CPE: Codeword Prediction Encoder

***Abstract*—The continuous scaling of the transistor length has resulted in significant increase in the soft error rate in combinational circuits. As a result, fault tolerant techniques that improve circuit reliability are the need of the hour. In the context of communication and storage, the study of novel techniques for reliable data transmission under unreliable hardware is an increasing priority. Traditional fault tolerant techniques analyze the circuit reliability issue from a static point of view neglecting the dynamic errors. This paper introduces a novel reliability driven fault tolerant methodology known as Codeword Prediction Encoder (CPE) for reliable LDPC encoding by augmenting extra logic to correct the dynamic errors introduced during the encoding process. A CAD framework known as CPE simulator is developed providing a unified platform that comprises of novel encoder and fault tolerant LDPC decoders. Our experiments on a set of encoders with different coding rates and different decoders indicate that the proposed framework can correct all errors under certain scenarios. On average, about 10K improvement in Soft Error Rate(SER) reduction is obtained.**

***Keywords*—Low Density Parity Check(LDPC), Fault tolerance, Forward Error Correction (FEC), Reliability, Soft errors.**

I. INTRODUCTION

Low density parity check (LDPC) codes are known to provide excellent error correction performance that closely approaches the Shannon capacity of noisy transmission channels [1], [2]. As a result, they have been adopted in many current and next generation wireless protocols such as IEEE 802.16E (WiMAX), IEEE 802.11 (Wi-Fi), and DVB-S2/T2/C2 standards, besides many other applications. With the shrinking advanced CMOS devices, transient error rate traditionally associated with memories is increasing there by reduced reliability of the fundamental logic gates [3], [4]. As a consequence, in the future systems of communication and storage, errors may not only come from the transmission channels, but also from the faulty hardware. Thus, the study of novel techniques for reliable data transmission using unreliable hardware is an increasing priority. Important work to cross the field of circuit design with the knowledge of error correction theory has been done by Taylor [5], [6] that used LDPC codes to build fault tolerant storage and computation architectures on unreliable systems. Following Taylor's approach, LDPC decoders on unreliable hardware have been widely investigated [7], [8]. As a main result, it was shown that when some of the decoder parameters (number of quantization levels, channel value, etc.) are carefully chosen, LDPC decoders are naturally robust to faulty hardware, with no need for additional circuitry. Unfortunately, it was also shown that LDPC encoders completely fail when they are built from unreliable gates [9]. The focus of the current work is thus on constructing reliable LDPC encoders built from unreliable gates.

This paper introduces a novel reliability driven fault tolerant methodology known as Codeword Prediction Encoder (CPE) for reliable LDPC encoding by augmenting extra logic to correct the errors introduced during the encoding process. The approach presented here can be seen as an expansion of the Check Symbols Generation [10] and the Parity Prediction Function [11], [12], where circuitry is added to a combinatorial network to generate extra bit to ensure parity. We formalize these approaches and extend them to take full advantage of the power of error correction codes to enable correction of the faults, and not just detection. The CPE simulator provides a unified platform which comprises of novel encoder and fault tolerant LDPC decoders.We employed encoders using regular LDPC codes with different column weights for the parity check matrix, namely $d_v = 3$ and $d_v = 4$, and different coding rates, namely $r = 1/2$ and $r = 3/4$. Also, different state-of-the-art reliability enhanced LDPC decoding mechanisms like Self-Corrected Min-Sum (SCMS), and Gallager B with Extended Alphabet(Gal-B) were used. Simulations results prove that it is possible to retrieve the original information by employing particular configurations of these encoders and decoders. In general, output BER is reduced by upto 10K times by adopting CPE mechanism as compared to transmitting data directly.

The remainder of the paper is organized as follows. Section II provides a general overview of LDPC and their decoding algorithm in the generic form. Section III presents the proposed CPE methodology with a detailed overview of the faulty encoder and the decoder. Section IV presents the CAD flow employed to test the methodology. Section V presents the experimental results based on the evaluations performed on different set of decoders. Section VI concludes the paper.

II. LDPC CODES AND ERROR MODELS

We consider the data transmission scheme depicted in Fig. 1. In this scheme, a binary information sequence u of length k has to be transmitted through a noisy channel. can be protected by adding some redundancy in the transmitted data. As u has to be perfectly recovered at the output of the channel, the information sequence has to be protected by adding some redundancy in the transmitted data. The data protection can be done with an LDPC code that encodes the information sequence u into a codeword x of length $n > k$. At the output of the channel, the received sequence y is passed through an LDPC decoder that aims at reconstructing u.

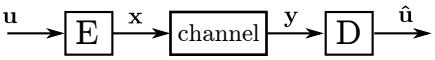


Fig. 1. Data transmission scheme

In this section, we first describe LDPC codes as well as the encoding and decoding operations. We then introduce the error model we consider for unreliable gates that constitute the encoder and the decoder.

A. LDPC codes

Low-density parity check (LDPC) codes are a class of linear block codes invented by Gallager [1]. An LDPC code is defined by its binary parity check matrix H of size $m \times n$, see Fig. 2 (a). A binary vector x of length n is a codeword of the LDPC code if it satisfies

$$Hx^T = 0, \tag{1}$$

where T is the transpose operator. For LDPC codes, the parity check matrix H is sparse, *i.e.*, it contains only a few non-zero components. In the following, we will denote by d_v and d_c the number of 1's in each row and in each column of H, respectively. Tanner also introduced a graphical representation of LDPC codes as shown in Fig. 2 (b). Tanner graphs are bipartite, which means that the nodes of the graph are separated into two distinct sets. The first set contains n variable nodes and the second set contains m check nodes. Edges are only connecting nodes of two different types, and there is an edge between variable node v and check node c if and only if there is a 1 at the corresponding position in the parity check matrix. At the end, the matrix representation of the LDPC code is used for encoding, while the graph representation is used for decoding, as we now describe.

B. LDPC Encoding and Decoding

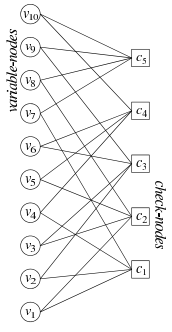
Once the LDPC parity check matrix H is fixed, it remains to construct the corresponding encoder that transforms any information sequence u into a codeword x that satisfies (1). From H, one can construct a generator matrix G of size $k \times n$, where $k = n - m$, that verifies $HG^{\mathrm{T}} = 0$. The encoding operation can then be realized from the generator matrix as

$$x = uG \tag{2}$$

Several solutions have been proposed to construct a generator matrix G from the parity check matrix H, see [13] for a review. Most of the usual solutions consider systematic encoding, for which the codeword x = $[u, p]$contains both the information sequence u and m parity bits given by p. In this case, the left hand side of the generator matrix G is the identity matrix of size $k \times k$.

$$H = \begin{array}{c} \begin{array}{cccccccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & v_8 & v_9 & v_{10} \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} \end{array} \begin{array}{c} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{array}$$

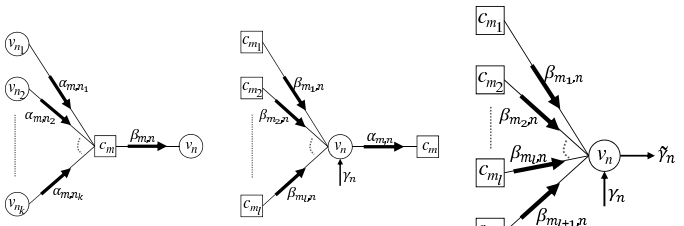
(a) parity check matrix



(b) Tanner graph

Fig. 2. LDPC Codes

After encoding, the codeword x is transmitted on the channel, which outputs y. If the channel is a Binary Symmetric Channel (BSC), the received word can be written as $y = x+e$, where e is the binary error pattern, and the sum above is computed modulo 2. From condition (1), one can compute the syndrome $z = Hx^T + Hy^T = He^T$. The decoding problem consists of finding the most probable vector e that explains the observation of the syndrome z. For LDPC codes, decoding can be implemented by message passing algorithms that exchange messages between variable-nodes and check-nodes, as shown in Fig. 3. Several LDPC decoders (Gallager B, Min-Sum, Belief Propagation, etc.) have been proposed, which consist of different processing rules and simplifications of the message-passing algorithm.



(a) Check-to-variable messages

(b) Variable-to-check messages

(c) Posteriori information

Fig. 3. LDPC message computation

LDPC codes were initially introduced under the assumption of reliable hardware. Here, in order to analyze the performance of LDPC codes under unreliable hardware, we introduce the error model we consider for the unreliable gates that are used in the encoder and in the decoder.

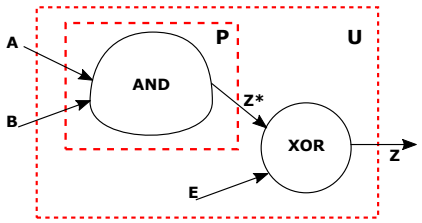


Fig. 4. Gate Error Model.

C. Gate Error Model

We represent unreliable gates by following the Von Neumann model. A Von Neumann erroneous gate is modeled as an ideal logic gate cascaded with an error injecting XOR gate. The error-injecting XOR gate determines the stochastic error behavior by toggling the gate output with a pre-defined probability. As an example, Fig. 4 graphically represents an unreliable AND gate. The noise variable E takes value 1 with a given probability p_g that represents the gate error probability. Under this model, it was shown that LDPC decoders built from unreliable gates are naturally robust to errors, with no need for additional circuit protection [8], [14]. Unfortunately, it was also shown that most of the standard LDPC encoders completely fail when they are built from unreliable gates [9]. Thus, we propose a novel robust LDPC encoding solution that consists of computing extra parity bits, as we now describe.

III. CODEWORD PREDICTION ENCODER (CPE)

Consider the systematic encoding operation described by (2). Fig. 5 represents the encoding error probability P_e with respect to the gate error probability p_g for various LDPC codes, with $k = 1000$, $d_v = 3$ and $r = 1/4, 2/5, 1/2, 5/8$, respectively. We see that the encoding error probability does not depend much on the coding rate. Fig. 5 also shows that the encoding error probability is dramatically increased with respect to the gate error probability p_g. For example, a gate error probability $p_g = 10^{-4}$ will give an encoding error probability $P_e = 10^{-2}$, which represents an increase by a factor 100. As a result, the high encoding error probability is going to combine with the channel noise, and at the end, the decoder will not be able to recover the original information sequence u from y. Hence there is a need to drastically reduce the encoding error probability before transmission on the channel.

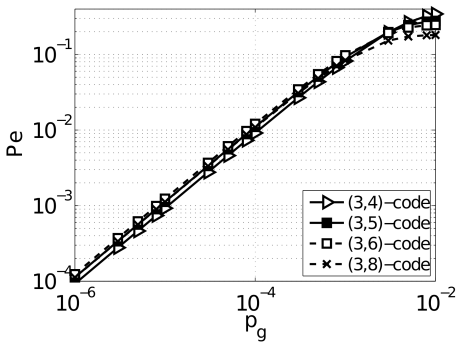


Fig. 5. Encoding error probability P_e with respect to gate error probability p_g. In the legend, the $(3, x)$-code represents the code with $d_v = 3$ and $d_c = x$.

As described in Fig. 6, a first solution consists of passing the noisy codeword $\tilde{x}$ through an LDPC decoder before channel transmission, in order to eliminate the encoding errors. This solution will be efficient only if the gate error probability lead to an encoding error probability lower than the correction capability of the LDPC code. As the encoding error probability can be high, we would like to go further than the correction capability of the LDPC code. We thus propose the Codeword Prediction Encoder (CPE) approach depicted in Fig 7 which consists of two methodologies: non-systematic and systematic.

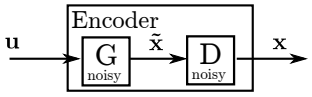
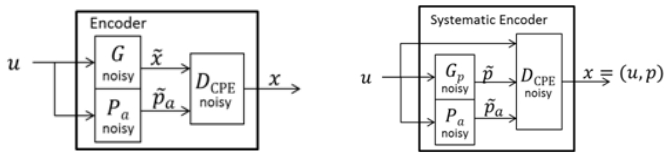


Fig. 6. First encoding solution

In case of non-systematic encoding as described in Fig 7(a), in addition to the parity bits p contained in $\tilde{x}$, we now compute m_a extra parity bits $\tilde{p_a}$ from u. The vector $\tilde{x_a} = [\tilde{x}, \tilde{p_a}]$ is called the augmented codeword. Before channel transmission, $\tilde{x_a}$ is passed through a different LDPC decoder, denoted by D_{CPE}, in order to eliminate the encoding errors. The extra parity bits $\tilde{p_a}$ serve only to help eliminate the encoding errors, and, after decoding, only $\tilde{x}$ is transmitted through the channel. Thus, both the LDPC code that produces $\tilde{x}$ and the one that produces $\tilde{x_a}$ have to lead to good decoding performance.



(a) Non-systematic Encoding *(b)* Systematic Encoding

Fig. 7. The CPE approach

The CPE approach for systematic encoding is illustrated in Fig 7(b). We denote by G_p the G sub-matrix of size $n-k \times k$, corresponding to parity bits. Thus, x= $[u, p]$, and the parity bits p can be computed by $p = u \cdot G_p$. In this case only the parity bits p can be affected by gate errors, while the data bits u are assumed to be error free. We also denote the circuit composed either by G and P_a (non-systematic case) or by G_p and P_a (systematic case) as G_{CPE}

The D_{CPE} decoder used at the encoding side makes use of both the original parity bits $\tilde{p}$ and of the extra parity bits $\tilde{p_a}$ to eliminate the encoding errors. If the LDPC codes are constructed carefully, this strategy will result in increased correction capabilities for the augmented codeword $\tilde{x_a}$ compared to the initial codeword $\tilde{x}$. For good code construction, an important condition is that the extra parity bits $\tilde{p_a}$ are independent of the original parity bits p, which means that $\tilde{p}$ and $\tilde{p_a}$ are computed from different combinations of bits from u. To guarantee the independence while insuring good decoding performance for both $\tilde{x_a}$ and $\tilde{x}$, we consider split-extended construction introduced in [15]. Precisely, the additional parity bits $\tilde{p_a}$ are computed in the same way as the "extended bits" in [15], and the D_{CPE} decoder utilizes the split-extended parity-check matrix defined therein.

At the end, the performance of the proposed CPE approach will depend on the choice of the code (rate, degrees, etc.) and of the considered LDPC decoder. In the following, before presenting the evaluation results for different codes and decoders, we proceed to describe how we implement the CPE methodology.

IV. CPE SIMULATOR & CAD AUTOMATION

The CPE simulator has been developed to validate the proposed CPE methodology and evaluate its performance. The CPE simulator automates the process of simulating the standard data transmission over noisy channels with circuits build using faulty error prone gates. Fig. 8 describes the complete CAD flow of the tool. It comprises of 2 main steps, namely: i) Pre-Processing ii)Testing using CPE simulator. This section describes the two steps.

A. Pre-Processing

The CPE simulator accepts all the input files namely, the two encoders composing the G_{CPE} encoder, and the LDPC generator matrix. A number of scripts were developed in order to perform pre-processing of all the input files before they can be run through the CPE simulator. In particular, the traditional verilog netlists that describe the encoding circuits have to be converted into internal proprietary format that is easy to parse by the CPE simulation engine. As a key point to understand the

whole process, we first present the internal netlist proprietary format that we have employed.

B. Netlist Format

Any gate level synthesized representation of digital circuit is represented as collection of gates commonly referred to as netlist. Since CPE simulator is a tool completely developed in C++, it cannot understand the terminology of gates. Hence, we have developed an internal proprietary format to represent the verilog netlists. Each gate in the circuit is converted into a node within the internal format. The following convention is used to represent all possible gates: 0 = NOT, 1 = AND, 2 = OR, 3 = XOR, 4 = NAND, 5 = NOR, 6 = XNOR A node X is called a predecessor of Y if the output of X is an input of Y (in graph terminology, there is a directed edge from X to Y). In this case, Y is said to be a successor of X. The indegree of a node is defined as the number of its predecessors input nodes must have indegree equal to zero The outdegree of a node is defined as the number of its successors output nodes must have outdegree equal to zero. In simple circuit design terminology, indegree and outdegree correspond to fan-in and fan-out of the gates. Nodes of type 0 must be of indegree = 1. Nodes of type 2-6 must be of indegree ≥ 2.

A number of consistency checks are performed in order to make sure the netlists adhere to the syntax. All the input nodes (numbered from 0 to $N_{\text{inputs}} - 1$) are checked to have indegree zero while the output nodes (numbered from N_{inputs} to N_{inputs} +N_{outputs}-1) must have outdegree zero. The processing order of internal and output nodes, i.e. which nodes must be processed in the first stage, which nodes must be processed in the second stage, etc is precomputed to facilitate the CPE simulator. Nodes processed in the first stage are those whose all predecessors are input nodes. Nodes processed in stage I ($I \geq 2$) are those whose predecessors are either input nodes or have been processed during stages 1, ..., I. If a processing order cannot be found, consistency check error is popped and the simulation is killed citing error in netlist files.

The internal format of circuit representation allows simulating the netlist in C++ through the CPE simulator, as we now describe.

C. CPE Simulator

In the CPE simulator itself, a source module generates the pseudo random input vectors. The output of the source module goes into the two encoder modules which propagates the faults through the circuits. The output of the encoders along with original databits is used by the D_{CPE} decoder module to generate the codeword that can be transmitted over the channel. Finally, output from the decoder goes into the BER/FER estimation module where it is compared to the original input generated by the SRC module.

In the CPE simulator, errors are injected by flipping the output of each gate with a pre defined error probability. For simplicity purpose, the same value of probability is used for all the gates, irrespective of their type or their position within the graph. The methodology employed to insert faults is called as "Gate output probabilistic mutant"- it alters the gate output with a given probability.

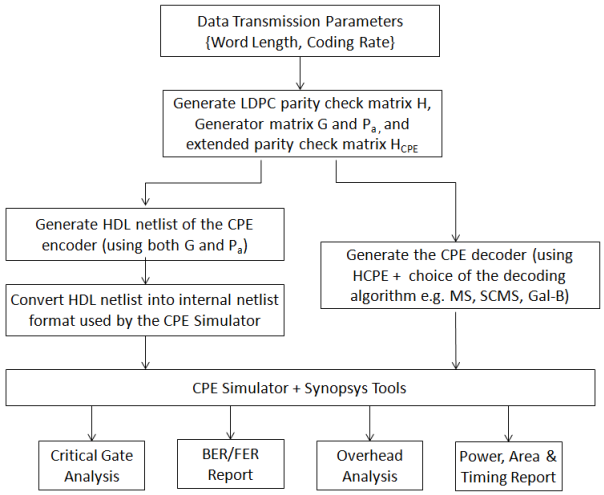


Fig. 8. The CPE CAD flow

Some particular gates of the circuit may be critical, in the sense that injecting only one error at the output of such gates may result in a very large number of errors at the output of the circuit. In the following, we explain how to identify and protect these gates by introducing the notion of Criticality Threshold (CT).

D. Criticality Threshold

A gate is critical when the output of the gate is propagated to a large number of outputs of the circuit. Such error event is characterized by a very high number of errors on the output of the circuit and causes decoding failures with very high probability. In order to identify the critical gates at the design time, we use the graphical description of the netlist introduced above, and proceed as described below.

The criticality degree of a node X, denoted by cdeg(X), is defined as the number of output nodes to which X is connected by at least one path. Thus, injecting an error in node X, may produce at most cdeg(X) errors on the output. In our simulations, we fix a criticality threshold (CT): Nodes X with cdeg(X) $>$ CT are considered to be "protected" (e.g. by increasing area), so as to make then reliable (error-free). Hence, errors are injected only in nodes X with cdeg(X) $<$ CT. For example, fixing CT = 5, infers that errors are injected only in those nodes that are connected to less than 5 output nodes (cdeg(X) $<$ 5) A particular case is CT $= -1$, which means that all nodes are error-prone (no "protected" nodes).

Now that we have presented the CPE simulator, we proceed to describe the experimental results we obtained.

V. Experimental Results

A number of encoders have been simulated using the CPE simulator employing different decoding algorithms. Two scenarios are under investigation: 1. Both encoder and decoder are assumed to be error prone. 2. Only the encoder block is faulty. The first scenario is the most generic and assumes both "encoder" and "decoder" to be fault prone. Given such situation only LDPC codes are usable for the coding schemes, the only known ECC for which fault tolerant decoder

exist. But, we also consider the "perfect decoder" case that presents an evident asymmetry between the "encoder" and the "decoder". This allows one to use coding schemes other than Low Density Parity Check Codes as ECC codes are not available for faulty decoder. From the encoder point of view, four different configurations are employed. Thus, we consider regular LDPC codes with different column weights for the parity check matrix, namely $d_v = 3$ and $d_v = 4$, and different coding rates, namely $r = 1/2$ and $r = 3/4$. From decoder perspective, we have employed three state-of-the-art reliability enhanced LDPC decoders within the CPE CAD flow: Min-Sum (MS), Self-Corrected Min-Sum (SCMS), and Gallager B with Extended Alphabet(Gal-B). Next, we present simulations results of the CPE scheme for different scenarios. Due to lack of space, we focus mainly on the symmetric scenario namely, both the encoder and the decoder are faulty. For the sake of completion, Fig. 12 presents the CPE performance assuming the decoder to be perfect.

TABLE I. CRITICAL GATE COUNT FOR DIFFERENT ENCODING SCHEMES

Encoder	G_{CPE} Node Count	CT=10	CT=20	CT=50
dv3-r12	44399	3373	1844	833
dv3-r34	28182	2288	1240	537
dv4-r12	45175	3424	1851	824
dv4-r34	27167	2112	1183	488

A. Critical Nodes

As defined in the previous section, criticality degree of a gate is defined as the number of erroneous outputs generated when the gate is in error (assuming that all the other gates are error-free). Tab. I lists the count of total & critical nodes within the four encoding schemes. The encoding scheme parameters are given in the first column while column two provides the count of total nodes within the G_{CPE} encoder. The following three columns list the count of critical nodes when critical threshold CT is set to 10, 20, and 50 respectively. Critical nodes are the ones which generate maximum number of errors on the output nodes of the encoder. They have to be safeguarded from external aggressions that would toggle the output value resulting in errors. We are currently looking at solutions to turn the critical gates into always reliable. One possibility would be to use modular redundancy for these gates which means that each gate is repeated N times (say 3 times) and a majority logic gate decides the output value. The other alternative is to make sure we define a different voltage island for all these critical gates so that they are powered up by higher voltage.

As expected, lower the value of critical threshold, higher the number of critical nodes within the encoder. As a tradeoff, a lower critical threshold is also expected to lead to lower encoding error probability. To illustrate this, we employed an encoder with $r = 3/4$ and $d_v = 4$ and D_{CPE} was set to Min-Sum model. As depicted in Fig. 9, the output BER value reduces with the critical threshold values. It infers that more the number of nodes safe guarded from possible soft errors, higher the possibility of retrieving the original information.

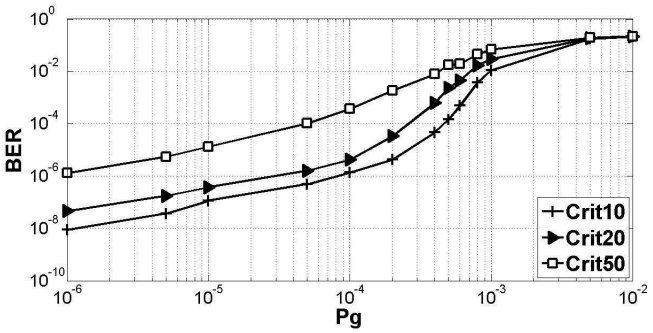


Fig. 9. Critical Threshold impact on Output BER

B. Impact of Decoder Configuration

We benchmark the reliability enhanced LDPC decoders with respect to their performance as well as their ability to effectively deal with the circuit fault-induced probabilistic behavior. For faulty decoders, we assume that the output of every variable and check node function computation is flipped with a probability $p = 10^{-3}$. For non-binary message alphabets, flipping the output value means that a value different from the correct one is selected uniformly at random from the alphabet. Fig. 10 highlights the output BER values for different faulty decoders and the default non-CPE approach when the critical threshold is set to 20. The encoder employed in this particular case has the following parameters $r = 3/4$ and $d_v = 3$. Clearly, SCMS and MS decoders provide the best performance by reducing the error rates to upto 10K times better than the default encoder. Gal-B provides upto 100 times improvement in terms of error correction. It also illustrate the performance of CPE compared to the default encoding mechanism. For example, for CT $= 10$ and for a gate error probability P_g = $1e^{-4}$, the BER of CPE is $5.83e^{-8}$, while the BER of the encoder without protection is $5.54e^{-3}$. This represents a significant improvement, by more than 5 orders of magnitude. Furthermore, by injecting errors only on non-critical gates, the performance of CPE fared much better than the default encoding mechanism.

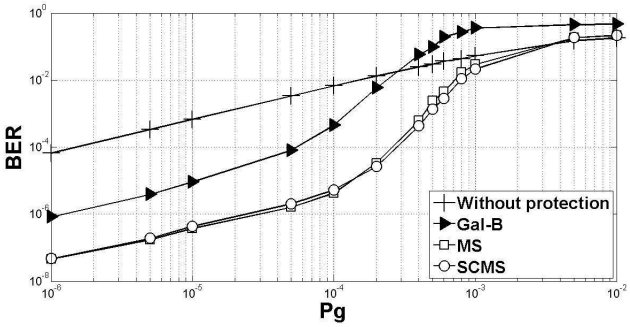


Fig. 10. Decoder Configuration impact on Output BER

For the encoder with $d_v = 4$ and $r = 1/2$, by setting the CT value to 10, it is possible to provide fault free information when the gate error rate is less than 10^{-4}. As depicted in Fig. 11, the CPE mechanism provides error free output for gate errors smaller than P_g = $6e^{-4}$. We employed MS decoding scheme and adopted an encoder with $r = 1/2$ and $d_v = 4$ for achieving

this kind of performance. For the sake of completion, Fig. 12 illustrates the similar scenario assuming a perfect decoder. In such case, we see that CPE performance improves marginally as compared to employing faulty decoder.

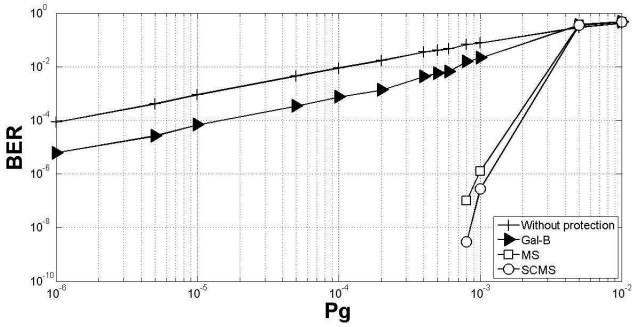


Fig. 11. CPE error free scenario employing faulty decoder

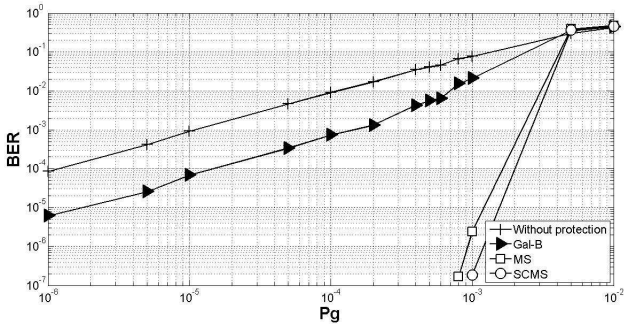


Fig. 12. CPE error free scenario employing perfect decoder

VI. Conclusion

A novel fault tolerant methodology known as Codeword Prediction Encoder (CPE) for reliable data transmission using unreliable hardware is proposed. The principle idea is to adopt additional LDPC encoder specifically to correct dynamic errors introduced during the encoding process. The CAD flow for CPE methodology is implemented and performance evaluation has been completely automated. Simulation results shows that performance of CPE is much better as compared to transmitting data by employing traditional encoding methodology. Performance evaluation using various fault tolerant LDPC decoders were discussed. It is shown that by employing Min-sum decoding mechanisms and a strong encoder $r = 1/2$ and $d_v = 4$, it is possible to correct all errors given that gate errors smaller than P_g = $6e^{-4}$. In general, CPE performance improvement of upto 10K is observed when compared to the normal encoding mechanism.

References

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

[2] D. J. MacKay and R. M. Neal, "Near shannon limit performance of low density parity check codes," *Electronics letters*, vol. 32, no. 18, pp. 1645–1646, 1996.

[3] S. Borkar, "Designing reliable systems from unreliable components: the challenges of transistor variability and degradation," *Micro, IEEE*, vol. 25, no. 6, pp. 10–16, 2005.

[4] C. Constantinescu, "Trends and challenges in vlsi circuit reliability," *Micro, IEEE*, vol. 23, no. 4, pp. 14–19, 2003.

[5] M. G. Taylor, "Reliable information storage in memories designed from unreliable components," *Bell System Technical Journal, The*, vol. 47, no. 10, pp. 2299–2337, Dec 1968.

[6] ——, "Reliable computation in computing systems designed from unreliable components," *Bell System Technical Journal, The*, vol. 47, no. 10, pp. 2339–2366, Dec 1968.

[7] B. Vasic and S. K. Chilappagari, "An information theoretical framework for analysis and design of nanoscale fault-tolerant memories based on low-density parity-check codes," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 54, no. 11, pp. 2438–2446, 2007.

[8] C. K. Ngassa, V. Savin, E. Dupraz, and D. Declercq, "Density evolution and functional threshold for the noisy Min-Sum decoder," *IEEE Transactions on Communications*, vol. 63, no. 5, pp. 1497–1509, May 2015.

[9] E. Dupraz and D. Declercq, "Evaluation of the robustness of LDPC encoders to hardware noise," in *IEEE BlackSeaCom conference*, 2015, pp. 1–5.

[10] M. R. Choudhury and K. Mohanram, "Low cost concurrent error masking using approximate logic circuits," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 32, no. 8, pp. 1163–1176, 2013.

[11] E. Sogomonjan and M. Goessel, "Design of self-parity combinational circuits for self-testing and on-line detection," in *Defect and Fault Tolerance in VLSI Systems, 1993., The IEEE International Workshop on*, Oct 1993, pp. 239–246.

[12] S. Manich, M. Nicolaidis, and J. Figueras, "Enhancing realistic fault secureness in parity prediction array arithmetic operators by i ddq monitoring," in *VLSI Test Symposium, 1996., Proceedings of 14th.* IEEE, 1996, pp. 124–129.

[13] T. Richardson and R. Urbanke, "Efficient encoding of low-density parity-check codes," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 638–656, 2001.

[14] C.-H. Huang, Y. Li, and L. Dolecek, "Gallager B LDPC Decoder with Transient and Permanent Errors," *IEEE Transactions on Communications*, vol. 62, no. 1, pp. 15–28, 2014.

[15] V. Savin, "Split-extended LDPC codes for coded cooperation," in *International Symposium on Information Theory and its Applications (ISITA)*, 2010, pp. 151–156.