Improving the Tolerance of Stochastic LDPC Decoders to Overclocking-Induced Timing Errors: A Tutorial and a Design Example

Xin Zuo, Isaac Perez-Andrade, Robert G. Maunder, Bashir M. Al-Hashimi and Lajos Hanzo
School of ECS, University of Southampton, SO17 1BJ, UK.
Email: {xz3g08, ip1g10, rm, bmah, lh}@ecs.soton.ac.uk, http://www-mobile.ecs.soton.ac.uk

***Abstract*—Channel codes such as Low-Density Parity-Check (LDPC) codes may be employed in wireless communication schemes for correcting *transmission errors*. This tolerance to channel-induced transmission errors allows the communication schemes to achieve higher transmission throughputs, at the cost of requiring additional processing for performing LDPC decoding. However, this LDPC decoding operation is associated with a potentially inadequate processing throughput, which may constrain the attainable transmission throughput. In order to increase the processing throughput, the clock period may be reduced, albeit this is at the cost of potentially introducing *timing errors*. Previous research efforts have considered a paucity of solutions for mitigating the occurrence of timing errors in channel decoders, by employing additional circuitry for detecting and correcting these overclocking-induced timing errors. Against this background, in this paper we demonstrate that stochastic LDPC decoders (LDPC-SDs) are capable of exploiting their *inherent* error correction capability for correcting not only transmission errors, but also timing errors, even without the requirement for additional circuitry. Motivated by this, we provide the first comprehensive tutorial on LDPC-SDs. We also propose a novel design flow for timing-error-tolerant LDPC decoders. We use this to develop a timing error model for LDPC-SDs and investigate how their overall error correction performance is affected by overclocking. Drawing upon our findings, we propose a modified LDPC-SD, having an improved timing error tolerance. In a particular practical scenario, this modification eliminates the approximately 1 dB performance degradation that is suffered by an overclocked LDPC-SD without our modification, enabling the processing throughput to be increased by up to 69.4%, which is achieved without compromising the error correction capability or processing energy consumption of the LDPC-SD.**

I. Introduction

Low-Density Parity-Check (LDPC) codes [1] are employed in wireless communication systems such as IEEE 802.11n/ac (WiFi) and IEEE 802.16e (WiMAX) [2] for correcting *transmission errors*, which are induced by channel effects such as noise, fading, interference and dispersion. This error correction capability facilitates reliable communication using higher transmission throughputs. However, this is achieved at the cost of requiring additional processing for performing LDPC decoding. This LDPC decoding operation is associated with a processing throughput, which potentially limits the transmission throughput, when employed for real-time communications. In order to increase the processing throughput of Application-Specific Integrated Circuit (ASIC) LDPC decoders [3], [4], the clock period of the hardware can be reduced, so that each LDPC decoding operation can be completed using less time. However, overclocking has the potential side effect of introducing *timing errors*, which occur whenever a signal is not yet ready to be clocked into a memory, by the end of the reduced clock period. As fabrication technology is developing to the sub-65nm regime, it is no longer possible to guarantee the reliability of nanoelectronic ASICs, due to process variation and other effects [5]. Error-tolerant electronic design is becoming a critical and essential enabling technology, having already been adopted in numerous applications [6]–[13].

The LDPC decoding algorithm employs an iterative exchange of bit value probabilities between Check Nodes (CNs) and Variable Nodes (VNs) [1], [14]–[16]. Conventionally, these probabilities are represented using the fixed-point twos-complement number representation, while the VNs and CNs perform corresponding fixed-point arithmetic operations [14], [17]–[21]. Previous efforts [8]–[10], [13], [22]–[25] have shown that fixed-point LDPC decoders (LDPC-FD) have an inherent, but limited, tolerance to timing errors. However, when timing errors affect the Most Significant Bits (MSBs) of the fixed-point probabilities, the error correction performance is significantly degraded [8]–[10]. Therefore, the designs of [9], [10] employed additional circuitry for detecting and/or correcting timing errors affecting the MSBs, although this imposes an additional overhead, which can limit the attainable processing throughput.

Against this background, this paper expands on and offers a thorough treatment of our previous work [13] on the first LDPC decoder that can employ its inherent error correction capability to provide tolerance to both channel-induced transmission errors and to overclocking-induced timing errors, which is achieved without requiring additional circuitry. Rather than employing LDPC-FDs, we consider *Stochastic* LDPC Decoders (LDPC-SDs) [3], [26]–[28]. As this describes in the first comprehensive tutorial on LDPC-SDs, the bit value probabilities exchanged between CNs and VNs in LDPC-SDs are

The authors wish to gratefully acknowledge the financial support of the ERSPC, Swindon UK under the auspices of grant EP/J015520/1 and EP/L010550/1, as well as of the TSB, Swindon UK under the auspices of grant TS/L009390/1 and of the European Research Council's Advanced Gellow Grant Beam-Me-Up. This work has been submitted to the IEEE for possible publication. Copyright may be transferred without notice, after which this version may no longer be accessible. The research data for this paper is available at http://dx.doi.org/10.5258/SOTON/388828.

represented by the fraction of bits having the value of 1 within but streams that are referred to as Bernoulli sequences [29]. All bits in these Bernoulli sequences have an equal and relatively low significance, granting them an inherent tolerance to timing errors [30]. The tolerance of LDPC-SDs to timing errors has been investigated in our previous work [13]. However in [13], we assumed that the supply voltage and hence the circuit propagation delays do not vary between clock cycles. By contrast, this paper considers the effect of supply voltage noise, which is a primary cause of timing errors when employing overclocking in practice. In particular, supply voltage noise models the effects of IR drop[1], inductive noise, cross talk, electrostatic discharges, particle strikes, switching noise and fabrication process variations [5], [11], [31]–[35]. Therefore, this paper offers a significantly more comprehensive study than was provided in our previous work of [13]. In particular, we detail a novel design flow, which may used for the systematic design of timing-error-tolerant LDPC decoders.

Fig. 1 illustrates the structure of this paper. We commence by explaining the motivation and relevant background in Section II. In particular, Section II provides this first comprehensive tutorial on the LDPC-SDs. Following this, Section III describes our novel design flow for timing error-tolerant LDPC decoders. Section IV models the variation in supply voltage and hence in circuit propagation delay between clock cycles in the LDPC-SD. We also derive a model of the causes and effects of different types of timing errors that are imposed upon the LDPC-SD in Section IV. We use this model to investigate the error-correcting performance of the LDPC-SD in the presence of each type of timing error separately. In this way, we characterise the most detrimental types of timing errors and use this to motivate the design of a novel modified LDPC-SD, as discussed in Section V. This modification redefines the functionality of the VNs, so that they can be implemented using a circuit schematic that has a significantly improved tolerance to timing errors. The error model is employed in the Bit Error Ratio (BER) analysis of Section VI, for characterizing the tolerance to both transmission errors and timing errors of the LDPC-SD and the modified LDPC-SD. We show that our modification does not compromise the error correction capability of the LDPC-SD in the absence of timing errors. We demonstrate that, in the case of the conventional LDPC-SD, the BER performance is not degraded by applying moderate overclocking and that even for aggressive overclocking, only a 1 dB performance degradation is incurred. Furthermore, we demonstrate that, despite requiring no extra circuitries, our modified LDPC-SD eliminates this 1 dB performance degradation that is incurred by the LDPC-SD when employing aggressive overclocking. This significantly improved tolerance to timing errors allows the processing throughput to be increased by up to 69.4% in practice, without compromising either the error correction capability or processing energy consumption of the LDPC-SD. Finally, we offer our conclusions in Section VII.

[1]The resistance of the power distribution network in integrated circuit systems may cause a drop in the power supply voltage, commonly referred to as the IR drop, where I is the current flowing in the network and R is the resistance.

Fig. 1. The structure of the paper.

II. Background

In this section, we review the challenges encountered in the design of previous timing error-tolerant LDPC decoders, which we will use to motivate our proposed LDPC-SD and its design flow. We begin in Section III by discussing previous work on timing-error-tolerant LDPC decoder design. Section II-B details the operation of LDPC decoders, including the decoding algorithm and its conventional fixed-point implementation. Finally, Section II-C offers the first comprehensive tutorial on the stochastic implementation of LDPC decoders.

A. Literature Review

Several previous research efforts have investigated the tolerance of LDPC decoder ASICs to timing errors and other types of processing errors [8]–[10], [13], [22]–[25]. In [8], it was demonstrated that the inherent error-correcting capability of LDPC-FDs may also grant them tolerance to timing errors, provided that they do not affect the MSBs of the fixed-point numbers used to represent bit probabilities. Analytical investigations of the error-correcting capability of LDPC decoders in the presence of processing errors were provided in [22]–[25]. The designs of [9], [10] employed additional circuitry for detecting and/or correcting timing errors affecting the MSBs of the fixed-point numbers, although this approach is associated with an additional processing overhead, which can limit the attainable processing throughput. Recently, LDPC decoders that are implemented using stochastic processing have attracted significant research interest [3], [27], [28], [36]–[47], owing to their low hardware complexity. Compared to the traditional implementations using fixed-point numbers, stochastic LDPC decoders are suited to timing-error-tolerant design, since all bits in the stochastic number representation have an equal and relatively-low significance. Owing to this, a timing error affecting any single bit in a stochastic number representation has only a small effect on the represented bit probabilities, which can be readily tolerated by the inherent

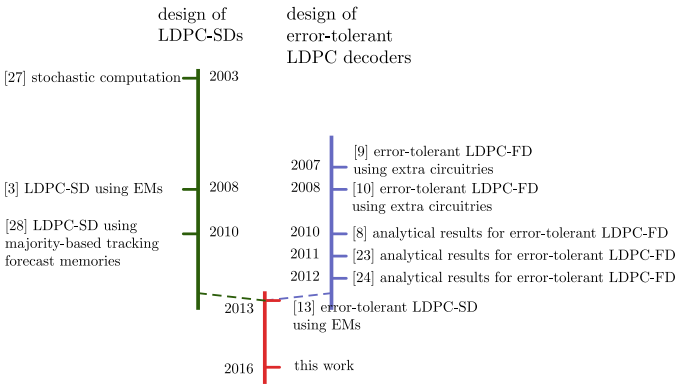


Fig. 2. Timelines of relevant publications.

error correction capability of the LDPC decoder. We investigated this tolerance to timing errors for the first time in [13], which modeled the timing errors by extracting gate-level timing characteristics from an 90 nm ASIC design, and inserting errors in a probabilistic manner. In this paper, we expand on our previous work of [13] by providing a comprehensive investigation into the causes and effects of timing errors on a stochastic LDPC decoder. We propose a number of enhancements to the stochastic LDPC decoder, in order to improve its tolerance to timing errors and we expand the analysis to consider supply voltage noise, as discussed in Section I. Furthermore, we propose a novel design flow for timing-error-tolerant LDPC decoders.

Fig. 2 lists the most relevant previous publications along two main timelines, namely the timeline of timing-error-tolerant LDPC decoder design and that of the stochastic implementation of LDPC decoders. Each timeline is represented by a vertical line with the downward direction representing the chronological order, where each knot on the vertical lines represent a publication discussed above.

B. LDPC Decoding Algorithm and Fully Parallel Implementation

In this section, we commence by describing the factor graph representation of LDPC codes, followed by introducing the fully-parallel scheduling of traditional LDPC-FDs. After this, we describe algorithm and structure used by each node within the factor graph.

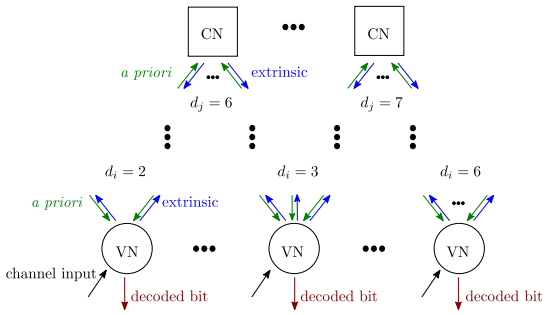


Fig. 3. Factor graph of a LDPC code.

1) Factor Graph and Fully Parallel Scheduling: The parametrization of an LDPC decoder is completely specified by its factor graph, which has a design that depends upon that of the corresponding LDPC encoder [15], [48], [49]. Fig. 3 illustrates the factor graph [15], [16], [50] of an LDPC code that uses n encoded bits to represent k information bits, where $n > k$. The factor graph is comprised of n VNs and $(n-k)$ CNs. The i^{th} VN has $(d_i + 1)$ ports, one of which is connected to the channel and the remaining d_i ports are connected to different CNs by edges within the factor graph, where d_i is referred to as the degree of the node. Similarly, the j^{th} CN has a degree of d_j and is connected to d_j different VNs by ports connected to edges within the factor graph, as shown in Fig. 3. Before LDPC decoding begins, the channel uses the corresponding ports of the VNs to provide an initial probability for each of the n LDPC-encoded bits adopting the binary value of 1. During LDPC decoding, the VNs and CNs work together to refine the probabilities associated with the n LDPC-encoded bits, which are then used to decide the values of the n decoded bits, as shown in Fig. 3. More specifically, the VNs and CNs iteratively exchange these probabilities along the edges of the factor graph. In a fully-parallel LDPC decoder, the VNs are activated in all odd clock cycles, while the CNs are activated in all even clock cycles. This may be achieved using register enable signals or clock gating, for example. This process continues until a fixed number of clock cycles is reached or until an early-stopping criterion is satisfied, such as having successfully recovered all n LDPC-encoded bits [16]. Following this, the k message bits may be extracted from the n decoded bits and output [16].

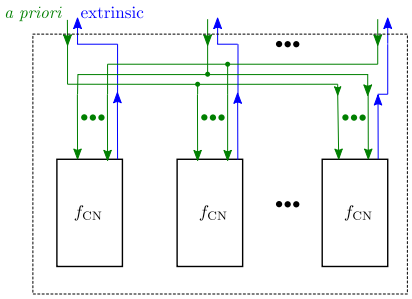


Fig. 4. Structure of a CN.

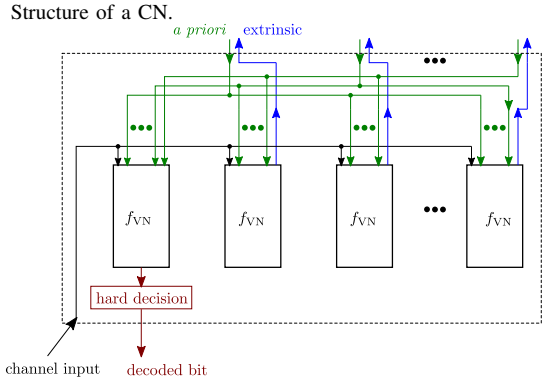


Fig. 5. Structure of a VN.

2) Structure of CNs and VNs: Whenever a VN or CN is activated, a bit probability is output on each of its ports. The probabilities that are output on a particular port of a VN or CN are calculated as a function of the probabilities that are

input to all of the other ports of that node, via the edges of the factor graph. The only exception to this is the probability that is output on the port of the VN connected to the channel, which is calculated as a function of the probabilities that are input to *all* ports of that VN, including the port connected to the channel. In the case of CNs and VNs, an output probability P_C that is obtained by combining two input probabilities P_A and P_B as given by

$$P_C = f_{\mathrm{CN}}(P_A, P_B) = P_A(1-P_B) + P_B(1-P_A), \quad (1)$$

$$P_C = f_{\mathrm{VN}}(P_A, P_B) = \frac{P_A P_B}{P_A P_B + (1-P_A)(1-P_B)}, \quad (2)$$

respectively [15]. These functions may be extended for more than two input probabilities, by recursively substituting (1) or (2) into itself. For example, these input probabilities may be combined according to

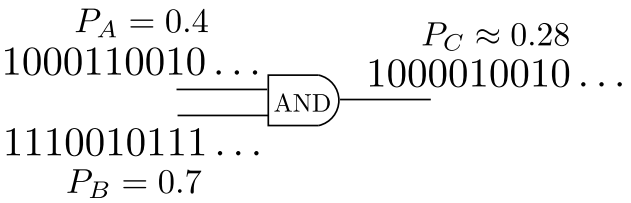
$$\begin{aligned} P_D &= f_{\mathrm{CN}}(P_A, P_B, P_C) \\ &= P_A(1 - f_{\mathrm{CN}}(P_B, P_C)) + f_{\mathrm{CN}}(P_B, P_C) \cdot (1-P_A), \end{aligned} \quad (3)$$

$$\begin{aligned} P_D &= f_{\mathrm{VN}}(P_A, P_B, P_C) \\ &= \frac{P_A \cdot f_{\mathrm{VN}}(P_B, P_C)}{P_A \cdot f_{\mathrm{VN}}(P_B, P_C) + (1-P_A)(1-f_{\mathrm{VN}}(P_B, P_C))} \\ &= \frac{P_A P_B P_C}{P_A P_B P_C + (1-P_A)(1-P_B)(1-P_C)}. \end{aligned} \quad (4)$$

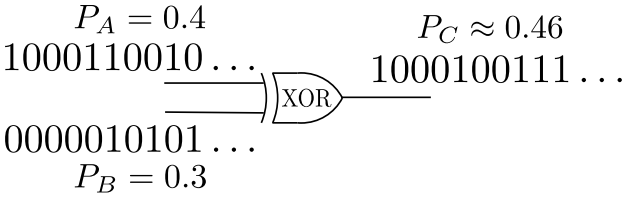
Fig. 4 and 5 depict the structures of an individual CN and VN, respectively. As shown in Fig. 4, the j^{th} CN uses d_j number of computation units to combine the d_j input probabilities, which are referred to as *a priori* probabilities. More specifically, f_{CN} is used to compute the so-called *extrinsic* probabilities. As shown in Fig. 5, the i^{th} VN is provided with a probability from the channel, as well as d_i *a priori* probabilities from the connected CNs. The VN uses the d_i computation units shown in Fig. 5 to compute f_{VN} and output an *extrinsic* probability back to each of the connected CNs via the corresponding edge. Furthermore, the VN uses an additional computation unit to calculate a probability for the channel port, which is then converted into a decoded bit having the binary value of 1, if the probability is greater than 0.5, or to the binary value of 0 otherwise.

C. Stochastic Computation in LDPC Decoders

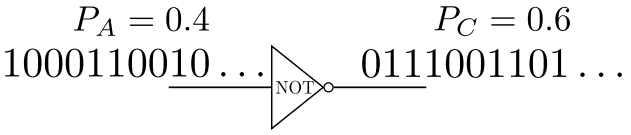
In order to implement f_{CN} and f_{VN} in particular ASIC designs, traditional implementations of LDPC-FDs use fixed-point representations of the probabilities that are iteratively exchanged between the nodes of the factor graph in successive clock cycles. However, the MSBs of these fixed-point representations are sensitive to timing errors, as discussed in Section I. By contrast, stochastic decoders represent the probabilities using only a single bit per clock cycle. Over the course of several successive clock cycles, the individual bits that are exchanged between a particular pair of nodes collectively form a Bernoulli sequence [26], [51]. Here, the exchanged probability is represented by the particular fraction of bits in this Bernoulli sequence that have a binary value of 1. For example, the probability of 0.7 may be expressed by a stochastic bit stream $1011110101\ldots$. However, the same probability of 0.7 may be represented by other streams



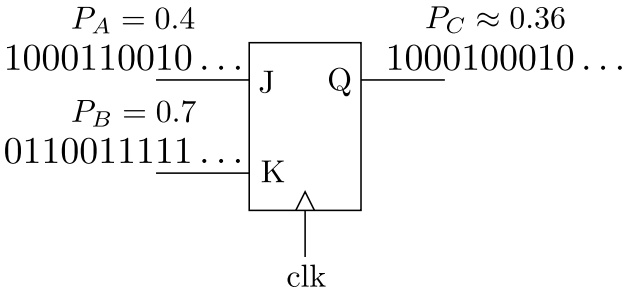
(a) Stochastic implementation for approximating $P_C = P_{A\cap B} = P_A \cdot P_B$.



(b) Stochastic implementation for approximating $P_C = P_{A\oplus B} = P_A(1-P_B) + P_B(1-P_A)$.



(c) Stochastic implementation of $P_C = P_{\bar{A}} = 1 - P_A$.



(d) Stochastic implementation for approximating $P_C = \frac{P_A}{P_A+P_B}$.

Fig. 6. Stochastic implementations of the computation used in LDPC decoding, as well as example Bernoulli sequences and the corresponding output Bernoulli sequences.

having the same fraction of 1s in a different order, such as $000110110111\ldots$. Accordingly, the stochastic implementation of (1) and (2), as well as of the CNs and VNs of Fig. 4 and 5, are explained in the following subsections.

1) Basic Stochastic Computation: In stochastic computation, different mathematical functions of probabilities can be evaluated using different logic gates. In particular, the intersection of two independent probabilities is given by their product $P_C = P_{A\cap B} = P_A \cdot P_B$, which can be implemented by using an AND logic gate, to combine the corresponding Bernoulli sequences, as exemplified in Fig. 6(a). Similarly, the difference between the union and intersection of two independent probabilities is given by $P_{A\oplus B} = P_{A\cup B} - P_{A\cap B} = P_A(1-P_B) + P_B(1-P_A)$, which can be implemented by using an eXclusive-OR (XOR) gate, as exemplified in Fig. 6(b). The complementary probability $P_C = P_{\bar{A}} = 1 - P_A$ can be obtained using a NOT gate to invert the Bernoulli sequence, as exemplified in Fig. 6(c). Finally, the normalized division $P_C = \frac{P_A}{P_A+P_B}$ of two probabilities can be implemented using

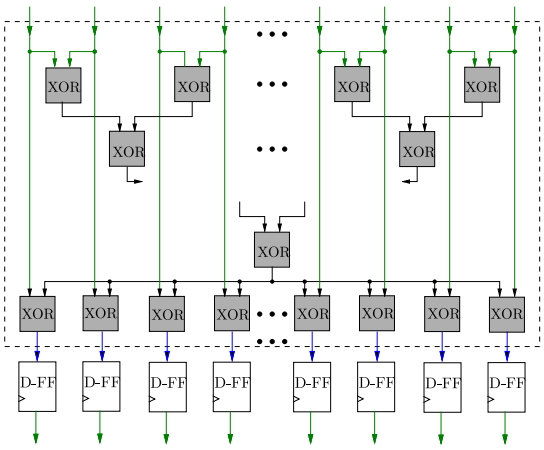


Fig. 7. Structure of a stochastic CN.

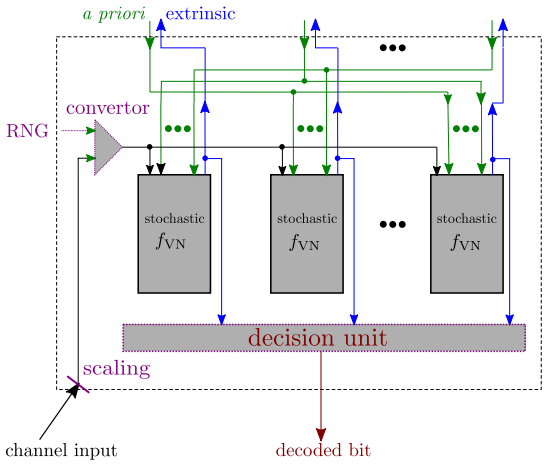


Fig. 8. Structure of a stochastic VN.

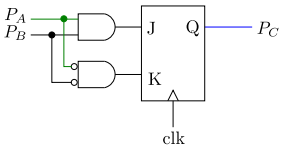
a JK-type Flip-Flop (JK-FF), which operates on the two corresponding Bernoulli sequences, as shown in Fig. 6(d). Note that while the gates of Fig. 6(a), 6(b) and 6(c) produce outputs that depend only on the current inputs, the JK-FF of Fig. 6(d) has memory, which implies that the output depends both on the current inputs and on the state of the memory that has been established based on the previous inputs. More explicitly, the output Q of the JK-FF is updated to the value of the input J, if it disagrees with the value of K. By contrast, if J and K are both equal to 0, then the state of the output Q is preserved. If J and K are both equal to 1, then the state of the output Q is toggled, as shown in the example of Bernoulli sequences of Fig. 6(d). Note that when the Bernoulli sequences are short, the outputs of the gates shown in Fig. 6 will represent a probability that only approximates the correct result. However, the approximations become increasingly accurate, as the length of the Bernoulli sequences are extended.

2) Stochastic Implementation of CNs: CNs in LDPC-SDs can be implemented by substituting the computation units of Fig. 4 by the stochastic implementation of the function f_{CN}. More specifically, in CNs having a degree of $d_j = 3$, the stochastic implementation of (1) is obtained using an XOR gate, as shown in Fig. 6(b). Likewise, the stochastic implementation of f_{CN} having more than two inputs in CNs having a degree of $d_j > 3$ can be implemented by recursively combining XOR gates, in analogy with (3). As shown in Fig. 4, a stochastic CN having a degree of d_j is required to perform parity check operations on all the *a priori* input bits provided by the connected VNs, by using d_j stochastic computation units, each of which combines its $d_j - 1$ inputs using a network of $d_j - 2$ XOR gates. However, the hardware complexity of this structure can be readily reduced by using the arrangement of Fig. 7 [16], [52], [53]. In this high-degree CN, an intermediate parity check result is obtained using a tree structure comprising $d_j - 1$ XOR gates, in order to find the XOR of *all* the input *a priori* bits. This parity check result is then XORed with each of the d_j input bits, in order to obtain the extrinsic bit that is output on each port. This improved structure therefore requires a total of $2d_j - 1$ XOR gates, which is lower than the $d_j(d_j - 2)$ XOR gates required by the structure of Fig. 4 for $d_j > 3$. Note that a D-type Flip-Flop (D-FF) is employed to suffer the extrinsic output bit provided by each port of a CN, as shown in Fig. 4. In this way, the operation of the CNs may be completed using a single clock cycle.

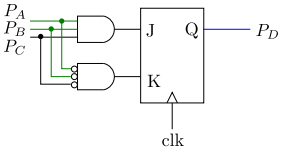
3) Stochastic Implementation of VNs: Fig. 8 illustrates the stochastic implementation of the VN of Fig. 5, where the shaded blocks represent the constituent components, namely (a) the convertor used for providing the computation units with stochastic bits that represent the channel input, (b) the computation units used for the stochastic implementation of the function f_{VN} for converting the *a priori* stochastic bits into the extrinsic stochastic bits and (c) the decision unit used for obtaining the decoded bit. In the following discussions, we will address the implementation of the each shaded blocks, addressing the differences with the corresponding components of VNs in conventional LDPC-FDs.

Channel Input Convertor: As shown in Fig. 8, the input from channel must be converted from a bit probability to a Bernoulli sequence, which generates a different stochastic bit in each clock cycle, which is supplied to each computation block. This convertor can be implemented as a comparator, which outputs a stochastic bit having the value of 1, if the channel input probability exceeds a random number generated with a uniform probability distribution, or outputs 0 otherwise [3]. This output is clocked into an output D-FF.

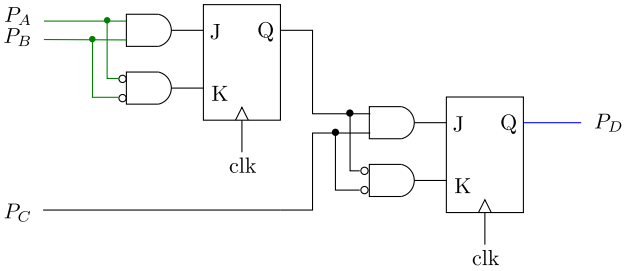
Computational Units: The implementation of the function f_{VN} can be constructed based on the stochastic arithmetic of Section II-C1. More specifically, (2) can be implemented by a JK-FF, where the J and K inputs are provided by Bernoulli sequences that represent the calculation of $P_A \cdot P_B$ and $(1-P_A)\cdot(1-P_B)$, respectively. As discussed previously, $P_A \cdot P_B$ can be obtained using the AND gate of Fig. 6(a), while the second term $(1 - P_A) \cdot (1 - P_B)$ can be obtained by using an AND gate combined with two NOT gates. The complete implementation of (2) is shown in Fig. 9(a), while Table I provides the corresponding truth table. Note that if the two stochastic input bits P_A and P_B have values that agree, then this value is passed to the output stochastic bit P_C, as a so-called regenerative bit. Otherwise, the value output in the previous clock cycle is preserved for the output P_C,



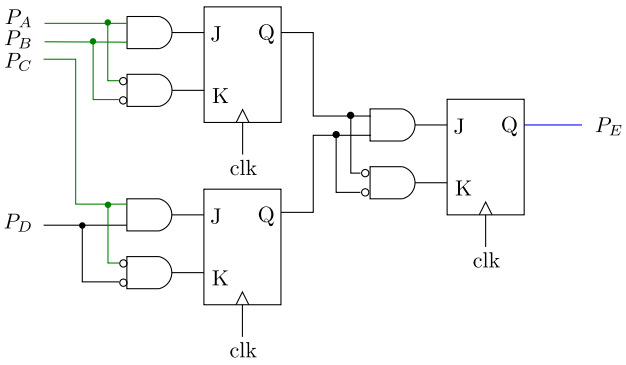
(a) Stochastic implementation of f_{VN} of (2) for VNs having a degree of $d_i = 2$.



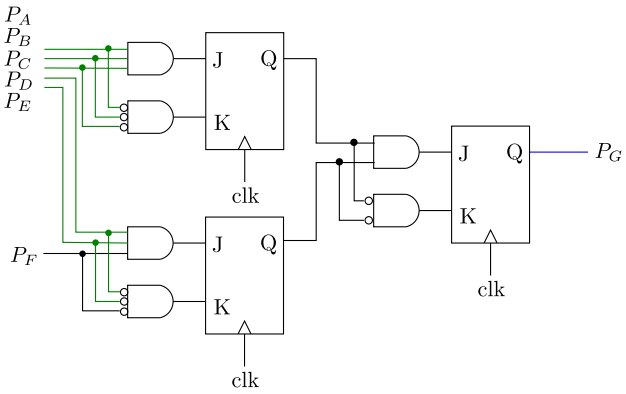
(b) Stochastic implementation of f_{VN} of (4) for VNs having a degree of $d_i = 3$.



(c) An anti-latching stochastic implementation of f_{VN} of (4) for VNs having a degree of $d_i = 3$.



(d) Anti-latching stochastic implementation of f_{VN} for VNs having a degree of $d_i = 4$.



(e) Anti-latching stochastic implementation of f_{VN} for VNs having a degree of $d_i = 6$.

Fig. 9. Stochastic implementations for the function f_{VN} of VNs having degrees of 2, 3, 4 and 6. [26], [28], [36], [54].

as a so-called conservative bit. Note that the JK-FF buffers the extrinsic output bits of a VN, allowing the operation of the VN to be completed in a single clock cycle. For VNs having higher degrees, the additional *a priori* stochastic bits that are input to each computation unit of Fig. 8 can be accommodated by increasing the number of inputs provided to the AND gates. This is shown in Fig. 9(b) for the case of VNs having a degree of $d_i = 3$, which operate on the basis

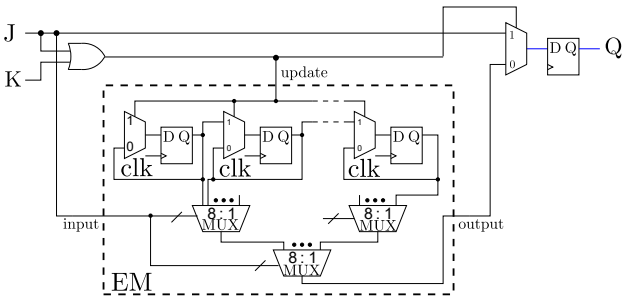


Fig. 10. The example structure of the EM, and the the circuit that allows it to replace the JK-FF in stochastic VNs.

of (4). However, as the number of inputs to an AND gate is increased, the probability that they will all adopt the value of 1 simultaneously diminishes. As a result, the outputs of the AND gates tend to get stuck at 0, causing the output of the connected JK-FF to also become stuck.

TABLE I
TRUTH TABLE FOR THE STOCHASTIC VN OF FIG. 9(A).

P_A	P_B	P_C	
0	0	0	regenerative bits
1	1	1	
1	0	previous P_C	conservative bits
0	1	previous P_C	

This problem is exacerbated by cycles in the factor graphs of LDPC codes, which comprise looping paths along the edges between connected CNs and VNs [50], [55]. In LDPC-SDs, these cycles can result in positive feedback loops, which can cause particular stochastic bits to get stuck at particular values over many successive clock cycles. This so-called latching problem typically prevents a LDPC-SD from converging to a valid decoding result, unless special measures are taken to mitigate the locked state of the nodes, as discussed below.

Anti-Latching Techniques: In particular, VNs having large degrees are most affected by the latching problem, since they typically form part of more cycles in the factor graph and because they pass their locked *extrinsic* bits to more CNs. Therefore, the latching problem can be addressed by redesigning the VNs having large degrees, at the cost of making them slightly more complex. In order to increase the switching activity and therefore mitigate the latching problem for stochastic VNs having high degrees, the implementations of Fig. 9(a) and 9(b) can be combined recursively, as exemplified in Fig. 9(d) for the case of a VN having a degree of $d_i = 4$ and in Fig. 9(e) for the case of a VN having a degree of $d_i = 6$. Similarly, as the layered implementation of Fig. 9(d) and Fig. 9(e), this anti-latching implementation can be applied to Fig. 9(b) for a VN having a degree of $d_i = 3$, which breaks down into the combination of the two circuits of Fig. 9(a), as shown in Fig. 9(c).

As a further step, the authors of [3], [28] proposed the replacement of the JK-FFs employed by VNs with circuits based on so-called Edge Memories (EMs), which comprise a shift register formed of several D-FFs, as shown in Fig. 10. The circuit of Fig. 10 behaves in analogy to a JK-FF and

may therefore replace the JK-FFs of Fig. 9. Note that the employment of the AND gates in Fig. 9 ensures that the inputs J and K of Fig. 10 cannot both have the value of 1 simultaneously. Therefore, the circuit of Fig. 10 does not have to implement the toggle functionality of a JK-FF. This means that a disagreement between the values of J and K can detected using the OR gate of Fig. 10. In this case, the MUX of Fig. 10 passes the value of the input J to the output D-FF, in analogy to the operation of a JK-FF. In this case, the update signal is used to shift the contents of the EM by one position and to shift in the value of the input J. By contrast, if the inputs J and K both adopt the value of 0, then a random bit is selected from the contents of the EM and passed to the output D-FF. In this way, a bit value from a previous clock cycle is output, in analogy to the behavior of a JK-FF, when preserving the value from the previous clock cycle. Since the EM typically stores a mixture of both 0s and 1s, the circuit of Fig. 10 allows a locked state to be escaped, in contrast to a JK-FF-based VN.

In VNs having degrees of 2,3,4 and 6, [3], [28] recommended the use of EMs comprising 32, 48, 48 and 64 D-FFs, respectively. However, in high-degree VNs adopting the layered structures of Fig. 9(e), the JK-FFs in the first layer may be replaced with so-called Intermediate Memories (IMs), rather than full EMs. These IMs have a similar structure as EMs, but only store 1 or 2 bits. More specifically, [3], [28] recommended the use of a pair of IMs comprising one or two D-FFs for VNs having degrees of 4 or 6, respectively. For VNs having a degree of 3, a single IM comprising one D-FF was recommended in [3], [28] for combining two of the three inputs to each computation unit. Meanwhile, IMs are not required for VNs having a degree of 2. Only small IMs are needed, since the outputs of these IMs feed into a large EM in the second layer, which can generate the VN's final output bit and mitigate the latching problem. Furthermore, in IMs, the output D-FF of Fig. 10 may be omitted, in order to reduce the number of clock cycles required for stochastic bits to propagate through the VN, as shown in Fig. 11 for the case of a VN having a degree of 6. Throughout the remainder of this paper, we assume the use of IMs and EMs in all LDPC-SDs discussions, unless specified otherwise.

As discussed previously, the mixture of 0s and 1s is that are stored by the EMs is essential for helping the VNs to escape the latching state. Furthermore, EMs are heavily relied on at the start of the iterative decoding process, when the inputs of the VNs are more likely to disagree with each other. Therefore, the initial values that are stored in EMs before the iterative decoding process begins plays an important role in the decoding of LDPC-SDs [3]. Motivated by this, [3] proposed to initialize the EMs of each VN with a Bernoulli sequence derived from the corresponding the input bit probability provided by the channel. More explicitly, before an iterative decoding process begins, the initialization signal shown in Fig. 11 is set to 1, for as many clock cycles as is required fill the largest EM with bits provided by the convertors of Fig. 8. After this initialization step, the initialization signals are reset to 0 and the iterative operation of the CNs and VNs in alternative clock cycles begins, whereupon the EM is controlled by the update signal discussed previously. It has been demonstrated that this initialization technique allows LDPC-SDs to achieve a similar BER performance as the LDPC-FDs [3].

So-called Noise-Dependent Scaling (NDS) [28] may also be used to help prevent the occurrence of the latching problem. This technique scales the bit probabilities provided by the demodulator before they are converted into stochastic bits and provided to the VNs, both during their initialization and during the iterative decoding process. In this way, the degree of the switching activity within the stochastic decoder can be maintained at a sufficiently high level, that mitigates the latching problem for all channel Signal-to-Noise Ratios (SNRs). In the case where BPSK modulation is employed for transmission over an AWGN channel, the scaled bit probability is obtained as

$$P = \frac{1}{1 + \exp(\frac{\alpha N_0}{y_{max}} \cdot \frac{4y}{N_0} \cdot \mathrm{Re}(y))}, \tag{5}$$

where y is the received BPSK symbol, N_0 is the AWGN power spectral density and $\frac{\alpha N_0}{y_{max}}$ is the noise-dependent scaling factor. Here, y_{max} is a predefined constant relating to the maximum value of $\mathrm{Re}(y)$ and α is a constant parameter of the scaling. By tuning the value of α, the BER performance of LDPC-SDs may be optimized [28]. According to [28], y_{max} has the value of 6 for BPSK transmission, where the optimal value of α is around 3.

Decision Unit and Termination of Decoding: The method used by the decision unit of Fig. 8 to generate a decision bit in each VN has a significant impact on the error correction performance of an LDPC-SD. However, neither the generation of the decision bits nor their impact on the error correction performance has been addressed in previous publications on LDPC-SDs. Therefore, we propose the employment of a JK-FF, to obtain the decision bits, which we have found to achieve a strong error-correction performance. As illustrated in Fig. 12, our decision unit is similar to the computation unit of Fig. 9(b), but replacing all *a priori* input bits, as well as the input bit from the channel, with the values of extrinsic output bits from the computation units of Fig. 8. If all the extrinsic bits agree with each other, then this value is passed to the output as the decoded bit. Otherwise, the value output in the previous clock cycle is preserved for the decision bit.

The iterative stochastic LDPC decoding process can be terminated, once a valid set of decoded bits is found, or when a predefined maximum number of decoding cycles is reached. The former termination condition relies on having a method for validating the decoding bits. In particular, the decoded bits may be considered to be valid, if their multiplication with the parity-check matrix produces an all-zero syndrome [3].

III. Motivation and Design Flow

In this section, we address the conventional design flow for LDPC decoders and discuss the modifications that are motivated for the design of timing-error-tolerant LDPC decoders. A typical design flow for ASIC implementations of LDPC decoders is illustrated in Fig. 13. The process starts from the specifications and proceeds step-by-step until the bottom level is reached, where the ASIC is fabricated. Here, the specification defines the design goals of the LDPC decoder,

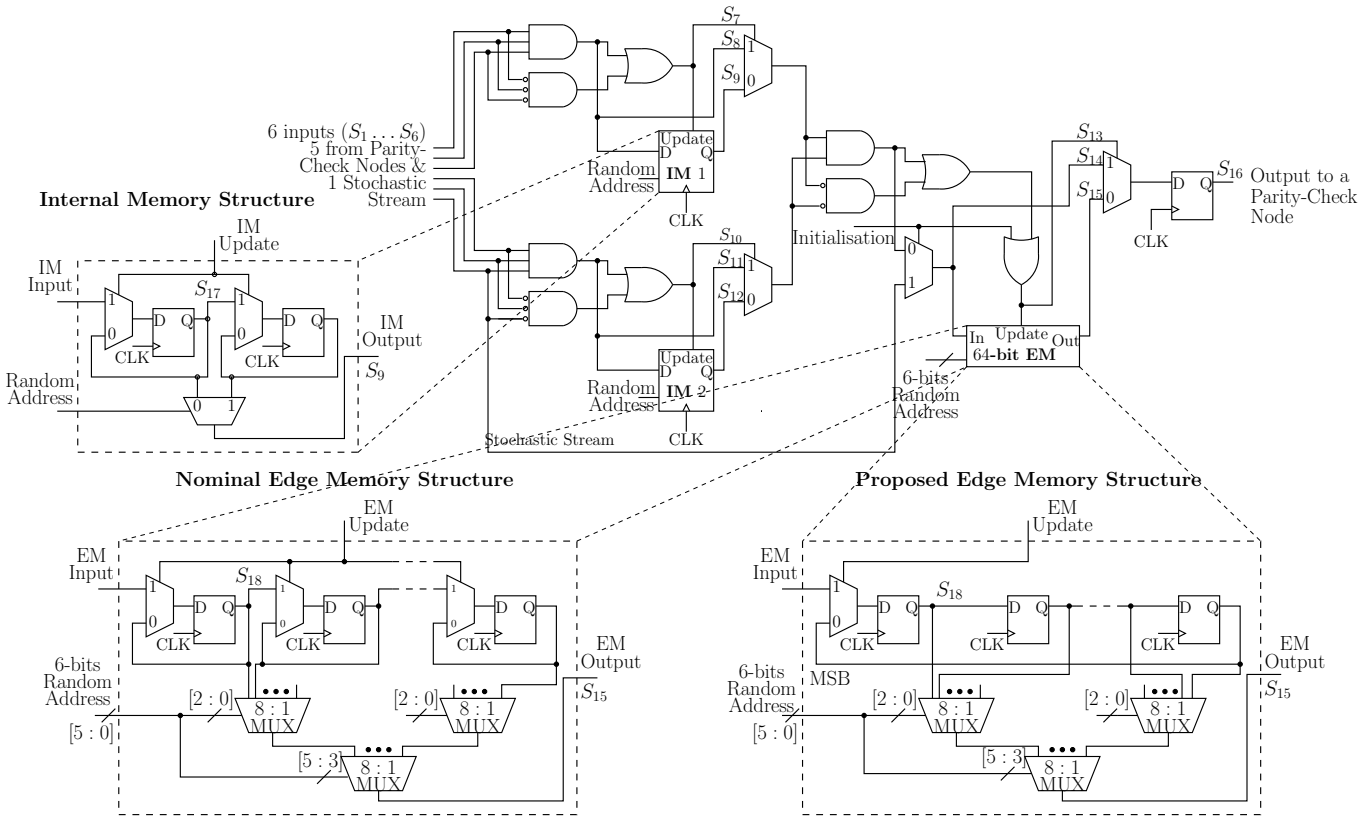


Fig. 11. Schematic of an anti-latching stochastic $d_i = 6$ VN. Corresponding schematics for $d_i = 2, 3$ and 4 VNs can be obtained combining the EM circuit of Fig. 10 with the VNs of Fig. 9(a), 9(b) and 9(d) [3, Figure 6].

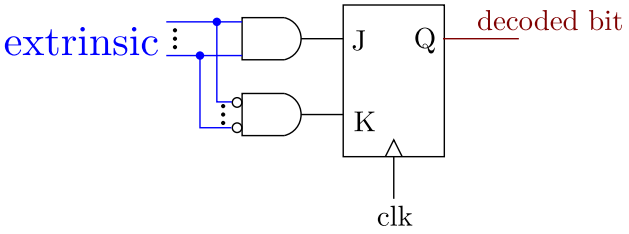


Fig. 12. The structure of the decision unit used to generate the decoded bit in a VN of an LDPC-SD.

which informs the selection of algorithmic parameters, such as block length, coding rate, and LDPC factor graph topology [50], [56]. Following this, an algorithm-level simulation of the iterative LDPC decoding process is employed to validate the algorithmic parameters and to identify the number of clock cycles required to achieve the desired BER. The simulation may be extended to also consider architectural parameters, such as fixed-point bit width, in order to characterize the effect on the BER. Following this, the algorithm can be converted into a Register Transfer Level (RTL) description, which precisely describes the behaviour of the decoder and allows synthesis. By linking a particular library of cells, the RTL behaviour description can be replaced by logic gates and interconnections, facilitating gate level simulation. The gates can be further replaced by transistors during placing and routing, with corresponding libraries. This facilitates transistor level simulation, which validates the operation and identifies the ASIC area, energy consumption and clock frequency. The above-mentioned design steps may be iterated several times, before finally taping out the design. As the design flow proceeds through each subsequent level, the design is enriched with more realistic simulation results, but also becomes more complex, time consuming and expensive.

Furthermore, the complexity of each level is significantly further increased when investigating the circuit behaviour under the influence of timing errors, which transform the synchronous digital operation of the circuit into asynchronous analogue operation. Consequently, it may become impossible to comprehensively simulate the timing error tolerance of an ASIC LDPC decoder at the lower design levels. As a result, the characteristics of the tape-out become unpredictable with a risk that the tape-out is unsuccessful. This risk of wasting all of the invested time, effort and expense, makes it undesirable to investigate timing error tolerance based only on measurements taken from a fabricated ASIC. Furthermore, the effect of the parameters chosen for the ASIC cannot be investigated, since these are typically hard-coded into a tape-out. Additionally, ASICs are not capable of offering insights into the internal operations of the design, in the way that simulations can. Furthermore, these experimental results may be influenced by numerous unpredictable factors, such as temperature variations, electromagnetic radiation and processing variations, which may obfuscate the results. Finally, these measurements cannot be readily reproduced by other researchers, unless they have access to the fabricated chip.

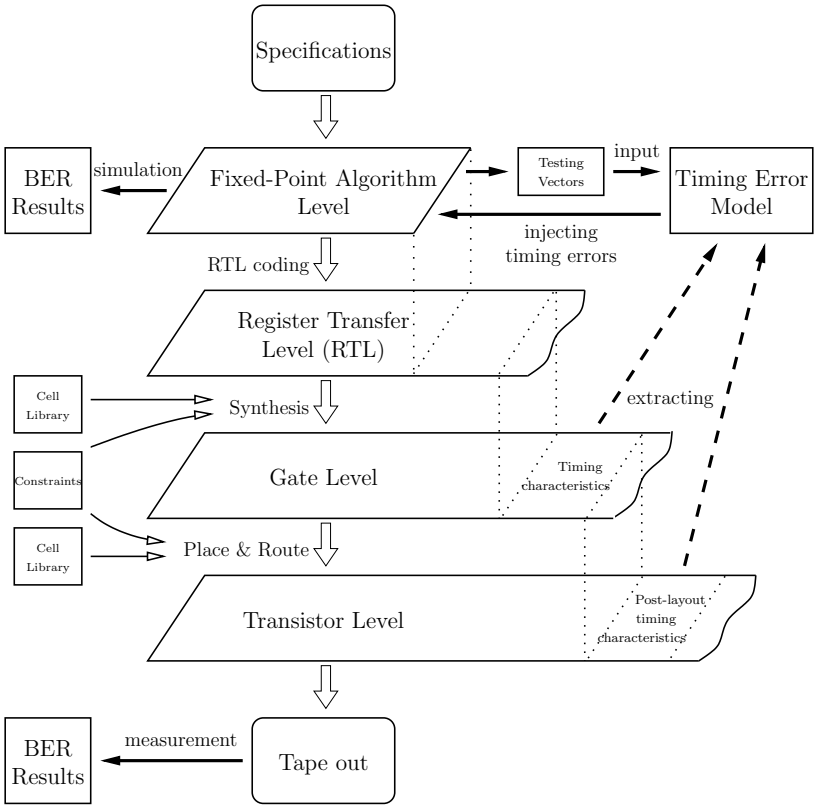


Fig. 13. The proposed design flow for LDPC decoder ASICs.

Owing to these issues, research is motivated in not only timing-error-tolerant LDPC implementations, but also in the design process for these implementations [6], [7], [11], [12]. This is particularly motivated now that fabrication technology has moved into the sub-65nm regime, where the reliability of nanoelectronic systems cannot be guaranteed, due to the increased effect of the IR drop, inductive noise, crosstalk, electrostatic discharges, particle strikes, switching noise and fabrication process variations [5], [11], [31]–[35].

Motivated by this, we propose a novel process for designing and characterizing our modified fully-parallel stochastic LDPC decoder, which has improved inherent tolerance to timing errors, compared to conventional designs. The arrows pointing upwards on in the right-hand side of Fig. 13 illustrates the proposed modification to the design flow, in contrast to the traditional design flow shown on the left-hand side. More specifically, we use simulations at the algorithm level to investigate the decoders' BER performance, but we incorporate a model of the causes and effects of timing errors, which is parametrized by characteristics obtained from the lower design levels of Fig. 13. Although the most realistic experimental results can only be obtained by taking measurements from a fabricated ASIC, this implies a huge amount of design time, effort and financial investment, which may not deliver the desired performance as described above. Therefore, our simulation-only approach de-risks the expensive and time-consuming fabrication of an ASIC, allowing confidence to be gained in a timing-error-tolerant design. This will pave the way for our future work, which will fabricate a timing-error-tolerant LDPC decoder ASIC. Furthermore, our simulations at the upper levels of the design flow are immune to random variation and are readily repeatable, using synthesis processing, SPICE simulation and Cadance simulation. Additionally, our approach allows the characterization of the effect of each design parameter on the circuit performance to be characterised, enabling the optimization of parameter values. We will demonstrate the proposed design flow throughout the remainder of this paper.

IV. Overclocking-Induced Timing Error Analysis

As described in Section I, overclocking causes timing errors, whenever there is insufficient time for a signal to propagate to the input of a memory, before its value is clocked into the memory. When aggressive overclocking is employed, the clock period T_{clk} is reduced below the nominal propagation delay t of some signals in a circuit, typically imposing timing errors. However, even moderate overclocking may cause timing errors, since this makes the circuit more sensitive to the late arrival of signals, owing to fluctuations in their propagation delay from one clock cycle to the next. These fluctuations may caused by power supply noise, which models the effects of IR drop, inductive noise, cross talk, electrostatic discharges, particle strikes, switching noise and fabrication process variations [5], [11], [31]–[35]. Formally, a timing error occurs for a particular signal when

$$t \times \delta > T_{\text{clk}}, \tag{6}$$

where δ characterizes the fluctuation in the propagation delay within the current clock cycle.

There are numerous methods that may be employed to quantify the nominal propagation delays t, their fluctuation δ and the occurrence of timing errors. The most realistic method for quantifying these circuit characteristics is to fabricate an ASIC and to measure them, as described in Section III. However we did not opt for this method, since these experimental results may be influenced by numerous uncontrollable factors, such as temperature variations, electromagnetic radiation and processing variations, which may obfuscate the results. Furthermore, these measurements cannot be readily reproduced by other researchers, if they do not have access to the chip. By contrast, SPICE simulation and manual calculations using component datasheets are immune to random variation and are readily repeatable, as described in Section III. In particular, these methods allow the isolated investigation of how the circuit characteristics are influenced by a particular parameter, such as supply voltage. Therefore, in the following sections of this paper, the values of delays t, their fluctuation δ and the occurrence of timing errors are modelled using either the data provided in the STMicroelectronics 90 nm datasheet [57] or SPICE simulations.

In Section IV-A, we characterize the nominal signal propagation delays t within stochastic VNs and CNs having various degrees. Following this, the fluctuation δ of signal propagation delays from one clock cycle to the next is characterized in Section IV-B. Finally, Section IV-C models the causes and effects of timing errors within LDPC-SDs, focussing on VNs having a degree of $d_i = 6$, since they are found to have the highest susceptibility to timing errors.

A. Nominal Signal Propagation Delays of Stochastic LDPC Decoders

In this section, we characterize the nominal signal propagation delays t of stochastic VNs and CNs having various degrees. More specifically, we consider VNs having degrees of $d_i \in \{2, 3, 6\}$ and CNs having degrees of $d_j \in \{6, 7\}$, as employed in the $k = 528, n = 1056$ WiMAX LDPC code [2]. Later, another $k = 2304, n = 1920$ WiMAX LDPC code containing VNs having a degree of 4 and CNs having a degree of 20 will also be considered. More particularly, we model the nominal propagation delay t of the signal arriving at each D-FF within these VNs and CNs. These are obtained as the maximum of the nominal propagation delays of all paths that end at the particular D-FF. Here, the nominal propagation delay of a path includes the output delay of the D-FF at the beginning of the path, the total delay of the combinational logic along the path and the setup time of the D-FF at the end of the path. We initially focus on the particular case of VNs having a degree of $d_i = 6$, before summarizing the analysis of the other nodes. For VNs having a degree of $d_i = 6$, we consider four sets of D-FF, which we refer to as the output D-FF, the IM1 D-FFs, the IM2 D-FFs and the EM D-FFs. In Fig. 11, the output D-FF provides the signal S_{16}, while the signals S_{17} and S_{18} are provided by IM1 and EM D-FFs, respectively. Note that VNs having a degree of $d_i = 3$ require only a single IM and hence they do not have any IM2 D-FFs [3, Figure 6], while VNs having a degree of $d_i = 2$ do not have any IMs and therefore do not have any IM1 D-FFs either.

The nominal propagation delay t of the signal arriving at a particular D-FF depends on the state of the VN or CN during both the previous and current clock periods. However, it is not feasible to simulate all the possible combinations of current and previous states. For example, the VN having a degree of $d_i = 6$ shown in Fig. 11 has 15 inputs, including control signals. In two consecutive clock cycles, these inputs will adopt one of $2^{2\times 15} \approx 10^9$ combinations of values. The amount of time required to simulate all of these combinations would be prohibitively excessive. Instead, our analysis focuses on the combinations of the particular signals that have the greatest effect on the operation and nominal propagation delay of the VNs. Unlike in the stochastic CNs, the flow of information within the VNs is controlled by multiplexers (MUXs), which have a significant impact on the operations of IMs, the EM and the final output. Owing to their importance, our analysis carefully considers the values of the MUX selector signals in both the current and the previous clock cycles. More specifically, when a MUX selector signal remains constant between consecutive clock cycles, the propagation delay of the MUX output depends only on the delay of the selected signal. By contrast, if the MUX selector signal is toggled in the current clock cycle, then the propagation delay of the MUX output is given by the maximum of the MUX selector signal's delay and the selected signal's delay. In the case of VNs having a degree of $d_i = 6$, we consider three MUX selector signals, which we refer to as the IM1, the IM2 and the EM MUX selector signals, that are respectively labelled as S_7, S_{10} and S_{13} in Fig. 11. Our analysis considers all $2^{2\times 3} = 64$ combinations of these MUX selector signals in both the current and previous clock cycles, offering a significant simplification, while capturing the main operation of the VN. Similarly, the IM1, IM2 and the EM MUX selector signals are considered for $d_i = 4$ VN, while we consider only the IM1 and the EM MUX selector signals for VN having a degree of $d_i = 3$, as well as only the EM MUX selector signal for the VN having a degree of $d_i = 2$, since they have only a single IM and no IMs, respectively.

Table II provides the nominal propagation delays associated with the input of each D-FF in stochastic VNs and CNs having various degrees. In Table II, 'toggle' indicates that the corresponding MUX selector signal of a VN has different values in the previous and current clock cycles. By contrast, '1' and '0' indicate that the MUX selector signal has maintained a constant value of either 1 or 0 in the previous and current clock cycles, respectively. Finally 'any' indicates that the nominal propagation path delay is not dependent on the corresponding MUX selector signal. Here, in order to simplify the analysis, we model the propagation delay of each gate using a constant value, which is obtained by considering its worst-case switching condition and the worst-case loading that is imposed by the particular set of gates that it drives. This approach is justified, since the loads imposed by the different pins of each gate vary by only about 1%. Furthermore, this worst-case approach guarantees a worst-case

upper bound for the performance degradation of Sections VI, since timing errors are most likely to occur for signals having long propagation delays. For providing deeper insights into the propagation delays and for obtaining reproducible results, the values of Table II were calculated manually using data provided in the STMicroelectronics 90 nm datasheet [57]. Note that the nominal path delay of CNs having a degree of $d_j = 20$ is omitted from Table II for simplicity, since it is only slightly higher than that of CNs having a degree of $d_j = 7$, when implemented using the CN structure of Fig. 7. Also note that the wire delay is neglected in our analysis of the nominal signal propagation delays of the LDPC-SD. This is justified, since 90 nm ASIC implementations of LDPC-SDs typically have dimensions that are no greater than a few millimeters [27]. Owing to this, the expected maximum wire delays may be of the order of tens of picoseconds [58], which is negligible compared to the propagation delays within the stochastic nodes.

B. Propagation Delay Fluctuation

As described in Section I, supply voltage noise can model the effects of IR drop, inductive noise, cross talk, electrostatic discharges, particle strikes, switching noise and fabrication process variations [5], [11], [31]–[35]. In this case, the supply voltage V_{DD} may be assumed to have a Gaussian distribution with a mean value that is equal to the nominal supply voltage μ and a standard deviation σ, which is related to the circuit layout, fabrication process and technology scale, in circuits containing a sufficiently high number of gates [59], [60]. Since the propagation delay of a particular type of gate is a decreasing function of the supply voltage V_{DD}, it can be modelled as an random variable that is independently selected from a particular Probability Density Function (PDF) in each successive clock cycle. We found that the time-normalized NAND gate delay PDF can accurately represent the PDFs of other types of logic gates, which is shown in Fig. 14 for $3\sigma/\mu \in \{0.01, 0.1, 0.3\}$ [60]. For example, when $3\sigma/\mu = 0.1$, the propagation delay of a NAND gate extends above its nominal value by 10% or more with a probability of around 1%. Correspondingly, the probability of a NOT gate's propagation delay extending by 10% or more above its nominal value is also around 1%, when $3\sigma/\mu = 0.1$. Hence, we employ the time-normalized NAND gate delay PDF to model the propagation delay distribution of all circuit components, in order to simplify the analysis of the following sections, as recommended in [60].

C. Effects of timing errors in Stochastic LDPC Decoders

As shown in Table II, the maximum nominal propagation delay t within stochastic VNs is 727.6 ps, rendering them more susceptible to timing errors than the CNs, which have a maximum propagation delay of 618.1 ps. In order to simplify our analysis, we assume that the normalized delay multiplier δ is never large enough and the clock period T_{clk} is never low enough to cause timing errors within the CNs. More specifically, we assume that $(620 \times \delta) > T_{\text{clk}}$ never happens. This assumption allows us to focus our attention on the specific

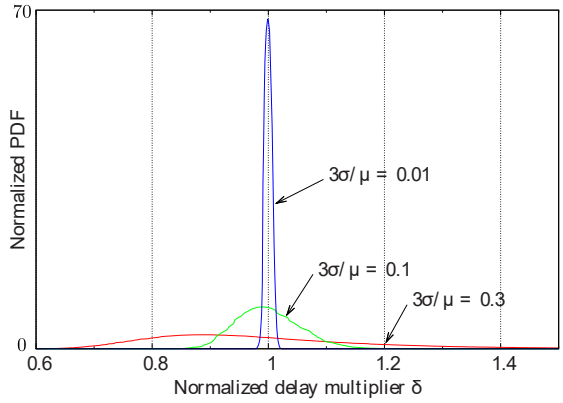
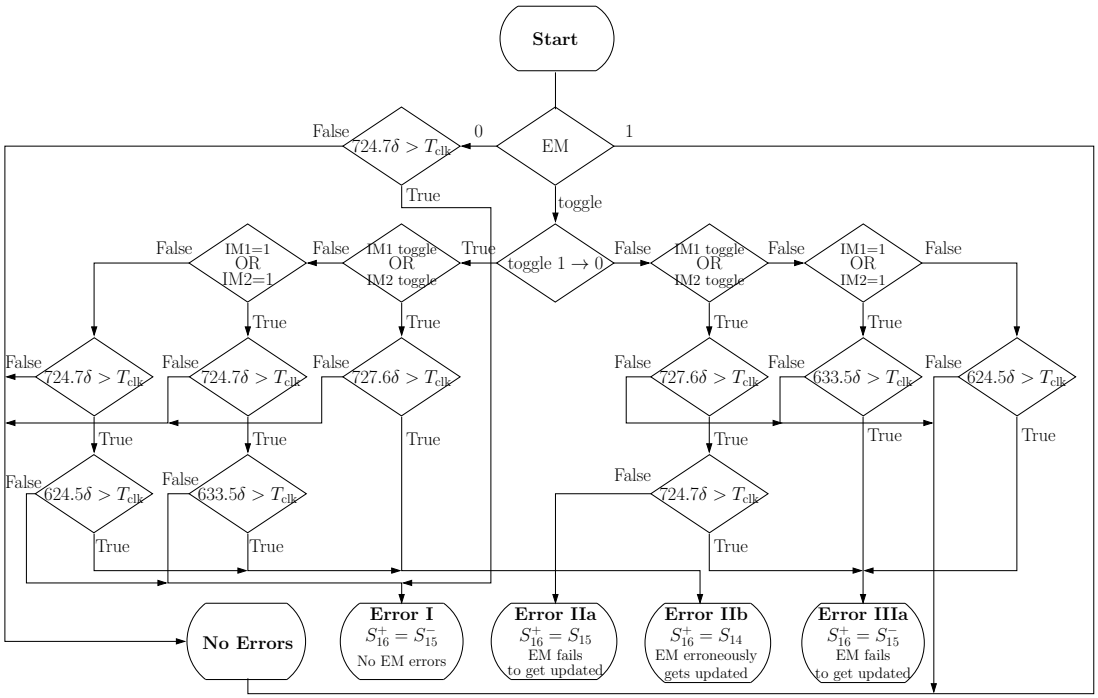


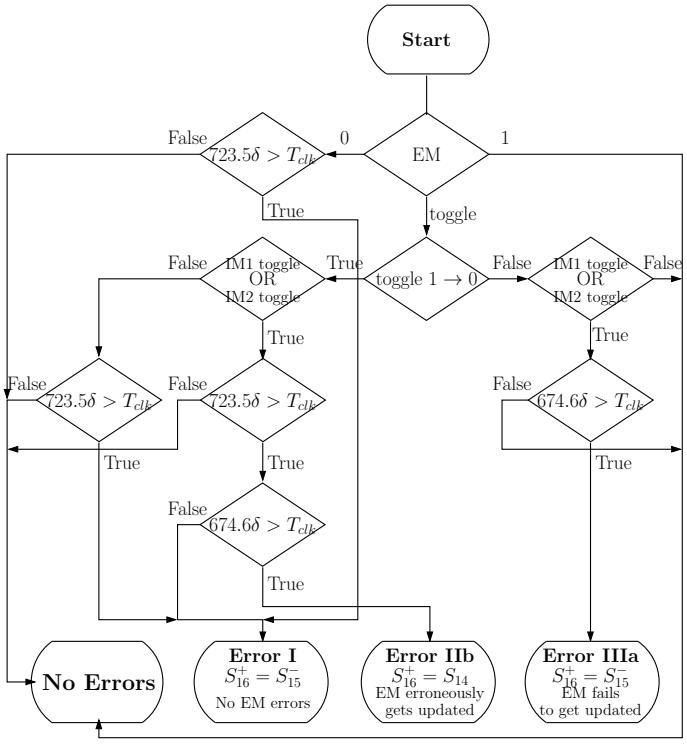
Fig. 14. Time-normalized NAND gate delay PDF for $3\sigma/\mu = \{0.01, 0.1, 0.3\}$ when employing STMicroelectronics 90 nm technology, where $\mu = 1$ V.

paths within the stochastic VNs that have nominal propagation delays t of at least 620 ps, as highlighted in bold within Table II. This approach is justified, since our experiments show that the inherent error tolerance of the LDPC-SD entirely breaks down, when timing errors occur in the CNs. As shown in Table II, nominal propagation delays t of at least 620 ps are manifested in $d_i = 2, 3, 4$ and 6 VNs. In order to simplify our discussions, we exemplify our analysis by focusing on $d_i = 6$ VNs of Fig. 11. This is because Section IV-A revealed that these VNs are associated with the longest propagation delays and hence have the highest susceptibility to timing errors. Note that the corresponding analysis carried out for VNs having lower degrees is similar, but simpler. In the case of the $d_i = 6$ VN of Fig. 11, only three types of timing errors need to be considered, as summarized by the flowchart of Fig. 15(a). Note that corresponding flowcharts are provided for VNs having degrees of $d_i = 2$ and 3 in Fig. 16 and 17, respectively. Due to the identical implementation in the first layer of Fig. 9(c) and 9(d), VNs having a degree of $d_i = 4$ have the similar timing characteristic to the VNs having a degree of $d_i = 3$, as shown in Table II, despite that two IMs instead of one are employed in the first layer of Fig. 9(c) in parallel. Therefore, the flowchart for VNs having a degree of $d_i = 4$ can be obtained by replacing the condition "IM1 toggle" in the flowchart of Fig. 17(a) with "IM1 or IM2 toggle". These flowcharts may be derived by combining (6) with the MUX selector signal conditions and with the nominal propagation delays t of Table II, as described for VNs having a degree of $d_i = 6$ in the following paragraphs for each type of timing error. More specifically, each value of t shown in Fig. 15(a), *e.g.* 674.6 ps, is obtained from the appropriate entry in Table II, which quantifies the timing characteristics of the corresponding path within the selected VN, as obtained during our simulations described in Section IV-A.

1) Timing Error Type I: In this type of timing error, the propagation of S_{15} is not completed before the end of the clock cycle, but the EM MUX selector signal S_{13} arrives on time. However, the EM MUX will not select the late S_{15} signal if $S_{13} = 1$, preventing the occurrence of a timing error. Therefore, $S_{13} = 0$ is a condition for a Type I error to occur.



(a) Before modification.



(b) After modification. Observe that a number of paths leading to type II and II timing errors have been removed, hence eliminating the type IIa errors.

Fig. 15. Flowcharts illustrating the causes and effects of timing errors in stochastic VNs having a degree of $d_i = 6$.

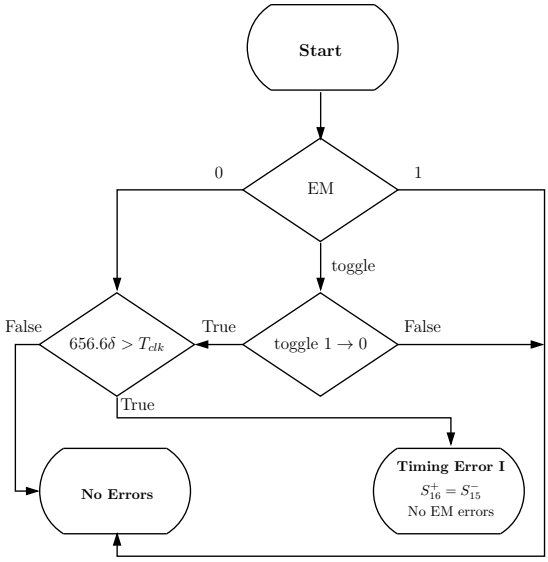
TABLE II
NOMINAL PROPAGATION DELAYS WITHIN THE VNS AND CNS OF THE LDPC-SD OF [3], WHEN EMPLOYING STMICROELECTRONICS 90 NM TECHNOLOGY. THE NOMINAL PROPAGATION DELAYS OF THE MODIFIED LDPC-SD ARE PROVIDED IN BRACKETS, WHERE THEY DIFFER.

Node	Updated D-FF	MUX selector signal			Nominal propagation delay t (ps)				
		EM	IM1	IM2	Degree 2	Degree 3	Degree 4	Degree 6	Degree 7
VN	EM		toggle	any	463.6(439.0)	**687.9(650.6)**	**687.9(650.6)**	**727.6(674.6)**	–
		toggle	1	toggle		595.4 (558.1)	**687.9(650.6)**	**727.6(674.6)**	
				1			595.4(558.1)	**633.5**(580.5)	
				0				**633.5**(580.5)	
			0	toggle		542.1 (504.8)	**687.9(650.6)**	**727.6(674.6)**	
				1			595.4(558.1)	**633.5**(580.5)	
				0			542.1(504.8)	**624.5** (571.5)	
		1	toggle	any	387.9	573.6	573.6	597.6	
			1	toggle		481.1	573.6	597.6	
				1			481.1	503.5	
				0				503.5	
			0	toggle		427.8	573.6	597.6	
				1			481.1	503.5	
				0			427.8	494.5	
		0	any	any		573.6	573.6	597.6	
	Output	toggle $0 \rightarrow 1$	toggle	any	463.6(439.0)	**687.9(650.6)**	**687.9(650.6)**	**727.6(674.6)**	
			1	toggle		595.4 (558.1)	**687.9(650.6)**	**727.6(674.6)**	
				1			595.4(558.1)	**633.5**(580.5)	
				0				**633.5**(580.5)	
			0	toggle		542.1 (504.8)	**687.9(650.6)**	**727.6(674.6)**	
				1			595.4(558.1)	**633.5**(580.5)	
				0			542.1 (504.8)	**624.5** (571.5)	
		toggle $1 \rightarrow 0$	toggle	any	**656.6(653.3)**	**724.7(723.5)**	**724.7(723.5)**	**727.6(723.5)**	
			1	toggle				**727.6(723.5)**	
				1				**724.7(723.5)**	
				0				**724.7(723.5)**	
			0	toggle				**727.6(723.5)**	
				1				**724.7(723.5)**	
				0				**724.7(723.5)**	
		1	toggle	any	387.9	573.6	573.6	597.6	
			1	toggle		481.1	573.6	597.6	
				1			481.1	503.5	
				0				503.5	
			0	toggle		427.8	573.6	597.6	
				1			481.1	503.5	
				0			427.8	494.5	
		0	any	any	**656.6(653.3)**	**724.7(723.5)**	**724.7(723.5)**	**724.7(723.5)**	
	IM1	any	toggle	any	N/A	393.0	393.0	417.0	
			1			301.1	301.1	323.5	
			0					323.5	
	IM2	any	any	toggle	N/A	N/A	393.0	417.0	
				1			301.1	323.5	
				0				323.5	
CN	Output	N/A	N/A	N/A	–	–	–	511.0	618.1

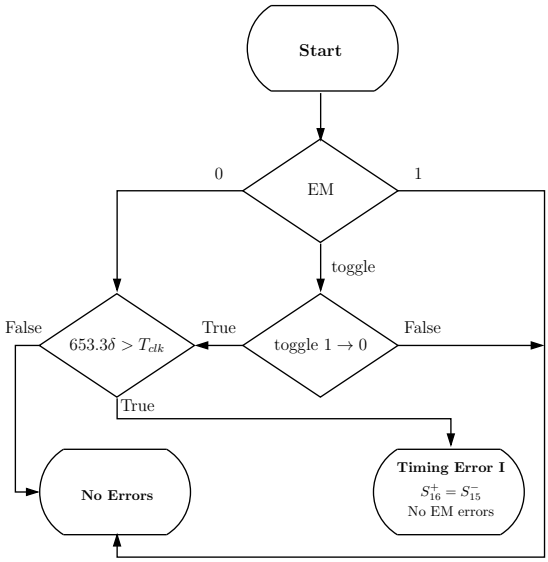
Instead of clocking the correct value of the signal S_{15} into the output D-FF, its value from the previous clock cycle S_{15}^{-} is latched. Note that timing errors are not inflicted upon the EM D-FFs, when a Type I error occurs, since the late S_{15} signal is not an input to the EM.

2) Timing Error Type II: Type II errors occur if S_{15} arrives on time, but S_{13} is toggled and arrives late. In this case, it is the previous value of S_{13}^{-} that controls both the updating of the EM D-FFs and the selection of the signal that is clocked into the output D-FF. Type II errors can be further classified into Type IIa and IIb errors, depending on the value of S_{13}^{-}. A Type IIa error occurs, when the EM MUX selector signal is toggled according to $S_{13}^{-} = 0$ and $S_{13} = 1$. In this case, the updating of the EM will be erroneously prevented and instead of S_{14}, it will be S_{15} that is clocked into the output D-FF. By contrast, $S_{13}^{-} = 1$ and $S_{13} = 0$ is associated with a Type IIb error, which results in S_{14} being erroneously clocked into the first EM D-FF, as well as into the output D-FF.

3) Timing Error Type III: Finally, the Type III errors occur, when S_{13} is toggled and arrives late, as well as S_{15} arriving late. Again, it is the previous value of S_{13}^{-} that controls the updating of the EM D-FFs and the selection of the signal that is latched into the output D-FF. Similarly, Type III errors can also be further classified into Type IIIa and IIIb errors, depending on the value of S_{13}^{-}. A Type IIIa error occurs when $S_{13}^{-} = 0$ and $S_{13} = 1$, causing the updating of the EM to be erroneously prevented and the wrong signal to be clocked into the output D-FF, instead of S_{14}. However, since S_{15} is late, it will not be clocked into the output D-FF as in the Type IIa error. Instead, its value in the previous clock cycle S_{15}^{-} will be clocked. By contrast, $S_{13}^{-} = 1$ and $S_{13} = 0$ is associated with a Type IIIb error, in which case the EM MUX will not select the late S_{15} signal. Instead, the timing error will cause S_{14} to be erroneously clocked into the first EM D-FF, as well as into the output D-FF. Note that Type IIIb errors have the same effect as Type IIb errors and so are merged into the type



(a) Before modification.



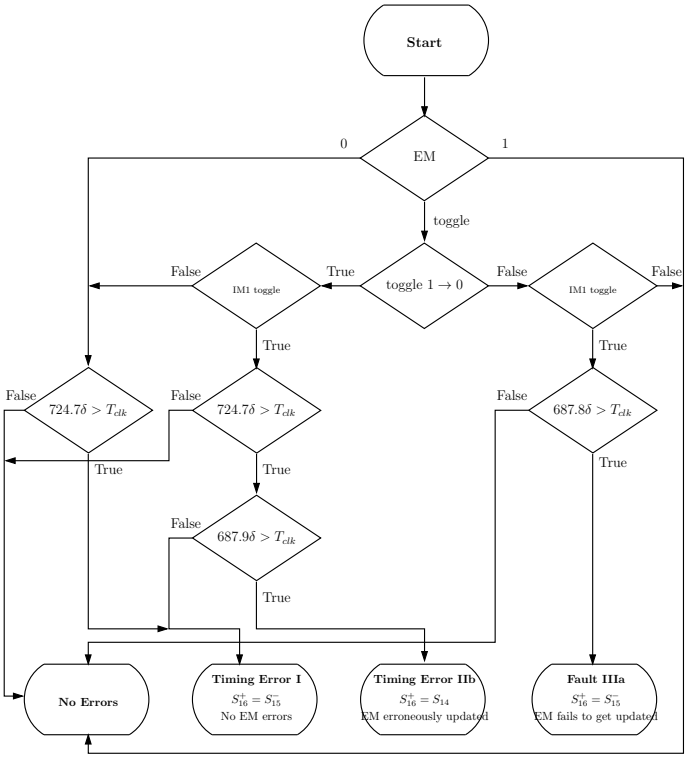
(b) After modification. Observe that the nominal propagation delay of the path has been reduced.

Fig. 16. Flowcharts illustrating the causes and effects of timing errors in stochastic VNs having a degree of $d_i = 2$.

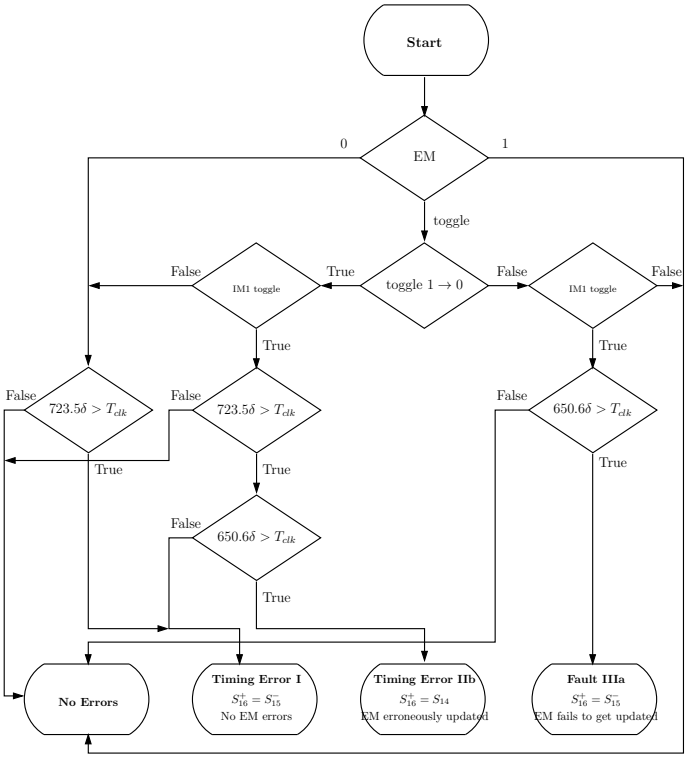
IIb outcome of Fig. 15(a).

D. Validation in SPICE

SPICE simulation has been shown to accurately predict practical measurements of circuit behaviour [61], [62], when operating in the presence of timing errors and other types of processing fault. However, since the SPICE simulation of the entire LDPC-SD would be impractical, our simulations considered only individual VNs and CNs in isolation. During these simulations, the clock period was scaled so that we could observe the occurrence of timing errors and validate the timing error model of Fig. 15(a). This approach is exemplified by the results of the VNs having a degree of $d_i = 6$ portrayed in Fig. 18. More specifically, Fig. 18 compares the ideal zero-delay response of the $d_i = 6$ VN with the simulated response



(a) Before modification.



(b) After modification. Observe that the nominal propagation delays of all paths have been reduced.

Fig. 17. Flowcharts illustrating the causes and effects of timing errors in stochastic VNs having a degree of $d_i = 3$. Note that the flowchart for VNs having a degree of $d_i = 4$ can be obtained by replacing the condition "IM1 toggle" with "IM1 or IM2 toggle".

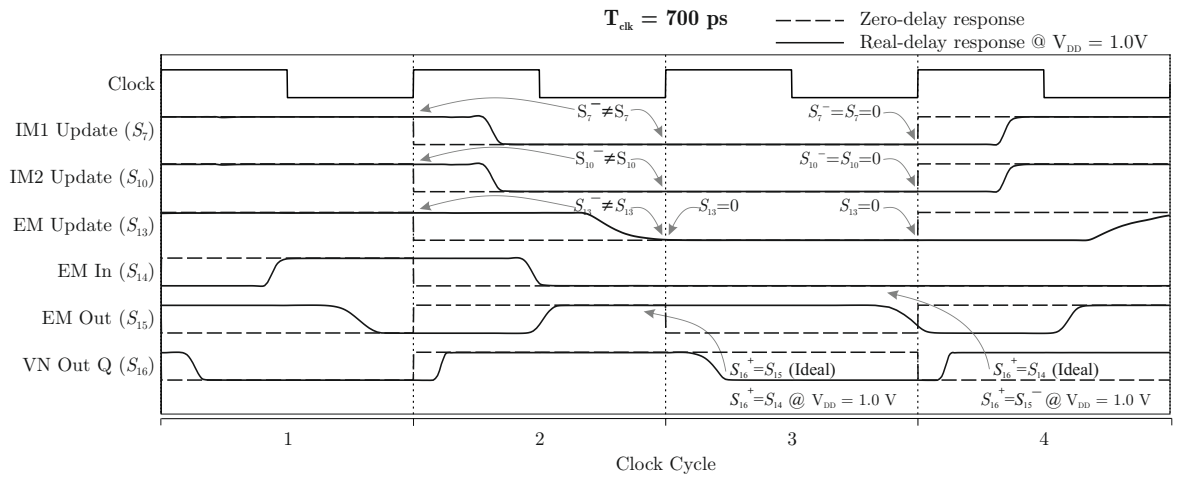


Fig. 18. SPICE simulation demonstrating the occurrence of Type IIb and Type I timing errors in a stochastic VN of [3] having a degree of $d_i = 6$, when employing STMicroelectronics 90 nm technology, where the supply voltage $V_{\rm DD}$ is set as 1.0 V, the clock period is set as 700 ps and random signals are used as the input.

for the case of implementation using STMicroelectronics 90 nm technology, with $V_{\rm DD} = 1.0$ V and $T_{\rm clk} = 700$ ps. As shown in Fig. 18, timing errors of Type IIb and I occur in clock cycles 2 and 3, respectively.

V. Modified Stochastic LDPC Decoder

As we will demonstrate later in Section VI, the LDPC-SD has a inherent tolerance to Type I timing errors. However, our results show that its BER performance is limited by the occurrence of Type II and III timing errors. In Section IV-C we showed that these types of timing errors are caused by the late arrival of the EM MUX selector signal, which is labeled S_{13} in Fig. 11. Motivated by this, Section V-A proposes a modification to the EM, which reduces the nominal propagation delay of S_{13} and mitigates the occurrence of Type II and III timing errors. The causes and effects of timing errors in the modified LDPC-SD are analysed in Section V-B, using a similar approach to that discussed in Section IV. The BER performance of the modified LDPC-SD will be characterised later in Section VI-B and compared with that of the LDPC-SD of [3], both in the absence and presence of timing errors.

A. Modified EM

As shown in Fig. 11, the EM MUX selector signal S_{13} also acts as the selector signal for a large number of MUXs within the EM structure of [3]. More specifically, 32, 48, 48 and 64 MUXs are employed within the EMs of VNs having degrees of $d_i = 2, 3, 4$ and 6, respectively. These MUXs generate a large capacitive load, which is the cause of the high nominal propagation delay of the EM MUX selector signal S_{13}.

In order to reduce this capacitive load and reduce the nominal propagation delay of the EM MUX selector signal S_{13}, we propose the modified EM structure of Fig. 11. Here, the EM MUX selector signal S_{13} acts as the selector signal for only one MUX within the EM structure. This is achieved by configuring the EM as a ring buffer, rather than as a shift register, as shown in Fig. 11. This is motivated by the observation that a ring buffer can fulfil the same role as a shift register in an EM, despite having a different operation. More explicitly, the role of an EM is to store recent VN decisions, so that when a decision cannot be made in future decoding cycles, one may be selected randomly from the EM. This role may be fulfilled by both a shift register and a ring buffer, since both replace older decisions with new ones, as they are clocked into the EM. Note however that in contrast to a shift register, the ring buffer does not necessary repalce the oldest decision when it is provided with a new one. Despite this, we will show in Section VI-B, that the replacement of the shift registers with ring buffers does not significantly affect the BER performance of the LDPC-SD in the absence of timing errors. Note that this modification eliminates all but one of the 2:1 MUXs that are employed in each EM. Owing to this, the total number of gates required to implement the $k = 528$ and $n = 1056$ IEEE 802.16e (WiMAX) LDPC decoder is reduced by 28.6%, from around 5.6×10^5 gates to around 4.0×10^5 gates. Furthermore, the proposed modification allows the OR gate that supplies the signal S_{13} to be replaced with a significantly smaller OR gate having a lower drive capability, owing to the reduced capacitive load that is imposed by these 2:1 MUXs.

B. Overclocking-Induced Timing Error Analysis

The same approach described in Section IV is employed here for quantifying the nominal propagation delays of every path within the modified LDPC-SD, as provided in brackets in Table II. It can be seen that many of the large nominal propagation delays are reduced in the modified LDPC-SD, owing to the reduced capacitive load that it imposes upon the EM MUX selector signal S_{13}. As shown in Table II, the number of highlighted entries having propagation delays exceeding that of the CN having a degree of $d_j = 7$ is reduced after the modification. Owing to this, a number of routes through the flowchart of Fig. 15(a) leading to Type II and III timing errors can be removed, as shown in the modified flowchart of Fig. 15(b). Consequently, the Type IIa timing errors are eliminated. Furthermore, higher values are

required for the normalized delay multiplier δ to satisfy the conditions for the remaining routes towards the Type II and III timing errors. As a result, Type II and III timing errors can be expected to occur significantly less often in the modified LDPC-SD, compared to the design of [3].

VI. Simulation Results and Discussions

In this section, we characterize the impact of timing errors upon the error correction capability of LDPC-SDs. Again, the most realistic method of characterizing this impact would be to fabricate an ASIC and to measure the occurrence of decoding errors. However, this is not preferred for the reasons discussed in Section IV. Instead, the BER results of Fig. 19 and 20 were obtained by applying the timing error model of Section IV-C within Monte-Carlo simulations written in C++. This has the advantage of allowing us to investigate the individual impact of the different types of timing errors that were introduced in Section IV-C, by disabling the occurrence of all other types of timing errors. This C++ simulation also allows the BER of the LDPC-SD to be characterised without incurring a significant overhead associated with simulating the occurrence of timing errors using SPICE, for example. Owing to this, a single CPU core requires about 24 hours of runtime to simulate the decoding of 10^9 message bits, which is typical of C++ BER simulations.

Our simulation parameters are summarized in Table III. In addition to the the STMicroelectronics 90 nm technology [57] discussed in Section IV-A, the Oklahoma State University FreePDK 45 nm technology [63] is also employed, for the sake of investigating timing errors in different fabrication scales. The BER performance of the LDPC-SD and the modified LDPC-SD of Section V are characterised for the case of employing Binary Phase Shift Keying (BPSK) for transmission over an Additive White Gaussian Noise (AWGN) channel, in order to facilitate comparisons with the results of [3]. In the following sections, we characterise the BER performance of two WiMAX LDPC codes, for which $k = 528, n = 1056$ and $k = 2304, n = 1920$. The former code has medium code rate of $1/2$ and a medium length of $k = 528$. By contrast, the latter code has the longest code length of $k = 2304$ and the highest code rate of $5/6$ supported in WiMAX, offering diverse application examples. Moreover, since our design is dependent on the characteristics of individual nodes, rather than on the interconnectivity among them, the BER plots of the two codes demonstrate that our design may be applied to various types of LDPC codes, provided that a similar timing analysis to that discussed in Section IV-A is completed for nodes having degrees different from those considered here. The number of memory elements used in the EMs are selected as 32, 48, 48 and 64 for VNs having degrees of $d_i = 2, 3, 4$ and 6, respectively, since these are the numbers employed in [3]. Likewise, for this reason, the IMs use $1, 1$ and 2 memory elements for VNs having degrees of $d_i = 3, 4$ and 6, respectively, while VNs having a degree of $d_i = 2$ contain no IMs [3]. The choices of $T_{\rm clk}$ and $3\sigma/\mu$ have been discussed previously in Sections IV and V-B. In each clock cycle, our simulations employed a different value for the normalized delay multiplier δ, which was randomly selected from the distribution of Fig. 14 associated with the selected values of $3\sigma/\mu$. The cause and effect of timing errors on each path within each VN in LDPC-SDs are revealed by identifying the appropriate path through the flowcharts of Fig. 15(a) and 15(b), by using the comparisons $t \times \delta > T_{\rm clk}$. All of the BER results are obtained by simulating the transmission of at least 10^6 codewords encoded from random information bits, then decoding the received codewords iteratively until early termination is triggered by the occurrence of an all-zero syndrome [3], or until the maximum affordable number of decoding cycles is reached. We limit the maximum number of decoding cycles to 2000, since this value is recommended for achieving the lowest BER shown in [3, Figure 12]. However, since the early termination cannot be anticipated in advance, the amount of time that is reserved for decoding each codeword is given by $2000T_{\rm clk}$. This corresponds to a processing throughput of $R_{\rm b} = k/(2000T_{\rm clk})$ information bits per second.

In Section VI-A, simulations were conducted for characterizing the BER performance of the LDPC-SD in the presence of timing errors, when employing diverse values of both the clock period $T_{\rm clk}$, as well as of the nominal supply voltage μ and its standard deviation σ. Following this, Section VI-B investigates and compares the BER performance of the novel modification proposed in Section V using similar simulations.

A. Inherent Timing Error Tolerance

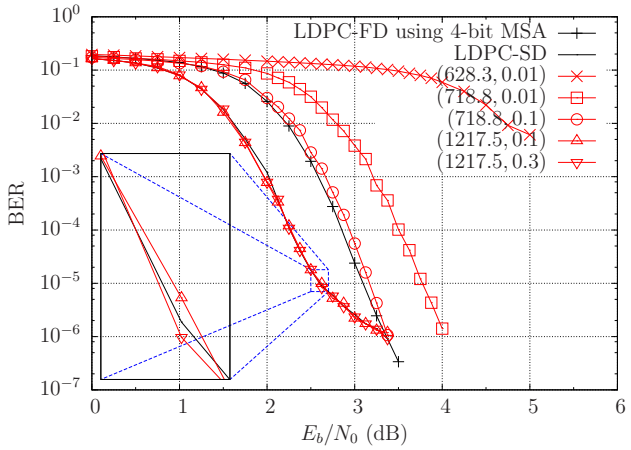
As described in Section IV-C, the analysis presented in the flowchart of Fig. 15(a) assumes that $\delta < T_{\rm clk}/620$, so that timing errors do not occur in the CNs. In order to satisfy this assumption in at least 99% of the clock cycles, the specific combinations of $(T_{\rm clk}, 3\sigma/\mu)$ that we consider for the LDPC-SD of [3] are $(628.3, 0.01)$, $(718.8, 0.01)$, $(718.8, 0.1)$, $(1217.3, 0.1)$ and $(1217.3, 0.3)$. Note that $T_{\rm clk} = 1217.3$ ps represents moderate overclocking, since this value exceeds 727.6 ps, which is the longest nominal propagation delay of Table II. By contrast, both $T_{\rm clk} = 628.3$ ps and 718.8 ps represent aggressive overclocking.

In order to consider the effect of different technology scales, the analysis described in the preceding sections for the STMicroelectronics 90 nm technology was also conducted for the Oklahoma State University FreePDK 45 nm technology [63]. In this case, our analysis assumes that $\delta < T_{\rm clk}/750$, which is satisfied in 99% of the clock cycles, when using the specific combinations for $(T_{\rm clk}, 3\sigma/\mu)$ of $(760.0, 0.01)$, $(869.5, 0.1)$, $(1171.2, 0.1)$, $(1472.8, 0.1)$ and $(1472.8, 0.3)$. Here, $T_{\rm clk} = 1472.8$ ps represents moderate overclocking, while $T_{\rm clk} = 1171.2$ ps, 869.5 ps and 760.0 ps represent aggressive overclocking.

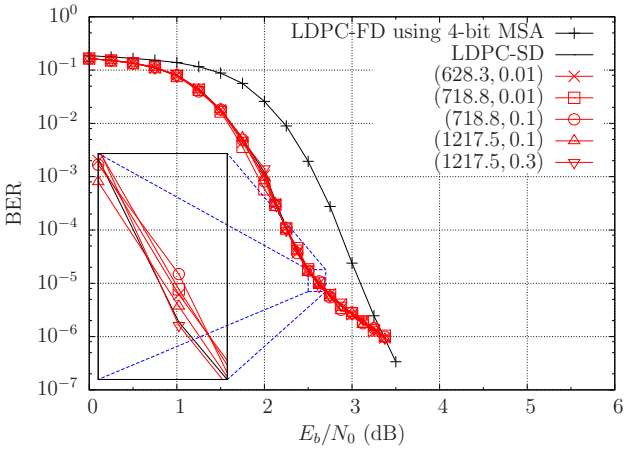
1) Tolerance to All Types of Timing Errors: Fig. 19(a) demonstrates the effect of all types of timing errors on the BER performance, when the LDPC-SD is implemented using 90 nm technology, while Fig. 20(a) provides the corresponding plot for 45 nm technology. The BER performance is also plotted for three benchmarkers, namely for the corresponding widely-used LDPC-FD using Min-Sum Algorithm (MSA) and a 4-bit fixed-point twos-complement number representation [14],

TABLE III
SIMULATION PARAMETERS.

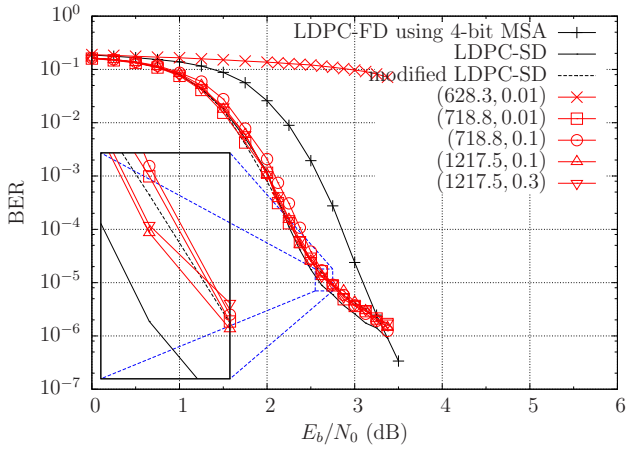
Manufacturer's datasheets	STMicroelectronics 90 nm [57] and FreePDK 45 nm [63]
$(T_{\text{clk}}, 3\sigma/\mu)$ for 90 nm	$(628.3, 0.01)$, $(718.8, 0.01)$, $(718.8, 0.1)$, $(1217.3, 0.1)$ and $(1217.3, 0.3)$
$(T_{\text{clk}}, 3\sigma/\mu)$ for 45 nm	$(760.0, 0.01)$, $(869.5, 0.1)$, $(1171.2, 0.1)$, $(1472.8, 0.1)$ and $(1472.8, 0.3)$
LDPC code	WiMAX LDPC $(1056, 528)$ $(2304, 1920)$ [2]
EM length	$32, 48, 48, 64$ bits for VNs having degree of $d_i = 2, 3, 4, 6$
IM length	$0, 1, 1, 2$ bits for VNs having degree of $d_i = 2, 3, 4, 6$
Max number of decoding cycles	2000
Clock cycles per decoding cycle	1



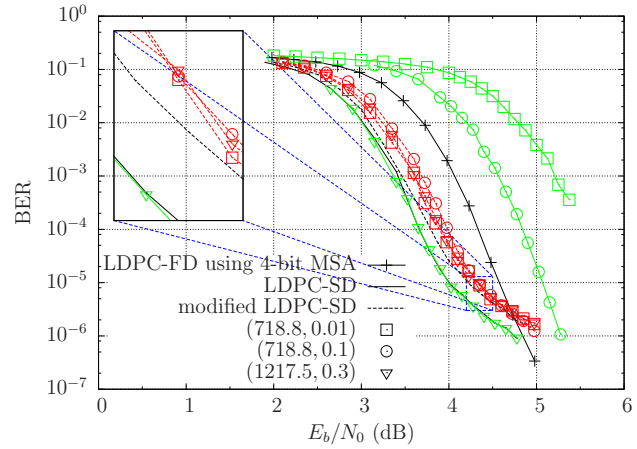
(a) LDPC-SD for decoding (1056,528) code, in the presence of Type I, II and III timing errors.



(b) LDPC-SD for decoding (1056,528) code, in the presence of only Type I timing errors.



(c) Modified LDPC-SD for decoding (1056,528) code, in the presence of Type I, II and III timing errors.



(d) LDPC-SD and modified LDPC-SD for decoding (2304,1920) code, in the presence of Type I, II and III timing errors.

Fig. 19. BER performance of the LDPC-SD and modified LDPC-SD for decoding (1056,528) and (2304,1920) WiMAX LDPC codes, using 90 nm technology.

[18], [20], [21], [64] and for the LDPC-SD in the absence of timing errors. In each case, iterative decoding is continued until convergence to a legitimate LDPC codeword is achieved.

As shown in Fig. 19(a) and 20(a), the BER performance of the LDPC-SD outperforms the 4-bit MSA benchmarker, confirming the high-performance operation of the LDPC-SD in the absence of timing errors. Furthermore, these figures show that when the moderate overclocking of $T_{\text{clk}} = 1217.3$ ps is employed for 90 nm and when $T_{\text{clk}} = 1472.8$ ps is employed for 45 nm, the resultant timing errors do not significantly degrade the performance of the LDPC-SD of [3]. It is only when employing the aggressive overclocking of $T_{\text{clk}} = 718.8$ ps for 90 nm and $T_{\text{clk}} = 1171.2$ ps for 45 nm, that the BER performance of the LDPC-SD becomes degraded by about 1 dB. When employing the aggressive overclocking of $T_{\text{clk}} = 628.3$ ps for 90 nm and $T_{\text{clk}} = 869.5$ ps or $T_{\text{clk}} = 760.0$ ps for 45 nm, the BER performance of the decoder is degraded so severely that it converges perceivably slower. Based on these observations, we consider the LDPC-SD to have a degree of inherent tolerance to timing errors, even though no additional circuitry is employed for detecting or correcting these errors.

2) Tolerance to Timing Error Type I : To further investigate the effects imposed by different types of timing

errors on the BER performance of the LDPC-SD of [3], we repeated the simulations of Fig. 19(a) and 20(a), but with the timing error types II and III turned off. This was achieved by simulating the *occurrence* of timing errors as usual, but only actually *imposing* those timing errors, if they were of Type I. As shown in Fig. 19(b) and 20(b), the resultant simulations yielded BER performances that are within 0.1 dB of those achieved by the benchmarkers. This indicates that it is the occurrence of Type II and III timing errors that causes the significant degradation of the BER in Fig. 19(b) and 20(b), when aggressive overclocking is employed. Furthermore, this demonstrates that the LDPC-SD has an inherent tolerance to Type I timing errors. It is these observations that motivated the modified LDPC-SD of Section V, which is designed to have an increased tolerance to Type II and III timing errors.

B. Improved Timing Error Tolerance

The simulations of Fig. 19(a) and 20(a) were repeated, but employing the modified LDPC-SD instead of the conventional LDPC-SD of [3], in order to characterize the corresponding improvement in BER performance, shown in Fig. 19(c) and 20(c). An additional benchmaker was introduced, namely the modified LDPC-SD in the absence of timing errors. In this case, Fig. 19(c) and 20(c) show that the BER performance of the modified LDPC-SD structure is similar to that of the conventional LDPC-SD of [3], despite having a different EM operation, as described in Section V-A. Furthermore, the BER results of Fig. 19(c) and 20(c) reveal that the modified LDPC-SD has a higher tolerance to timing errors than the conventional LDPC-SD of [3]. As shown in Fig. 19(c), even the aggressive overclocking of $T_{\text{clk}} = 718.8$ ps for 90 nm technology fails to impose a significant degradation on the BER performance of the modified LDPC-SD. Likewise, a significant BER degradation is avoided, when employing the aggressive overclocking of $T_{\text{clk}} = 1171.2$ ps for the 45 nm technology, as shown in Fig. 20(c). It may be also observed from Fig. 19(d) and 20(d), that our modification facilitates a similar BER improvement for the (2304,1902) code of the WiMAX LDPC family, which has a different code rate and length. In summary, it may be observed that the proposed modification eliminates the 1 dB performance degradation that is suffered by the LDPC-SD, when employing aggressive overclocking.

C. Processing Throughput

An alternative comparison can be made by observing that for 90 nm, the BER performance of the modified LDPC-SD recorded for $(T_{\text{clk}}, 3\sigma/\mu) = (718.8, 0.1)$ is similar to that of the conventional LDPC-SD of [3] for $(T_{\text{clk}}, 3\sigma/\mu) = (1217.5, 0.1)$. Therefore, the proposed modification may be deemed to offer a 41% clock period reduction. This corresponds to a 69.4% increase in processing throughput, namely from $R_{\text{b}} = 225.2$ Mbit/s using the conventional LDPC-SD to $R_{\text{b}} = 367.3$ Mbit/s for the modified design, when employing the upper limit of 2000 clock cycles to decode every encoded codeword. However, this upper limit is typically only hit at low channel SNR values. At higher channel SNR values, the stochastic decoding process is able to reach a syndrome of zero using significantly fewer clock cycles. When this happens, the decoding of the current encoded codeword can be stopped early and the decoding of the next codeword can commence without significantly degrading the BER performance. Indeed, Fig. 21 shows that the modified decoder is capable of achieving a processing throughput as high as 3.8 Gbit/s at a channel SNR per bit of $E_b/N_0 = 5$ dB, which is about 60% higher than throughput reported for the conventional LDPC-SD of [3]. Note that this average number of decoding cycles is obtained based on the simulation of 10^6 codewords.

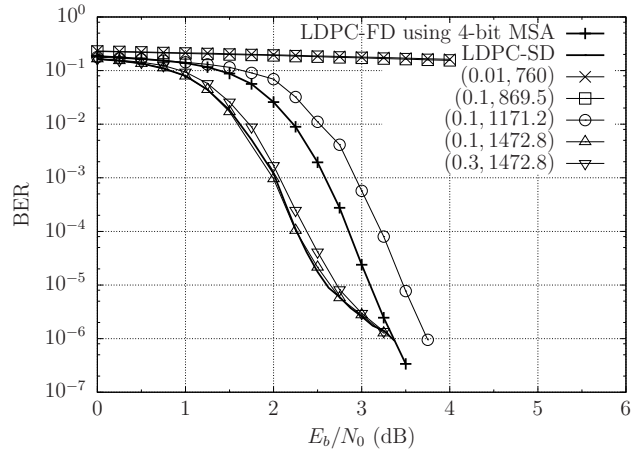
Similarly, for 45 nm, the BER performance of the modified LDPC-SD recorded for $(T_{\text{clk}}, 3\sigma/\mu) = (1171.2, 0.1)$ is similar to that of the conventional LDPC-SD of [3] for $(T_{\text{clk}}, 3\sigma/\mu) = (1472.8, 0.1)$, which corresponds to a 20% clock period reduction. This represents a 25.5% increase in processing throughput, namely from $R_{\text{b}} = 179.3$ Mbit/s using the conventional LDPC-SD of [3] to $R_{\text{b}} = 225.2$ Mbit/s for the modified design.

D. Processing Energy Consumption

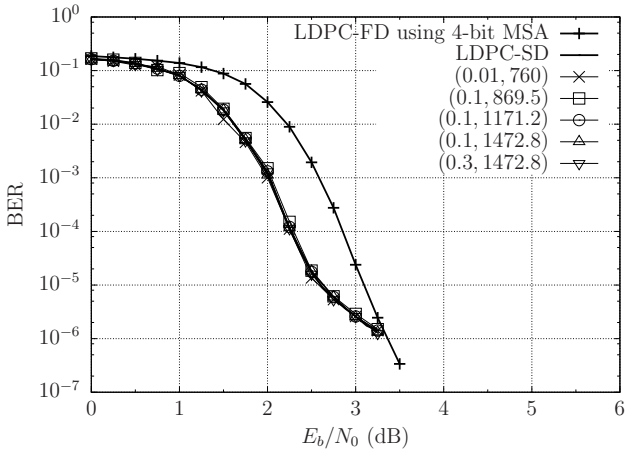
Table IV quantifies the processing energy consumption of individual $d_i = 2, 3, 6$ VNs and $d_j = 6, 7$ CNs, within the conventional LDPC-SD of [3] and the modified LDPC-SD. These results were obtained by using SPICE simulations, when employing various T_{clk}. The STMicroelectronics 90 nm technology [57] and the FreePDK 45 nm technology [63] are designed to work at nominal power supply voltages of 1.0 V and 1.1 V, respectively. The same values of the scaled clock period T_{clk} that were selected and justified in Sections IV and V were used to conduct the simulation for energy consumption estimation. The total LDPC-SD energy consumption was estimated by multiplying the individual VN and CN energy consumptions by the total number of nodes having the corresponding degree in the WiMAX LDPC (1056,528) code. As shown in Table IV, neither overclocking, nor the modification of Section V-A has a significant effect on the energy consumption of the LDPC-SD. Therefore, we can conclude that the attained processing throughput increases of up to 69.4% were achieved without significantly increasing the processing energy consumption.

VII. Conclusions

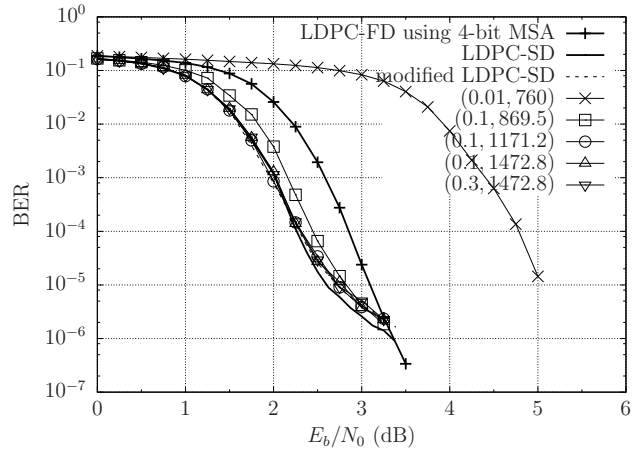
In this paper, we have provided the first comprehensive tutorial on LDPC-SDs and we have proposed a novel design flow for timing-error-tolerant LDPC decoders. Using this design flow, we have demonstrated that LDPC-SDs are capable of exploiting their inherent error correction capability, to correct not only transmission errors, but also timing errors. We have characterized the causes and effects of timing errors within LDPC-SDs by developing a timing error model, which we have validated using SPICE simulations. Drawing upon our findings, we proposed a modified LDPC-SD, having an improved timing error tolerance. In a particular practical scenario, we demonstrated that this modification eliminates the 1 dB performance degradation suffered by the LDPC-SD of [3], which allows the processing throughput to be increased by up



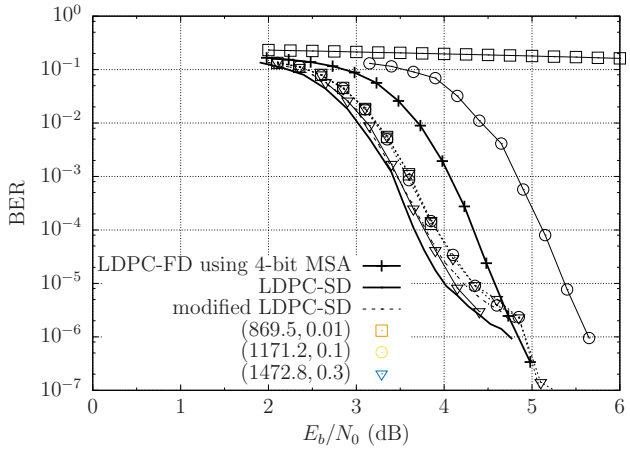
(a) LDPC-SD for decoding (1056,528) code, in the presence of Type I, II and III timing errors.



(b) LDPC-SD for decoding (1056,528) code, in the presence of only Type I timing errors.



(c) Modified LDPC-SD for decoding (1056,528) code, in the presence of Type I, II and III timing errors.



(d) LDPC-SD and modified LDPC-SD for decoding (2304,1920) code, in the presence of Type I, II and III timing errors.

Fig. 20. BER performance of the LDPC-SD and modified LDPC-SD for decoding (1056,528) and (2304,1920) WiMAX LDPC codes, using 45 nm technology.

TABLE IV
ENERGY CONSUMPTION FROM SPICE SIMULATIONS.

		Energy consumption per 2 clock cycles (pJ)						
		ST90 nm, $\mu = 1.0$ V			FreePDK45 nm, $\mu = 1.1$ V			
		$T_{clk} = 628.3$ ps	718.8 ps	1217.5 ps	760 ps	869.5 ps	1171.2 ps	1472.8 ps
CNs	$d_j = 6$	0.297	0.302	0.303	0.262	0.264	0.275	0.286
	$d_j = 7$	0.325	0.332	0.336	0.282	0.287	0.288	0.288
Conventional VNs	$d_i = 2$	1.47	1.48	1.51	1.52	1.55	1.58	1.59
	$d_i = 3$	3.22	3.34	3.38	3.19	3.22	3.26	3.31
	$d_i = 6$	8.25	8.36	8.58	8.36	8.39	8.45	8.65
Conventional	Total	3.83×10^3	3.90×10^3	3.97×10^3	3.84×10^3	3.88×10^3	3.92×10^3	3.99×10^3
Modified VNs	$d_i = 2$	1.53	1.54	1.54	1.43	1.44	1.44	1.44
	$d_i = 3$	3.10	3.32	3.43	3.10	3.13	3.37	3.38
	$d_i = 6$	8.64	8.88	8.90	8.08	8.10	8.38	8.40
Modified	Total	3.89×10^3	4.03×10^3	4.08×10^3	3.70×10^3	3.72×10^3	3.88×10^3	3.88×10^3

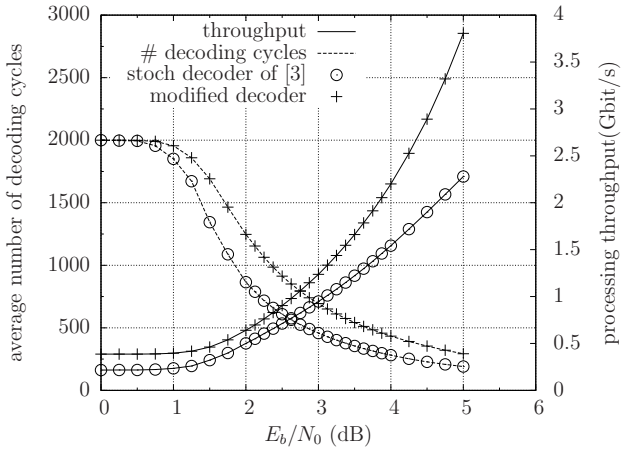


Fig. 21. Processing throughput (information bits per second) and average number of decoding cycles used for decoding the (1056,528) code, at different E_b/N_0, using 90 nm technology, when the overclocking is adopted as $(T_{\text{clk}}, 3\sigma/\mu) = (1217.5, 0.1)$ and $(718.8, 0.1)$, for the LDPC-SD and the modified LDPC-SD, respectively.

to 69.4% in practice, due to the significantly improved tolerance to timing errors. This is achieved without the requirement for additional circuitry or the associated processing energy consumption. Note that since the timing analysis and the timing error model of Sections IV and V are particular to VNs and CNs having specific degrees and are independent of the interconnectivity of the VNs and CNs, our methodology may be readily applied to other LDPC decoders having different block lengths and coding rates. Additionally, our methodology may be applied to other stochastic decoders that rely on EMs, such as the turbo decoder of [65]. Our future work will fabricate timing-error-tolerant channel decoder ASICs and compare practical measurements of their performance with simulations.

REFERENCES

[1] R. Gallager, "Low-density parity-check codes," *IRE Transactions on Information Theory*, vol. 8, pp. 21 –28, Jan. 1962.

[2] IEEE 802.16e. The IEEE 802.16 Working Group. [Online]. Available: http://www.ieee802.org/16/.

[3] S. S. Tehrani, S. Mannor, and W. J. Gross, "Fully parallel stochastic LDPC decoders," *IEEE Transactions on Signal Processing*, vol. 56, pp. 5692–5703, Nov. 2008.

[4] A. Darabiha, *VLSI Architecture for Multi-Gbps Low-Density Parity-Check Decoders*. PhD thesis, University of Toronto, Toronto, ON, Canada, 2008.

[5] S. Ghosh and K. Roy, "Parameter variation tolerance and error resiliency: New design paradigm for the nanoscale era," *Proceedings of the IEEE*, vol. 98, pp. 1718–1751, Nov. 2010.

[6] B. Shim and N. Shanbhag, "Energy-efficient soft error-tolerant digital signal processing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 14, pp. 336–348, April 2006.

[7] R. Abdallah and N. Shanbhag, "Error-resilient low-power Viterbi decoder architectures," *IEEE Transactions on Signal Processing*, vol. 57, pp. 4906–4917, May 2009.

[8] V. C. Gaudet, "Low-power LDPC decoding by exploiting the fault-tolerance of the sum-product algorithm," in *Contemporary Mathematics*, vol. 523, pp. 165–171, Jun. 2010.

[9] M. Alles, T. Brack, and N. Wehn, "A reliability-aware LDPC code decoding algorithm," in *IEEE 65th Vehicular Technology Conference (VTC), Dublin, Ireland*, pp. 1544–1548, Apr. 2007.

[10] M. May, M. Alles, and N. Wehn, "A case study in reliability-aware design: A resilient LDPC code decoder," in *Design, Automation and Test in Europe, Munich, Germany*, pp. 456 –461, Mar. 2008.

[11] D. Ernst, N. S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge, "Razor: a low-power pipeline based on circuit-level timing speculation," in *Proceedings of 36th Annual IEEE/ACM International Symposium on Microarchitecture.*, pp. 7–18, Dec 2003.

[12] R. Uppu, R. Uppu, A. Singh, and A. Chatterjee, "A high throughput multiplier design exploiting input based statistical distribution in completion delays," in *26th International Conference on VLSI Design and 12th International Conference on Embedded Systems (VLSID) , Hyatt Regency, Pune, India*, pp. 109–114, Jan. 2013.

[13] I. Perez-Andrade, X. Zuo, R. Maunder, B. Al-Hashimi, and L. Hanzo, "Analysis of voltage- and clock-scaling-induced timing errors in stochastic LDPC decoders," in *IEEE Wireless Communications and Networking Conference (WCNC), Shanghai, China*, pp. 4293–4298, Apr. 2013.

[14] J. Chen, A. Dholakia, E. Eleftherious, M. P. C. Fossorier, and X. Hu, "Reduced-complexity decoding of LDPC codes," *IEEE Transactions on Communications*, vol. 53, pp. 1288–1299, Aug. 2005.

[15] F. Kschischang, B. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, pp. 498–519, Feb. 2001.

[16] X.-Y. Hu, E. Eleftheriou, D.-M. Arnold, and A. Dholakia, "Efficient implementations of the sum-product algorithm for decoding LDPC codes," in *IEEE Global Telecommunications Conference, San Antonio, USA*, vol. 2, pp. 1036–1036E, Nov. 2001.

[17] D. Oh and K. Parhi, "Min-sum decoder architectures with reduced word length for LDPC codes," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 57, no. 1, pp. 105–115, 2010.

[18] L. Ping and W. K. Leung, "Decoding low density parity check codes with finite quantization bits," *IEEE Communications Letters*, vol. 4, pp. 62–64, Feb. 2000.

[19] G. Montorsi and S. Benedetto, "Design of fixed-point iterative decoders for concatenated codes with interleavers," *IEEE Journal on Selected Areas in Communications*, vol. 19, pp. 871–882, May 2001.

[20] T. Zhang, Z. wang, and K. Parhi, "On finite precision implementation of low density parity check codes decoder," in *IEEE International Symposium on Circuits and Systems, Sydney, Australia*, vol. 4, pp. 202–205, May 2001.

[21] X. Zuo, R. Maunder, and L. Hanzo, "Design of fixed-point processing based LDPC codes using EXIT charts," in *2011 IEEE Vehicular Technology Conference (VTC Fall), San Francisco, CA, USA*, pp. 1–5, Sep. 2011.

[22] L. Varshney, "Performance of LDPC codes under faulty iterative decoding," *IEEE Transactions on Information Theory*, vol. 57, no. 7, pp. 4427–4444, 2011.

[23] S. Yazdi, C.-H. Huang, and L. Dolecek, "Optimal design of a gallager B noisy decoder for irregular LDPC codes," *IEEE Communications Letters*, vol. 16, pp. 2052–2055, Dec. 2012.

[24] S. Tabatabaei Yazdi, H. Cho, and L. Dolecek, "Gallager B decoder on noisy hardware," *IEEE Transactions on Communications*, vol. 61, pp. 1660–1673, May 2013.

[25] C.-H. Huang, Y. Li, and L. Dolecek, "Gallager B LDPC decoder with transient and permanent errors," *IEEE Transactions on Communications*, vol. 62, pp. 15–28, Jan. 2014.

[26] V. Gaudet and A. Rapley, "Iterative decoding using stochastic computation," *Electronics Letters*, vol. 39, pp. 299–301, Feb. 2003.

[27] S. Tehrani, A. Naderi, G.-A. Kamendje, S. Hemati, S. Mannor, and W. Gross, "Majority-based tracking forecast memories for stochastic LDPC decoding," *IEEE Transactions on Signal Processing*, vol. 58, no. 9, pp. 4883–4896, 2010.

[28] S. Tehrani, W. Gross, and S. Mannor, "Stochastic decoding of LDPC codes," *IEEE Communications Letters*, vol. 10, pp. 716–718, Oct. 2006.

[29] B. Gaines, *Advances in Information Systems Science*, ch. 2. New York: Plenum, 1969.

[30] N. R. Shanbhag, R. A. Abdallah, R. Kumar, and D. L. Jones, "Stochastic computation," in *47th ACM/IEEE Design Automation Conference (DAC)*, pp. 859 –864, June 2010.

[31] R. Ahmadi and F. Najm, "Timing analysis in presence of power supply and ground voltage variations," in *International Conference on Computer Aided Design*, pp. 176–183, Nov 2003.

[32] M. Alioto, G. Palumbo, and M. Pennisi, "Understanding the effect of process variations on the delay of static and domino logic," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 18, pp. 697–710, May 2010.

[33] M. Gupta, J. Oatley, R. Joseph, G.-Y. Wei, and D. Brooks, "Understanding voltage variations in chip multiprocessors using a distributed power-delivery network," in *Design, Automation Test in Europe Conference Exhibition*, pp. 1–6, April 2007.

[34] S. Beer, J. Cox, R. Ginosar, T. Chaney, and D. Zar, "Variability in multistage synchronizers," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, pp. 2957–2969, Dec 2015.

[35] N. Ahmed, M. Tehranipoor, and V. Jayaram, "A novel framework for faster-than-at-speed delay test considering IR-drop effects," in *IEEE/ACM International Conference on Computer-Aided Design*, pp. 198–203, Nov 2006.

[36] S. Tehrani, S. Mannor, and W. Gross, "Survey of stochastic computation on factor graphs," in *37th International Symposium on Multiple-Valued Logic (ISMVL)*, p. 54, May 2007.

[37] S. S. Tehrani, C. Winstead, W. J. Gross, S. Mannor, S. L. Howard, and V. C. Gaudet, "Relaxation dynamics in stochastic iterative decoders," *IEEE Transactions on Signal Processing*, vol. 58, pp. 5955–5961, Nov. 2010.

[38] A. Naderi, S. Mannor, M. Sawan, and W. J. Gross, "Delayed stochastic decoding of LDPC codes," *IEEE Transactions on Signal Processing*, vol. 59, pp. 5617–5626, Nov. 2011.

[39] G. Sarkis and W. J. Gross, "Efficient stochastic decoding of non-binary LDPC codes with degree-two variable nodes," *IEEE Communications Letters*, vol. 16, pp. 389–391, March 2012.

[40] A. Rapley, V. Gaudet, and C. Winstead, "On the simulation of stochastic iterative decoder architectures," in *Canadian Conference on Electrical and Computer Engineering*, p. 1868, May 2005.

[41] S. Tehrani, S. Mannor, and W. Gross, "An area-efficient FPGA-based architecture for fully-parallel stochastic LDPC decoding," in *IEEE Workshop on Signal Processing Systems*, pp. 255–260, 2007.

[42] C. Winstead, V. Gaudet, A. Rapley, and C. Schlegel, "Stochastic iterative decoders," in *Proceedings of International Symposium on Information Theory*, p. 1116, Sep 2005.

[43] V. C. Gaudet and W. J. Gross, "Switching activity in stochastic decoders," in *40th IEEE International Symposium on Multiple-Valued Logic (ISMVL)*, p. 167, May 2010.

[44] N. Onizawa, W. J. Gross, T. Hanyu, and V. C. Gaudet, "Clockless stochastic decoding of low-density parity-check codes," in *IEEE Workshop on Signal Processing Systems (SiPS)*, pp. 143–148, Oct. 2012.

[45] N. Onizawa, V. C. Gaudet, T. Hanyu, and W. J. Gross, "Asynchronous stochastic decoding of Low-Density Parity-Check codes," in *42nd IEEE International Symposium on Multiple-Valued Logic (ISMVL)*, pp. 92–97, May 2012.

[46] K.-L. Huang, V. Gaudet, and M. Salehi, "A Markov chain model for edge memories in stochastic decoding of LDPC codes," in *45th Annual Conference on Information Sciences and Systems (CISS)*, pp. 1–4, March 2011.

[47] K.-L. Huang, V. Gaudet, and M. Salehi, "Output decisions for stochastic LDPC decoders," in *48th Annual Conference on Information Sciences and Systems (CISS)*, pp. 1–5, March 2014.

[48] R. Tanner, "A recursive approach to low complexity codes," *IEEE Transactions on Information Theory*, vol. 27, pp. 533–547, Sep. 1981.

[49] H.-A. Loeliger, "An introduction to factor graphs," *IEEE Signal Processing Magazine*, vol. 21, pp. 28–41, Feb. 2004.

[50] T. Richardson, M. Shokrollahi, and R. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE transcations on Information Theory*, vol. 47, pp. 619–637, Feb. 2001.

[51] J. P. H. A. Alaghi, "Survey of stochastic computing," *ACM Transactions on Embedded Computing Systems*, 2012.

[52] J. Zhao, F. Zarkeshvari, and A. Banihashemi, "On implementation of min-sum algorithm and its modification for decoding low-density parity-check (LDPC) codes," *IEEE Transactions on Communications*, vol. 53, pp. 549–554, Apr. 2005.

[53] E. Yeo, B. Nikolic, and V. Anantharam, "Architectures and implementations of low-density parity check decoding algorithms," in *Midwest Symposium on Circuits and Systems*, vol. 3, pp. III–437, Aug. 2002.

[54] W. J. Gross, V. C. Gaudet, and A. Milner, "Stochastic implementation of LDPC decoders," in *Conference Record of the Thirty-Ninth Asilomar Conference on Signals, Systems and Computers*, pp. 713–717, Oct. 2005.

[55] T. Richardson and R. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Transactions on Information Theory*, vol. 47, pp. 599–618, Feb. 2001.

[56] M. Mansour and N. Shanbhag, "High-throughput LDPC decoders," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 11, no. 6, pp. 976–996, 2003.

[57] STMicroelectronics, *User Manual & Databook*, May 2006.

[58] X. Qi, S. Lo, Y. Luo, A. Gyure, M. Shahram, and K. Singhal, "Simulation and analysis of inductive impact on VLSI interconnects in the presence of process variations," in *Proceedings of the IEEE Custom Integrated Circuits Conference*, pp. 309–312, 2005.

[59] T. Enami, S. Ninomiya, and M. Hashimoto, "Statistical timing analysis considering spatially and temporally correlated dynamic power supply noise," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 4, pp. 541–553, 2009.

[60] F. Martorell, M. Pons, A. Rubio, and F. Moll, "Error probability in synchronous digital circuits due to power supply noise," in *International Conference on Design Technology of Integrated Systems (DTIS) in Nanoscale Era*, pp. 170–175, Sep. 2007.

[61] Synopsys, "HSPICE: The gold standard for accurate circuit simulation," 2014. [Online]. Available: http://www.synopsys.com/Tools/Verification/AMSVerification/CircuitSimulation/HSPICE/Documents/hspice_ds.pdf.

[62] S. Zhong, S. Khursheed, B. Al-Hashimi, and W. Zhao, "Efficient variation-aware delay fault simulation methodology for resistive open and bridge defects," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, pp. 798–810, May 2014.

[63] A variation-aware 45nm design flow for the Semiconductor Research Corporation, jointly developed by Oklahoma State University and North Carolina State University [Online]. Available: http://vlsiarch.ecen.okstate.edu/flows/OSUFreePDK45/.

[64] G. Masera, F. Quaglio, and F. Vacca, "Finite precision implementation of LDPC decoders," *IEE Proceedings-Communications*, vol. 152, pp. 1098–1102, Dec. 2005.

[65] Q. T. Dong, M. Arzel, C. Jego, and W. J. Gross, "Stochastic decoding of turbo codes," *IEEE Transactions on Signal Processing*, vol. 58, pp. 6421–6425, Dec. 2010.

Xin Zuo received the B.Eng. degree in Telecommunication Engineering from the Beijing University of Posts and Telecommunications (BUPT), Beijing in 2008 and the M.Sc. degree with distinction in Radio Frequency Communication Systems from the University of Southampton, Southampton, in 2009. He is currently working towards the Ph.D. degree in the Wireless Communications Group in the University of Southampton. His research interests include the error-tolerant ASIC system design and the implementation of fully-parallel channel decoders.

Isaac Perez-Andrade was awarded a BSc in Electronics and Computer Science from ITESM, Mexico, in 2009 and an MSc in System-on-Chip from the University of Southampton, UK, in 2011. He is currently working towards his PhD degree at the University of Southampton. His research interests include low-power error-tolerant VLSI architectures for wireless communications and iterative decoding.

Robert G. Maunder has studied with Electronics and Computer Science, University of Southampton, UK, since October 2000. He was awarded a first class honors BEng in Electronic Engineering in July 2003, as well as a PhD in Wireless Communications in December 2007. He became a lecturer in 2007 and an Associated Professor in 2013. Rob's research interests include joint source/channel coding, iterative decoding, irregular coding and modulation techniques. For further information on this research, please refer to http://users.ecs.soton.ac.uk/rm.

Bashir M. Al-Hashimi (M'99-SM'01-F'09) is a Professor of Computer Engineering and Dean of the Faculty of Physical Sciences and Engineering at University of Southampton, UK. He is ARM Professor of Computer Engineering and Co-Director of the ARM-ECS research centre. His research interests include methods, algorithms and design automation tools for energy efficient of embedded computing systems. He has published over 300 technical papers, authored or co-authored 5 books and has graduated 31 PhD students.

Lajos Hanzo (FREng, FIEEE, FIET, Eurasip Fellow, RS Wolfson Fellow) received his degree in electronics in 1976 and his doctorate in 1983. In 2009 he was awarded the honorary doctorate "Doctor Honoris Causa" by the Technical University of Budapest. During his 38-year career in telecommunications he has held various research and academic posts in Hungary, Germany and the UK. Since 1986 he has been with the School of Electronics and Computer Science, University of Southampton, UK, where he holds the chair in telecommunications. He has successfully supervised about 100 PhD students, co-authored 20 John Wiley/IEEE Press books on mobile radio communications totaling in excess of 10,000 pages, published 1,500+ research entries at IEEEXplore, acted both as TPC and General Chair of IEEE conferences, presented keynote lectures and has been awarded a number of distinctions. Currently he is directing a 100-strong academic research team, working on a range of research projects in the field of wireless multimedia communications sponsored by industry, the Engineering and Physical Sciences Research Council (EPSRC) UK, the European 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 and he offers a range of industrial courses. Lajos is a Fellow of the Royal Academy of Engineering, of the Institution of Engineering and Technology, and of the European Association for Signal Processing. He is also a Governor of the IEEE VTS. During 2008-2012 he was the Editor-in-Chief of the IEEE Press and a Chaired Professor also at Tsinghua University, Beijing. He has 22,000+ citations. For further information on research in progress and associated publications please refer to http://www-mobile.ecs.soton.ac.uk.