IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 6, 2021, accepted October 4, 2021, date of publication October 6, 2021, date of current version October 15, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3118544

Syndrome-Based Min-Sum vs OSD-0 Decoders: FPGA Implementation and Analysis for Quantum LDPC Codes

JAVIER VALLS[1], FRANCISCO GARCIA-HERRERO[2], NITHIN RAVEENDRAN[3], (Member, IEEE), AND BANE VASIĆ[3], (Fellow, IEEE)

[1]Instituto de Telecomunicaciones y Aplicaciones Multimedia, Universitat Politecnica de Valencia, 46022 Valencia, Spain
[2]ARIES Research Center, Universidad Nebrija, 28040 Madrid, Spain
[3]University of Arizona, Tucson, AZ 85721, USA

Corresponding author: Francisco Garcia-Herrero (fgarciahe@nebrija.es)

The work of Bane Vasić was supported by NSF under Grant CCF-1855879, GrantCCSS-2027844, GrantCCSS-2052751, and Grant NS-FERC 1941583.

ABSTRACT Quantum processors need to improve their reliability to scale up the number of qubits and increase the number of algorithms that can execute. To reduce the logical error rate of the quantum systems, the use of error correction codes and decoders has been established as a low-cost and feasible approach, with good results from a theoretical perspective, for mid and long-term architectures. While most of the authors are focused on the algorithms to improve the correction capability of quantum computers, without taking into account a fundamental implementation aspect for their deployment in a real system, *i.e.*, their latency must be bounded to avoid the qubit decoherence, only a few propose hardware architectures and they just include time estimations of their decoding latency. However, a real implementation has not been shown yet. In this work, we analyze from the point of view of hardware implementation two algorithmic options based on quantum low-density parity-check (QLDPC) codes: a) belief propagation min-sum decoders combined with codes with good error-floor behavior and b) belief propagation min-sum decoders concatenated with ordered statistics decoders (OSDs) for codes with early error-floor. The bounds for the maximum clock frequency required by the decoders to decode within the qubit coherence time are established as a parameter to show if a practical implementation is possible with the present or near future FPGA technology. Furthermore, real implementation results for a Xilinx FPGA device are provided, showing that some solutions can meet the timing constraints set up by the state-of-the-art quantum processors.

INDEX TERMS Quantum error correction, syndrome based decoding, ordered statistics, FPGA devices.

I. INTRODUCTION

Quantum error correction codes have been deeply studied for more than three decades [1]–[3]. Several solutions based on different codes such as: Calderbank-Shor-Steane (CSS) codes [4], Shor codes [5], 2D color codes, 3D color codes [6], surface codes [7] and quantum low-density parity-check (QLDPC) codes [8] have been proposed to correct errors in quantum processors. In addition, different decoding algorithms based on techniques like Blossom decoder [9], message-passing [10] and artificial intelligence (neural-networks) [11] have been described showing the benefits and the error-correction capacity of each method.

The associate editor coordinating the review of this manuscript and approving it for publication was Derek Abbott.

Even though these codes and decoders show several orders of magnitude of improvement of the output error rate of the quantum processors, from theoretical analysis, most of the solutions do not provide physical implementations and hence, it remains unclear that if in a real system these solutions are fast enough to handle with the qubit decoherence (which is the perturbation of the superposition states of the qubits by their interaction with the environment) [12]. This implies that the time to perform the error correction is bounded. Due to this, the implementation of any quantum error correction system has to consider the time budget of a single error correction round which is set between hundreds of nanoseconds [13] and several microseconds [14], to ensure that quantum information remains stable.

Another fundamental parameter is scalability of the number of qubits [15]. The error correction hardware needs to be feasible for not only tens or hundreds, but for thousands of qubits. Only with a large number of qubits, improvements of the quantum over classical systems will be highlighted [16].

Among error correction codes that can overcome these challenges, sparse quantum stabilizer codes - QLDPC codes are gaining prominence [8], [17]. Since the theoretical result that QLDPC codes promise scalable quantum computation with a finite overhead, asymptotically good codes and their efficient decoders have been proposed [18]. Compared to surface codes and color codes of similar lengths, QLDPC codes boast better code rates with fault-tolerant thresholds supported by low-complexity iterative decoding algorithms [19]. Recent breakthroughs achieved in improved scaling of minimum distance of QLDPC codes [20] pushes for further improvement of iterative decoders and more importantly, efficient decoder implementations that are scalable.

A. CHALLENGES FOR THE IMPLEMENTATION OF QLDPC DECODERS

In terms of decoder implementation, numerous works have been done for the classical counterparts of QLDPC codes [21]– [22], obtaining good hardware results in different areas with high specifications in throughput and area-power consumption, such as optical communications, storage systems or post-quantum cryptography applications [23].

However, from the hardware point of view, there is one important difference that does not make most of the classical architectures directly applicable to quantum systems: most of the existing solutions are based on improving speed/throughput in the decoder and do not consider latency, or at least the latency restrictions are not critical. This problem can be understood by comparing the equation of throughput, $T = (N \cdot f_{\max})/(\mathrm{It} \cdot \tau)$, and latency, $L = (\mathrm{It} \cdot \tau)/f_{\max}$, for the parallel decoders, where N is the length of the codeword in terms of bits, It is the number of iterations, τ the number of clock cycles per iteration and $f_{\max}$ the maximum frequency of the device. Classical LDPC decoders for high-speed try to maximize $f_{\max}$ by introducing a large number of pipeline registers to reduce the critical path and hence increase τ, but they compensate this increase of τ using long LDPC codes, with a large N parameter (which also have very good error correction properties for classical decoders). For quantum systems, the only inputs that the decoder receives are the quantum syndromes, which are sent in parallel, so N does not have any effect on the input latency, in terms of clock cycles. On the other hand, the decoder needs to be very fast to avoid decoherence effects, so only parallel architectures meet the latency requirements in terms of clock cycles. Unfortunately, for parallel decoders, N does not have any effect on the number of clock cycles. The number of pipeline registers is the one that determines the number of clock cycles per iteration, τ, and contributes to improve the $f_{\max}$. The parameter N only may have a negligible effect in routing and hence in the maximum frequency, but for the quantum decoders the critical parameter is τ. So, if the number of pipeline registers was increased, τ is also increased, and τ increases faster than $f_{\max}$. In other words, triplicating the pipeline registers it is not possible to triplicate $f_{\max}$ for problems with routing. Hence, latency is not optimum for classical decoders, which is the main constraint of the quantum ones.

Finally, it is crucial to perform implementations of these decoders in the selected hardware platforms, as the timing limitations cannot be derived only from the synthesis results or the estimation of the delay by counting the number of gates involved in the critical path. As we show in the following sections of this paper, the most limiting component of $f_{\max}$ is routing, which can only be calculated by the development tools with the after place and route results.

B. RELATED WORKS

QLDPC codes, similar to classical LDPC codes apply message-passing decoding based on belief-propagation (BP) algorithms obtaining good performance [24], [25]. However, some QLDPC codes fail introducing an early degradation, the error-floor. The error-floor is the effect of not improving the logical error rate of the quantum system with an error correction decoder even when the physical error rate of the quantum processor improves. To solve this problem, the most studied solution, proposed in [19] and [26], is the concatenation of a classical algorithm called ordered statistics decoding (OSD) [27] to a BP-based decoder, *i.e.* to apply OSD when BP fails, which is a different approach from other classical works where the OSD is used offline to optimize some parameters of the BP decoder [28]. This improves the error correction in most QLDPC codes, but there are great differences in its final effect depending on the code. For some codes there is a slight improvement of the total error correction capacity of less than one order of magnitude on the output error-rate, for others, OSD completely removes the error floor.

The main problem of a derived implementation of OSD is that the number of rounds is exponentially linked to the data bits from the code. This problem was reported in [19], where OSD, although interesting to set a boundary in terms of error correction capacity, is labeled as *impractical* for real-time implementations. As an alternative, OSD order 0 (OSD-0), which is a one round OSD algorithm, is proposed. Although it is evident that exponential complexity is avoided, under our best knowledge no implementation of OSD-0 algorithm for quantum processors has been reported yet. Therefore, the question that remains open is if OSD-0 can meet the timing constraints of the quantum processors or not, as it involves costly steps such as matrix inversion.

Under the best knowledge of the authors, a practical hardware architecture of a quantum error correction decoder has not been published yet. Most of the papers published are focused on the algorithms to achieve the target correction capability and only a few, [29], [30], consider the implementations aspects to achieve the time budget to avoid the qubit decoherence, and they include an estimation of the latency of their potential implementation.

C. PAPER ORGANIZATION

In this paper, two decoding algorithms will be evaluated from a VLSI perspective, comparing its performance and calculating what should be the expected latency and clock frequency in real hardware implementations. The first one is the syndrome-based min-sum (SB-MS) decoder [31], which is an iterative BP-based solution with a good trade-off between correction and physical constraints. However, SB-MS is not applicable to different classes of stabilizer codes in general due to error floor-problems. The second one is SB-MS concatenated with OSD-0, to improve the waterfall performance and alleviate the error floor problems. However, this performance improvement comes with increasing the total latency time due to computationally intense operations involved. The objective of the paper is to obtain timing, area and power results to evaluate if it is possible to apply these algorithms with the present technology to a quantum processor meeting the constraints of the qubit decoherence.

The rest of the document has the following structure: in Section II, the background concepts about complexity and correction capacity are summarized; in Section III, different architectures for SB-MS and OSD-0 will be analyzed. The results of these architectures are shown in Section IV, and in Section V the main conclusions of the paper are highlighted.

II. BACKGROUND

A. QUANTUM ERROR CORRECTION AND CHANNEL MODEL

Despite classical error correction decoders, quantum error correction (QEC) decoders do not take as initial information the corrupted message from the channel [16], $c+e$, where c is the codeword vector and e the error introduced by the channel. Due to the nature of qubits, data cannot be read without altering it, for this reason, the main solution to introduce error correction is to use the information that comes from the syndromes, S, calculated at the quantum processor [32]. In that sense, instead of looking for a correct codeword ($\tilde{c}$) through the search of a candidate that satisfies the parity check equations, the objective of the error-correction system is to search for an error pattern ($\tilde{e}$) that generates the same syndromes as the quantum system. By using the syndromes, the qubits that contain data are not altered and the correct error pattern can be just added to the distorted codeword recovering the information. The main differences of both classical and quantum decoders can be seen in the block diagrams from Fig.1.

Another important change is the error model of the quantum system. The errors that modify the codeword in a quantum processor can be defined by a depolarizing channel model that inserts bit-flips, phase-flips or a combination of bit and phase flips [16]. The bit-flips are defined by Pauli errors of type X, with a probability of p_X. The phase-flips are modeled as Pauli errors of type Z with a probability of error of p_Z. A combination of bit-flips and phase-flips in the errors is described by a Pauli error of type Y with probability

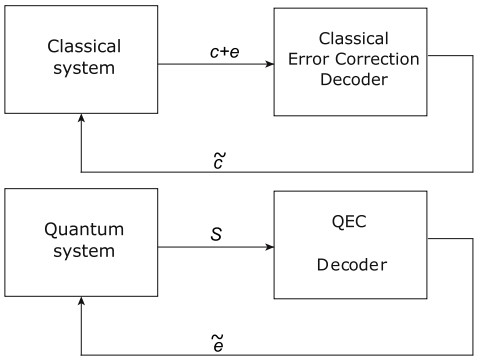

FIGURE 1. Block diagram of the classical LDPC decoder vs the QLDPC decoder.

p_Y. If the error model assumes a symmetric depolarizing error channel, the probability of error of the model is $p = p_X/3 + p_Z/3 + p_Y/3$.

B. QUANTUM LOW-DENSITY PARITY-CHECK DECODERS

Unlike classical LDPC codes, which are defined by just one $M \times N$ parity-check matrix H, QLDPC codes are defined by two different matrices: H_X to correct the X-errors (or bit-flips) and H_Z to correct the Z-errors (or phase-flips) from the depolarizing channel. The constructions of these matrices need to satisfy the symplectic inner product constraint arising from the stabilizer formalism based on the Pauli-to-binary isomorphism (for more details about code construction we refer to [16] and [24]). In this document, for simplicity, we will refer to H_X and H_Z as H indistinctly, without loss of generality. However, in Section IV, there will be an impact of the architecture depending on the code is dual containing ($H_X = H_Z$) or non-dual containing ($H_X \neq H_Z$). QLDPC codes are usually named as (N, $N-$ rank(H)) QLDPC. The code can be graphically represented as a bipartite Tanner graph, with two types of nodes: check-nodes that correspond to the rows of parity-check equation; and variable-nodes that correspond to the columns or the information of each of the bits in the codeword. Each check-node (variable-node) of the code has a degree of check-node d_c (degree of variable-node d_v), which is equal to the non-zero elements of a row (column).

SB-MS is a message-passing algorithm derived from BP that exchanges two types of messages, the ones generated in the check-node, $\sigma_{i,j}$, and the ones computed in the variable-node, $R_{i,j}$. The algorithm tries to estimate the error pattern in the variable-nodes to match the input syndromes. In terms of hardware, there are two types of nodes that exchange information and the complexity of the derived decoder will be limited by the code parameters, *i.e.* degree of the check-node d_c, degree of variable node d_v, number of parity-check equations M and number of total bits in the code N [33]. The operations computed by the check nodes and the variable nodes are described in equations (1) and (2), where S_i is the syndrome for the parity-check equation i and $\mathcal{N}_i$ and $\mathcal{M}_j$ are defined

as the sets consisting of all the non-zero elements of a row i (check-node) and a column j (variable-node), respectively, $\oplus$ is the mod2 addition (one-bit XOR operation) and α is a scaling factor to improve the convergence of the algorithm. The hard decision operation (HD) included in equations (1) and (3) is defined as the conversion from a soft-decision message to a logical one. Usually, it is equivalent to the bit that represents the sign, considering the positive sign as a logic 0 and the negative sign as a logic 1. The scaling factor is obtained by performing Monte Carlo simulations, as happened with classical LDPC decoders, to optimize the logical error rate of the processor under a depolarizing channel. The value of the factor is constrained between 0 and 1 and is usually selected as $\alpha = 2^{-a} + 2^{-b}$ with $a, b \in \{1, 2, 3\}$. In this way, we can perform the products by just shifting the bits and adding two shifted words. The shifting process is implemented with wires, so no additional hardware is required. In the worst case, only one adder is implemented, avoiding the use of multipliers that increase the hardware resources and the critical path. L_j is the reliability of e_j, which is the value of the error in the bit j. As e_j is unknown, L_j is initialized to a constant, usually to $1 - 2p/3$, but as all the values are the same for all the j indexes, it can be initialized to $+1$ to keep messages small without any performance loss.

$$\sigma_{i,j} = \min_{j' \in \mathcal{N}_i \backslash j} (|R_{i,j'}|) \times \prod_{j' \in \mathcal{N}_i \backslash j} \mathrm{HD}(R_{i,j'}) \oplus S_i \tag{1}$$

$$R_{i,j} = L_j + \alpha \cdot \sum_{i' \in \mathcal{M}_j \backslash i} \sigma_{i',j} \tag{2}$$

The algorithm will iterate applying recursively these equations until one solution is found or a maximum number of computations is reached. The solution has to satisfy equations (3) and (4).

$$\tilde{e_j} = \mathrm{HD}(L_j + \alpha \cdot \sum_{i' \in \mathcal{M}_j} \sigma_{i',j}) \tag{3}$$

$$S = \tilde{e} H^T \tag{4}$$

As it will be shown in the next section, the solution with lower latency consists of a parallel implementation of one iteration, mapping (1), (2) and (3) for $M \times N$ nodes, but at the same time has the problem of routing between processing units, which makes difficult to ensure hundreds of nanoseconds of latency.

C. OSD IMPACT ON QLDPC ERROR CORRECTION PERFORMANCE

Recent work [19] and [26] conclude that decoding algorithms derived from BP (such as SB-MS) do not provide a correction capacity good enough for all QLDPC codes. Some codes such as the (882, 24) QLDPC or the (1024, 30) QLDPC exhibit an early error-floor when BP-based algorithms are applied. The difference between the physical error rate of the quantum processor without QEC is almost the same as the one obtained after introducing the decoder. To solve this limitation OSD is concatenated, improving the problem of the early error-floor.

However, not all the codes need the support of OSD to obtain a good error correction performance. Other codes such as (254,28), (7938, 578), (882,48) or (126,28) QLDPC[1] show a good error correction capacity with BP-based algorithms like SB-MS and the inclusion or not of the OSD algorithm does not entail a significant improvement. In Fig.2, some codes are included to show the gap between SB-MS and SB-MS with OSD concatenated. As it can be seen, the difference between SB-MS and SB-MS with OSD is progressively increased with higher reliability physical error rates (PER). For example, for the (126,28) QLDPC the output error rate is about $1.3{\cdot}10^{-5}$@PER=10^{-2} including OSD and $2.6{\cdot}10^{-5}$@PER=10^{-2} without OSD, less than one order of magnitude. For the (255,32) QLDPC the difference is more significant $1.6{\cdot}10^{-4}$@PER=$2{\cdot}10^{-2}$ without OSD vs $3.6{\cdot}10^{-5}$@PER=$2{\cdot}10^{-2}$, almost one order of magnitude. The largest difference can be found for codes like (625,25) and (900,36) QLDPC where the gap is almost two orders of magnitude [19]. For these codes, the impact of OSD is more significant due to the bad decoding of SB-MS algorithm. However, there are alternatives for similar number of qubits i.e., (126,28) and (255,32) QLDPC codes, as alternative to (625,25) and (900,36) QLDPC codes respectively, that only with SB-MS behave better than SB-MS+OSD. But the key is: i) what is the cost of OSD? and ii) is it feasible to implement this algorithm to fit in the time window set by the qubits decoherence? Finally, it is also important to take into account that the output of the OSD algorithm only has two possible options, a logical error or the correct error pattern, as the output obtained is just binary, as it will be shown in the next section, OSD does not provide any soft information. For this reason, the output of the OSD algorithm cannot be improved in terms of correction by using any other method concatenated, so all the decoding failures turn into logical errors. While the output of SB-MS decoder, if it does not find the error pattern, is just a decoder failure that includes soft information from the messages that can be processed by other concatenated decoders.

D. OSD AND OSD-0 COMPLEXITY

OSD was originally introduced in [27] for classical codes to perform post-processing based on an exhaustive search which requires from matrices inversion in an iterative process that scales with $2^{N-\mathrm{rank}(H)}$. This same algorithm, as it was said before, was evaluated to set a boundary for QLDPC codes to check if concatenation can improve BP-based decoders. Nevertheless, a recent study also focused on the combination of BP and OSD [19] reported that, searching over all configurations that are required in the original OSD soon becomes intractable for large codes, concluding that a more realistic approach is just to apply the simplest version of the OSD decoder, named as OSD-0. Note that for the (255,32) and

[1]All these codes were simulated using a CPU Intel Core-i7-6700-HQ CPU@2.6 GHz using a software model developed in Matlab R2015b. The results of some of these simulations are included in Fig.2 and 5.

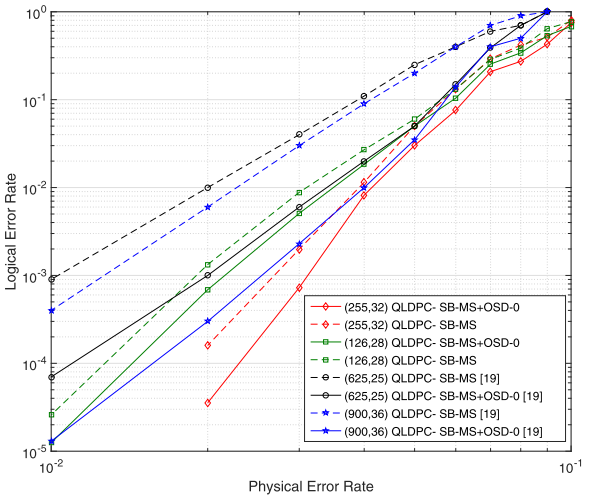

FIGURE 2. Performance of SB-MS and SB-MS+OSD for different QLDPC codes under a depolarizing channel. The physical error rate (PER) is determined by the reliability of the quantum processor's physical components modeled by the depolarizing channel. The logical error rate is the reliability obtained by the quantum processor after the decoder. The codes (256,32) and (128,28) are simulated with $\alpha = 0.75$ for the SB-MS decoder and codes (625,25) and (900,36) are simulated with $\alpha = 1$.

(126,28) QLDPC codes, whose performance is in Fig.2, 2^{32} and 2^{28} rounds are required with the original OSD in the worst case, respectively. In principle, this complexity does not justify the improvement in the error rate, because, among other things, it is impossible to implement an OSD decoder that reaches an acceptable decoding latency. Although in [19] there is not a hardware analysis or implementation, there is a clear demonstration of why OSD-0 is the only presumable version of OSD that can be integrated into a real-time system, for this reason, is the only that we will consider in this document.

The objective of OSD and OSD-0 is to take the reliability value of $\tilde{e}_j$ from equation (3), $L_j + \alpha \cdot \sum_{i' \in \mathcal{M}_j} \sigma_{i',j}$, when BP-based decoders have failed to converge and look for the error pattern with the highest probability to satisfy equation (4). To do so, the steps of OSD-0 are:

1) Sort $L_j + \alpha \cdot \sum_{i' \in \mathcal{M}_j} \sigma_{i',j}$ from the most probable indexes of j to have an error ($\tilde{e}_j = 1$) to the less ($\tilde{e}_j = 0$). Keep the sorted set in $\mathcal{J}_{\text{OSD}}$.
2) Reorder the columns of H according to $\mathcal{J}_{\text{OSD}}$ obtaining H_{OSD}.
3) Build a square matrix with the first rank(H) linearly independent columns of H_{OSD} and the rank(H) independent rows, to look for an invertible matrix H'_{OSD}.
4) Compute the inversion matrix of H'_{OSD}: ${H'_{\text{OSD}}}^{-1}$.
5) Calculate an error pattern that satisfies the syndromes: $\tilde{e}'_{\text{OSD}} = {H'_{\text{OSD}}}^{-1} \cdot S$.
6) Build the final error pattern $\tilde{e}_{\text{OSD}}$ in which the j locations that correspond to columns not included on H'_{OSD} are equal to 0 and the j locations included in H'_{OSD} are equal to $\tilde{e}'_{\text{OSD}}$.
7) Undo the reordering of step 2), to correct the right indexes of the codeword.

It is easy to see that OSD-0 is not an iterative algorithm, but it entails complex operations like the sort of N elements in step 1) or a dynamic inversion of the matrix in step 4).

III. ARCHITECTURES FOR SB-MS AND OSD-0 ALGORITHMS AND LATENCY ANALYSIS

In this section, the hardware architectures for both SB-MS and OSD algorithms will be introduced to evaluate if the timing constraints of the qubits decoherence can be accomplished in different scenarios.

A. ARCHITECTURE FOR A SB-MS DECODER

In order to accomplish the latency requirements we propose a parallel architecture, which consists on the implementation of one iteration of the SB-MS algorithm. Besides, the parallel broadcasting techniques such as the ones in [34], are applied as described next.

The complete decoder will implement M check-node units (CNUs) such the ones from Fig. 3, and N variable-node units (VNUs) like the one in Fig. 4. All the cells will be interconnected according to the parity-check matrix of the code. If the code is dual containing two decoders with exactly the same architecture are required; if the code is non dual containing the two decoders have different interconnection between CNUs and VNUs for H_X and H_Z, but the number of wires will be the same. However, in both cases their complexity and number of wires will be the same.

Each CNU (the i-th one) receives as input d_c values of $R_{i,j}$ with $j \in \mathcal{N}_i$ and the corresponding syndrome S_i. This unit has two different parts:

- The computation of the parity check equation to obtain the syndrome S_i, which is implemented by a tree of XOR gates that compute all the HD($R_{i,j}$) (sg($R_{i,j}$) in Fig.3 and 4) and the syndrome. After the tree of XOR gates, the $R_{i,j'}$ with $j' \in \mathcal{N}_i \setminus j$ is calculated by subtracting to the total its own contribution. This sub-unit implements $\prod_{j' \in \mathcal{N}_i \setminus j} \text{HD}(R_{i,j'}) \oplus S_i$ from equation (1), generating d_c hard-decisions (1-bit per hard-decision) that are sent to the corresponding VNUs (sg($\sigma_{i,j}$) in Fig.3 and 4). Moreover, d_c signals of one bit are also exchanged to indicate if the connected VNU contains the first minimum or not (sl($\sigma_{i,j}$) in Fig.3 and 4).
- The computation of the magnitude of the exchanged messages is obtained looking for the first and the second minimum of the received $R_{i,j}$ values. Once these minimums are obtained, they are scaled by the factor α. This factor improves the convergence speed of the algorithm. In order to simplify the implementation, the set of possible values are reduced to $\alpha = 2^{-a} + 2^{-b}$ with $a, b \in \{1, 2, 3\}$, so hardwired multiplication based on one shifter and one adder is implemented instead of a complete multiplier. In order to reduce the routing congestion of the architecture, instead of sending $d_c - 1$

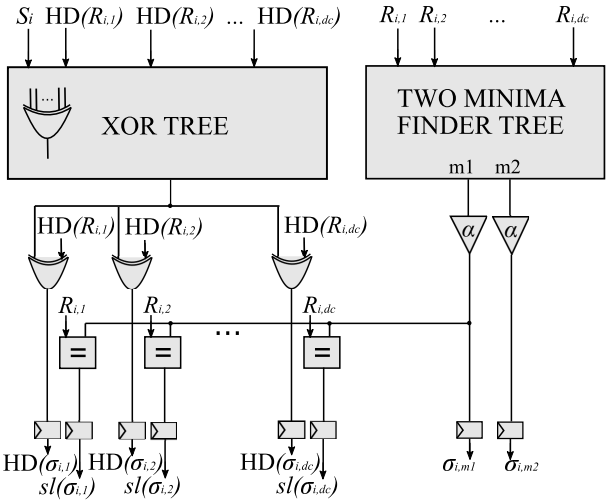

FIGURE 3. Architecture for check-node unit, Equation (1).

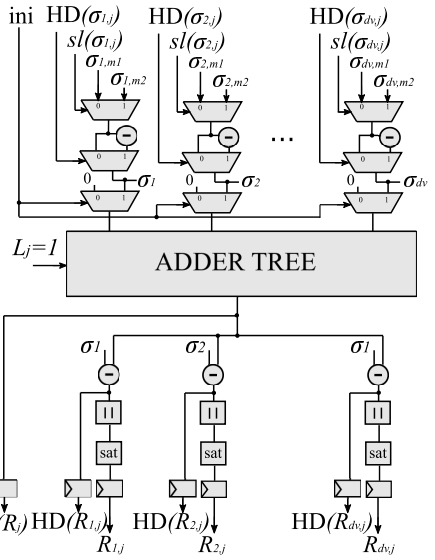

FIGURE 4. Architecture for variable-node unit, Equation (2), and error pattern, Equation (3).

messages with the first minimum and one message with the second one, a broadcasting technique is adopted. Therefore, only two values are sent to all the connected VNUs to reduce the wiring, so instead of $d_c \times \gamma$ wires between units there are $2 \times \gamma$ wires (and hence $2 \times \gamma$ outputs), where γ is the number of bits of the quantized messages. The selection of what message has the second minimum to implement $\min_{j' \in \mathcal{N}_i \setminus j} (|R_{i,j'}|)$ from equation (1) is performed inside the VNU unit as detailed next.

Each VNU (the j-th one) has $2 \times d_v$ inputs of γ bits as each of the d_v connected CNUs sends two minimums plus d_v hard-decisions (d_v bits) and d_v bits to select between the first and the second minimum. In the first step, the architecture selects if it needs the first or the second minimum. After this, the next stage, which initializes to zero the first iteration, combines the hard-decision and the magnitude to compute with a tree of adders $L_j + \alpha \cdot \sum_{i \in \mathcal{M}_j} \sigma_{i,j}$ from equation (2). As it is explained in the previous section L_j is set to a constant for all the values of j, *i.e.*, $L_j = 1$, because the reliability of the error message is unknown in the first iteration and all the bits have the same reliability. After the tree, each message eliminates its own contribution from the total, via a subtracter, to implement $R_{i,j}$ values in equation (2). During the last stage, the values are saturated to control the growth of the data path. The messages $R_{i,j}$ are split into hard-decision and magnitude and send to the connected CNUs. The outputs of both sub-units, CNUs and VNUs, are registered to limit the critical path of the circuit.

According to the previous descriptions, it is easy to derive that the total latency per iteration is equal to two clock cycles, one to reach the outputs of the CNU and another to complete the operations in the VNU. So, assuming that the syndromes are available in parallel at the input, the global delay is equal to $D_{\max} = 2 \times It_{\max}/f_{\max}$, where $It_{\max}$ is the maximum number of iteration and $f_{\max}$ is the maximum frequency of the architecture. It should be noted that the $f_{\max}$ is limited by the delay of the CNU and that of the routing cables, which are quite congested due to the large number of cells to be connected.

With this information, a range of required frequencies for the hardware implementation can be computed taking as reference the time budget from real quantum processors. Although this range can be modified by the technology of the processor itself, the number of qubits and the encoding of the information, it allows hardware designers to have an order of magnitude to evaluate if the architectures are feasible to be applied in real systems or not. For example, taking as starting point the most restrictive time budget found in literature [13], [14] that is 400 ns,[2] the maximum frequency of the circuit needs to be $f_{\max} = 2 \times 20/400\,\text{ns} = 100$ MHz, assuming 20 iterations. If both H_X and H_Z are decoded with the same device, *i.e.* for dual containing codes, twice this speed need to be reached, so it would be higher than 200 MHz.

This maximum frequency could seem reduced and easy to obtain with the actual FPGA technology, however, although it is plausible is not easy to get it under any type of circumstances or code parameters, as there is a long critical path, with just two pipeline registers in the whole decoder, which depends on d_c and d_v. Besides, the routing congestion increases with N, M, d_c and d_v, and limits the maximum frequency by more than 50% compared to the limitation of logic depth, as it will be shown in Section IV.

B. ARCHITECTURES FOR OSD-0 ALGORITHM

Several architectures for OSD-0 have been proposed for classical error correction [35]– [36], and as the algorithm

[2]This time is reported as the period required to compute a syndrome generation cycle, and according to [13], a real hardware implementation of a decoder must be able to execute faster than syndrome data are generated as a prerequisite for tractable fault tolerant computation.

is exactly the same for QEC, the classical hardware design should be evaluated for the quantum scenario.

From the steps described in Section II, the most critical one is the inversion of H'_{OSD} (step 4). The operations involved in this step are going to establish the bottleneck of the OSD-0 architecture.

The inversion of the matrix does not have any low-complexity or suboptimal approach as happens with, for example, the sorting process of the incoming messages (step 1). To perform it, the Gaussian elimination with Galois Field GF(2) needs to be implemented. The two main architectures to perform Gaussian elimination in an efficient way are introduced in [37] and [38]. The first one [37] is made of a network that connects all the elements from the matrix in parallel, so it is a network with $M \times N$ nodes, and pivots columns and rows to choose the columns and rows to eliminate. Although it is a parallel implementation, the algorithm needs at least $2M$ rounds and at most $(M^2 + M/2)$ to compute ${H'_{\text{OSD}}}^{-1}$. So, in the best scenario, the latency of the Gaussian elimination implementation is determined by $D_{\max} = 2M/f_{\max}$.

In Table 1, we summarize the maximum frequency for different codes and for the latency constraint used as a reference. As it can be seen the value of $f_{\max}$ required by the (882,24) QLDPC code is not reachable by actual FPGA technology. Furthermore, it is important to notice that we only consider the delay introduced by the Gaussian elimination required by OSD-0, but the time budget is even less as the delay of the QLDPC decoder has to be also considered and the same for the rest of the steps of OSD-0, here we assume them as negligible just to establish an optimistic boundary.[3] As it is mentioned in Section II, this code needs OSD-0 to avoid error-floor, however, as it is analyzed here a real-time implementation, in this case, is not possible, so it is discarded as a real solution.

Even in the best case the speed needed for the (126,28) QLDPC code is too high for an FPGA device. Besides, for this case, the OSD-0 introduces less than one order of magnitude of improvement that hardly justifies the major drawback of this implementation, the high requirement of resources and routing congestion, since every matrix element is instantiated and connected via global signals as analyzed in [36]. For this reason, hardware designers usually apply other approaches.

In [38] an efficient architecture for Gaussian elimination with fewer resources and without routing congestion is proposed. This architecture implements a systolic array that is connected to a small number of processors that are neighbour nodes in the matrix. This reduces the total number of wires and global connections increasing the maximum frequency, but at a cost of increasing the number of clock cycles required to $3M + N - 2$. Repeating the analysis performed with the parallel implementation the equation that links the maximum frequency with the delay is $D_{\max} = (3M + N - 2)/f_{\max}$. As it can be seen in Table 2 results are even more restrictive with this approach making unrealistic to expect a real implementation being tightened up to the time budget imposed by the decoherence.

[3]Note that this is a very optimistic estimation as for example, for the sorting there are multiple direct implementations of latency N or implementations with a large critical path due to the logical depth of the derived parallel trees of comparators. Other alternatives with lower latency, such as using a threshold to select the most probable indexes of j to have an error, can be set, assuming some performance loss of the correction capacity.

TABLE 1. Maximum frequency approximation for the Gaussian elimination parallel architecture [37] to meet the timing constraints of the quantum system.

QLDPC code	M	$f_{\max}$ @ $D_{\max}$=400 ns
(126, 28)	63	315 MHz
(255, 32)	112	560 MHz
(882, 24)	441	2.2 GHz

TABLE 2. Maximum frequency approximation for the Gaussian elimination systolic architecture [38] to meet the timing constraints of the quantum system.

QLDPC code	M	$f_{\max}$ @ $D_{\max}$ = 400 ns
(126, 28)	63	787 MHz
(255, 32)	112	1.5 GHz
(882, 24)	441	5.5 GHz

According to all the previous analysis, it can claim that OSD-0 derived architectures are too complex or too slow due to the high number of operations and iterative calculations involved, and they are not a realistic approach to decode QEC. The frequencies that FPGA devices should reach are not feasible with the existing technology and hence it is not possible to meet latency constraints because within this same time budget of 400 ns other operations such as the sorting of the indexes according to the probability values (step 1 of OSD-0) or the reordering of the indexes (step 7) should also be performed. Besides, before applying OSD-0, a SB-MS decoder is applied, and due to its iterative nature already consumes 80% of the time budget, as it will be shown in next section. For this reason, it is more reasonable to focus the efforts on optimizing BP-based decoders for QLDPC codes that show good behavior in error correction. Moreover, it is important to remark that OSD-0 provides a binary error pattern without any soft-information, turning a decoding failure into a logical error, while BP-based decoders keep the soft information from the messages after a decoding failure, so further processing can be added to improve the final error correction capacity. In the next section real results for implementation of SB-MS decoder for a code that does not show any error degradation without OSD-0 are included.

IV. IMPLEMENTATION RESULTS

In this section, hardware results for the SB-MS decoder are included. To perform the implementation and set up a lower boundary, the (255,32) QLDPC code was selected, as: i) it is the one with more restrictive parameters, from the hardware point of view (it has high d_c and d_v) and; ii) it is one of the codes with better error correction capacity. These codes are designed following the methods described in [39].

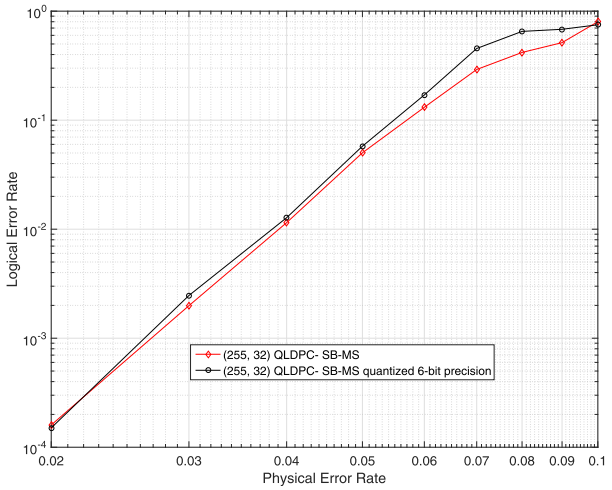

FIGURE 5. Performance of a full-precision SB-MS and a six-bit quantized version of SB-MS for the (255,32) QLDPC code under a depolarizing channel with $\alpha = 0.75$.

The code parameters, $M = 112$, $d_c = 10$ and $d_v = 5$ are representative to evaluate the routing congestion of the design and compute a realistic $f_{\max}$, not only limited by the logic involved in the critical path but also by the wiring, which can be used as a reference for the implementation of other simpler QLDPC codes.

About the performance, we show in Fig. 2 that the error correction capacity is good enough without OSD-0, and for this reason, it can work as a standalone solution.

The decoder was implemented on an FPGA device xcvu095ffva2104-2 using a hardware description language (VHDL) and Xilinx's Vivado Design Suite. A finite precision model was performed with Matlab. As it is shown in Fig. 5, the differences in error correction capacity between the quantized version with 6-bit messages and the full-precision one are almost negligible. The decoder was verified with ModelSim comparing the outputs of Matlab's golden model of 6-bit precision to the outputs of the hardware architecture.

The area results for the implementation are included in Table 3 and the layout can be found in Fig.6, where it can be seen that the decoder does not occupy the whole chip, allowing the wires between the different computational units and the logic to distribute optimally to reduce congestion and critical path, increasing speed. Proof of this fact is that the decoder can achieve an $f_{\max} = 125$ MHz (clock period of 8 ns, with 78.7% of delay due to the routing and 21.3% due to the logic), which is equivalent to a total latency of 320 ns @ $It_{\max} = 20$. This latency is slightly smaller (80 ns less) than the time budget reported in [13], making this parallel architecture a promising candidate for real implementations of QLDPC codes for the QEC step, as with two cores of this architecture it can correct X and Z errors for dual and non-dual containing codes, just changing the wiring between processing units in the last case. It is important to highlight that other serial implementations that process each CNU at the time, will allow higher convergence in the decoding but would increase the total latency M times, not fulfilling the time constraints of the quantum system. Derived from the latency results, throughput can be obtained as $T = (N \cdot f_{\max})/(It \cdot \tau) = (256 \cdot 125\text{MHz})/(20 \cdot 2) =$ 800 Mbps. Note that other implementations with a larger number of pipeline stages would obtain a higher throughput, however, for these systems, the critical parameter is latency, not speed/throughput.

Concerning the dynamic power consumption, 0.63 W are spent on the signals between cells and 0.48 W is spent on logic, validating the claim that about half of the complexity is located on the wiring and the rest on the logic processors (CNUs and VNUs). The design has a static power consumption of 0.918 W. These results show that the proposed solution, with a total power consumption of 2.1 W, can be considered as a low-power co-processor to increase reliability of the quantum system.

TABLE 3. Area resources for the FPGA implementation of the SB-MS decoder for the (255,32) QLDPC code.

Resource	Used	Available	Utilization %
CLB LUTs	49783	537600	9.26
LUT as Logic	49783	537600	9.26
CLB Registers	10848	1075200	1.01
CARRY8	2256	67200	3.36
F7 Muxes	336	268800	0.13

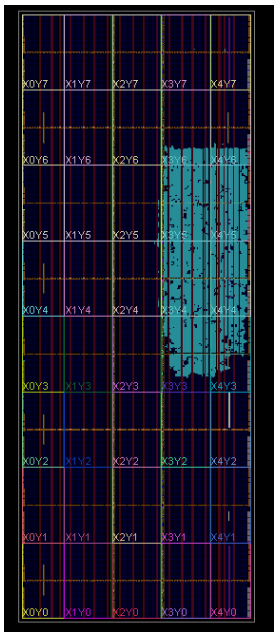

FIGURE 6. Layout of the SB-MS FPGA's implementation.

V. CONCLUSION

In this work, the feasibility of the hardware implementation of the SB-MS algorithm and its concatenation with the OSD-0 to achieve the timing constraints required to avoid the qubit

decoherence has been evaluated. The main conclusions are that a) it is not possible to implement the OSD-0 algorithm with the current or near-future VLSI technology mainly due to its sequential nature, and b) the SB-MS algorithm can be implemented accomplishing the timing requirement of QEC and is a solution with the potential to be scaled to a higher number of qubits. To reinforce our conclusions, an SB-MS decoder has been implemented in a Virtex FPGA device for the (255,32) QLDPC code achieving a latency of 320 ns with 20 iterations, which is within the required time budget. Under the best knowledge of the authors, this is the first real implementation reported for an iterative QEC syndrome decoder. Finally, given that OSD-0 is not a practical solution for QLDPC decoders, future research is focused on improving the performance of message-passing decoding algorithms to solve problems like error-floor, rather than trying to concatenate other post-processing steps with higher complexity like OSD.

Future work will try to reduce more the maximum latency to obtain some extra margin in terms of timing, and thus prevent further limitations that may appear during the integration in a quantum system. Although the final objective of this research line is to integrate the error-correction step on a real system, nowadays devices do not work with the number of qubits that handle the decoders of this paper, so it will be necessary to wait for the next generation of quantum computers. In the meantime, more steps in this direction will be required to design the hardware implementations of error-correction solutions that meet both the latency and logical-error rates of future quantum computers.

ACKNOWLEDGMENT

Bane Vasić has disclosed an outside interest in Codelucida to the University of Arizona. Conflicts of interest resulting from this interest are being managed by The University of Arizona in accordance with its policies.

REFERENCES

[1] A. K. Calderbank, E. M. Rains, P. W. Shor, and N. J. A. Sloane, "Quantum error correction and orthogonal geometry," *Phys. Rev. Lett*, vol. 78, nos. 3–20, pp. 405–408, 1997, doi: 10.1103/PhysRevLett.78.405.

[2] A. R. Calderbank, E. M. Rains, P. W. Shor, and N. J. A. Sloane, "Quantum error correction via codes over GF(4)," *IEEE Trans. Inf. Theory*, vol. 44, no. 4, pp. 1369–1387, Jul. 1998.

[3] I. B. Djordjevic, "Cavity quantum electrodynamics (CQED)-based quantum LDPC encoders and decoders," *IEEE Photon. J.*, vol. 3, no. 4, pp. 727–738, Aug. 2011.

[4] D. Gottesman, "Class of quantum error-correcting codes saturating the quantum Hamming bound," *Phys. Rev. A, Gen. Phys.*, vol. 54, pp. 1862–1868, Sep. 1996, doi: 10.1103/PhysRevA.54.1862.

[5] P. W. Shor, "Scheme for reducing decoherence in quantum computer memory," *Phys. Rev. A, Gen. Phys.*, vol. 52, pp. 2493–2496, Oct. 1995, doi: 10.1103/PhysRevA.52.R2493.

[6] A. G. Fowler, "Two-dimensional color-code quantum computation," *Phys. Rev. A, Gen. Phys.*, vol. 83, no. 4, Apr. 2011, Art. no. 042310, doi: 10.1103/PhysRevA.83.042310.

[7] A. Kubica, M. E. Beverland, F. Brandão, J. Preskill, and K. M. Svore, "Three-dimensional color code thresholds via statistical-mechanical mapping," *Phys. Rev. Lett.*, vol. 120, no. 18, May 2018, Art. no. 180501, doi: 10.1103/PhysRevLett.120.180501.

[8] J.-P. Tillich and G. Zemor, "Quantum LDPC codes with positive rate and minimum distance proportional to the square root of the blocklength," *IEEE Trans. Inf. Theory*, vol. 60, no. 2, pp. 1193–1202, Feb. 2014, doi: 10.1109/TIT.2013.2292061.

[9] V. Kolmogorov, "Blossom V: A new implementation of a minimum cost perfect matching algorithm," *Math. Program. Comput.*, vol. 1, no. 1, pp. 43–67, Jul. 2009.

[10] Z. W. E. Evans and A. M. Stephens, "Message passing in fault-tolerant quantum error correction," *Phys. Rev. A, Gen. Phys.*, vol. 78, no. 6, Dec. 2008, Art. no. 062317, doi: 10.1103/PhysRevA.78.062317.

[11] P. Baireuther, T. E. O'Brien, B. Tarasinski, and C. W. J. Beenakker, "Machine-learning-assisted correction of correlated qubit errors in a topological code," *Quantum*, vol. 2, pp. 48–58, Jan. 2018, doi: 10.22331/q-2018-01-29-48.

[12] D. Ristè, L. C. G. Govia, B. Donovan, S. D. Fallek, W. D. Kalfus, M. Brink, N. T. Bronn, and T. A. Ohki, "Real-time processing of stabilizer measurements in a bit-flip code," *NPJ Quantum Inf.*, vol. 6, no. 1, pp. 71–77, Dec. 2020, doi: 10.1038/s41534-020-00304-y.

[13] A. Holmes, M. R. Jokar, G. Pasandi, Y. Ding, M. Pedram, and F. T. Chong, "NISQ+: Boosting quantum computing power by approximating quantum error correction," in *Proc. ACM/IEEE 47th Annu. Int. Symp. Comput. Archit. (ISCA)*, May 2020, pp. 556–569.

[14] S. Varsamopoulos, K. Bertels, and C. G. Almudever, "Comparing neural network based decoders for the surface code," *IEEE Trans. Comput.*, vol. 69, no. 2, pp. 300–311, Feb. 2020, doi: 10.1109/TC.2019.2948612.

[15] D. P. DiVincenzo, "The physical implementation of quantum computation," *Fortschritte Phys.*, vol. 48, nos. 9–11, pp. 771–783, 2000.

[16] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*, 10th ed. Cambridge, U.K.: Cambridge Univ. Press, 2011.

[17] Y.-J. Wang, Z.-Y. Xiao, X.-Y. Xiong, J.-Z. Yan, C. Wang, and S. Shi, "A rate scalable construction of non-homogeneous quantum LDPC codes of CSS type based on balanced incomplete block design," *IEEE Commun. Lett.*, vol. 25, no. 5, pp. 1408–1411, May 2021.

[18] O. Fawzi, A. Grospellier, and A. Leverrier, "Constant overhead quantum fault tolerance with quantum expander codes," *Commun. ACM*, vol. 64, no. 1, pp. 106–114, Jan. 2021, doi: 10.1145/3434163.

[19] J. Roffe, D. R. White, S. Burton, and E. Campbell, "Decoding across the quantum low-density parity-check code landscape," *Phys. Rev. Res.*, vol. 2, no. 4, Dec. 2020, Art. no. 043423, doi: 10.1103/PhysRevResearch.2.043423.

[20] A. Grospellier, L. Grouès, A. Krishna, and A. Leverrier, "Combining hard and soft decoders for hypergraph product codes," *Quantum*, vol. 5, pp. 432–449, Apr. 2021, doi: 10.22331/q-2021-04-15-432.

[21] X. Chen, J. Kang, S. Lin, and V. Akella, "Memory system optimization for FPGA-based implementation of quasi-cyclic LDPC codes decoders," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 58, no. 1, pp. 98–111, Jan. 2011.

[22] P. Hailes, L. Xu, R. G. Maunder, B. M. Al-Hashimi, and L. Hanzo, "A survey of FPGA-based LDPC decoders," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1098–1122, 2nd Quart., 2016.

[23] D. Zoni, A. Galimberti, and W. Fornaciari, "Efficient and scalable FPGA-oriented design of QC-LDPC bit-flipping decoders for post-quantum cryptography," *IEEE Access*, vol. 8, pp. 163419–163433, 2020.

[24] Z. Babar, P. Botsinis, D. Alanis, S. X. Ng, and L. Hanzo, "Fifteen years of quantum LDPC coding and improved decoding strategies," *IEEE Access*, vol. 3, pp. 2492–2519, 2015.

[25] M. K. Roberts and R. Jayabalan, "An improved low-complexity sum-product decoding algorithm for low-density parity-check codes," *Frontiers Inf. Technol. Electron. Eng.*, vol. 16, no. 6, pp. 511–518, Jun. 2015.

[26] P. Fuentes, J. Etxezarreta Martinez, P. M. Crespo, and J. Garcia-Frias, "Degeneracy and its impact on the decoding of sparse quantum codes," *IEEE Access*, vol. 9, pp. 89093–89119, 2021.

[27] M. P. C. Fossorier and S. Lin, "Soft-decision decoding of linear block codes based on ordered statistics," *IEEE Trans. Inf. Theory*, vol. 41, no. 5, pp. 1379–1396, Sep. 1995.

[28] W. Xue, T. Ban, and J. Wang, "A modified normalized min-sum algorithm for LDPC decoding using order statistics," *Int. J. Satell. Commun. Netw.*, vol. 35, no. 2, pp. 163–175, Mar. 2017, doi: 10.1002/sat.1173.

[29] S. Varsamopoulos, B. Criger, and K. Bertels, "Decoding small surface codes with feedforward neural networks," *Quantum Sci. Technol.*, vol. 3, no. 1, Nov. 2017, Art. no. 015004, doi: 10.1088/2058-9565/aa955a.

[30] C. Chamberland and P. Ronagh, "Deep neural decoders for near term fault-tolerant experiments," *Quantum Sci. Technol.*, vol. 3, no. 4, Jul. 2018, Art. no. 044002, doi: 10.1088/2058-9565/aad1f7.
[31] M. P. C. Fossorier, M. Mihaljevic, and H. Imai, "Reduced complexity iterative decoding of low-density parity check codes based on belief propagation," *IEEE Trans. Commun.*, vol. 47, no. 5, pp. 673–680, May 1999.
[32] N. Raveendran, M. Bahrami, and B. Vasic, "Syndrome-generalized belief propagation decoding for quantum memories," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–6, doi: 10.1109/ICC.2019.8761366.
[33] K. Gunnam, J. M. Catala Perez, and F. Garcia-Herrero, "Algorithms and vlsi architectures for low-density parity-check codes: Part 2—Efficient coding architectures," *IEEE Solid StateCircuits Mag.*, vol. 9, no. 1, pp. 23–28, 2017.
[34] A. Darabiha, A. C. Carusone, and F. R. Kschischang, "Block-interlaced LDPC decoders with reduced interconnect complexity," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 55, no. 1, pp. 74–78, Jan. 2008.
[35] S. Scholl and N. Wehn, "Hardware implementation of a Reed–Solomon soft decoder based on information set decoding," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2014, pp. 1–6, doi: 10.7873/DATE.2014.222.
[36] S. Scholl, C. Stumm, and N. Wehn, "Hardware implementations of Gaussian elimination over GF(2) for channel decoding algorithms," in *Proc. Africon*, Sep. 2013, pp. 1–5, doi: 10.1109/AFRCON.2013.6757620.
[37] A. Bogdanov and M. C. Mertens, "A parallel hardware architecture for fast Gaussian elimination over GF(2)," in *Proc. 14th Annu. IEEE Symp. Field-Program. Custom Comput. Mach.*, Apr. 2006, pp. 237–248.
[38] B. Hochet, P. Quinton, and Y. Robert, "Systolic solution of linear systems over GF(p) with partial pivoting," in *Proc. IEEE 8th Symp. Comput. Arithmetic (ARITH)*, May 1987, pp. 161–168.
[39] A. A. Kovalev and L. P. Pryadko, "Quantum Kronecker sum-product low-density parity-check codes with finite rate," *Phys. Rev. A, Gen. Phys.*, vol. 88, no. 1, Jul. 2013, Art. no. 012311, doi: 10.1103/PhysRevA.88.012311.

JAVIER VALLS received the degree in telecommunication engineering from the Universidad Politecnica de Catalunya, in 1993, and the Ph.D. degree in telecommunication engineering from the Universitat Politecnica de Valencia, Spain, in 1999. He has been with the Department of Electronics, Universitat Politecnica de Valencia, since 1993, where he is currently a Professor. His current research interests include the design of FPGA-based systems, computer arithmetic, VLSI signal processing, and digital communications.

FRANCISCO GARCIA-HERRERO received the B.Sc. degree in telecommunication engineering from the Escuela Politecnica Superior de Gandia, Spain, in 2008, and the M.S. and Ph.D. degrees in electrical engineering from the Universitat Politecnica de Valencia, Spain, in 2010 and 2013, respectively. He has worked as a Lecturer and a Researcher at several universities, including the European University Miguel de Cervantes and the Universitat Politecnica de Valencia. He is currently an Associate Professor and a Researcher with the Universidad Antonio de Nebrija. His research interests include hardware and algorithmic optimizations of error-control decoders and fault-tolerance electronics in communication and storage systems.

NITHIN RAVEENDRAN (Member, IEEE) received the B.E. degree (Hons.) in electrical engineering from BITS Pilani, India, in 2011, and the M.Sc. (Engg.) degree from the Indian Institute of Science, Bengaluru, in 2015. He is currently pursuing Ph.D. degree in electrical and computer engineering with the University of Arizona, Tucson, AZ, USA. His research interests include classical and quantum error correction, iterative decoding techniques, signal processing, and coding and information theory.

BANE VASIĆ (Fellow, IEEE) is currently a Professor in electrical and computer engineering and mathematics with the University of Arizona and the Director of the Error Correction Laboratory. He is also an Inventor of the soft error-event decoding algorithm, and the Key Architect of a detector/decoder at Bell Labs data storage read channel chips which were regarded as the best in industry. His pioneering work on structured low-density parity check (LDPC) error correcting codes and invention of codes has enabled low-complexity iterative decoder implementations. Structured LDPC codes are today adopted in a number of communications standards and data storage systems. He is known for his theoretical work in error correction coding theory and codes on graphs which has led to characterization of the hard decision iterative decoders of LDPC codes and design of decoders with best error-floor performance known today. He is also a Co-Founder of Codelucida, a startup company developing advanced error correction solutions for communications and data storage. He is a Fulbright Scholar, a da Vinci Fellow, and the Past Chair of IEEE Data Storage Technical Committee.