figure 21a presents the BER of the modified schemes SR-4, SR-5, TFM-4, TFM-5 when operated at $V_{DD} = 1.20$ V. Note that the BER performances of SR-2, SR-3, TFM-2 and TFM-3 are not included in Figure 21a, since SR-2 and SR-3 offer similar BER performance to that of SR-4. Similarly, TFM-2 and TFM-3 offer similar BER performance to that of TFM-4. However, schemes SR-2 and SR-3 and schemes TFM-2 and TFM-3 offer different chip area, latency, throughput and energy consumption, when compared to SR-4 and TFM-4, respectively, as described above in Section VI.

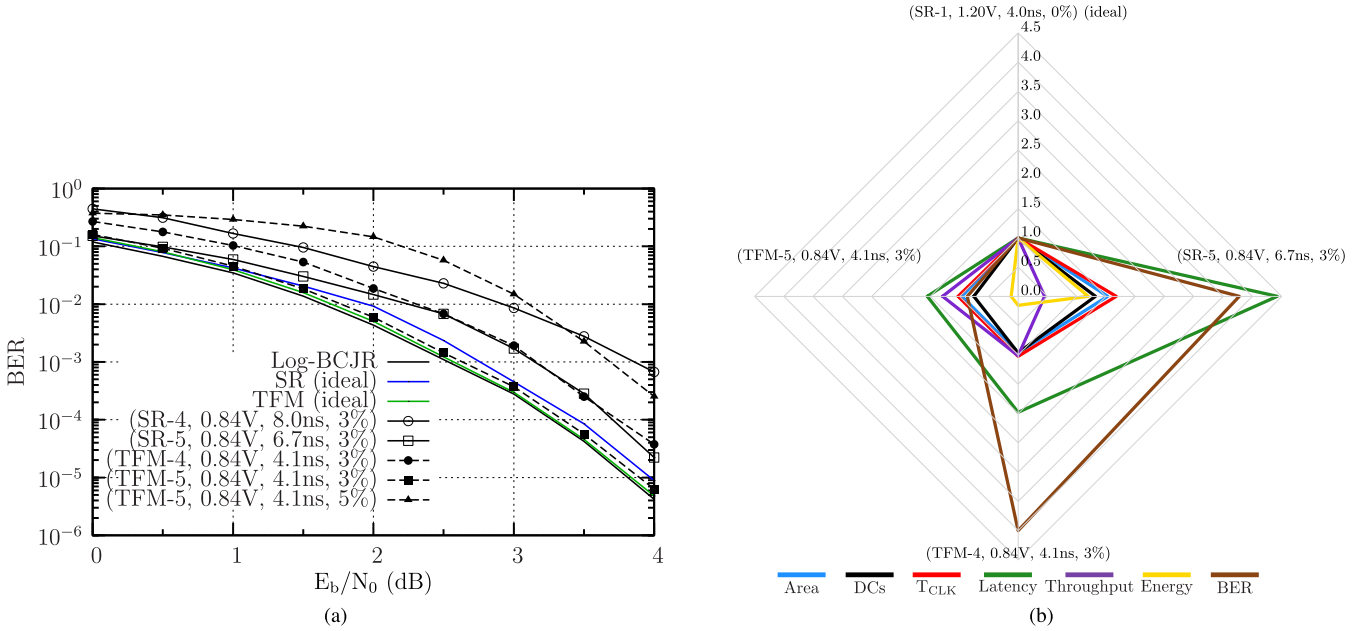


FIGURE 22. BER and hardware and performance of the modified STDs in the presence of timing errors and power supply variations when $V_{DD} = 0.84$ V: (a) BER performance of the modified STDs operated at $V_{DD} = 0.84$ V. (b) Hardware performance of the modified STDs operated at $V_{DD} = 0.84$ V and $E_b/N_0 = 3.0$ dB.

Figure 21a demonstrates that the proposed STDs offer an enhanced tolerance to timing errors. As an example of this, Figure 21a shows that (SR-4, 1.20 V, 4.0 ns, 3%) exhibits an E_b/N_0 degradation of about 0.14 dB respect to the ideal BER performance of SR-1. Similarly, (TFM-4, 1.20 V, 4.0 ns, 5%) offers similar BER performance of that of the TFM-based STD in the absence of timing errors. However, schemes SR-4 and TFM-4 exhibit an E_b/N_0 degradation of about 0.75 dB and 0.5 dB, when the three standard deviation percentage of power supply noise is increased to 5% and to 7%, as shown in Figure 21a in (SR-4, 1.20 V, 4.0 ns, 5%) and (TFM-4, 1.20 V, 4.0 ns, 7%), respectively,. Moreover, Figure 21a demonstrates that the inclusion of the pipeline stage described in Section V-D improves the BER performance of the STDs by preventing the occurrence of timing errors during the estimation of the decoded bit $b_{1,k}$. This is shown in the BER performance of (SR-5, 1.20 V, 4.0 ns, 5%), which exhibits similar BER performance of SR-1. Likewise (TFM-5, 1.20 V, 4.0 ns, 7%) presents similar error correction capabilities to that of (SR-1, 1.20 V, 4.0 ns, 0%), which corresponds to the state-of-the-art STD of [36] with no timing errors. This corresponds to an E_b/N_0 degradation of about 0.1 dB, when compared to TFM-1 in the absence of timing errors.

In addition to the BER performance, Figure 21b shows the chip area, average number of DCs, latency, throughput and energy consumption per decoded bit of the proposed STDs, normalized relative to the benchmark SR-1 in the absence of timing errors, when the different STDs operate at $V_{DD} = 1.20$ V and $E_b/N_0 = 3.0$ dB. We also present the hardware performance of TFM-1 in the absence of timing errors as a benchmark. Figure 21b shows that the average number of DCs required by (SR-4, 1.20 V, 4.0 ns, 3%), (SR-4, 1.20 V, 4.0 ns, 5%) and (SR-5, 1.20 V, 3.6 ns, 5%) is 1.23×, 1.29× and 1.10× that of SR-1 in the absence of timing errors, respectively. This, in addition to the decoding of two frames in alternate clock cycles, is reflected in increased latencies and energy consumptions. However, (SR-5, 1.20 V, 3.6 ns, 5%) exhibits a similar throughput to that of SR-1 in the absence of timing errors, albeit at the cost of an increased BER. Figure 21b shows that TFM-based decoders in the presence of timing errors offer a reduced number of DCs and latencies, which facilitate increased throughputs and reduced energy consumptions, when compared to the benchmark SR-1 in the absence of timing errors. More specifically, the latency, throughput and energy consumption of (TFM-5, 1.20 V, 2.2 ns, 7%) are 0.83×, 2.42× and 0.25× those of SR-1 in the absence of timing errors, respectively.

Figure 22a demonstrates that the BER performance of the different STDs is severely affected when they are operated at $V_{DD} = 0.84$ V and in the presence of power supply variations, owing to the quadratic dependency of delays on the power supply. More specifically, Figure 22a shows that (SR-4, 0.84 V, 8.0 ns, 3%) exhibits an E_b/N_0 degradation of about 1.1 dB, compared to the ideal BER performance of the SR-1. The introduction of the pipeline stage into SR-4 reduces the E_b/N_0 degradation to 0.3 dB, which can be observed in the BER performance of (SR-5, 0.84 V, 6.7 ns, 3%). A similar trend is encountered in the TFM-based STDs, where (TFM-4, 0.84 V, 4.1 ns, 3%) exhibits a E_b/N_0 degradation of about 0.5 dB respect to the ideal BER performance of TFM-1. The introduction of the pipeline stage into TFM-5 enhances the BER performance,

as shown in the BER plot of (TFM-5, 0.84 V, 4.1 ns, 3%), which exhibits an E_b/N_0 degradation of only about 0.1 dB. However, when the percentage of power supply variation is increased to 5% (TFM-5, 0.84 V, 4.1 ns, 5%) exhibits a E_b/N_0 degradation of about 1.0 dB. Figure 22b presents the hardware performance of the modified STDs when they operate at $V_{DD} = 0.84$ V and $E_b/N_0 = 3.0$ dB. Here, we present hardware performance results for SR-1 operated at $V_{DD} = 1.20$ V and in the absence of timing errors as benchmark. Note that schemes (SR-4, 0.84 V, 8.0 ns, 3%) and (TFM-5, 0.84 V, 4.1 ns, 5%) are not considered in Figure 22b, owing to their increased BER. Figure 22b shows that (SR-5, 0.84 V, 6.7 ns, 3%) increases the number of DCs, latency, energy consumption and BERs performance, when compared to (SR-5, 1.20 V, 4.0 ns, 0%). Scheme (TFM-4, 0.84 V, 4.1 ns, 3%) exhibits a throughput similar to that of SR-1 in the absence of timing errors. However, this modified scheme presents an increased latency and BER. By contrast, the average number of DCs, latency, throughput and energy consumption of (TFM-5, 0.84 V, 4.1 ns, 3%) are 0.75×, 1.55×, 1.28× and 0.12× those of (SR-1, 1.20 V, 4.0 ns, 0%), respectively.

Owing to the hardware implementation results of Table 1, and Figures 21b and 22b and to the BER performance of Figures 21a and 22a, we recommend the employment of TFM-5 in the presence of timing errors.

VIII. REDUCED-LATENCY STOCHASTIC TURBO DECODER

The different STD implementations presented in Section IV require a large number of DCs for successfully decoding a frame, hence resulting in relatively poor processing latencies, throughputs and energy efficiencies. As mentioned in Section I, this drawback may prevent the employment of STDs in practical low-latency next-generation Mission-Critical Machine-Type Communication (MCMTC) systems, such as those required by vehicular traffic safety and control [19]. To overcome this problem, the authors of [36] proposed exponential transformations [67] for the implementation of STDs. This technique uses stochastic computing to perform the Taylor's expansion of the exponential function of BSs. In this way, the additions of BSs are transformed into multiplications of exponentially transformed BSs, which can be performed with the aid of AND gates, as detailed in Section III-A. However, the result of the multiplication of the exponentially transformed BSs must be converted back into the conventional BS representation, by employing a logarithmic transformation. The order of the Taylor's series determines not only the accuracy of the exponential transformation, but also the hardware complexity of its implementation, as detailed in [60]. As an example of this, a NOT gate may be employed for the first order approximation of both exponential and logarithmic functions. By contrast, the second order approximation requires one DFF and two NAND gates for the exponential function, as well as one DFF, two AND gates and a 2-input MUX for the logarithmic function. The error correction capability of the STD is improved by using higher order approximations, but the hardware complexity grows rapidly. Motivated by this, the authors of [36] applied the second order exponential transformation to a 200-bit STD. This technique reduced the maximum number of DCs from 250×10^3 to 32×10^3, without degrading the error correction performance of the STD, albeit at the cost of increasing the hardware complexity of the design. In order to further reduce the number of DCs, the STD of [37] employed the multiple-stream decoding technique, which was originally introduced for the stochastic decoding of cortex codes in [34]. This technique increases the degree of parallelism for the STD by representing each probability with $\rho \geq 2$ BSs. The exponential transform-based multiple-stream decoding of the 200-bit STDs in conjunction with $\rho = 32$ allowed the reduction of the number of DCs from 32×10^3 to 1×10^3,* albeit at the cost of increasing the hardware complexity by a factor of $\rho = 32$. Hence, this technique may be deemed unsuitable for practical STD implementations.

The authors of [38] and [39] proposed modified BS representations for the implementation of fully-parallel LogBCJR decoders. More specifically, these contributions proposed sign-magnitude BSs for representing Logarithmic Likelihood Ratios (LLRs). Here, each LLR is represented using one BS for determining its sign and one or more BSs for determining its magnitude. As an example of this, [38] employs 2-bit BS-based LLRs in the range of $[-1, +1]$. In [39], the authors proposed a sliding-window method for converting BSs into bit-serial sign-magnitude LLRs. In each DC, the three most-recent bits of a BS are combined to provide an LLR comprising one sign bit and two magnitude bits. Therefore, each LLR is represented in the range of $[-3, +3]$. Additionally, this implementation relies on 3-bit fixed-point adders and 4-bit fixed-point comparators for the addition and max operations of the sign-magnitude BSs, respectively. These techniques significantly reduce the number of DCs required for successfully decoding a frame and hence yield substantial throughput gains. However, these designs may be considered to be half-stochastic STDs or low-precision serially-operated fixed-point LogBCJR decoders, rather than true-stochastic decoders. Owing to its reliance on fixed-point numbers, the half-stochastic STD of [39] does not benefit from the inherent tolerance of true STDs to timing errors.

Motivated by this, in this section we propose a Reduced-Latency Stochastic Turbo Decoder (RLSTD), which achieves an ultra-low latency without relying on fixed-point numbers. More specifically, the following sections detail the proposed improvements of the STD of [36], which first of all significantly reduce the number of DCs required for successfully decoding a frame, whilst improving the characteristics of its hardware implementation. Each of these enhancements is detailed in the following subsections.

*The number of DCs is reported as 1×10^3 in [37]. However, in the more detailed description of the multiple-stream STD given in the PhD thesis of [60], the same architecture and implementation is reported to employ 8×10^3 DCs.

A. APPROXIMATE STOCHASTIC ADDERS

As described in Section II, the turbo decoding algorithm requires the addition of probabilities, as shown in Equations (3) and (6). In stochastic computing, BSs can only represent probabilities in the range of [0,1]. However, the addition of M probabilities may not give a result falling in the closed interval [0,1]. To overcome this problem, the addition of M probabilities may be performed by scaling the operands by a factor of M, so that we have $P_{add} = \sum_{i=1}^{M}[P_i/M]$, which represents the mean of the M probabilities. In stochastic computing, this may be performed using an M-input MUX, as described in Section III-A. In this configuration, the MUX outputs the value of one of the M input BSs, which is randomly selected in each DC. However, this means that the other $(M-1)$ BSs do not directly contribute to the output BS of the MUX. As a result of this, the length of the outgoing BS P_{add} is required to be M times longer than that of the input BSs, in order for P_{add} to achieve the same precision as the input BSs [37].

As an alternative to the employment of MUXs as stochastic adders, OR gates may be employed for performing approximate additions, as detailed in [13]. This has the advantage of granting all M input BSs influence over the output BS, hence reducing the length required for the output to achieve the same precision as the inputs. However, the probability P_{or} represented by the BS output by a two-input OR gate is given by $P_{\text{or}} = P_A + P_B - P_A \cdot P_B$, where P_A and P_B correspond to the probabilities represented by the two input BSs S_A and S_B, respectively. Therefore, the addition P_{or} exhibits an error that is proportional to the product $P_A \cdot P_B$, although this will become negligible, if either P_A or P_B has a very small value.

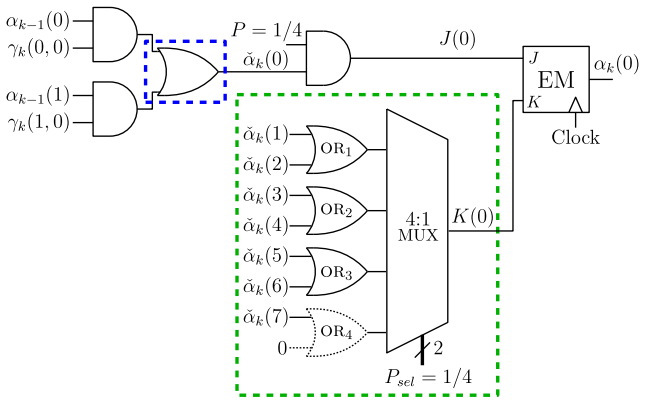


FIGURE 23. Stochastic realization of $\alpha_k(0)$ employing OR gates as approximate adders.

Figure 23 shows how OR gates may be employed to perform approximate additions in an STD. More specifically, Figure 9 exemplifies the stochastic implementation of the forward recursion $\alpha_k(s)$ of (3) for the case where $s = 0$, as it was previously shown in Figure 9. The blue box of Figure 23 at the top left corner shows the employment of a 2-input OR gate to approximate the two-term addition $\check{\alpha}_k(0) = [\alpha_{k-1}(0)\gamma_k(0,0) + \alpha_{k-1}(1)\gamma_k(1,0)]$. Similarly, the green box at the bottom right corner of Figure 23 shows the employment of OR gates for the approximate addition of 8 BSs. Note that an M-input OR gate outputs the value of 1 if any of its M inputs adopts this value. As a result of this, the employment of an 8-input OR gate required for providing $K(0)$ in Figure 23 may result in this BS becoming stuck at 1. Owing to this, the green box of Figure 23 implements the addition $K(0) = [\sum_{i=1}^{7}[\check{\alpha}_k(i)] + 0]/4$ using four 2-input OR gates and a single 4:1 MUX. The division by 4 in $K(0)$ requires the corresponding division by 4 to be employed in $J(0) = \check{\alpha}_k(0)/4$, in order to preserve $\alpha_k(0) = J(0)/[J(0) + K(0)]$. This may be achieved using an AND gate for multiplying $\check{\alpha}_k(0)$ with a BS representing the probability of 1/4, as shown in Figure 23. Similar structures to that of Figure 23 may be employed for the implementation of the forward recursion $\alpha_k(s)$ for all other states $s \in [1, 7]$, as well as for the backward recursion $\beta_{k-1}(s')$ of (4). Note that the OR gate labeled as or_4 in Figure 23 is drawn with dotted lines for indicating that this gate may be eliminated, since one of its inputs has the constant logical value of 0. However, this OR gate is indeed required for the approximate additions that may be used for the calculation of the *extrinsic* probabilities of (5) and for the calculation of the APP of (6), as shown in Figures 24 and 25, respectively.

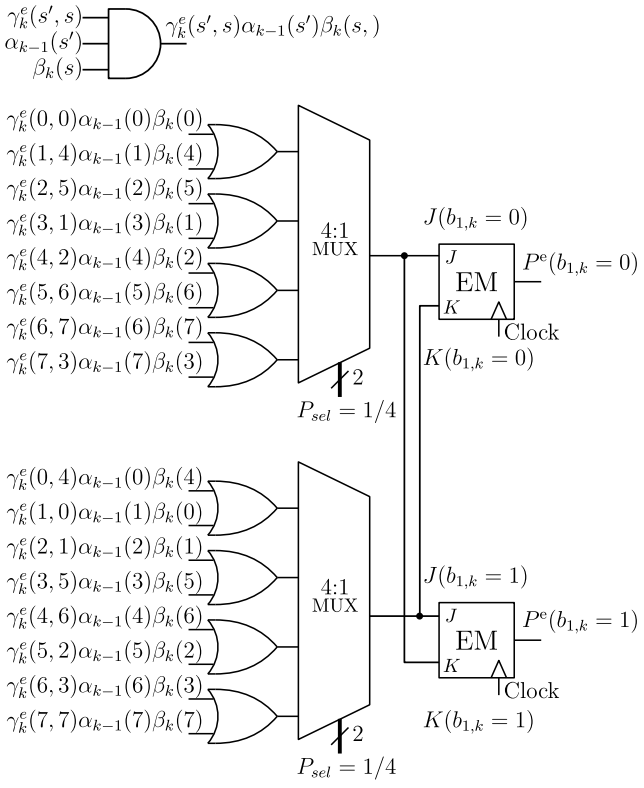


FIGURE 24. Estimation of the *extrinsic* probabilities in the RLSTD.

As we will demonstrate in Section IX, the employment of the approximate adders imposes only an imperceptible BER performance degradation on the RLSTD. By contrast,

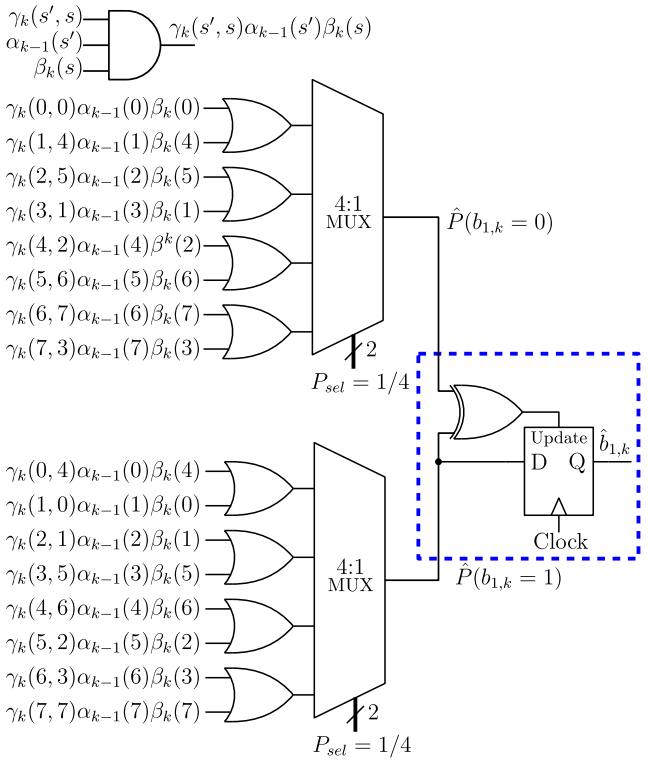


FIGURE 25. Estimation of the hard-decision bit $\hat{b}_{1,k}$ in the RLSTD.

the approximate stochastic adders reduce the chip area, the average number of DCs required, the latency and the energy consumption, while increasing the throughput of the RLSTD.

B. REDUCED-COMPLEXITY TRACKING-FORECAST MEMORY

As detailed in Section VI, TFM-based EMs offer significant improvements in both the BER and the hardware implementation performance of STDs. This may be attributed to the TFM's enhanced capability for tracking changes in the regenerative bit's probability and to the relatively low hardware complexity of TFMs. As described in Section V-C, TFMs employ (14) for quantifying the moving average probability $P(t+1)$ of the regenerative bit $J(t)$ having the value of 1 according to

$$P(t+1) = \begin{cases} P(t) - \phi P(t) & J(t) = 0 \\ P(t) + \phi \bar{P}(t) & J(t) = 1. \end{cases}$$

The relaxation parameter ϕ determines the significance given to the regenerative bit and its value can be chosen by obtaining a compelling trade-off between the BER and hardware efficiency, where we use $\phi = 2^{-4}$ in the TFM-based STDs presented in Section IV. However, in this section, we propose the use of $\phi = 2^{-1}$, since this facilitates a significant further reduction in the hardware implementation complexity of TFMs. In this case, the probability of the TFM can be expressed as

$$P(t+1) = \begin{cases} P(t) - \dfrac{P(t)}{2} = \dfrac{P(t)}{2} & J(t) = 0 \\ P(t) + \dfrac{\bar{P}(t)}{2} = P(t) + \dfrac{1 - P(t)}{2} & \\ \qquad = \dfrac{P(t)}{2} + \dfrac{1}{2} & J(t) = 1. \end{cases} \quad (15)$$

We refer to the resultant scheme as Reduced-Complexity Tracking Forecast Memory (RCTFM), which may be implemented using the arrangement shown in the green box of Figure 26. Here, the fixed-point TFM represents $P(t)$ using a 9-bit unsigned fixed-point number, where the MSB has a significance of 2^{-1} and the Least Significant Bit (LSB) has a significance of 2^{-9}. The multiplication $P(t)/2$ can be realized by shifting the contents of the fixed-point TFM one position to the right. Moreover, the conditional addition of the constant $1/2$ can be realized by updating its MSB with the incoming regenerative bit $J(t)$. Figure 26 compares the hardware complexity of the proposed RCTFM to those of the TFMs employed in schemes TFM-1 to TFM-5 of Section IV. In particular, the hardware implementation complexity of the RCTFM is reduced by avoiding the employment of the fixed-point adder/subtractor and the set of XOR gates employed by the TFM shown in the blue box of Figure 26.

The operation of the RCTFM-based EM follows the same principles as the SR-based and TFM-based EMs described in Sections III-B and V-C, respectively. Moreover, the RCTFM operates as a combination of an SR and a TFM. To elaborate further, the RCTFM adopts the functionality of an SR, when $J \neq K$. In this situation, the DFFs of the RCTFM will shift their contents one position and the incoming regenerative bit $J(t)$ will be stored in the first DFF, in analogy to the behavior of an SR-based EM. Here, the clock-gating latch and the AND gate are employed to enable the clock signal of the DFFs only when the signal Update is asserted, owing to $J \neq K$. By contrast, if $J = K$, the unsigned fixed-point probability $P(t)$ stored in the RCTFM is compared to a pseudo-random number, in order to determine the outgoing bit of the RCTFM, in analogy to the operation of a TFM-based EM. Here, the outgoing bit assumes the value of 1 if the probability stored in the RCTFM is larger than or equal to the 9-bit pseudo-random number. The contents of the RCTFM-based EM can be initialized prior the beginning of the decoding process, in order to improve the attainable BER performance of the STD. In our investigations detailed in the following sections, the RCTFMs are initialized to store the probability $P = 0.5$, which can be achieved in a single clock cycle by setting the MSB of each RCTFM to logic 1 and the rest of the bits to logic 0.

C. OUTPUT DECISION

The STD presented in Section IV employs a 4-bit saturated up/down counter for the estimation of the decoded bit $\hat{b}_{1,k}$. The saturated counter stores a binary value that behaves similarly to a fixed-point representation of an LLR [26], with its

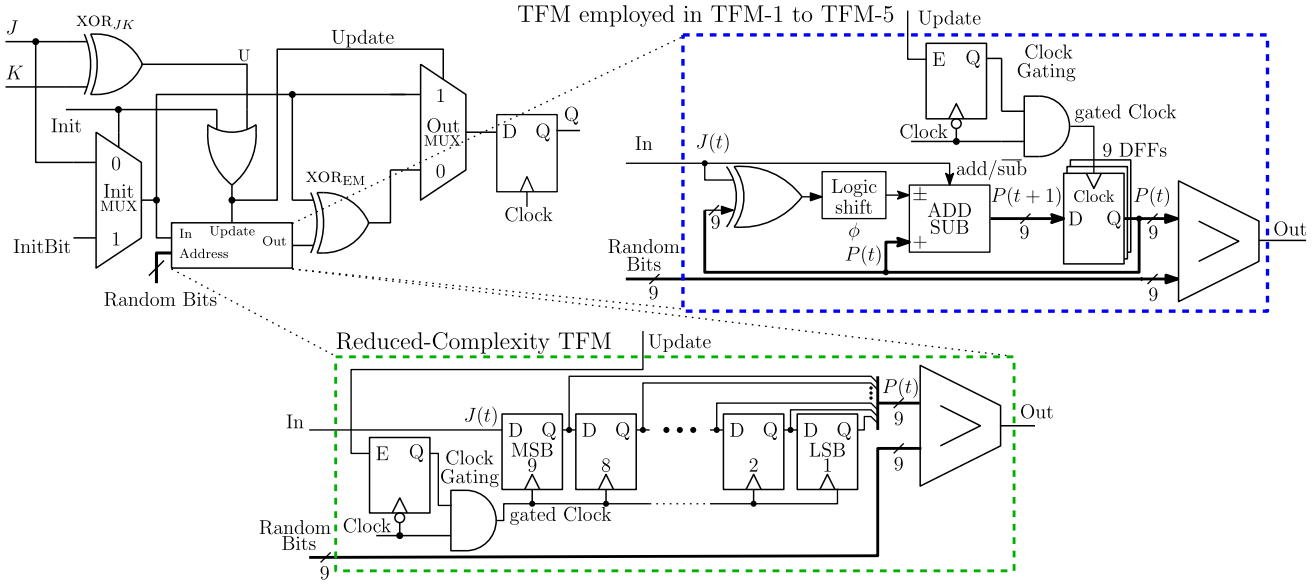


FIGURE 26. Reduced complexity TFM associated with $\phi = 2^{-1}$, compared to the TFM structure employed in TFM-based STDs of Section IV.

MSB determining the hard-decision bit $\hat{b}_{1,k}$, as described in Section IV-D. The width of the saturated counter determines both its capability to track changes in the represented LLR and the precision of its representation of this LLR. In this way, a wider counter provides a higher precision for the represented LLR, as well as a more robust mechanism against rapid variations in the value of this LLR. This is particularly useful at low channel SNR values, where the LLR represented may be expected to fluctuate between consecutive DCs, owing to the low reliability of the decoding process in this SNR region. By contrast, smaller counter widths offer a lower precision for the represented LLR and are more susceptible to changes within the BSs. Despite this, low-precision counters may be employed at high SNR values, owing to the high reliability of the decoding process, as detailed in [26]. In spite of these trade-offs, we propose the employment of only a single DFF for determining the hard-decision bit $\hat{b}_{1,k}$ of the RLSTD, as shown in Figure 25. Here, the output DFF takes the value of 0 if $\hat{P}(b_{1,k} = 0) = 1$ and $\hat{P}(b_{1,k} = 1) = 0$. Similarly, the output DFF takes the value of 1 if $\hat{P}(b_{1,k} = 0) = 0$ and $\hat{P}(b_{1,k} = 1) = 1$. This is achieved by updating the output of the DFF with the bit of the BS representing $\hat{P}(b_{1,k} = 1)$, when $\hat{P}(b_{1,k} = 0) \neq \hat{P}(b_{1,k} = 1)$, as shown in the blue box of Figure 25. By contrast, the DFF will not update its contents, if $\hat{P}(b_{1,k} = 0) = \hat{P}(b_{1,k} = 1)$.

Section IX will demonstrate that the error correction capability of the RLSTD is not degraded by having only a single output DFF. Moreover, Section X will demonstrate that the single output DFF reduces the hardware complexity of the STD.

IX. ERROR CORRECTION CAPABILITIES OF THE REDUCED-LATENCY STOCHASTIC TURBO DECODER

In this section, we characterize the error correction capability of the proposed RLSTD of Section VIII and compare it to that of various benchmarkers. In order to allow direct comparison with the results of [36] and [39], Figure 27 presents the BER performance achieved for different STDs employing 50-bit and 200-bit frames, a coding rate of 1/3, 8 states, tailbiting, S-Random interleavers, the state-transition diagram of Figure 4c and an NDS associated with $\eta = 1$ and $\psi = 2$, as described in Section III-B. All results assume BPSK transmission over an AWGN channel. Figure 27 presents BER plots for the RLSTD described in Section VIII, when allowing a maximum of 10×10^3 DCs and when employing early stopping of the decoder's iterations upon achieving convergence. We also present BER plots for four benchmarkers, namely the floating-point LogBCJR and maxLogBCJR turbo decoders, when allowing a maximum of 8 iterations, as well as for the schemes SR-1 and TFM-1 described in Sections IV and V, respectively, when allowing a maximum of 100×10^3 DCs. Figure 27 demonstrates that the RLSTD exhibits a similar BER performance to that

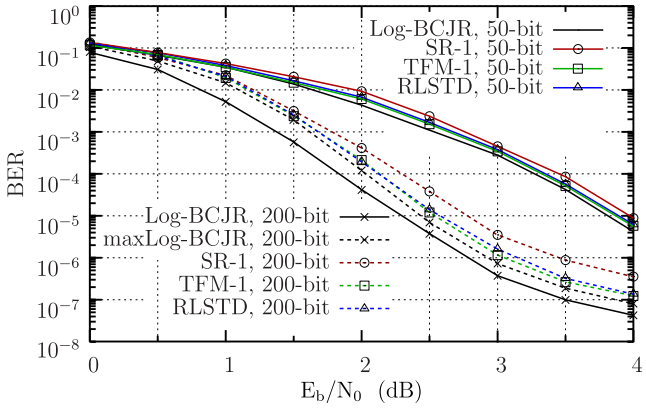


FIGURE 27. BER performance of the RLSTD, as well as of various benchmarkers.

TABLE 2. Hardware complexity comparison of the SR-1, TFM-1 and RLSTD schemes.

Module	Scheme	AND2	OR2	XOR2	MUX2	MUX4	MUX8	DFFs
γ	SR-1	8	0	0	0	0	0	0
	TFM-1	8	0	0	0	0	0	0
	RLSTD	8	0	0	0	0	0	0
α	SR-1	40	8	16	280	0	8	264
	TFM-1	40	8	16	24	0	8	73
	RLSTD	32	40	16	24	8	0	73
β	SR-1	40	8	16	280	0	8	264
	TFM-1	40	8	16	24	0	8	73
	RLSTD	32	40	16	24	8	0	73
Ext	SR-1	32	2	16	68	0	2	66
	TFM-1	32	2	16	2	0	2	20
	RLSTD	32	10	16	2	2	0	20
δ	SR-1	32	0	0	0	0	2	4
	TFM-1	32	0	0	0	0	2	4
	RLSTD	32	8	1	0	2	0	1
TOTAL	SR-1	**152**	**18**	**48**	**628**	**0**	**20**	**598**
	TFM-1	**152**	**18**	**48**	**50**	**0**	**20**	**170**
	RLSTD	**136**	**98**	**49**	**50**	**20**	**0**	**167**

of the TFM-1 scheme, which confirms that the proposed modifications do not degrade the attainable error correction performance of the RLSTD. More specifically, the 50-bit RLSTD exhibits near-optimal decoding performance, when compared to the floating-point LogBCJR decoder. However, the 200-bit RLSTD exhibits an E_b/N_0 degradation of up to 0.2 dB and 0.5 dB, when compared to the sub-optimal maxLogBCJR and to the LogBCJR turbo decoders, respectively. Note however that similar trends may be observed for the 200-bit TFM-1 STD described in Section V.

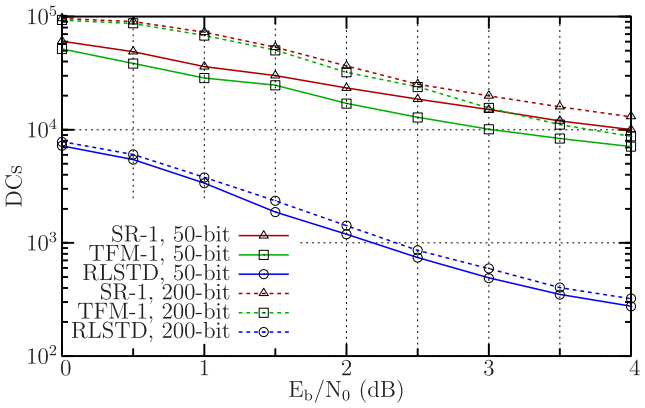


FIGURE 28. Average number of DCs for successfully decoding a frame for the RLSTD.

Figure 28 presents the average number of DCs required by the 50-bit and the 200-bit STDs for successfully decoding a frame, when operated at different E_b/N_0 values and when compared to the benchmarkers of SR-1 and TFM-1 presented in Sections IV and V, respectively. Figure 28 demonstrates that the proposed RLSTD reduces the average number of DCs by an order of magnitude, when compared to both the SR-1 and to the TFM-1 schemes. This may be attributed to the increased switching activity owing to the employment of OR gates as approximate adders and to the employment of a single DFF for providing the decision of the decoded bit. Figures 27 and 28 demonstrate that the proposed RLSTD significantly reduces the number of DCs required for successfully decoding a frame, without degrading its error correction capability. As a benefit of this, the latency, the throughput and the energy efficiency of the STD are significantly enhanced, as we will demonstrate in Section X.

X. HARDWARE IMPLEMENTATION OF THE REDUCED-LATENCY STOCHASTIC TURBO DECODER

Table 2 presents the hardware complexity of the RLSTD in terms of the number of basic logic gates and DFFs employed per decoded bit, when compared to the hardware complexity of the SR-1 and TFM-1 schemes of Table 1 used as benchmarkers. Table 2 demonstrates that the number of 2-input MUX gates employed in the TFM-1 and RLSTD schemes is significantly reduced, when compared to those of SR-1 described in Section IV. This may be attributed to SR-1 employing 32 MUXs for the update operation of each of the 18 SR-based EMs and to the employment of 2-input MUXs for the addition of BSs. Similarly, the number of DFFs employed in the TFM-1 scheme detailed in Section V and in the RLSTD scheme described in Section VIII is significantly reduced, owing to the employment of 9-bit TFMs. By contrast, the RLSTD scheme employs a higher number of 2-input OR gates, when compared to both SR-1 and TFM-1, owing to the employment of these logic gates for the approximate addition of BSs. However, Table 3 demonstrates that the increased number of OR gates does not increase the chip area or the overall gate count of the RLSTD.

Table 3 compares the hardware efficiency of different STDs, when using TSMC 90 nm technology. We present results for the rate 1/3, 8-state, tailbiting RLSTD, TFM-1 and SR-1 schemes, using S-Random interleavers and the state-transition diagram of Figure 4c, in order to allow direct comparison with the results of Table 1 in Section VI.

TABLE 3. Hardware efficiency of different STDs.

	RLSTD			TFM-1		SR-1 [36]		[39]
Algorithm	Stochastic			Stochastic		Stochastic		Half-stoch.
Area per bit (mm^2)	0.0062			0.0129		0.026		-
Gate count per bit	2.2 K			4.6 K		8.6 K		3.4 K
T_{CLK} Frequency	1.7 ns 588 MHz			2.2 ns 454 MHz		4.0 ns 250 MHz		1.8 ns 550 MHz
Frame length (bits)	50	200	2048	50	200	50	200	2048
BER @ E_b/N_0	10^{-5} 3.8 dB	10^{-5} 2.6 dB	10^{-5} 1.25 dB	10^{-5} 3.8 dB	10^{-5} 2.55 dB	10^{-5} 3.95 dB	10^{-5} 2.75 dB	10^{-5} 1.25 dB
Average DCs	200	800	6×10^3	7.5×10^3	19×10^3	10×10^3	20×10^3	280
Latency (μs)	0.34	1.36	10.2	16.5	41.8	40	80	0.504
Throughput (Mbps)	147	147	200	3.0	4.7	1.2	2.5	4000
Area eff. (bps/gates)	1336	334	44	13	5.1	2.8	1.5	574
Area eff. ($Mbps/mm^2$)	474	118	15.7	4.65	1.82	0.92	0.48	-
Energy eff. (nJ/bit)	0.76	3.04	31.13	41.25	104	240	480	-

Table 3 characterizes the various STDs in terms of their diverse characteristics, including the chip area per decoded bit, clock period, number of equivalent NAND gates, number of DCs, latency, throughput, as well as area and energy efficiency, when using TSMC 90 nm technology. Additionally, we employ NDS associated with $\eta = 1$ and $\psi = 2$ for the proposed RLSTD as well as for the SR-1 and TFM-1 schemes, as described in Section III-B. As described in Section VI, the results of Table 3 were obtained from the physical layout generated by the automatic place and route of the RLSTD, TFM-1 and SR-1 STDs using Cadence SoC Encounter. These results were obtained for the cases, where the supply voltage of the STDs is set to 1.20 V in the absence of power supply variations. Both the critical clock period and the energy consumption are obtained from Synopsys PrimeTime. The average number of DCs, latency, throughput and energy efficiency were obtained from post-layout gate-level simulations, with the extracted parasitics and annotated delays without timing errors, when using early stopping and allowing a maximum of 100×10^3 DCs for both SR-1 and TFM-1, as described in Section VI, as well as a maximum of 10×10^3 DCs for the proposed RLSTD described in Section VIII. We assume that the different STDs operate at their critical clock period, for BPSK transmission over an AWGN channel having an SNR, where a BER of 10^{-5} is achieved. We also present the hardware efficiency of the half-stochastic STD, as reported in [39]. Note that the hardware results of [39] correspond to the synthesis of a 2048-bit fully-parallel decoder using TSMC 90 nm technology. However, the authors of [39] did not quantify the area or energy consumption of this half-stochastic STD implementation, hence the corresponding characteristics cannot be shown in Table 3.

Table 3 demonstrates that the reduced number of DCs offered by the RLSTDs facilitate reduced latencies, increased throughputs as well as improved area and energy efficiencies, when compared to the benchmarkers SR-1 and TFM-1. As shown in Table 3, the proposed RLSTD has the lowest gate count and area per bit among all schemes considered. The proposed RLSTD enables the highest clock frequency and the lowest average number of DCs, when the frame length is 50 bits, resulting in the lowest latency among all schemes considered. This is particularly attractive in MCMTCs, where emergency or control messages comprising as low as tens of bits must be reliably communicated with ultra-low latency.

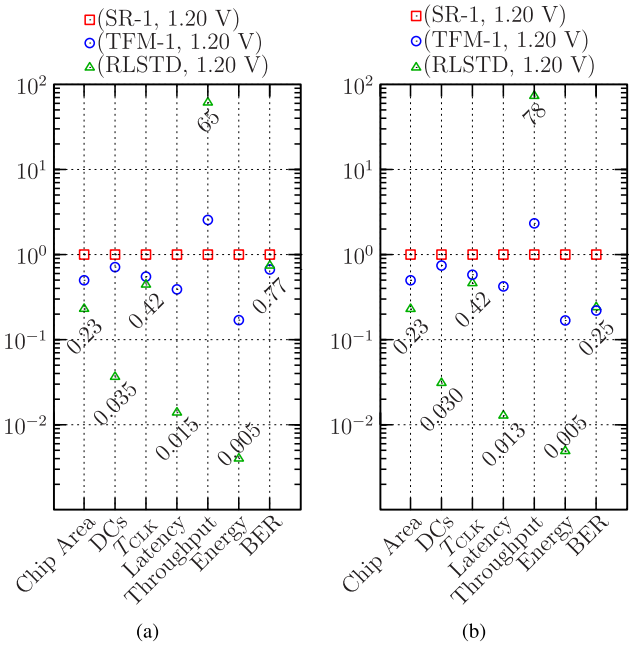


FIGURE 29. Hardware implementation results for different STDs, normalized relative to SR-1, when operated at $V_{DD} = 1.20$ V and when $E_b/N_0 = 3.0$ dB. (a) 50-bit STDs. (b) 200-bit STDs.

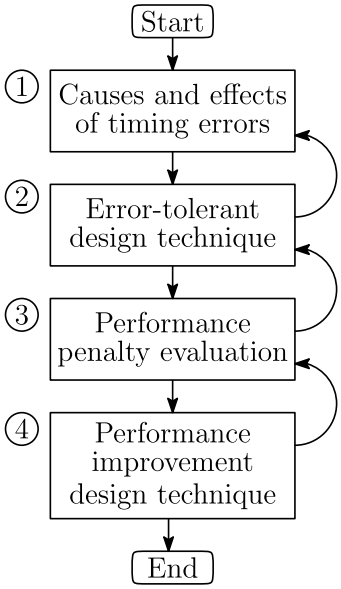


FIGURE 30. Design flow of error-tolerant iterative decoders.

Our simulations suggest that the number of DCs required by the proposed RLSTD scales approximately linearly with the frame length. We performed BER simulations of the RLSTD having a frame length of 2048 and an SNR of $E_b/N_0 = 1.25$ dB, which we found to result in the desired BER of 10^{-5}. Here, the average number of DCs required by the RLSTD is 6×10^3. Owing to this, the throughput and area efficiency of the RLSTD having a frame length of 2048 bits are significantly lower than those of the half-stochastic decoder of [39], as shown in Table 3. However, the half-stochastic decoder employs a fixed number of 280 DCs. Therefore, the latency of this design remains constant regardless of the frame length, as opposed to the RLSTD implementation. As a result, for a frame length of 50 bits, the latency of the half-stochastic decoder is 1.8 ns $\times$ 280 $=$ 0.504 μs and its throughput is 50/0.504 μs $=$ 99 Mbps, which are inferior to those of the proposed RLSTD. Based on these results, we recommend the employment of the proposed RLSTD for short-frame-length, ultra-low-latency applications, such as MCMTCs.

The hardware efficiency of the RLSTD is further detailed in Figure 29, which compares the hardware implementation trade-offs associated with the 50-bit and 200-bit versions of the TFM-1 and RLSTD schemes, relative to the benchmark scheme SR-1, for the case where $E_b/N_0 = 3.0$ dB. Figure 29a shows that the average number of DCs required by the proposed RLSTD scheme is as low as 0.035 times the corresponding number required by SR-1. As a result of this, the latency, throughput and energy consumption of the modified RLSTD are 0.015×, 65× and 0.005× those of the benchmark scheme SR-1, respectively. Similar trends may be observed for the hardware implementation results of the 200-bit STDs, as shown in Figure 29b. Here, the proposed RLSTD exhibits a latency that is just 0.013× the latency of SR-1, which increases the throughput by 78× and reduces the energy consumption by a factor of 0.005×.

XI. DESIGN GUIDELINES AND CONCLUSIONS

In this paper, we have presented two different sets of modifications to the state-of-the-art STD of [36], which significantly improve its timing error tolerance and processing latency. This has been achieved by considering the close relationship between the different trade-offs involved in the hardware implementation of the STD, as listed in Figure 2. To elaborate further, the implementation of iterative decoders is typically oriented towards the optimization of a particular design objective. For example, a particular design may focus on achieving a low chip area, to the detriment of all other design objectives. More specifically, only the design constraints and parameters that affect the overall chip area of the design may be considered and optimized during the design process. However, this approach fails to consider other characteristics of the hardware implementation, such as its energy efficiency, latency, throughput, error correction capability or timing error tolerance. Hence, the resultant implementation may not achieve the desired hardware specifications or BER performance. Motivated by this trend, we conceived the design approach of Figure 30 for improving the tolerance of STDs to timing errors. In this case, the design guidelines of Figure 30 may be interpreted as follows.

1) As described in Section V and observed in Figure 20, power supply variations was identified as the most detrimental cause of both timing errors and metastability in the STD. Hence we have to conceive measures to mitigate the catastrophic propagation of metastabiliy through the decoder.
2) The employment of synchronizers is recommended as the error-tolerant design technique for mitigating the catastrophic propagation of metastability, as detailed in Section V-A.

3a) However, as portrayed in Table 1 and Figure 17, the employment of synchronizers increased the chip area, the latency as well as the energy consumption and reduced the throughput.

4a) Therefore, an additional set of EMs may be advocated for enabling the simultaneous decoding of two received frames for the sake of improving the throughput of the STD, as described in Section V-B.

3b) However, observe in Table 1 and Figure 17 in Section VI that the additional set of EMs increased the chip area, the latency and the energy consumption of the design.

4b) Therefore clock gating is proposed in Section V-B for reducing the chip area, latency and energy consumption of the STD.

4c) Additionally, in Section V-C, TFM-based EMs were proposed for reducing the chip area, the latency as well as the energy consumption of the design and for improving both the throughput as well as the BER performance, as portrayed in Table 1 and Figure 17.

4d) Finally, in Section V-D, pipelining was recommended for further improving the BER performance of the STD in the presence of timing errors, as observed in Figures 21 and 22.

Building on this, we have characterized the trade-offs among the chip area, energy efficiency, latency, throughput and error correction capabilities of different timing-error tolerant STDs, when they operate at two different nominal supply voltages. Our simulations in Figures 21 and 22 that the proposed STD (TFM-5, 1.20 V, 2.2 ns, 7%) offers the same BER performance as the state-of-the-art STD (SR-1, 1.20 V, 4.0 ns, 0%) design of [36], despite suffering from power supply variations, while increasing the throughput by a factor of 2.42, reducing the latency by a factor of 0.83 and consuming only 0.25× the energy of that of (SR-1, 1.20 V, 4.0 ns), without increasing the chip area. Furthermore, this trade-off analysis technique may be applied to the design of other timing-error-tolerant iterative decoder implementations in order to determine the most desirable configuration.

We have also proposed modifications to the state-of-the-art STD that significantly reduce the average number of DCs required for successfully decoding a frame. This has been achieved with the aid of OR gates for the approximate stochastic addition of BSs, as well as a reduced-complexity TFM and a single output DFF involved for the estimation of the decoded bit, as described in Section VIII. As a result of these modifications, the proposed RLSTD improves the latency, throughput and energy efficiency of the state-of-the-art STD by an order of magnitude, without imposing an area extension and without degrading the error correction capabilities of the STD, as shown in Figures 27 and 28. Our simulations in Figure 29 show that the proposed 50-bit RLSTD exhibits an improved BER, reduces the number of DCs by a factor of 0.035×, the latency by a factor of 0.015× and the energy consumption by a factor of 0.005×, while increasing the throughput 65× and employing only 0.23× the chip area of the state-of-the-art SR-1 [36]. Similar trends were found for the proposed 200-bit RLSTD, which offers a throughput that is 78× the throughput of SR-1. Based on the results presented in Section X, we conclude that the proposed RLSTD is particularly suited for short-frame-length, low-latency communication systems, such as those required in next-generation MCMTCs.

Finally, our future work will combine the proposed techniques for improving both the timing-error tolerance and the processing latency of the STD. More specifically, we will employ the design flow of Figure 30 for enhancing the robustness of the proposed RLSTD, by determining the causes and effects of timing errors and by applying as well as critically appraising the corresponding error-tolerant design techniques.

REFERENCES

[1] R. G. Gallager, "Low-density parity-check codes," *IRE Trans. Inf. Theory*, vol. 8, no. 1, pp. 21–28, Jan. 1962.

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

[3] *IEEE Standard for Wireless Metropolitan Area Networks*, IEEE Standard 802.16, 2012.

[4] *IEEE Standard for Wireless Local Area Networks*, IEEE Standard 802.11, 2012.

[5] *DVB-S2 Standard*, ETSI EN Standard 302 307, 2009.

[6] *CDMA Standard*, Standard 3GPP2, 2002.

[7] *UMTS Standard*, ETSI TS Standard 125 331, 2011.

[8] *LTE Standard*, 3GPP TS Standard 36.212, 2011.

[9] A. Darabiha, A. C. Carusone, and F. R. Kschischang, "A 3.3-Gbps bit-serial block-interlaced min-sum LDPC decoder in 0.13-μm CMOS," in *Proc. IEEE Custom Integr. Circuits Conf. (CICC)*, San Jose, CA, USA, Sep. 2007, pp. 459–462.

[10] A. Darabiha, A. C. Carusone, and F. R. Kschischang, "Block-interlaced LDPC decoders with reduced interconnect complexity," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 55, no. 1, pp. 74–78, Jan. 2008.

[11] E. Yeo and V. Anantharam, "Iterative decoder architectures," *IEEE Commun. Mag.*, vol. 41, no. 8, pp. 132–140, Aug. 2003.

[12] A. E. Pusane, A. J. Felstrom, A. Sridharan, M. Lentmaier, K. S. Zigangirov, and D. J. Costello, Jr., "Implementation aspects of LDPC convolutional codes," *IEEE Trans. Commun.*, vol. 56, no. 7, pp. 1060–1069, Jul. 2008.

[13] B. R. Gaines, "Stochastic computing systems," in *Advances in Information Systems Science*. New York, NY, USA: Plenum, 1969, ch. 2, pp. 37–172.

[14] R. Ahmadi and F. N. Najm, "Timing analysis in presence of power supply and ground voltage variations," in *Proc. Int. Conf. Comput.-Aided Design (ICCAD)*, San Jose, CA, USA, Nov. 2003, pp. 176–183.

[15] D. Ernst *et al.*, "Razor: A low-power pipeline based on circuit-level timing speculation," in *Proc. 36th Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, San Diego, CA, USA, Dec. 2003, pp. 7–18.

[16] N. Ahmed, M. Tehranipoor, and V. Jayaram, "A novel framework for faster-than-at-speed delay test considering IR-drop effects," in *Proc. Int. Conf. Comput.-Aided Design (ICCAD)*, San Jose, CA, USA, Nov. 2006, pp. 198–203.

[17] M. S. Gupta, J. L. Oatley, R. Joseph, G.-Y. Wei, and D. M. Brooks, "Understanding voltage variations in chip multiprocessors using a distributed power-delivery network," in *Proc. Design, Autom., Test Eur. Conf. Exhibit.*, Apr. 2007, pp. 1–6.

[18] M. Alioto, G. Palumbo, and M. Pennisi, "Understanding the effect of process variations on the delay of static and domino logic," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 5, pp. 697–710, May 2010.

[19] O. N. C. Yilmaz, Y.-P. E. Wang, N. A. Johansson, N. Brahmi, S. A. Ashraf, and J. Sachs, "Analysis of ultra-reliable and low-latency 5G communication for a factory automation use case," in *Proc. IEEE Int. Conf. Commun. Workshop (ICCW)*, London, U.K., Jun. 2015, pp. 1190–1195.

[20] V. C. Gaudet and A. C. Rapley, "Iterative decoding using stochastic computation," *IET Electron. Lett.*, vol. 39, no. 3, pp. 299–301, Apr. 2003.

[21] A. C. Rapley, C. Winstead, V. C. Gaudet, and C. Schlegel, "Stochastic iterative decoding on factor graphs," in *Proc. 3rd Int. Symp. Turbo Codes Rel. Topics*, Brest, France, Sep. 2003, pp. 507–510.

[22] C. Winstead, V. C. Gaudet, A. Rapley, and C. Schlegel, "Stochastic iterative decoders," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Adelaide, SA, Australia, Sep. 2005, pp. 1116–1120.

[23] W. J. Gross, V. C. Gaudet, and A. Milner, "Stochastic implementation of LDPC Decoders," in *Proc. Conf. Rec. 39th Asilomar Conf. Signals, Syst. Comput.*, Pacific Grove, CA, USA, Oct./Nov. 2005, pp. 713–717.

[24] S. S. Tehrani, W. J. Gross, and S. Mannor, "Stochastic decoding of LDPC codes," *IEEE Commun. Lett.*, vol. 10, no. 10, pp. 716–718, Oct. 2006.

[25] S. S. Tehrani, S. Mannor, and W. J. Gross, "An area-efficient FPGA-based architecture for fully-parallel stochastic LDPC decoding," in *Proc. IEEE Workshop Signal Process. Syst. (SiPS)*, Oct. 2007, pp. 255–260.

[26] S. S. Tehrani, S. Mannor, and W. J. Gross, "Fully parallel stochastic LDPC decoders," *IEEE Trans. Signal Process.*, vol. 56, no. 11, pp. 5692–5703, Nov. 2008.

[27] S. S. Tehrani, A. Naderi, G.-A. Kamendje, S. Mannor, and W. J. Gross, "Tracking forecast memories in stochastic decoders," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Taipei, Taiwan, Apr. 2009, pp. 561–564.

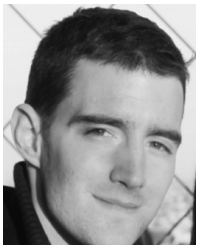
[28] F. Leduc-Primeau, S. Hemati, W. J. Gross, and S. Mannor, "A relaxed half-stochastic iterative decoder for LDPC codes," in *Proc. IEEE Global Telecommun. Conf. (GLOBECOM)*, Honolulu, HI, USA, Nov./Dec. 2009, pp. 1–6.

[29] G. Sarkis and W. J. Gross, "Reduced-latency stochastic decoding of LDPC codes over GF(q)," in *Proc. Eur. Wireless Conf. (EW)*, Lucca, Italy, Apr. 2010, pp. 994–998.

[30] S. S. Tehrani, A. Naderi, G.-A. Kamendje, S. Hemati, S. Mannor, and W. J. Gross, "Majority-based tracking forecast memories for stochastic LDPC decoding," *IEEE Trans. Signal Process.*, vol. 58, no. 9, pp. 4883–4896, Sep. 2010.
[31] A. Naderi, S. Mannor, M. Sawan, and W. J. Gross, "Delayed stochastic decoding of LDPC codes," *IEEE Trans. Signal Process.*, vol. 59, no. 11, pp. 5617–5626, Nov. 2011.
[32] C. Ceroici and V. C. Gaudet, "FPGA implementation of a clockless stochastic LDPC decoder," in *Proc. IEEE Workshop Signal Process. Syst. (SiPS)*, Belfast, Northern Ireland, Oct. 2014, pp. 1–5.
[33] S. S. Tehrani, C. Jego, Z. Bo, and W. J. Gross, "Stochastic decoding of linear block codes with high-density parity-check matrices," *IEEE Trans. Signal Process.*, vol. 56, no. 11, pp. 5733–5739, Nov. 2008.
[34] M. Arzel, C. Lahuec, C. Jego, W. J. Gross, and Y. Bruned, "Stochastic multiple stream decoding of cortex codes," *IEEE Trans. Signal Process.*, vol. 59, no. 7, pp. 3486–3491, Jul. 2011.
[35] T.-H. Chen and J. P. Hayes, "Design of stochastic Viterbi decoders for convolutional codes," in *Proc. IEEE 16th Int. Symp. Design Diagnostics Electron. Circuits Syst. (DDECS)*, Karlovy Vary, Czech Republic, Apr. 2013, pp. 66–71.
[36] Q. T. Dong, M. Arzel, C. Jego, and W. J. Gross, "Stochastic decoding of turbo codes," *IEEE Trans. Signal Process.*, vol. 58, no. 12, pp. 6421–6425, Dec. 2010.
[37] Q. T. Dong, M. Arzel, and C. Jego, "Design and FPGA implementation of stochastic turbo decoder," in *Proc. IEEE 9th Int. New Circuits Syst. Conf. (NEWCAS)*, Bordeaux, France, Jun. 2011, pp. 21–24.
[38] J. Hu, Y. Deng, J. Chen, and X. Ling, "High speed turbo decoder design based on stochastic computation," in *Proc. Int. Conf. Commun. Circuits Syst. (ICCCAS)*, vol. 1. Chengdu, China, Nov. 2013, pp. 235–239.
[39] J. Chen and J. Hu, "High throughput stochastic log-MAP turbo-decoder based on low bits computation," *IEEE Trans. Signal Process.*, vol. 20, no. 11, pp. 1098–1101, Nov. 2013.
[40] M. Alles, T. Brack, and N. Wehn, "A reliability-aware LDPC code decoding algorithm," in *Proc. IEEE 65th Veh. Technol. Conf. (VTC-Spring)*, Dublin, Ireland, Apr. 2007, pp. 1544–1548.
[41] V. C. Gaudet, "Low-power LDPC decoding by exploiting the fault-tolerance of the sum-product algorithm," in *Contemporary Mathematics*. Providence, RI, USA: AMS, 2010, pp. 165–171.
[42] E. P. Kim and N. R. Shanbhag, "Energy-efficient LDPC decoders based on error-resiliency," in *Proc. IEEE Workshop Signal Process. Syst. (SiPS)*, Oct. 2012, pp. 149–154.
[43] Y. Tang, C. Winstead, E. Boutillon, C. Jego, and M. Jezequel, "An LDPC decoding method for fault-tolerant digital logic," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Seoul, Korea, May 2012, pp. 3025–3028.
[44] C. Winstead, Y. Tang, E. Boutillon, C. Jego, and M. Jezequel, "A space-time redundancy technique for embedded stochastic error correction," in *Proc. 7th Int. Symp. Turbo Codes Iterative Inf. Process. (ISTC)*, Gothenburg, Sweden, Aug. 2012, pp. 36–40.
[45] J. Geldmacher and J. Gotze, "EXIT-optimized index assignments for turbo decoders with unreliable LLR transfer," *IEEE Commun. Lett.*, vol. 17, no. 5, pp. 992–995, May 2013.
[46] C. L. K. Ngassa, V. Savin, and D. Declercq, "Faulty stochastic LDPC decoders over the binary symmetric channel," in *Proc. Int. Symp. Turbo Codes Iterative Inf. Process. (ISTC)*, Bremen, Germany, Aug. 2014, pp. 112–116.
[47] J. Andrade *et al.*, "On the performance of LDPC and turbo decoder architectures with unreliable memories," in *Proc. 48th Asilomar Conf. Signals, Syst. Comput.*, Pacific Grove, CA, USA, Nov. 2014, pp. 542–547.
[48] C.-H. Huang, Y. Li, and L. Dolecek, "Belief propagation algorithms on noisy hardware," *IEEE Trans. Commun.*, vol. 63, no. 1, pp. 11–24, Jan. 2015.
[49] M. May, M. Alles, and N. Wehn, "A case study in reliability-aware design: A resilient LDPC code decoder," in *Proc. Design, Autom. Test Eur. Conf. Exhibit. (DATE)*, Munich, Germany, Mar. 2008, pp. 456–461.
[50] C. Winstead and S. Howard, "A probabilistic LDPC-coded fault compensation technique for reliable nanoscale computing," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 56, no. 6, pp. 484–488, Jun. 2009.
[51] R. A. Abdallah and N. R. Shanbhag, "Error-resilient low-power Viterbi decoder architectures," *IEEE Trans. Signal Process.*, vol. 57, no. 12, pp. 4906–4917, Dec. 2009.
[52] I. Perez-Andrade, X. Zuo, R. G. Maunder, B. M. Al-Hashimi, and L. Hanzo, "Analysis of voltage- and clock-scaling-induced timing errors in stochastic LDPC decoders," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Shanghai, China, Apr. 2013, pp. 4293–4298.
[53] B. Sedighi, N. P. Anthapadmanabhan, and D. Suvakovic, "Timing errors in LDPC decoding computations with overscaled supply voltage," in *Proc. Int. Symp. Low Power Electron. Design (ISLPED)*, 2014, pp. 201–206.
[54] I. Levi and A. Fish, "Dual mode logic—Design for energy efficiency and high performance," *IEEE Access*, vol. 1, pp. 258–265, May 2013.
[55] J. B. Anderson and S. M. Hladik, "Tailbiting MAP decoders," *IEEE J. Sel. Areas Commun.*, vol. 16, no. 2, pp. 297–302, Feb. 1998.
[56] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate (Corresp.)," *IEEE Trans. Inf. Theory*, vol. 20, no. 2, pp. 284–287, Mar. 1974.
[57] B. D. Brown and H. C. Card, "Stochastic neural computation. I. Computational elements," *IEEE Trans. Comput.*, vol. 50, no. 9, pp. 891–905, Sep. 2001.
[58] A. Alaghi and J. P. Hayes, "Survey of stochastic computing," *ACM Trans. Embedded Comput. Syst.*, vol. 12, no. 2s, May 2013, Art. ID 92.
[59] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 498–519, Feb. 2001.
[60] Q. T. Dong, "Le principe de calcul stochastique appliqué au décodage des turbocodes: Conception, implémentation et prototypage sur circuit FPGA," Ph.D. dissertation, Dept. Electron., Télécom Bretagne, Brest, France, Dec. 2011.
[61] R. Ginosar, "Metastability and synchronizers: A tutorial," *IEEE Des. Test Comput.*, vol. 28, no. 5, pp. 23–35, Sep. 2011.
[62] S. Beer, J. Cox, R. Ginosar, T. Chaney, and D. M. Zar, "Variability in multistage synchronizers," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 12, pp. 2957–2969, Dec. 2015.
[63] G. Albertengo and R. Sisto, "Parallel CRC generation," *IEEE Micro*, vol. 10, no. 5, pp. 63–71, Oct. 1990.
[64] Y. Huo, X. Li, W. Wang, and D. Liu, "High performance table-based architecture for parallel CRC calculation," in *Proc. IEEE Int. Workshop Local Metropolitan Area Netw. (LANMAN)*, Apr. 2015, pp. 1–6.
[65] P. N. Whatmough, S. Das, D. M. Bull, and I. Darwazeh, "Circuit-level timing error tolerance for low-power DSP filters and transforms," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 21, no. 6, pp. 989–999, Jun. 2013.
[66] S. Purohit and M. Margala, "Investigating the impact of logic and circuit implementation on full adder performance," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 7, pp. 1327–1331, Jul. 2012.
[67] C. L. Janer, J. M. Quero, J. G. Ortega, and L. G. Franquelo, "Fully parallel stochastic computation architecture," *IEEE Trans. Signal Process.*, vol. 44, no. 8, pp. 2110–2117, Aug. 1996.

ISAAC PEREZ-ANDRADE received the B.Sc. degree in electronics and computer science from ITESM, Mexico, in 2009, and the M.Sc. degree in system-on-chip from the University of Southampton, U.K., in 2011, where he is currently pursuing the Ph.D. degree. His research interests include low-power error-tolerant VLSI architectures for wireless communications and iterative decoding.

SHIDA ZHONG received the B.Sc. degree in electronics engineering from Shenzhen University, Shenzhen, China, in 2008, and the M.Sc. and Ph.D. degrees in electronics and electrical engineering from the University of Southampton, U.K., in 2009 and 2013, respectively. He is currently a Research Fellow with the School of Electronics and Computer Science, University of Southampton. His research interests include design for testing, fault modeling, FPGA and ASIC implementations for high-throughput, and low-power and error tolerant decoders.



ROBERT G. MAUNDER (S'03–M'08–SM'12) received the B.Eng. degree in electronics engineering and the Ph.D. degree in wireless communications from the Department of Electronics and Computer Science, University of Southampton, U.K., in 2003 and 2007, respectively. He became a Lecturer in 2007, and an Associate Professor in 2013. His research interests include joint source/channel coding, iterative decoding, irregular coding, and modulation techniques.

BASHIR M. AL-HASHIMI (M'99–SM'01–F'09) is currently a Professor of Computer Engineering and the Dean of the Faculty of Physical Sciences and Engineering with the University of Southampton, U.K. He is an ARM Professor of Computer Engineering and the Co-Director of the ARM-ECS Research Centre. He has authored over 300 technical papers, authored or co-authored five books, and has graduated 31 Ph.D. students. His research interests include methods, algorithms, and design automation tools for energy efficient of embedded computing systems.

LAJOS HANZO (F'04) received the degree in electronics in 1976, the Ph.D. degree in 1983, the Doctor Honoris Causa degree from the Technical University of Budapest, in 2009, and the D.Sc. degree. During his 38-year career in telecommunications, he has held various research and academic positions in Hungary, Germany, and U.K. Since 1986, he has been with the School of Electronics and Computer Science, University of Southampton, U.K., as the Chair in Telecommunications. He has 23 000+ citations. He has successfully supervised 100 Ph.D. students, co-authored 20 *Mobile Radio Communications* (John Wiley/IEEE Press) books totaling in excess of 10 000 pages, authored over 1500 research entries at the IEEE Xplore, acted as the TPC Chair and General Chair of the IEEE conferences, presented keynote lectures, and received a number of distinctions. He is directing 100 strong academic research teams, working on a range of research projects in the field of wireless multimedia communications sponsored by the industry, the Engineering and Physical Sciences Research Council, U.K., the European Research Council's Advanced Fellow Grant, and the Royal Society's Wolfson Research Merit Award. He is an enthusiastic supporter of industrial and academic liaison. He offers a range of industrial courses. He is a fellow of the Royal Academy of Engineering, the Institution of Engineering and Technology, and the European Association for Signal Processing. He is also a Governor of the IEEE VTS. From 2008 to 2012, he was the Editor-in-Chief of the *IEEE Press* and a Chaired Professor with Tsinghua University, Beijing. He has 22 000+ citations.