Memristive circuits for LDPC decoding

Jussi H. Poikonen[1,2], Eero Lehtonen[2], Mika Laiho[2], Jonne K. Poikonen[2]
[1]Department of Communications and Networking (Comnet), Aalto University, Espoo, Finland
[2]Technology Research Center (TRC), University of Turku, Turku, Finland

***Abstract*—We present design principles for implementing decoders for low-density parity check codes in CMOL-type memristive circuits. The programmable nonvolatile connectivity enabled by the nanowire arrays in such circuits is used to map the parity check matrix of an LDPC code in the decoder, while decoding operations are realized by a cellular CMOS circuit structure. We perform detailed performance analysis and circuit simulations of example decoders, and estimate how CMOL and memristor characteristics such as the memristor OFF/ON resistance ratio, nanowire resistance, and the total capacitance of the nanowire array affect decoder specification and performance. We also analyze how variation in circuit characteristics and persistent device defects affect the decoders.**

I. Introduction

In this work, we consider the use of an emerging technology — *CMOL-type memristive circuits* — in decoding error correcting *LDPC codes*. This is a computational task of practical significance especially in digital communication and data storage systems. We begin by briefly outlining the considered technologies and their currently perceived significance, the motivation for the considered area of computing, and the specific objectives of this work.

A memristor — short for *memory resistor* — is a two-terminal resistive component, whose resistance changes as a function of the voltage across or the current through it. Theoretical concepts regarding memristors, which were postulated by Leon Chua in 1971 [1], can be used to explain the dynamical properties of various emerging memory devices based on resistive switching, such as resistive RAM and phase-change memories [2]; in this work we assume the term memristor refers to all such devices. Various different physical realizations of memristors have been reported, for example in [3]–[6]. A natural circuit topology for the use of memristors is a nanowire crossbar structure, where a memristive device is located at each crossing of two wires. This topology allows the realization of extremely dense non-volatile random-access memories, as each memory cell consists of a single memristor, whose feature size can already be scaled down to a few tens of nanometers [3]. Memristive memory technology has been reported to be approaching commercial viability as a replacement for Flash and DRAM memories [7], [8].

The *CMOL (CMOS/molecular hybrid)* [9] architecture facilitates the realization of a programmable communication network on top of CMOS integrated circuits by interfacing active CMOS components with passive memristive crossbars. Besides memory circuits, existing proposals for practical applications of CMOL circuits have focused especially on implementations of neuromorphic systems consisting of memdevice-based synapses and CMOS-based artificial neurons [10]–[12]. Memristive devices and systems can also be used to implement Boolean logic directly within memory arrays, as considered recently for example in [13]–[16]. CMOL-type realization of parallelized memristor logic is considered in [17].

In this work we utilize a specific benefit of the CMOL architecture: it allows programmable, nonvolatile, and area efficient connectivity between CMOS processing elements. A general drawback in the CMOL architecture is that addressing several memristors simultaneously in the associated nanowire crossbar is limited. However, we show that in the presented LDPC decoding approach this is not a significant limitation, as the considered decoding circuits consist of independent parallel processing cells, and require mainly global signaling.

LDPC decoding is one of the key enablers of near-capacity information transfer in modern digital communication systems and memory circuits. Practical application of these codes is facilitated by *iterative decoding*, which in essence allows a shift from a complex global decoding problem to operating a large number of relatively simple parallel processing elements with complex interconnections. In general, added interconnection complexity in LDPC coding improves code performance, but presents significant implementation challenges in energy-efficient decoder hardware realizations. In this work we demonstrate how the CMOL architecture can be used to implement reconfigurable iterative decoding with simple CMOS designs, yielding low power and energy consumption using a 130 nm CMOS technology.

Compared to conventional purely CMOS-based LDPC decoders, the considered designs benefit from removal of the physical routing between parallel processing elements, and related routing memory and programming, from the CMOS layer. Another benefit of this approach is that performing signal processing related to error correction coding directly within, for example, CMOL-type resistive memory circuits should be more efficient in terms of design, circuit area, and energy efficiency than passing all stored data to corresponding external processing circuits.

It should be noted that CMOL circuits are not yet a mature technology, which means that comprehensive measurements on their characteristics and performance are not currently available. In the following we apply published empirical data on memristive device characteristics when available, and consider analytically the effects of currently not well known circuit characteristics. Specifically, we assume memristor char-

acteristics from empirical results published in [6], where also integration of a nanowire array on top of a CMOS circuit was demonstrated. The computing approach considered in this work is not restricted to this single memristor technology, however; in principle, any resistive switching devices whose characteristics match the requirements outlined in the following could be applied.

A. Main objectives

A significant property of the CMOL architecture considered in this work is that it can be used to realize circuits which enable computations directly within memory structures. Examples of such circuits were recently considered for example in [12], where various CMOL-based memristive associative memories utilizing 1–2 nanowire layers for memory and computing purposes were presented. The primary objective of this work is to show how a single additional nanowire layer and relatively simple additional logic in the CMOS cells can be used to add basic LDPC decoding functionality to a CMOL-type memory/computing circuit. For example, the CMOS cells in the associative memory circuits presented in [12] already contain the components required for the current sum -based decoder presented in the following; in the CMOS layer its implementation thus requires mainly different control signal configurations and some additional switching logic. It is worth noting that the mathematical operations performed by the decoder considered in the following can also be used for LDPC encoding.

Secondary objectives of this work are:

- Where empirical data on CMOL circuit characteristics is not available, to approximate the numerical circuit and device parameter values required in practice to yield useful results. Specifically, in Sections II and III we consider how technology characteristics such as nanowire and memristor capacitance and OFF–ON resistance ratio limit the performance of the considered decoders. In Section V we analyze how inaccurate memristor programming, varying nanowire resistance, limited processing accuracy, and device defects affect decoding performance.
- To estimate the benefits and drawbacks of the CMOL architecture compared to traditional CMOS approaches in implementing decoding algorithms. In order to do this, we present in Section IV performance estimates and HSPICE simulations of the considered decoding circuits.

On the other hand, the focus of this work is not to produce a comprehensive study on CMOS cell designs required for different types of iterative decoding algorithms. Rather, we outline a basic design principle which can be used to add error correction functionality to CMOL circuits, and which may be used in later studies to implement decoders of varying complexity. To estimate the baseline performance of the considered design approach, we analyze and implement simple bit flipping iterative LDPC decoders with practical significance for example in ultra low power applications such as bio-compatible electronics [18] and energy-efficient wireless communication systems [19], [20]. Furthermore, such decoders are used for example in fibre-optic systems, where the communication channel can be approximated to be binary symmetric, and hard decision demodulation is assumed by default [21].

II. Background and assumed device characteristics

A. Principles of memristive devices

A memristor is in this work defined as a nonvolatile programmable resistor whose resistance can be changed by applying a voltage across, or current through, the device. The number of reliably programmable resistance states of a memristor depends on its fabrication, and the resulting dynamic device characteristics. For example, bistable devices called *binary memristors* have been reported in [4], [22], while an *analog memristor* with a seemingly continuous range of memristances is presented in [23].

In this work, we utilize binary memristors. The physical characteristics of the devices utilized in this work are based on those demonstrated empirically in [6]. For the decoding algorithms considered in the rest of this paper, it is not necessary to change the resistance states of memristors after initially programming them acccording to a specific LDPC code. Note, however, that the considered architecture allows for configuring the decoder according to any LDPC code within the dimensions of the circuit.

We define the memristor as a voltage controlled two-terminal bistable linear device having the on-resistance R_{ON} and the off-resistance R_{OFF}, where R_{OFF} is assumed to be much larger than R_{ON}. Correspondingly we say that a memristor is either in a conducting (ON) or a nonconducting (OFF) state. A memristor is programmed to ON-state by applying a voltage larger than a threshold voltage V_{TH} across it; correspondingly applying a voltage more negative than $-V_{\mathrm{TH}}$ programs the memristor to OFF-state.

In Section III-B we also consider the use of rectifying memristors, that is, memristors which pass significant current only in one direction. Such devices have been reported empirically for example in [5], [6]. These are programmed to OFF-state identically to nonrectifying devices by applying a voltage smaller than $-V_{\mathrm{TH}}$ across them. Memristor programming is a fundamental operation in CMOL circuits, and is not discussed in detail in this paper. However, we do analyze the effect of inaccurate programming, or variation in the resistance states R_{ON} and R_{OFF} on decoding performance in Section V.

We assume non-volatile memristors, which need to be programmed only once per LDPC code to be decoded. Thus this programming can be performed using accurate but relatively time-consuming methods such as cyclic programming, as discussed in [24], or the variation-tolerant algorithm presented in [25]. In the latter, the authors demonstrate memristor programming with less than 1 % deviation from the target conductances. Note that non-volatile binary memristors have been demonstrated for example in [4], where the reported devices retain their resistance state more than 10 years at 85 °C.

Memristors are naturally fabricated within a nanowire crossbar where a memristor is placed at each crossing of two nanowires [26]. The nanowires are driven by CMOS circuitry

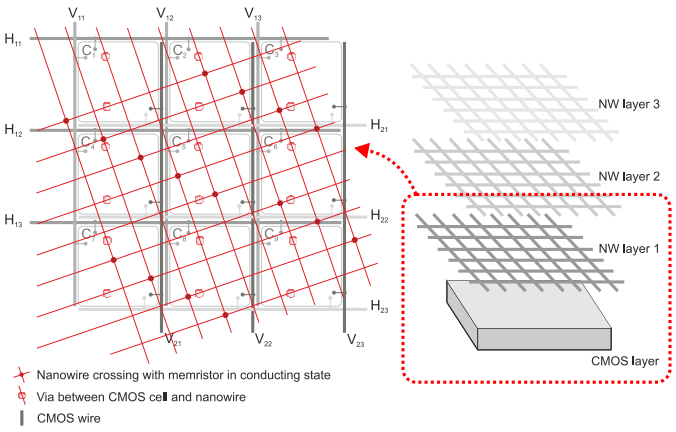

Fig. 1. Conceptual structure of a CMOL circuit with multiple vertically stacked nanowire layers. Note that in the left inset, vias are depicted only between the CMOS cells and the first nanowire layer; in practice, separate vias are realized for each layer. The CMOS cells may use the different nanowire layers for various purposes, for example for storing data. In this work we show how a single nanowire layer can be used to implement LDPC decoding. Designs for the CMOS cells denoted here by C_i are presented in Figures 5 and 9.

which is proposed to be located physically below the nanowire crossbar [9]. This CMOS/molecular hybrid (CMOL) architecture makes memristive nanowire crossbars ideally suited for facilitating programmable communication and memory within a CMOS chip.

B. CMOL circuits

An example of a CMOL-type CMOS/memristor hybrid architecture [9], [27] is illustrated in Fig. 1. It consists of a CMOS layer stacked vertically with a number of nanowire layers, or memristive crossbars. For simplicity, we consider in this work only one nanowire layer connected to the CMOS cells. The CMOS layer is used for signal restoration, gain and processing, and it is interfaced with the memristive crossbar — which acts as a memory layer and as a programmable communication network for the chip — by vertical vias. The CMOS layer is divided into cells whose design depends on the functionality of the system. For example, if a CMOL architecture is used as a random access memory, the CMOS cells comprise merely of pass-transistors, which are used to select two mutually perpendicular nanowires and thus the memristor located between them. CMOS cell designs facilitating LDPC decoding in the CMOL architecture are demonstrated in Section III, and illustrated in Figures 5 and 9.

As illustrated in Fig. 1, each CMOS cell can drive one horizontal and one vertical nanowire, and thus two CMOS cells are required to address a single memristor located at the crossing of a horizontal and a vertical nanowire. To simultaneously control only these two CMOS cells, four CMOS wires are needed — thus the CMOL architecture generally requires double addressing compared for example to an array of CMOS cells. Note also that the nanowire array is slanted with respect to the CMOS cell array to allow N vertical and N horizontal nanowires — or N^2 memristors — to be controlled by N CMOS cells. A limitation of this architecture is that it is not possible to address an arbitrary set of memristors; for example, to program two memristors simultaneously, it is necessary to drive two horizontal and two vertical nanowires, and there are four memristors located at the crossings of these wires. However, the decoding circuits described in this article require mainly global control signals, and it is not necessary to address individual memristors during the decoding, which means that the considered decoding algorithms are well suited for the CMOL architecture.

C. Device parameters

1) Memristors: We assume memristor characteristics corresponding to the devices presented empirically in [6]. Specifically, we consider the use of rectifying memristors located at the crossings of perpendicular nanowires of width 50 nm, separated vertically by a 20 nm memristive switching layer of amorphous silicon. The reported ratio of memristor ON- and OFF-resistances is of the order of 10^3 with ON-resistances of order 10^5 Ω and OFF-resistances of order 10^8 Ω. In this work we assume nominal resistance values $R_{\text{ON}} = 500$ kΩ and $R_{\text{OFF}} = 500$ MΩ; the effect of variation in these values caused by inaccurate programming or device variation is analyzed in Section V. As will be shown in Section III, the ratio of the memristors' OFF- and ON-resistances limits the maximum LDPC code block size decodable in parallel using a single nanowire crossbar. Therefore, it is desirable to have as large an OFF—ON resistance ratio as possible.

2) Nanowire capacitance: Currently, there is little empirical data available on memristor capacitances. We estimate the parasitic capacitance of a single memristor as a parallel plate capacitance

$$C_{\text{pp}} = \varepsilon_0 \varepsilon_{\text{aSi}} A / d_v \approx 13 \text{ aF}, \tag{1}$$

where $\varepsilon_0 \approx 8.85419 \cdot 10^{-12}$ is the vacuum permittivity, $\varepsilon_{\text{aSi}} \approx 11.8$ is the relative permittivity of amorphous silicon, $A = (50 \text{ nm})^2$ is the area of the memristor, and $d_v = 20$ nm is the vertical plate distance, or dielectric thickness.

Furthermore, we approximate the capacitance between two parallel nanowires as

$$C_w = \pi \varepsilon_0 \varepsilon_{\text{SiO}_2} L / \cosh^{-1}(d_h / t), \tag{2}$$

where L is the length of a nanowire, and it is assumed that a nanowire's thickness $t = 50$ nm is the same as its width. The distance between parallel nanowires is d_h, and we assume that the nanowires are separated by a silicon dioxide substrate with relative permittivity $\varepsilon_{\text{SiO}_2} \approx 3.9$. Note that the distance d_h depends on the area of the CMOS cells in the CMOL structure; as presented in Section IV-A, our estimate for the sizes of the example cells implemented with 130 nm technology is approximately 50 μm $\times$ 50 μm. We can thus approximate the distance between parellel nanowires as $d_h \approx 50$ μm$/\sqrt{N}$, where N is the number of memristors along a nanowire. We estimate the total capacitance of a nanowire as

$$C_{\text{tot}} = N C_{\text{pp}} + 2 C_w, \tag{3}$$

taking into account N parallel plate capacitances and capacitances to two neighboring parallel wires. Fig. 2 illustrates the magnitudes of the estimated capacitances as functions of N, when we assume that the nanowire length $L = N d_h$.

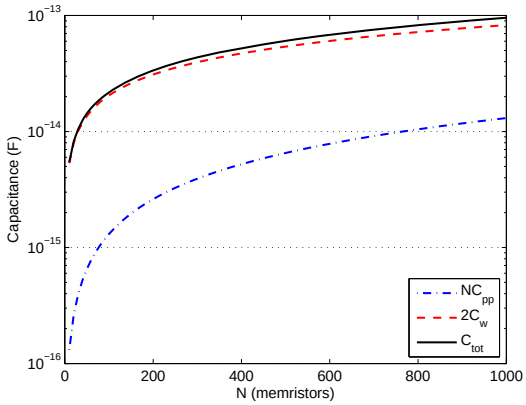

Fig. 2. Estimated nanowire capacitance as a function of the LDPC codeword length N.

This total capacitance estimate is used in Section IV-B to approximate the decoding speed and corresponding decoder energy consumption.

3) Nanowire resistance: The decoding methods presented in Section III are essentially based on measuring simultaneously the resistances of several memristors on a nanowire. The resistances of the nanowires affect these measurements, as the total resistance depends on the length of the associated current path, and will vary over the decoder circuit. We estimate the maximum resistance from nanowires — corresponding to a current path between diagonally opposite corners of the nanowire array — as

$$R_{\text{nw,max}} = 2Nd_h\rho_{\text{Ag}}/A, \tag{4}$$

where Nd_h is again the length of a single nanowire and silver nanowires with resistivity $\rho_{\text{Ag}} = 1.6 \cdot 10^{-8}$ Ω-m are assumed, as well as a rectangular nanowire cross-section with area $A = (50\text{ nm})^2$. If we assume that the maximum nanowire length is of the order of 1 mm, this maximum resistance will be of the order of 10 kΩ, which is 2 % of the assumed memristor ON-resistance. For the considered decoding approaches, it is not the maximum nanowire resistance, but the deviation from mean resistance which may cause decoding errors. This deviation is always smaller than $R_{\text{nw,max}}/2$, which corresponds in this case to less than 1 % of the memristor ON-resistance. Worst-case effects of nanowire resistance variation are considered in more detail in Section V.

D. LDPC codes and decoding

A Low-Density Parity Check code is described completely by its *parity check matrix* H. The columns of H represent the bits b_i – often called *variables* – in an LDPC codeword, while the rows of H specify a number of *parity checks*: $H(k,l) = 1$ when bit l is included in the k:th parity check, and each check consists of an XOR operation over the included bits. We denote the *row weight*, or number of input bits participating in the k:th parity check by ρ_k, or simply ρ, when all row weights are assumed to be equal. Correspondingly, the column weight, or number checks corresponding to bit l, is denoted by γ_l or γ. The LDPC code construction problem is to find good parity check matrices H in terms of theoretical error correction capability and coding/decoding performance. Detailed information on LDPC codes can be found for example in [28].

By definition, the parity check matrices of LDPC codes have *low density* – all rows and columns are *sparse*, which allows for efficient decoding. In accordance with Shannon's random coding principle, good LDPC codes can be generated by selecting H pseudorandomly according to some constraints on the row and column densities. However, encoding randomly generated codes is complex, and analysis of the properties and theoretical performance of such codes is difficult. It is thus beneficial to have random-like LDPC codes which can be constructed algebraically, and contain a regular strucure such as check matrix rows which are cyclic shifts of each other. Especially quasi-cyclic (QCLDPC) codes [29], where H consists of a number of circulant submatrices, are currently widely used, as they are relatively simple to construct and implement, and provide good error correction performance. These are also well suited for the decoders considered in this work, as will be shown.

Bit flipping algorithms are the simplest approach to iterative decoding of LDPC codes. As an example of bit flipping, consider an LDPC codeword where one bit b_k is erroneous. Suppose that in the parity check matrix H of the considered code, the weight, or number of ones, of the k:th column is γ_k. The single error will then cause γ_k parity checks to fail. In a well designed LDPC code, no other bit is involved in all the same parity checks as b_k, meaning that the number of unsatisfied checks per bit will be larger for bit k than for any other bit. A bit flipping decoder may look for the largest number of unsatisfied parity checks, and in this case flip bit b_k, producing the correct codeword. Allowing for the possibility of several erroneous bits in a codeword, this same procedure can be implemented in an iterative fashion, if the code is well designed.

Let $\mathbf{x}(0)$ be the undecoded input vector. For $n \geq 0$, the mathematical operations performed by the decoders described in the following are

$$\mathbf{c}(n) = \mathbf{x}(n)H^T \pmod 2, \tag{5}$$

$$\mathbf{x}(n+1) = \mathbf{x}(n) + f_\Theta(\mathbf{c}(n)H) \pmod 2, \tag{6}$$

where $\mathbf{c}(n)$ is a row vector containing the parity check results for all rows of H, and $f_\Theta(\mathbf{v})$ is a vector whose ith element is 1 if the ith element of $\mathbf{v}$ is greater than or equal to Θ and 0 otherwise.

The threshold Θ can be selected for example to flip a fixed fraction of the bits corresponding to the largest numbers of unsatisfied checks. Also, it is possible to weigh the bits in the codeword according to external reliability information, and use this to find the most probable erroneous bits at each iteration [30], [31]. Such decoding approaches the principle of belief propagation, where the decoder operates with continuous-valued estimates of the likelihoods of bit values instead of quantized bits. This requires more information on the transmission channel than bit flipping, and updating the likelihood values at each iteration is relatively complicated. However, belief-propagation produces near-optimal decoding.

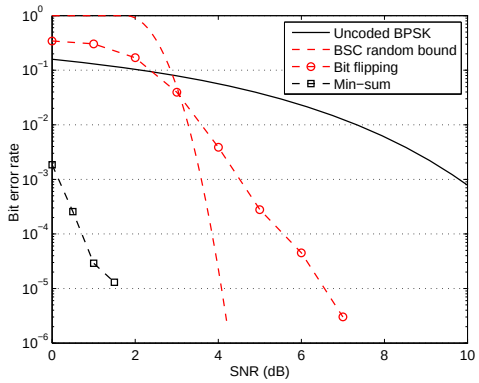

Fig. 3. Bit error rates versus signal-to-noise ratio corresponding to uncoded binary phase shift keying (BPSK), the random coding bound [32] for the binary symmetric channel (BSC), and a randomly generated LDPC code [28] of codeword length 500 bits with $\rho = 6,\ \gamma = 3$, decoded with bit flipping (BF) and min-sum (MS) algorithms.

In terms of bit error rate (BER) versus signal-to-noise ratio (SNR), the performance of an LDPC code is crucially dependent on the size and structure of the parity check matrix, and on the algorithm used for decoding. An example of this is presented in Fig. 3, where the bit error rate of an LDPC code of length 500 bits is shown for bit flipping (BF) and min-sum (MS) decoding. For comparison, the random coding bound [32] is shown; this represents the optimal BER performance of a code with this length when the related communication channel is binary symmetric, that is, no soft channel information is available. The asymptotic distance between the BF decoding performance and the random coding bound demonstrates that the structure of the simulated code is not optimal. On the other hand, the distance between the random coding bound and the MS decoding performance shows that if soft channel information is available, its use in more complicated decoding algorithms provides improvement in BER performance. A related tradeoff is that near-optimal decoders typically consume significantly more power and energy per decoded bit than for example bit flipping decoders. Therefore the choice of decoder type generally depends on the application-specific energy budget and power limitations. Please note that the discussion above applies directly to the performance comparisons presented in Section VI.

In the following we outline two approaches to decoding LDPC codes in a CMOL circuit structure. In the first, computations are based on measuring current sums in the CMOS cells, and in the second, computations are performed using digital logic. To be efficient, the digital approach requires the decoded LDPC code to have certain characteristics specified in Section III-B; QC-LDPC codes are particularly well suited for this decoder.

In both of these decoding approaches, the $M \times N$ parity check matrix H of the LDPC code is represented by the states of memristors at the crossings of M horizontal and N vertical nanowires, that is, memristor $m_{i,j}$ is in a low resistance (ON) state if $H(i,j) = 1$, and in a high resistance (OFF) state otherwise. We assume in the following that the CMOL circuit has been initialized in this way according to a given parity check matrix, and do not explicitly consider the circuitry required for programming the memristors. The states of the memristors do not change during the following decoding procedures.

III. CMOS DECODER CELL DESIGNS

Let us first consider the basic principle according to which the decoder circuits, discussed in following sections, operate. As illustrated in the example of Fig. 4, the parity check matrix is mapped into the states of memristors in a nanowire crossbar. Assume that we now keep the horizontal nanowires at a fixed voltage, while driving to a higher voltage the vertical nanowires corresponding to logical ones in an input codeword. This results in a current flow through every memristor located at a driven vertical wire. At horizontal wire k, the sum current can be written as

$$I_k = V_{\text{mem}}\mathbf{x}(H_k^T/R_{\text{ON}} + (\mathbf{1}^T - H_k^T)/R_{\text{OFF}}) \quad (7)$$

$$\Rightarrow I_k R_{\text{ON}}/V_{\text{mem}} = \mathbf{x}(H_k^T + (1 - H_k^T)R_{\text{ON}}/R_{\text{OFF}}), \quad (8)$$

where H_k is the kth row of H, $\mathbf{1}$ is a row vector of ones, and V_{mem} is the nonzero voltage across a memristor. If the length of an input codeword is N, then

$$\mathbf{x}H_k^T \leq I_k R_{\text{ON}}/V_{\text{mem}} \leq \mathbf{x}H_k^T + NR_{\text{ON}}/R_{\text{OFF}}. \quad (9)$$

Assuming that $NR_{\text{ON}}/R_{\text{OFF}} < 1$, the parity check of row k is obtained as $\lfloor I_k R_{\text{ON}}/V_{\text{mem}} \rfloor \pmod 2$, which is in the following computed using a parity A/D converter explained below.

Similarly to the above, in the second phase of decoding, vertical nanowires are kept at a fixed voltage, while the horizontal wires corresponding to failed parity checks are driven to a higher voltage. As above,

$$\mathbf{c}H^l \leq I^l R_{\text{ON}}/V_{\text{mem}} \leq \mathbf{c}H^l + MR_{\text{ON}}/R_{\text{OFF}}, \quad (10)$$

where H^l is the lth column of H, and I^l is the current at vertical nanowire l. Thresholding $I^l R_{\text{ON}}/V_{\text{mem}}$ yields an approximation of (6). In the following subsection we describe a simple CMOS circuit which implements the above specified operations. Note that the condition

$$NR_{\text{ON}}/R_{\text{OFF}} < 1 \Leftrightarrow N < R_{\text{OFF}}/R_{\text{ON}} \quad (11)$$

sets a fundamental limitation on length of a codeword decodable using the considered approach in a monolithic, or non-segmented, nanowire array.

A. Current sum -based decoder cell

Fig. 5 illustrates the structure of a CMOS cell which can be used in bit flipping decoding of an arbitrary LDPC code. The operating principle of this circuit resembles that of the search operation specified and demonstrated for a memristive autoassociative content addressable memory in [12]. A single iteration in the decoding process is divided into two phases which in the following are referred to as *row checks* and *bit flips*.

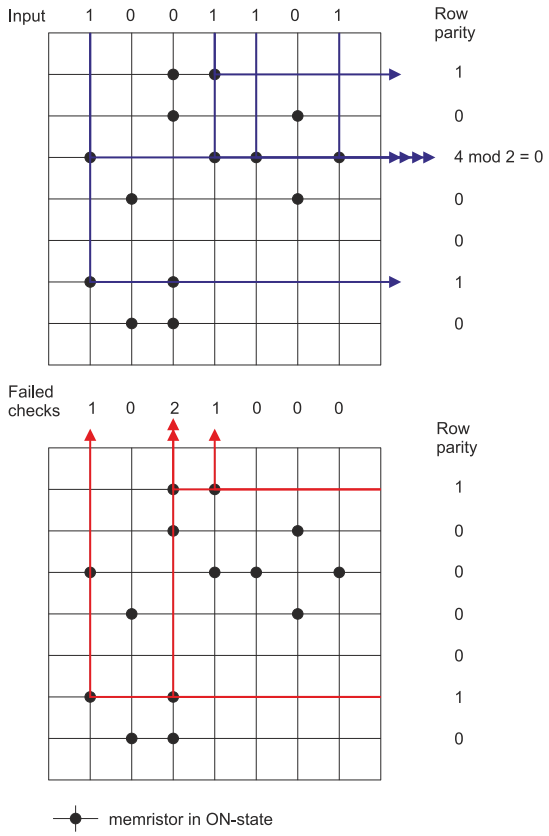

Fig. 4. Principle of the considered LDPC decoding. Top: in the parity check phase, modulo-2 sums of currents flowing through memristors in ON-state are computed in parallel for all rows of the nanowire array. Bottom: in parallel for all columns of the nanowire array, bit flip decisions are made based on the numbers of failed parity checks from the first decoding phase.

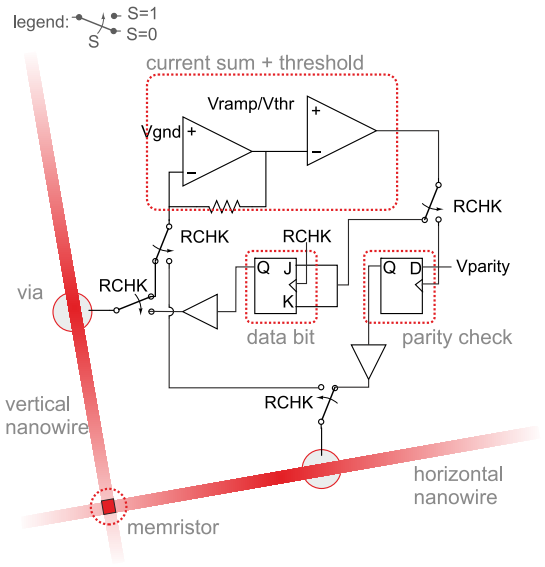

Fig. 5. Operating principle of the analog decoding cell. The decoding phase is selected by the global control signal RCHK: when RCHK=1, row checks are computed, and when RCHK=0, bit flipping is performed.

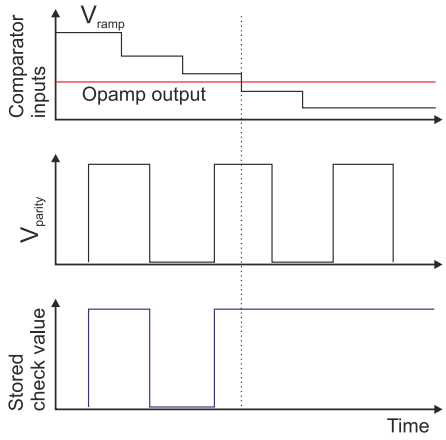

Fig. 6. Principle of the ramp A/D conversion. Top: the comparator inputs are the opamp output and the ramp voltage; the latter is selected to distinguish between opamp outputs corresponding to the currents I_k specified in (9). Middle: V_{parity} is selected as the parity of the current value of the ramp signal. Bottom: the output of the comparator gates the D flip flop; once the ramp voltage falls below the opamp output at the time indicated by the dotted vertical line, the current logical value of V_{parity} is stored as the result of the parity check.

In the first decoding phase, the global signal RCHK $= 1$. Each CMOS cell drives its vertical nanowire to V_{DD} if the value of the codeword bit stored in the JK flip flop is 1, otherwise the vertical nanowire is driven to V_{gnd}. Simultaneously, the negative feedback of the op-amp keeps the horizontal nanowires at V_{gnd}, and the incoming currents from the vertical nanowire drivers are summed and converted into a voltage by the transresistance opamp circuit. To obtain the parity check values for each row, a *parity analog-to-digital conversion* is performed by comparing the opamp output voltage corresponding to the row current sum against a ramp voltage signal denoted as V_{ramp}, which goes through the possible row sum values. The result of this comparison is used to gate a D flip flop, whose input V_{parity} varies according to the ramp signal parity. The principle of this parity A/D conversion and the corresponding control signals are illustrated in Fig. 6. As this process is performed in all CMOS cells simultaneously, this configuration realizes the modulo 2 matrix multiplication specified in equation (5).

In the second decoding phase, the global signal RCHK $= 0$. The CMOS cells drive the horizontal nanowires according to the stored check bit values, obtained in the previous decoding phase. Again, the resulting sum currents at the vertical nanowires — representing the numbers of unsatisfied checks per bit — are measured and thresholded. If the measured voltage is below the global threshold voltage V_{thr}, the codeword bit stored in the JK flip flop is flipped. Over the whole array of cells this realizes the operation specified in equation (6). In each decoding iteration, the threshold voltage is the same for all CMOS cells, but may vary between decoding iterations. Selection of predefined threshold voltage sequences is discussed in more detail for example in [31].

Note that in memristive crossbar architectures, as considered in this work, *sneak path current leakage* must be taken into account. This term refers to undesired current passed through memristors connected to nanowires which are not driven to a specific voltage; such currents may disrupt the operation of the circuit. In the current sum -based decoder design described above, sneak path leakage is prevented as in [12] by always

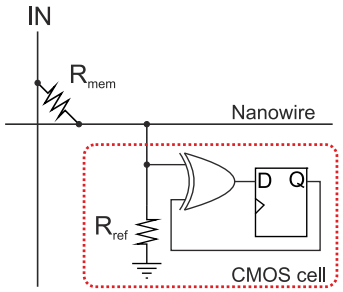

Fig. 7. Principle of computing parity at one horizontal nanowire using digital logic. At the CMOS cell, resistance R_{ref} is selected to enable voltage division so that when the vertical nanowire is driven to a high voltage (IN $= V_D$) and the memristor is in ON-state ($R_{\text{mem}} = R_{\text{ON}}$), the voltage at the XOR gate input connected to the horizontal nanowire is higher than the switching threshold of the XOR gate. Otherwise the voltage at the XOR input remains close to ground. The other input of the XOR gate is the previous result of the XOR operation (initially zero). By sequentially driving all vertical nanowires connected to this horizontal nanowire, the row parity is computed. When inactive, the vertical nanowires are at high impedance (IN = HZ).

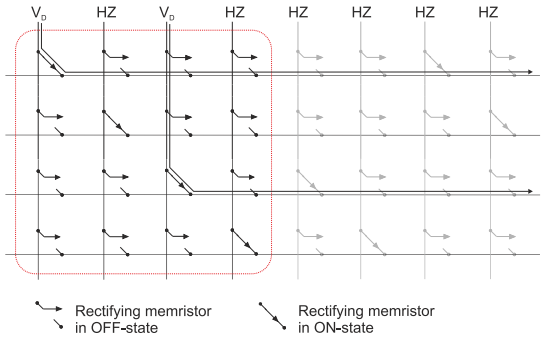

Fig. 8. Example of a block-based LDPC code structure allowing parallelization of parity check computation over vertical nanowires. The parity check matrix is divided into submatrices, or blocks, of size 4×4, where the row weight of each submatrix is one. The parity checks are now performed by allowing vertical nanowires to be driven only in one code block at a time. Thus the current at any horizontal nanowire may only be close to zero — corresponding to a low voltage at the corresponding CMOS cell's XOR gate input — or close to that passed by a single memristor in ON-state — corresponding to a high voltage at the XOR input. In this example, input bits one and three have logical value one, which results in significant current at horizontal nanowires one and three, as the first code block is as identity matrix. The memristors must be rectifying, as otherwise stray currents would flow from horizontal wires to vertical wires in high impedance state through memristors in ON-state.

connecting all nanowires either to the virtual ground voltage V_{gnd} or to V_{DD}, ensuring that currents are generated only according to the desired computations.

B. Digital decoder cell

The main advantage of the opamp-based processing described above is that it allows summations in current domain, in parallel at all rows or columns of the nanowire array. A drawback in this is the analog-to-digital conversion required to process the sum currents: especially in the parity check phase there is a tradeoff between the maximum number of different current sum levels — corresponding to the row weight ρ of the LDPC code — and the complexity and operation speed of the CMOS cell. As the row parity can be defined as a sequence of logical XOR operations, it is natural to consider its computation using digital logic.

Fig. 7 illustrates the principle of a digital decoding cell with a single vertical and a single horizontal nanowire. A reference resistor R_{ref} is used to implement voltage division so that when the vertical nanowire is driven with a large voltage and the memristor is in ON-state, the XOR gate input connected to the horizontal nanowire is at a high voltage, corresponding to logical 1. The other input of the XOR gate is connected to a flip flop which maintains the result of the previous XOR operation. By sequentially driving to a high voltage V_D all vertical nanowires corresponding to input bits with logical value 1, while keeping all other vertical wires at high impedance, the parity over all horizontal nanowires can be computed in parallel. Note however that in this case the memristors must be rectifying, as otherwise undesired currents would be passed from horizontal nanowires to vertical nanowires in high impedance state, preventing correct operation.

The main problem with this approach is that while it is possible to compute the parity of all rows in parallel, N steps are required to compute the row parities. However, this drawback can be significantly reduced with suitable LDPC code structures such as quasi-cyclic codes, where the parity check matrix H consists of $\rho \times \gamma$ submatrices obtained as cyclic shifts of an $L_B \times L_B$ identity matrix. As demonstrated in Fig. 8 with $L_B = 4$, with such coding, all vertical nanowires within each block of L_B bits can be driven simultaneously according to their associated input bits. Within a block, only one memristor in ON-state per horizontal nanowire can be driven, which ensures the correct logical value at the XOR input. Thus, only ρ sequential XOR steps are required to compute the parity for all rows.

Note that with QCLDPC codes the parity check matrix H and its transpose H^T have the same structure. This means that the above described approach to computing the row parities can be applied directly in computing the total numbers of failed parity checks in parallel over all vertical nanowires. In this case the single XOR gate is replaced with a counter and comparison to a threshold. Furthermore, due to the use of rectifying memristors it is necessary in this case to drive horizontal nanowires corresponding to check bits with logical value 1 to a low voltage V_{SS}, while the reference resistor is connected to a high voltage V_D. Finally, V_D should generally be smaller than the memristor programming threshold voltage V_{TH} to avoid changing the memristors' states during decoding. In the following simulations we assume $V_{\text{D}} = 0.8V_{\text{DD}} = 0.96$ V.

Fig. 9 shows the structure of a digital decoding cell which operates according to the principle described above. Resistors R_1 and R_2 are implemented using NMOS and PMOS transistors in saturation. The target resistance values for these are specified in Section IV-B. Selection of the cells which drive their nanowires at any given time is in Figure 9 controlled by the signal SEL. In practice, small local memory registers and some additional logic can be used to realize this selection.

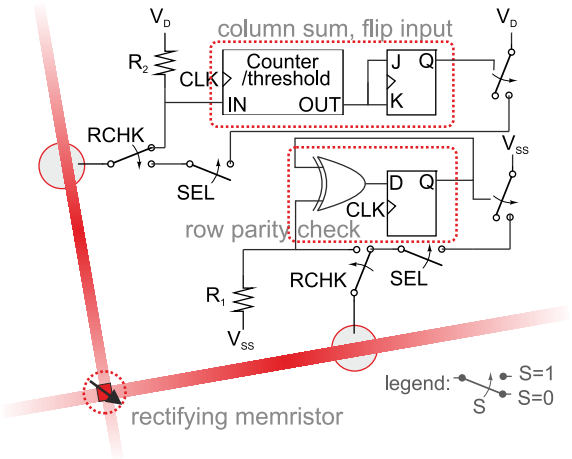

Fig. 9. Operating principle of the digital decoding cell. The global signal RCHK functions as in the analog decoder cell of Fig. 5. The local control signal SEL is used to select whether or not the cell drives its vertical (horizontal) nanowire in the row check (bit flip) phase. The JK and D flip flops store the input bits and parity check bits, respectively.

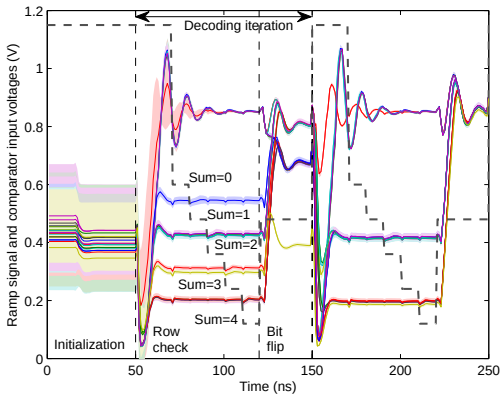

Fig. 10. Simulation example of analog decoding. Mean comparator input voltages (solid lines) and their standard deviations (shaded areas) obtained from 10 Monte Carlo simulations are plotted for 20 decoding cells along with the global ramp/threshold signal $V_{\text{ramp}}/V_{\text{thr}}$ (dashed line). In the first 50 ns, the input bits are programmed to the cells; during this time the comparator inputs are undetermined, which explains the large standard deviations of the input voltages. After this initialization, two decoding iterations are shown. The input vector contains one erroneous bit, which is seen in the first row check phase as odd-valued comparator inputs. The single error is corrected in the first bit flip phase, and in the second row check phase all comparator inputs are even-valued.

IV. Simulations and Performance Analysis

In the following we first present simulation results for the CMOS cells described in the previous section. The main objective in these simulations is to verify that the circuits operate as intended, and to obtain area and power consumption estimates for the considered CMOS cells, corresponding to a specific technology. In the second subsection below, we estimate analytically the average power and energy consumption of the considered decoding approaches as functions of the code parameters and assumed hardware characteristics.

A. Circuit simulations using a 130 nm CMOS technology

Decoding circuits corresponding to LDPC codeword lengths 20, 40, and 80 bits were simulated using 130 nm CMOS technology with $V_{\text{DD}} = 1.2$ V. Memristors were modeled using fixed-valued resistors and capacitors selected according to the device characteristics specified in Section II-C.

Fig. 10 illustrates the operation of the analog decoding cell outlined in Section III-A. Decoding is here demonstrated using a QCLDPC code with a 15×20 parity check matrix with row weight $\rho = 4$ and column weight $\gamma = 3$, with one erroneous bit in the input codeword. Fig. 10 shows the comparator input voltages in all 20 CMOS cells over two decoding iterations. The ramp signal used as a threshold in both the parity check and the bit flip phase is depicted as the dashed curve. The colored line plots are averages over 10 Monte Carlo simulations taking device mismatch into account; the sample standard deviation of each voltage curve is presented by a shaded area. Here we do not consider the variances of the memristors' ON and OFF resistances; the effect of inaccuracy in memristor programming with respect to the accuracy required for correct operation of the analog decoding cell is analyzed in detail in Section V. Fig. 10 shows how the decoder corrects the input codeword in a single iteration, with a decoding iteration duration of 100 ns. This consists of 70 ns for the parity check phase, and 30 ns for the bit flip phase.

Note that in the considered example with row weight four, the separation between voltage levels with the presented decoder circuit is larger than 100 mV, and typically the row weights in well performing LDPC codes are of order 10 or less. For example, in [33] quasi-cyclic LDPC codes with row weight 8 which perform within 1 dB of the optimal random code bound are presented. Thus in practically significant decoding cases the required voltage separation is of the order of several tens of millivolts, achieved even by the presented simple CMOS cell design. Of course, practical circuits will contain also other sources of error than the considered device mismatch. In Section V-A we determine the required resolution, taking into account the worst-case effects of memristor programming inaccuracy, nonzero nanowire resistance, and statistical variation in the input Hamming weight in the current sum -based decoder.

The digital decoding cell of Fig. 9 was also simulated using 130 nm technology. The memristor characteristics were assumed to be the same as with the analog decoding cell, with the additional assumption that the memristors are rectifying. The counter/threshold logic block was realized in this example as a three-bit adder and register whose carry overflow bit is used as the flipping indicator corresponding to a global three-bit threshold signal. With the digital decoder, circuit simulations corresponding to a QCLDPC code with row weight 4, column weight 3, and codeword length 20 were performed. With this code length, the duration of both the XOR and the counter/threshold operation over a block of $L_B = N/4$ bits was 2.5 ns. As one full decoding iteration consists here of four sequential XOR operations and three additions, the simulated minimum duration of one decoding iteration is 17.5

ns. A 400 MHz clock frequency is sufficient for both of the considered decoding approaches; in the opamp-based analog decoder this clock frequency allows for switching between the required ramp voltage levels shown in Fig. 10.

The simulated average power consumption of the opamp-based CMOS cell is approximately 78 μW; most of this power is consumed by the opamp and subsequent comparator. The average power consumption of the digital decoding cell is approximately 14 μW. Circuit areas of the CMOS cells were estimated using a layout tool to place all transistors without wiring inside a minimal area. For the opamp-based cell, this minimum area was 25 μm $\times$ 25 μm; to approximate the total area requirement we double these dimensions, obtaining an area estimate of 50 μm $\times$ 50 μm, or 2500 μm^2. The individual components requiring most circuit area are the opamp and the associated feedback resistor. For the digital decoding cell, the corresponding minimal area estimate is 22 μm $\times$ 22 μm, yielding an approximate total area of 2000 μm^2.

B. Performance analyses

In the following we estimate analytically the minimum power consumption and energy per decoded bit corresponding to the decoding principles described in Section III. The power consumption in the full CMOL decoder circuit can be divided into power dissipated in the memristors and resistive CMOS cell elements along the measured current path, and power consumed by other circuit elements in the CMOS cells. The CMOS cell power consumption depends on the specific implementation and technology used in the cells; in the following we assume the values given in the previous subsection.

For estimating the power consumption in the memristors, in the following we assume that the CMOL circuit is configured according to an $M \times N$ parity check matrix with constant row and column weights ρ and γ, respectively, and that a bit in the input codeword has value 1 with probability 0.5. These assumptions are made only to simplify the following analytical expressions; the results are easily generalized for arbitrary parity check matrices and input codewords. We first consider the performance of the opamp-based decoder, and repeat the analysis for the digital decoder.

1) Opamp-based decoding cell: From (7), the mean current at row k of the nanowire array in the parity check phase is

$$\begin{aligned}\mathbb{E}\left[I_k\right] &= V_{\text{mem}}\left(\mathbb{E}\left[\mathbf{x}H_k^T\right]/R_{\text{ON}} + \mathbb{E}\left[\mathbf{x}(\mathbf{1}^T - H_k^T)\right]/R_{\text{OFF}}\right)\\ &= \frac{1}{2}V_{\text{mem}}\left(\rho/R_{\text{ON}} + (N-\rho)/R_{\text{OFF}}\right).\end{aligned}$$

As this mean current is the same for all rows, we may leave out the index k. Since vertical nanowires are driven to voltage V_{DD}, and there are in total M active rows in the nanowire array, the mean total power consumption of the opamp-based decoder can be written as

$$\mathbb{E}\left[P_{\text{check}}\right] = N\,\mathbb{E}\left[P_{\text{cell}}\right] + \frac{1}{2}MV_{\text{DD}}V_{\text{mem}}\left(\frac{\rho}{R_{\text{ON}}} + \frac{N-\rho}{R_{\text{OFF}}}\right), \tag{12}$$

where $\mathbb{E}\left[P_{\text{cell}}\right]$ is the mean power consumed by a single CMOS cell and $V_{\text{mem}} = V_{\text{DD}} - V_{\text{gnd}}$ is the voltage across each memristor connected to a vertical nanowire whose input data bit has value 1.

With similar reasoning, the mean total power consumed in the second decoding phase is

$$\mathbb{E}\left[P_{\text{flip}}\right] = N\,\mathbb{E}\left[P_{\text{cell}}\right] + N_{\text{fail}}V_{\text{DD}}V_{\text{mem}}\left(\frac{\rho}{R_{\text{ON}}} + \frac{N-\rho}{R_{\text{OFF}}}\right), \tag{13}$$

where N_{fail} is the number of failed parity checks, that is, rows driven to V_{DD}.

Based on the above, we obtain the mean energy required per decoded bit as

$$\mathbb{E}\left[E_{\text{bit}}\right] = \left(\mathbb{E}\left[P_{\text{check}}\right]T_{\text{check}} + \mathbb{E}\left[P_{\text{flip}}\right]T_{\text{flip}}\right)N_{\text{iter}}/N, \tag{14}$$

where T_{check} is the duration of the first decoding phase, T_{flip} is the duration of the second decoding phase, and N_{iter} is the number of decoding iterations performed — this number depends on the bit error probability of the input data. We take into account the increase of capacitance along with code length by selecting the phase durations as $T_{\text{check}} = 70$ ns $+ \tau_{\text{check}}$ and $T_{\text{flip}} = 30$ ns $+ \tau_{\text{flip}}$, according to the simulated decoding phase durations and time constants $\tau = R_{\text{Thevenin}} \cdot C_{\text{tot}}$ consisting of the Thevenin resistance of the current path in the two decoding phases, and the total nanowire capacitance as specified in Section II-C.

2) Digital decoding cell: In the digital decoding circuit, mean current at a given nanowire is equal either to the current passed through a block of wires with no memristor in ON-state driven, or to the current passed through a block with a single memristor in ON-state driven. The mean block resistances equivalent to these cases in the parity check phase are

$$\begin{aligned}&\mathbb{E}\left[R_{\text{B},0}\right] = 2R_{\text{OFF}}/L_B \text{ and}\\ &\mathbb{E}\left[R_{\text{B},1}\right] = 2R_{\text{ON}}R_{\text{OFF}}/\left(2R_{\text{OFF}} + R_{\text{ON}}(L_B - 1)\right).\end{aligned}$$

The total resistance between the driven voltage V_D and ground for a single nanowire is then $R_{\text{B},0} + R_{\text{ref}}$ or $R_{\text{B},1} + R_{\text{ref}}$, where R_{ref} is the reference resistor from Fig. 7.

According to our initial assumptions, in the parity check phase, the block resistances $R_{\text{B},0}$ and $R_{\text{B},1}$ occur with equal probability. Thus the mean total power consumption over all horizontal nanowires can be estimated as

$$\begin{aligned}&\mathbb{E}\left[P_{\text{check}}\right] \approx\\ &N\,\mathbb{E}\left[P_{\text{cell}}\right] + \frac{1}{2}MV_{\text{D}}^2\left(\frac{1}{\mathbb{E}\left[R_{\text{B},1}\right] + R_{\text{ref}}} + \frac{1}{\mathbb{E}\left[R_{\text{B},0}\right] + R_{\text{ref}}}\right).\end{aligned} \tag{15}$$

Note, however, that this expression is only an approximation, as it is a nonlinear function of mean resistance values. Furthermore, to maximize the voltage swing at the XOR gate input, the target value for the reference resistor is selected as

$$R_{\text{ref}} = \sqrt{\mathbb{E}\left[R_{\text{B},0}\right]\mathbb{E}\left[R_{\text{B},1}\right]}. \tag{16}$$

Following a similar reasoning as in the above, the mean total power consumption in the second decoding phase can be approximated as

$$\begin{aligned}&\mathbb{E}\left[P_{\text{flip}}\right] \approx\\ &N\,\mathbb{E}\left[P_{\text{cell}}\right] + N_{\text{fail,block}}V_{\text{D}}^2\left(\frac{\rho}{R_{\text{ON}} + R_{\text{ref}}} + \frac{N-\rho}{R_{\text{OFF}} + R_{\text{ref}}}\right),\end{aligned} \tag{17}$$

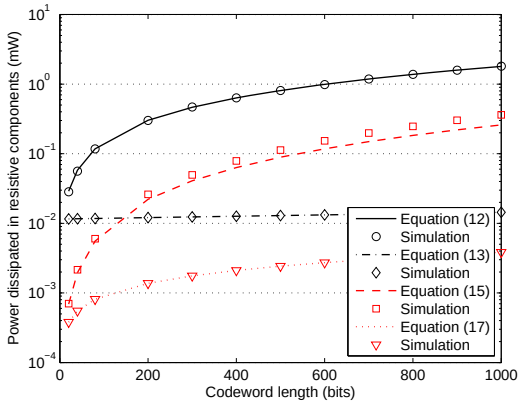

Fig. 11. Analytical estimates of the mean power dissipated in memristors and reference resistors in the CMOS cells, compared to corresponding numerical simulation results obtained as averages over 100 different input vectors and error patterns.

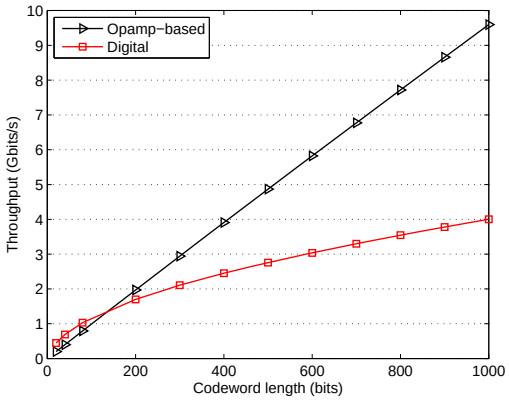

Fig. 13. Decoder maximum throughput estimates obtained using the total nanowire capacitance defined in Section II-C. The shown values correspond to a single decoding iteration; throughput estimates for larger numbers of iterations can be obtained by dividing these results by the number of iterations.

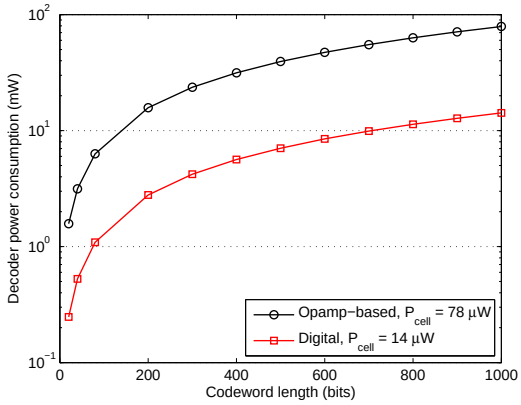

Fig. 12. Mean total power consumption estimates obtained from (12), (13), (15), and (17), including CMOS cell power dissipation estimates P_{cell} obtained with HSPICE for a 130 nm technology.

where $N_{\text{fail,block}}$ is the mean number of failed parity checks in a code block of length L_B, and we assume that $N_{\text{fail,block}}$ is small compared to L_B. The mean energy consumption per bit is obtained also for the digital decoder from (14). The increase of capacitance along with code length is again taken into account, selecting the phase durations now as $T_{\text{check}} = 2.5$ ns $+ \tau_{\text{check}}$ and $T_{\text{flip}} = 2.5$ ns$+\tau_{\text{flip}}$. Again, $\tau = R_{\text{Thevenin}} \cdot C_{\text{tot}}$ is computed according to the Thevenin resistance of the current path in the decoding phases, and the total nanowire capacitance.

3) Performance estimation: In the following we show examples of performance estimates obtained using the analytical results presented above. In these examples, QCLDPC codes with row weigth $\rho = 4$ and column weight $\gamma = 3$, and lengths varying from 20 bits to 1000 bits are assumed. To estimate the power consumed in the second decoding phase, in the following we assume that each codeword contains one erroneous bit, which produces γ failed parity checks, and is corrected in one decoding iteration.

Fig. 11 illustrates the magnitudes and accuracies of the resistive power dissipation estimates (assuming $\mathbb{E}[P_{\text{cell}}] = 0$) provided by (12), (13), (15), and (17). The analytical estimates are compared to results from numerical simulations, where average power dissipations were determined by computing the currents and total resistances corresponding to 100 randomly chosen input vectors and error patterns. As noted above, (15) does not produce an exact estimate of the mean power dissipation, but the estimation error is relatively small. The other analytical estimates are accurate. It is also evident that the resistive power dissipation in the digital decoder is smaller than in the opamp-based decoder, because a smaller number of nanowires is driven at any given time in the digital decoder than in the opamp-based decoder.

Fig. 12 shows the mean total power dissipation for both decoding approaches, obtained by combining the estimates produced by (12), (13), (15), and (17), taking into account the relative durations of the decoding phases and the CMOS cell power consumption estimates obtained with HSPICE for a 130 nm technology.

Throughput estimates of the two decoding approaches are presented in Fig. 13. These results illustrate how the operating speed of the blockwise sequential digital decoding is reduced as the codeword length — and corresponding total nanowire capacitance defined in Section II-C — increases. With the code lengths considered in these examples, the throughput of the opamp-based decoder is limited mainly by the operating speed of the operational amplifier. This means that the throughput increases along with codeword length, as more data is processed in parallel in each decoding iteration, while the duration of an iteration remains approximately constant. Note that the presented throughput estimates are optimistic in the sense that they do not include the time required for storing the input bits in the CMOS cells.

Finally, Fig. 14 presents the estimated energy consumption per decoded bit for the considered examples. Results are shown with and without the simulated CMOS cell power consumption terms in (12), (13), (15), and (17). These results demonstrate that the digital decoder is more energy efficient than the opamp-based decoder, despite the larger throughputs obtainable with the latter.

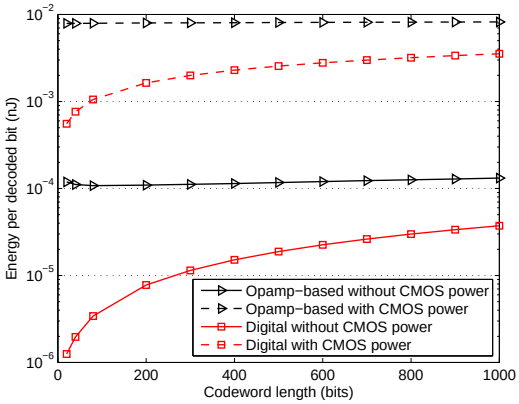

Fig. 14. Decoding energy consumption estimates according to (14) for both considered decoder cell designs with and without simulated CMOS cell power consumption.

To summarize the above discussed results, we demonstrated that with the considered LDPC codes, the digital decoder — taking into account resistive power dissipation and simulated CMOS cell power consumption — is estimated to have the better overall performance of the two designs. It achieves average power consumptions of order 1–10 mW, throughputs of order 1 Gbit/s, and energy efficiencies of order 1 pJ/bit. It should be noted that here we focus strictly on the core decoding circuits, ignoring peripheral circuitry and operations.

The power consumption with both of the considered decoding approaches is dominated by the power consumption of the CMOS circuit elements. There is margin for improvement in the energy efficiency of these decoders using more refined circuit designs. For example, the opamp utilized here was specifically designed for fast and robust operation instead of low power consumption.

The above presented analytical performance estimates can be applied with any LDPC code. In Section VI we compare estimated performance metrics of the considered digital decoding approach with similar LDPC decoders presented in recent publications. However, we first consider the effects of circuit nonidealities on decoding reliability.

V. Effects of Nonidealities

A. Allowable variation in memristances and nanowire resistance with limited-precision processing

Let us consider LDPC codes of length $N = 500$ with row weight $\rho \leq 8$, and denote the number of driven vertical nanowires in the parity check phase by N_D. This is equal to the Hamming weight of an input codeword. Assuming that input data statistically follows the binomial distribution with $p = 0.5$, it follows that $P(N/2 - 70 \leq N_D \leq N/2 + 70) \approx 1 - 10^{-10}$. Furthermore, let us denote by N_{ON} the number of memristors in ON-state at vertical nanowires corresponding to codeword bits 1 at a given horizontal nanowire, and by $I_{N_{\text{ON}}}$ the corresponding nominal current,

$$I_{N_{\text{ON}}} = N_{\text{ON}} I_{\text{ON}} + (N_D - N_{\text{ON}}) I_{\text{OFF}},$$

where I_{ON} and I_{OFF} refer to the nominal currents through memristors in ON and OFF state, respectively, at vertical nanowires corresponding to input bit 1.

We account for variation in circuit characteristics affecting the parity check phase in the opamp-based decoder as follows. We assume that memristors can be programmed to ON-state with a relative conductance error less than or equal to α. We define the OFF–ON resistance ratio K to measure the achievable ratio of resistances in the two states. Thus we here assume that a memristor in OFF-state has resistance equal to or larger than KR_{ON}, but otherwise does not have to be accurately programmed, that is, the OFF-resistance can take any value from KR_{ON} to infinity. Finally, we denote the maximum nanowire resistance relative to the nominal memristor ON-resistance by $\beta = R_{\text{nw,max}}/R_{\text{ON}}$.

Due to the above described variation in circuit characteristics and random variation of N_D, the current at a given horizontal nanowire will differ from the nominal value. We denote the corresponding nonideal currents as $\hat{I}_{N_{\text{ON}}}$, and estimate the worst-case maximum and minimum currents as

$$\max(\hat{I}_{N_{\text{ON}}}) = N_{\text{ON}} I_{\text{ON}}(1+\alpha) + (N/2+70-N_{\text{ON}}) I_{\text{OFF}}, \quad (18)$$

$$\min(\hat{I}_{N_{\text{ON}}}) = \frac{N_{\text{ON}} I_{\text{ON}}(1-\alpha)}{1+(1-\alpha)\beta}. \quad (19)$$

In (18), every driven ON-state memristor is assumed to be in a maximally conducting state, the input Hamming weight is larger than average by the assumed maximum 70, and the total nanowire resistance is assumed to be zero. In (19), we assume that each memristor in ON-state is programmed to the least conducting state possible, that all memristors in OFF-state have infinite resistance, and that the nanowire resistance for every current path is equal to $R_{\text{nw,max}}$. Note that the above assumed values for total nanowire resistance are severely pessimistic.

To guarantee correct operation of the decoder in the parity check phase, the following must hold:

$$\min_{N_{\text{ON}}}(\min(\hat{I}_{N_{\text{ON}}}) - \max(\hat{I}_{N_{\text{ON}}-1})) = \min(\hat{I}_\rho) - \max(\hat{I}_{\rho-1}) \geq (I_\rho - I_0)/2^B, \quad (20)$$

where B is the bit resolution of the analog-to-digital converter in the decoding cell.

To demonstrate the accuracy requirements of the opamp-based decoder cell to achieve practically error-free operation, we fix the A/D converter resolution to values $B = 4$ and $B = 6$, and plot for varying row weight ρ the maximum allowable relative ON-state programming error α for line resistance values $0.01R_{\text{ON}}$, $0.02R_{\text{ON}}$, and $0.04R_{\text{ON}}$. This analysis is illustrated in Figure 15. The results show that with the value estimated for $R_{\text{nw,max}}$ in Section II-C3, an A/D converter with four bit resolution is sufficient to guarantee correct operation for a code with row weight 8, if the maximum relative memristor programming error is less than approximately 0.6 %. Increased A/D conversion accuracy allows for more error in memristor programming and larger nanowire resistance. We also show the results of the above analysis with an OFF–ON resistance ratio 2000; increasing this ratio effectively

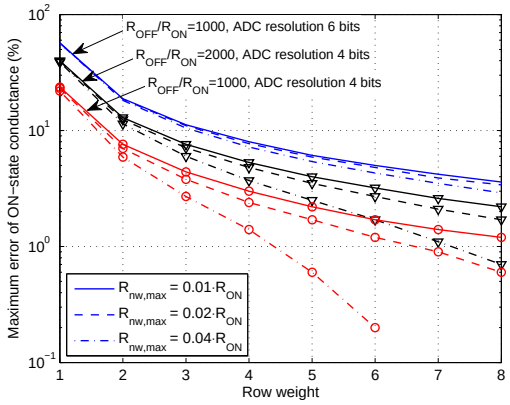

Fig. 15. Estimation of circuit parameter values with which a parity check may fail in less than one codeword per ten billion, assuming four and six bit A/D converter resolutions and OFF–ON resistance ratios 1000 and 2000 for a current sum -based decoding cell with an LDPC code of length 500. Note that the correspondence between line type and nanowire resistance specified in the legend applies to all curves.

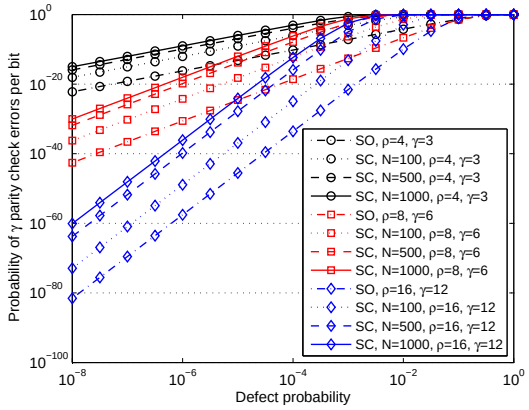

Fig. 16. Probability of γ parity check errors per bit, caused by stuck open (SO) or stuck closed (SC) -type defects.

reduces the A/D conversion resolution required for error-free operation. Note that memristor programming with less than 1 % error has been demonstrated empirically for example in [25] with target resistances of order 10 kΩ.

The above described analysis applies with minor modifications also to the digital decoder cell. As in this case the decoder is assumed to process smaller submatrices with fixed row weighth 1, the decoder operates practically without errors despite much larger variation in nanowire resistance and memristor programming. For example, the decoder operates correctly with probability larger than $1 - 10^{-10}$ on code blocks of length 250 even with maximum relative ON-state programming error of order 10 % and maximum nanowire resistance of order 100 kΩ.

B. Stuck open and stuck closed -type defects

In addition to inaccurate programming, the considered decoders may be affected by device defects in the nanowire array. We model these as memristors that are stuck open — assumed to be permanently in OFF or logical 0 -state — with probability p_{so}, or stuck closed — permanently in ON or logical 1 -state — with probability p_{sc}.

The probability $p_{\text{e,so}}$ that a stuck open -type defect will cause at least one error in the $M \times N$ parity check matrix of any LDPC code is

$$p_{\text{e,so}} = 1 - (1 - p_{\text{so}})^{M\rho}, \tag{21}$$

where ρ is again the mean row weight of the code. The corresponding error probability $p_{\text{e,sc}}$ for stuck closed -type defects is

$$p_{\text{e,sc}} = 1 - (1 - p_{\text{sc}})^{M(N-\rho)}. \tag{22}$$

The probabilities specified in (21) and (22) are significant even with very low defect probabilities p_{so} and p_{sc}. However, it should be noted that these are not equal to the probability of bit error in the output of the decoder. For example, the probability of a stuck open defect causing a change on one row of H, which may cause an erroneous parity check depending on the input data, is

$$p_{\text{pce,so}} = 1 - (1 - p_{\text{so}})^{\rho}. \tag{23}$$

If we assume that each such error always causes an erroneous parity check, the number of detected failed parity checks per bit, denoted by N_{pce}, follows a binomial distribution,

$$N_{\text{pce}} \sim B(\gamma, p_{\text{pce}}). \tag{24}$$

With relatively low error rates, it is reasonable to assume that an erroneous bit in the input will cause γ failed parity checks. With this assumption, we can assume that the flipping threshold is equal to γ, and a correct bit in the input will be classified erroneously by the decoder due to stuck open defects with probability

$$p_{\text{be,so}} = (1 - (1 - p_{\text{so}})^{\rho})^{\gamma}. \tag{25}$$

The above described reasoning applies also for stuck closed -type defects, resulting in

$$p_{\text{pce,sc}} = 1 - (1 - p_{\text{sc}})^{N-\rho}, \tag{26}$$

and

$$p_{\text{be,sc}} = (1 - (1 - p_{\text{sc}})^{N-\rho})^{\gamma}. \tag{27}$$

Fig. 16 illustrates the probabilities specified in (25) and (27) for code lengths 100, 500, and 1000, and various code parameters. Note that increasing the row and column weights of a code significantly reduces the probability of γ erroneous parity checks corresponding to a correct input bit.

VI. Comparisons to existing decoders

In Table I, the digital memristive decoder is compared to a state-of-the-art digital CMOS implementation [18] of similar bit flipping decoding as considered in this work. In [18], the authors consider ultra-low-power LDPC decoders realized with 65 nm technology; they also predict that a digital sub-threshold implementation of the considered decoder will provide an approximately 30-fold reduction in power consumption. Power consumption and throughput in [18] are given for a code of length 512 bits at bit error rate (BER)

equal to 10^{-5} after decoding; for this code, the corresponding signal-to-noise power ratio is reported approximately as 7 dB. For comparison with the results presented in [18], we apply a QCLDPC code of length 500 bits.

The considered code reaches a BER of 10^{-5} at an SNR of approximately 8 dB, and at this operation point, $\mathbb{E}[N_{\text{iter}}] \approx 2.5$ and $\mathbb{E}[N_{\text{fail}}] \approx 4$. Please note that for the BF decoders discussed here, comparing BER vs. SNR performances is not relevant, since — ignoring the possibility of hardware malfunctions, which were in this work considered in Section V — the BER performance of bit flipping is essentially not dependent on the hardware implementation of the algorithm. Based on the operating parameters above and equations (15), (17), and (14), we obtain the performance measures presented in Table I. These results indicate that the digital memristive decoding approach yields an energy consumption per decoded bit approximately 1.8 times that reported in [18], but also a throughput approximately 5.5 times that of the reference. Note that as in this work, the results presented in [18] are based on simulations of the considered decoding circuitry, not measurements of fabricated circuits.

It is also relevant to note that the decoder reported in [18] is designed for a fixed LDPC code. It is not trivial to design integrated decoder circuits which operate efficiently with arbitrary LDPC codes, as the routing and corresponding data scheduling between decoding nodes can become very complicated, especially with large code sizes and codes with high row or column weights. To demonstrate the energy performance gain of the considered simple bit flipping decoding with respect to a more complicated decoding algorithm in a programmable decoder, in Table I we also compare performance estimates obtained for the digital CMOL-based decoder with the LDPC decoder circuit presented in [34]. This comparison is justified in the sense that the reference decoder is also fully programmable, utilizes the same kind of quasi-cyclic LDPC codes as assumed in this work, and is implemented with 130 nm CMOS technology. However, the reference decoder performs min-sum (MS) decoding which, as illustrated in Fig. 3, is significantly better in terms of error correction capability than the bit flipping considered in this work. The related performance tradeoffs were discussed in more detail in Section III.

However, the performance estimates presented in Table I indicate that the bit flipping decoder yields a data throughput of order 10 times that reported in [34], and power consumption and energy per decoded bit approximately 30 % and 2 %, respectively, of that of the reference. Of course, some implementation margin must be assumed in all of these figures, as our results are based on simulations and analytical estimates, while the reference results are measured.

VII. Conclusions

A. Further work

The decoder design principle considered in this work provides many topics for further studies. A direct continuation of this work would be to refine the simple CMOS cell designs considered, especially with the aim of reducing power consumption. Also, the considered decoding principle can be used to implement more complicated decoding algorithms such as soft bit flipping [30], [31], allowing a tradeoff between CMOS cell complexity and bit error rate performance.

Finally, although we have not discussed memristive stateful logic in this paper, the topic is relevant also to the work performed here. For example, the digital decoding approach considered in this work operates using a similar principle as multi-input memristive stateful logic operations, whose principles and practical realization are described in detail in [16], [17]. It would be relevant to investigate further the design of LDPC decoders using memristive stateful logic. Based on the work presented in this paper, we find that a stateful logic decoder implementation using the same kind of monolithic nanowire crossbar as in this work is not reasonable, since the necessary XOR logic can be implemented more efficiently as CMOS gates. However, the use of partitioned nanowire crossbars as considered in [17] might produce interesting possibilities for error correction in memory.

B. Summary

We have presented design principles for implementing decoders for low-density parity check codes in CMOL-type memristive circuits. The CMOL circuit architecture is designed to yield programmable, nonvolatile, and area efficient connectivity between CMOS processing elements; in this work, the programmable connections were used to map the parity check matrix of an LDPC code in the decoder. The considered LDPC decoder examples are based on matrix-vector multiplication, implemented using two different approaches: analog correlation based on Ohm's law and Kirchoff's current law, and corresponding digital processing, achieved by setting specific constraints on the LDPC code structure.

We performed detailed performance analysis and circuit simulations of the example decoders considered in the work, and estimated how CMOL and memristor characteristics will affect decoder operation. Most significantly, the maximum codeword or code block length is limited by the ratio of the memristors' OFF and ON resistances. We also estimated how the operation speed of the decoders depends on the total capacitance in the nanowire array, and analyzed how variation in memristance values, nonzero nanowire resistances, and persistent device defects affect the computation of correlations. Note that these analyses are relevant also in other parallel CMOL-type computing applications.

Regarding the main objectives of this work, based on currently available empirical data on CMOL and memristor characteristics, we found that LDPC decoding in CMOL-type circuit architectures is viable, and enables programmable, energy efficient decoding. In CMOL architectures where multiple vertically stacked nanowire arrays are available, as assumed for example in [12], the considered approach facilitates adding error correction functionality with relatively little increase in CMOL circuit complexity or cell area.

Acknowledgements

This work was supported by the Academy of Finland (253596, 258831, 264914, 140108). We thank the anonymous

TABLE I

COMPARISON OF THE CONSIDERED MEMRISTIVE DECODING, USING THE DIGITAL CMOL CELL DESIGN SPECIFIED IN SECTION III-B, WITH CONVENTIONAL CMOS DECODERS. NOTE THAT THE TWO DIFFERENT SETS OF PERFORMANCE ESTIMATES GIVEN FOR THE MEMRISTIVE DECODER CORRESPOND TO THE TWO DIFFERENT CODES USED IN THE REFERENCES. THESE PERFORMANCE ESTIMATES ARE OBTAINED USING THE FORMULAS SPECIFIED FOR THE DIGITAL MEMRISTIVE DECODER IN SECTION IV.

Reference	Winstead et al., 2012 [18]	This work	Shih et al., 2009 [34]	This work
Decoding	**BF (digital)**	**BF (digital)**	**MS (programmable)**	**BF (digital)**
Technology	**65 nm**	**130 nm**	**130 nm**	**130 nm**
Code length	512	500	1536 (max.)	1536
Circuit area (mm^2)	N/A	1	4.9	3.1
Iterations	N/A	2.5 (at BER 10^{-5})	N/A (reported 2–8)	2
Throughput (Gbps)	**0.2**	**1.1**	**0.086 (reported max)**	**1.6**
Power (mW)	**0.67**	**7.0**	**58**	**18**
Energy efficiency (pJ/bit)	**3.5**	**6.4**	**670**	**11**

reviewers for their comments and suggestions.

REFERENCES

[1] L. O. Chua, "Memristor - the missing circuit element," *IEEE Transactions on Circuit Theory,*, vol. CT-18, no. 5, pp. 507–519, 1971.

[2] L. Chua, "Resistance switching memories are memristors," *Applied Physics A*, vol. 102, no. 4, pp. 765–783, 2011. [Online]. Available: http://dx.doi.org/10.1007/s00339-011-6264-9

[3] S. H. Jo, T. Chang, I. Ebong, B. B. Bhadviya, P. Mazumder, and W. Lu, "Nanoscale memristor device as synapse in neuromorphic systems," *Nano Letters*, vol. 10, pp. 1297–1301, 2010.

[4] M.-J. Lee, C. B. Lee, S. R. Lee, M. Chang, J. H. Hur, Y.-B. Kim, C.-J. Kim, D. H. Seo, S. Seo, U.-I. Chung, I.-K. Yoo, and K. Kim, "A fast, high-endurance and scalable non-volatile memory device made from asymmetric Ta_2O_{5-x}/TaO_{2-x} bilayer structures," *Nature Materials*, vol. 10, no. 8, pp. 625–630, 2011.

[5] K.-H. Kim, S. Hyun Jo, S. Gaba, and W. Lu, "Nanoscale resistive memory with intrinsic diode characteristics and long endurance," *Applied Physics Letters*, vol. 96, no. 5, pp. 053 106–053 106–3, Feb. 2010.

[6] K.-H. Kim, S. Gaba, D. Wheeler, J. M. Cruz-Albrecht, T. Hussain, N. Srinivasa, and W. Lu, "A functional hybrid memristor crossbar-array/CMOS system for data storage and neuromorphic applications," *Nano Letters*, vol. 12, no. 1, pp. 389–395, 2012.

[7] J. Nickel, "Memristor materials engineering: From flash replacement towards a universal memory," in *IEEE IEDM Advanced Memory Technology Workshop*, 2011.

[8] H. Nazarian, "Crossbar resistive memory: the future technology for NAND Flash," 2013, White paper. [Online]. Available: http://www.crossbar-inc.com/assets/img/media/Crossbar-RRAM-Technology-Whitepaper-080413.pdf

[9] K. K. Likharev and D. B. Strukov, "CMOL: Devices, circuits, and architectures," in *Introducing Molecular Electronics*, G. Cuniberti, G. Fagas, and K. Richter, Eds. Berlin: Springer, 2005, pp. 447–478.

[10] K. Likharev, A. Mayr, I. Muckra, and O. Türel, "CrossNets: high-performance neuromorphic architectures for CMOL circuits," *Annals Of The New York Academy Of Sciences*, vol. 1006, pp. 146–163, 2003.

[11] G. S. Snider, "Self-organized computation with unreliable, memristive nanodevices," *Nanotechnology*, vol. 18, 2007.

[12] E. Lehtonen, J. H. Poikonen, M. Laiho, and P. Kanerva, "Large-scale memristive associative memories," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 3, pp. 562–574.

[13] E. Lehtonen, J. H. Poikonen, and M. Laiho, "Two memristors suffice to compute all Boolean functions," *Electronics Letters*, no. 3, p. 239, 2010.

[14] J. H. Poikonen, E. Lehtonen, and M. Laiho, "On synthesis of Boolean expressions for memristive devices using sequential implication logic," *IEEE Transactions on computer-aided design of Integrated Circuits and Systems*, vol. 31, no. 7, pp. 1129–1134, 2012.

[15] S. Kvatinsky, G. Satat, N. Wald, E. Friedman, A. Kolodny, and U. Weiser, "Memristor-based material implication (imply) logic: Design principles and methodologies," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. PP, no. 99, pp. 1–1, 2013.

[16] E. Lehtonen, J. H. Poikonen, and M. Laiho, "Memristive stateful logic," in *Memristor Networks*, A. Adamatzky and L. Chua, Eds. Springer, 2014.

[17] E. Lehtonen, J. Tissari, J. H. Poikonen, M. Laiho, and L. Koskinen, "A cellular computing architecture for parallel memristive stateful logic," *Microelectronics Journal*, 2014 (in press).

[18] C. Winstead and J. N. Rodrigues, "Ultra-low-power error correction circuits: Technology scaling and sub-v_T operation." *IEEE Transactions on Circuits and Systems–II: Express briefs*, vol. 59-II, no. 12, pp. 913–917, 2012.

[19] K. Ganesan, P. Grover, and J. Rabaey, "The power cost of over-designing codes," in *Signal Processing Systems (SiPS), 2011 IEEE Workshop on*, Oct 2011, pp. 128–133.

[20] K. Ganesan, Y. Wen, P. Grover, A. Goldsmith, and J. Rabaey, "Choosing "green" codes by simulation-based modeling of implementations," in *Global Communications Conference (GLOBECOM), 2012 IEEE*, Dec 2012, pp. 3286–3292.

[21] B. Smith, "Error-correcting codes for fibre-optic communication systems," Ph.D. dissertation, 2011.

[22] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *Nature*, vol. 453, pp. 80–83, 2008.

[23] S. H. Jo, K. H. Kim, and W. Lu, "Programmable resistance switching in nanoscale two-terminal devices," *Nano Lett.,*, vol. 9, pp. 496–500, 2009.

[24] M. Laiho and E. Lehtonen, "Arithmetic operations within memristor-based analog memory," in *Proceedings of the 12th International Workshop on Cellular Nanoscale Networks and Their Applications (CNNA) 2010*, 2010.

[25] F. Alibart, L. Gao, B. D. Hoskins, and D. B. Strukov, "High precision tuning of state for memristive devices by adaptable variation-tolerant algorithm," *Nanotechnology*, vol. 23, no. 7, p. 075201, 2012. [Online]. Available: http://stacks.iop.org/0957-4484/23/i=7/a=075201

[26] D. B. Strukov and R. S. Williams, "Four-dimensional address topology for circuits with stacked multilayer crossbar arrays," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 106, no. 48, pp. 20 155–20 158, 2009.

[27] G. S. Snider and R. S. Williams, "Nano/CMOS architectures using a field-programmable nanowire interconnect," *Nanotechnology*, vol. 18, no. 3, 2007.

[28] D. J. C. MacKay, *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2003.

[29] K. Yu, L. Shu, and M. P. C. Fossorier, "Low-density parity-check codes based on finite geometries: a rediscovery and new results," *Information Theory, IEEE Transactions on*, vol. 47, no. 7, pp. 2711–2736, Nov 2001.

[30] X. Wu, C. Ling, M. Jiang, E. Xu, C. Zhao, and X. You, "New insights into weighted bit-flipping decoding," *Communications, IEEE Transactions on*, vol. 57, no. 8, pp. 2177–2180, Aug 2009.

[31] J. Cho, J. Kim, and W. Sung, "VLSI implementation of a high-throughput soft-bit-flipping decoder for geometric LDPC codes," *IEEE Transactions on Circuits and Systems–I*, vol. 57, no. 5, pp. 1083–1094, May 2010. [Online]. Available: http://dx.doi.org/10.1109/TCSI.2010.2047743

[32] R. Gallager, "The random coding bound is tight for the average code (corresp.)," *Information Theory, IEEE Transactions on*, vol. 19, no. 2, pp. 244–246, Mar 1973.

[33] G. Liva, W. Ryan, and M. Chiani, "Quasi-cyclic generalized LDPC codes with low error floors," *Communications, IEEE Transactions on*, vol. 56, no. 1, pp. 49–57, January 2008.

[34] X.-Y. Shih, C.-Z. Zhan, and A.-Y. Wu, "A real-time programmable LDPC decoder chip for arbitrary QC-LDPC parity check matrices," in *Solid-State Circuits Conference, 2009. A-SSCC 2009. IEEE Asian*, Nov 2009, pp. 369–372.