Error-Detection Schemes for Analog Content-Addressable Memories

Ron M. Roth

***Abstract*—Analog content-addressable memories (in short, a-CAMs) have been recently introduced as accelerators for machine-learning tasks, such as tree-based inference or implementation of nonlinear activation functions. The cells in these memories contain nanoscale memristive devices, which may be susceptible to various types of errors, such as manufacturing defects, inaccurate programming of the cells, or drifts in their contents over time.**

The objective of this work is to develop techniques for overcoming the reliability issues that are caused by such error events. To this end, several coding schemes are presented for the detection of errors in a-CAMs. These schemes consist of an encoding stage, a detection cycle (which is performed periodically), and some minor additions to the hardware. During encoding, redundancy symbols are programmed into a portion of the a-CAM (or, alternatively, are written into an external memory). During each detection cycle, a certain set of input vectors is applied to the a-CAM. The schemes differ in several ways, e.g., in the range of alphabet sizes that they are most suitable for, in the tradeoff that each provides between redundancy and hardware additions, or in the type of errors that they handle (Hamming metric versus L_1 metric).

***Index Terms*—Analog computation, Content-addressable memory, Error-detecting codes, Lee metric, Memristive devices.**

I. Introduction

For integers $a \le b$, denote by $[a : b]$ the integer subset $\{z \in \mathbb{Z} : a \le z \le b\}$ and by $[a : b\rangle$ the set $[a : b-1]$; we will use the shorthand notation $[b\rangle$ for $[0 : b\rangle$.

Let $q \ge 2$ be a positive integer which will stand for the *alphabet size*. A *q-comparator* is a circuit that implements the bivariate function $\mathrm{L}_q : [q\rangle \times [q\rangle \to \{0, 1\}$ which is defined by

$$\mathrm{L}(x, \vartheta) = \mathrm{L}_q(x, \vartheta) = \begin{cases} 1 & \text{if } x \le \vartheta \\ 0 & \text{otherwise} \end{cases} . \qquad (1)$$

The argument x is called the *input* and ϑ is called the *threshold*.[1]

Nanoscale analog content-addressable memories (in short, a-CAMs) have been recently introduced in [17] and proposed as accelerators for machine-learning applications, including fast inference on decision trees [23], and computation of various functions: analog-to-digital conversion, nonlinear activation functions [30],[33], and bivariate functions [32]. Such a-CAMs use q-comparators as building blocks, which are arranged in $m \times n$ arrays with each entry $(i, j) \in [m\rangle \times [n\rangle$ implementing the function $x \mapsto \mathrm{L}_q(x, \vartheta_{i,j})$, for some threshold $\vartheta_{i,j}$.[2] The input to the a-CAM array is a vector $\boldsymbol{x} = (x_j)_{j \in [n\rangle} \in [q\rangle^n$, with x_j serving as the input to all the comparators along column j. Each row ("match line") in the array computes the conjunction ("and", "$\wedge$") of the outputs of the comparators along the row, and the m results form the output vector, $\boldsymbol{v} \in \{0, 1\}^m$, of the array. The nonzero entries in $\boldsymbol{v}$ are referred to as "matches," and in practice one might be interested in finding all the matches (i.e., knowing the whole vector $\boldsymbol{v}$), or just their number (namely, the Hamming weight of $\boldsymbol{v}$), or the index i of the first match in $\boldsymbol{v}$. The comparator in each position (i, j) contains a memristive device whose conductance is set, during the programming stage, to (a value which is proportional to) the threshold $\vartheta_{i,j}$.[3] These conductances are assumed to change much less frequently than the input vector $\boldsymbol{x}$.

Figure 1 presents a schematic diagram of (our abstract model of) an a-CAM. It consists of an $m \times n$ array of cells, where each cell consists of a comparator (represented by a triangular shape) and a memory device (namely, a memristor) which stores the threshold value that is associated with the cell. Each comparator has two q-ary input terminals and one output binary terminal. One input terminal is fed by the memory device of the cell, while the other is connected through a column conductor to an entry of the input vector $\boldsymbol{x}$. The output terminals of the n comparators along each row $i \in [m\rangle$ are "and"ed together to produce the match line value, ML_i. Under normal operation of the a-CAM, the output value v_i of row i equals ML_i. (The red components in the figure are additions to the a-CAM to be suggested below in this work.)

The operation of an a-CAM is prone to errors from several sources, such as: inaccuracies in the conductance values during programming, drifts in conductance values over time, noise while reading, or manufacturing defects of cells (e.g., short cells or open cells). In this work, we present several schemes for detecting errors in the programmed thresholds in an a-CAM array. For the case of (ordinary) binary CAMs, the error-handling problem was addressed in [20],[21],[28], where the prevailing proposed solution involved a hardware

Ron M. Roth is with the Computer Science Department, Technion, Haifa 3200003, Israel. This work was done in part while visiting Hewlett Packard Laboratories, Milpitas, CA. This work was supported in part by the U.S. Army Research Office under Agreement W911NF2110355, and by Grant 1713/20 from the Israel Science Foundation. Email: ronny@cs.technion.ac.il.

[1] In practice, the threshold ϑ is also allowed to take a negative value, in which case $x \mapsto \mathrm{L}(x, \vartheta)$ is always 0, and x is allowed to take an "infinite" value (in fact, any $x > q-1$ will suffice), for which $\vartheta \mapsto \mathrm{L}(x, \vartheta)$ is always 0.

[2] In the design of [17] (which is used in the mentioned applications), some of the comparators implement functions where the inequality in the right-hand side of (1) is reversed to "$x \ge \vartheta$". Conceptually, this can be realized by an implementation of $x' \mapsto \mathrm{L}(x', q-1-\vartheta)$ with the input $x' \leftarrow q-1-x$.

[3] Thus, the thresholds in the design of [17] are in effect real values, and so are the entries of the input vector $\boldsymbol{x}$ (as they are set therein by voltages). However, in the mentioned intended applications, the thresholds are quantized, and in most cases so are the inputs. The schemes to be presented in this work will use (by design) discrete inputs over the same alphabet, $[q\rangle$, of the thresholds. Hence our description of a-CAMs as consisting of comparators with discrete thresholds and inputs.

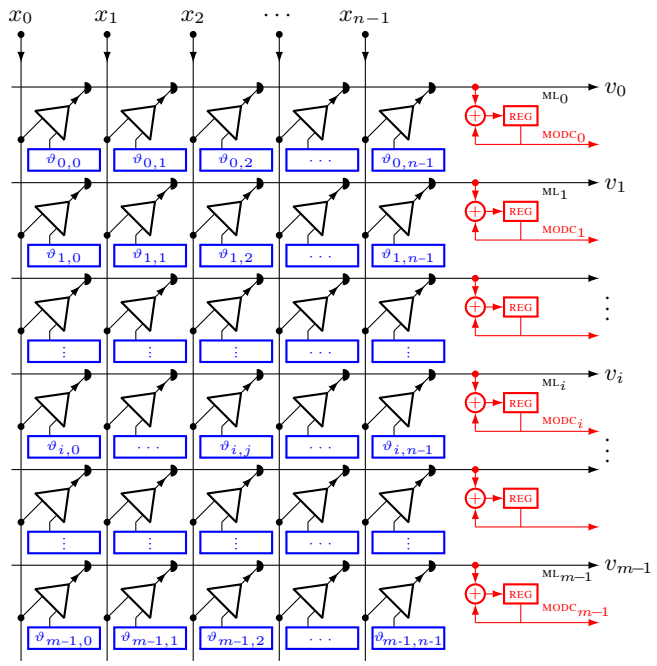

Fig. 1. Schematic diagram of an a-CAM.

modification of the sense amplifiers along each match line. As for ternary CAMs (TCAMs), it was shown in [16] that such an approach would require no less redundancy (i.e., overhead) than just having multiple copies of the contents of the TCAM, thereby resulting in a prohibitively large redundancy. The error-detection scheme in our previous work [3],[4] proposed a different approach for error handling in TCAMs. That work, along with our more recent papers on coding schemes for vector–matrix multipliers [26],[27], inspired, in part, the schemes to be presented in the sequel; however, substantial changes are warranted due to the different operation of an a-CAM. To the best of our knowledge, our results herein are the first attempt to address the error-detection issue in a-CAMs.

Similarly to the general paradigm of operation of [3],[4], all our schemes entail:

- some modification to the a-CAM hardware,
- an encoding stage while programming the a-CAM array, and—
- an error detection cycle, which is performed periodically and consists of applying to the array a certain set of pre-selected input *test vectors*.

For the schemes to work, *redundancy symbols* should be added to the array, on top of the *task-driven thresholds*, namely, the thresholds that are determined by the computation task. These redundancy symbols are calculated during the encoding stage and can be stored in one of two ways:

- *Internal redundancy:* The redundancy symbols are programmed as thresholds in certain columns in the array that have been designated as *redundancy columns*.
- *External redundancy:* The redundancy symbols are stored in an external "ordinary" memory (e.g., an SRAM).

We will consider both these scenarios in our schemes, starting with the internal-redundancy approach (this approach was also used in our prior work [3],[4],[26],[27]). The external-redundancy approach does require an additional component in the hardware design, yet, as we will see, it offers some advantages in terms of error detection.

During each detection cycle, all the rows that contain incorrect threshold values are identified, assuming that the number of erroneous thresholds per row does not exceed some prescribed number τ. The parameter τ is selected so that the probability of the event of misdetecting an erroneous row is sufficiently small according to the design specifications. As is typically done in storage and communication systems, τ can be decided upon either by computation (when there is a sound probabilistic model of the error events in the hardware), or through empirical measurements.

Our schemes differ in the trade-off that they offer between several attributes, including:

- amount of changes and additions to the hardware,
- range of values of the alphabet size q,
- number of redundancy columns, and—
- time complexity (or latency) of the detection cycle.

Like in [3],[4], most of our schemes will take advantage of the parallel operation of the a-CAM in that all rows are fed with the input $\boldsymbol{x}$ and produce their respective match line outputs concurrently; as such, the time complexity of the detection cycle in these schemes will not depend on the number of rows in the a-CAM.

We find it convenient to regard the threshold values $\vartheta_{i,j}$ as elements in the ring, $\mathbb{Z}_q$, of integers modulo q. The a-CAM is then viewed as an $m \times n$ array over $\mathbb{Z}_q$, with $\boldsymbol{\vartheta}_i = (\vartheta_{i,j})_{j \in [n\rangle}$ standing for the vector of thresholds along row $i \in [m\rangle$. The encoding will generally be carried out so that each $\boldsymbol{\vartheta}_i$ forms a codeword of some code (subset) $\mathcal{C}$ of $\mathbb{Z}_q^n$. For a prescribed k (which depends on the code $\mathcal{C}$), the first k entries in $\boldsymbol{\vartheta}_i$ contain the task-driven threshold values, which are selected freely (during the programming phase) according to the desired computation task, while the remaining $r = n-k$ entries form the redundancy part and are determined by an encoder for $\mathcal{C}$ (again, during the programming phase); clearly, we seek to minimize r subject to the prescribed number of errors that we wish to be able to detect.

In our reference to errors so far, we have (implicitly) assumed the Hamming metric, where an error event means that a threshold value is incorrect, irrespective of the (integer) difference between the correct and erroneous values. This metric is the most conservative as it can model a variety of sources of errors, such as malfunctioning of a comparator or drifts in the programmed threshold values. If only drift errors are expected, then there is a merit in considering the L_1-metric instead, thereby ending up with detection schemes that require less redundancy. Under this metric, an error event is defined as changing a threshold value by ± 1 (larger changes are interpreted as having multiple errors at the same threshold). We will consider both the Hamming metric and the L_1-metric (or, more precisely, the Lee metric, which is the L_1-metric counterpart for $\mathbb{Z}_q$).

This work is organized as follows. In Section II, we summarize some definitions and properties relating to codes. When the alphabet size q is a prime (in which case $\mathbb{Z}_q$ is a field),

these properties are well known and can be found in any coding textbook (see, for example, [25]). For our purposes, however, we will formulate them to suit also the case where q is not a prime.

In Section III, we present two detection schemes for the Hamming metric: the bit-interleaving scheme (Construction A in Section III-A) and the shift-and-count scheme (Construction B in Section III-B). Their exposition will first be made for the internal-redundancy scenario; the external-redundancy approach will then be discussed in Section III-C.

In Section IV, we present two coding schemes for the Lee metric: the Gray conversion scheme, which is based on Gray codes (Construction C in Section IV-A), and the Lee-metric shift-and-count scheme (Construction D in Section IV-B).

In all the detection schemes of Sections III and IV, the number of test vectors grows linearly with the alphabet size q and super-linearly with the number of columns n, yet does not depend on the number of rows m in the a-CAM. In contrast, in Section V we present a detection scheme in which the number of test vectors grows linearly with m yet does not depend on the alphabet size and grows only logarithmically with the number of columns. This scheme uses the circuitry that already exists in the a-CAM for reading the cells (since the programming phase of an a-CAM is typically accompanied with a read-after-write). As one of its building blocks, the scheme will require a code with certain (non-standard) properties; constructions of such codes, along with bounds on their parameters, will be presented in Section VI.

Section VII contains a summary of the parameters of the various coding schemes that are presented in this work.

II. Definitions and notation

We will consider codes $\mathcal{C} \subseteq \mathbb{Z}_q^n$ that are *linear* and *systematic* over $\mathbb{Z}_q$; namely, there exists a $k \times n$ matrix G over $\mathbb{Z}_q$, referred to as the generator matrix, such that

- *(the linear property)*
$$\mathcal{C} = \left\{ \boldsymbol{c} = \boldsymbol{u}G \,:\, \boldsymbol{u} \in \mathbb{Z}_q^k \right\},$$
with operations carried out in $\mathbb{Z}_q$, and—
- *(the systematic property)* G contains an invertible $k \times k$ submatrix over $\mathbb{Z}_q$.

In such a code, the rows of G form a basis of $\mathcal{C}$; thus, $|\mathcal{C}| = q^k$. The parameters n, k, and $r = n - k$ are called the length, dimension, and redundancy of $\mathcal{C}$, respectively. By possibly permuting the columns of G and applying invertible linear operations on the rows of G, we can obtain a generator matrix of the form

$$\left(\; I_k \;\middle|\; A \; \right), \tag{2}$$

where I_k is the $k \times k$ identity matrix. The encoding in this case can be easily carried out by the mapping

$$\boldsymbol{u} \mapsto \boldsymbol{u}G = (\boldsymbol{u} \,|\, \boldsymbol{u}A), \tag{3}$$

namely, $\boldsymbol{u}$ forms part of the codeword. It is for this reason that we require the systematic condition (on top of linearity, which will be useful for the detection process as well): to exploit the full functionality of the a-CAM, the encoding scheme should not impose any constraints on the first (say) k task-driven threshold values in each row.

Remark 1. The linear and systematic properties imply that $\mathcal{C}$ is a free $\mathbb{Z}_q$-module, since it has a basis [31, p. 70]. However, there are free modules which are (linear but) not systematic, such as the code over $\mathbb{Z}_6$ which is spanned by the rows of the 2×3 matrix:

$$\begin{pmatrix} 1 & 2 & 2 \\ 2 & 0 & 1 \end{pmatrix}.$$

The rows of this matrix are linearly independent over $\mathbb{Z}_6$, yet none of its 2×2 submatrices is invertible over this ring. This means that while the 36 linear combinations of the two rows form distinct vectors in $\mathbb{Z}_6^3$, none of the projections of these vectors onto any two coordinates will produce all the 36 pairs in $\mathbb{Z}_6^2$. We note that when q is a prime, linearity always implies the systematic property. □

The minimum (Hamming) distance of $\mathcal{C}$, denoted by d, is the smallest number of positions on which any two distinct codewords differ; equivalently, it is the smallest Hamming weight of any nonzero codeword in $\mathcal{C}$. If a codeword is subject to any pattern of no more than $\tau = d - 1$ errors, then such an event can always be detected. Also, if a codeword is subject to any pattern of no more than $\lfloor (d-1)/2 \rfloor$ errors, then the codeword can always be recovered.[4] The length n, dimension k, and minimum distance d of a systematic linear code will usually be written as a triple $[n, k, d]$.

A parity-check matrix of a code $\mathcal{C} \subseteq \mathbb{Z}_q^n$ is an $r \times n$ matrix over $\mathbb{Z}_q$ such that $\mathcal{C}$ forms its right kernel:

$$\mathcal{C} = \left\{ \boldsymbol{c} \in \mathbb{Z}_q^n \,:\, H\boldsymbol{c}^\top = \boldsymbol{0}_r \right\}$$

(where $\boldsymbol{0}_r$ stands for the all-zero column in $\mathbb{Z}_q^r$ and $(\cdot)^\top$ denotes transposition). Every systematic linear $[n, k{=}n{-}r, d]$ code has an $r \times n$ parity-check matrix that contains an invertible $r \times r$ submatrix; in particular, when G has the form (2), then

$$\left(\; -A^\top \;\middle|\; I_r \; \right)$$

is such a matrix. Conversely, any $r \times n$ matrix over $\mathbb{Z}_q$ which contains an invertible $r \times r$ submatrix is a parity-check matrix of a systematic linear $[n, k{=}n{-}r, d]$ code over $\mathbb{Z}_q$. The minimum distance of such a code can be characterized through any of its parity-check matrices H as follows: it is the largest integer d such that every $d - 1$ columns in H are linearly independent over $\mathbb{Z}_q$.

Given a row vector $\boldsymbol{y} \in \mathbb{Z}^n$, the syndrome vector of $\boldsymbol{y}$ with respect to an $r \times n$ parity-check matrix H of a code $\mathcal{C} \subseteq \mathbb{Z}_q^n$ is the column r-vector

$$\boldsymbol{\sigma} = H\boldsymbol{y}^\top.$$

Thus, the codewords of $\mathcal{C}$ are all the vectors in $\mathbb{Z}_q^n$ whose syndrome is all-zero.

The notion of systematic linear codes extends in a straightforward manner to codes over the integer ring $\mathbb{Z}$ (except that in this case the code will have infinite size).

[4] In this work, however, we focus on detection only, which corresponds to just locating the a-CAM rows that contain errors. The recovery (correction) process is then carried out by re-programming the erroneous rows.

III. Coding schemes

In this section, we present two error-detection coding schemes for a-CAM arrays, under the Hamming metric. Throughout, m and n stand for the number of rows and columns in the array, respectively, and τ is the largest number of errors that we expect to have in a row (and we wish to detect). Each scheme uses a certain systematic linear $[n, k{=}n{-}r, d{=}\tau{+}1]$ code $\mathcal{C}$. In Construction A (Section III-A) the code is binary, while in Construction B (Section III-B) the code is over $\mathbb{Z}_q$. In each coding scheme, the encoding algorithm will be identical for all rows and will be carried out by an encoder of the form (3). In addition, we will exploit the nature of operation of the a-CAM in that the detection process will apply the same set of (pre-selected) input test vectors—in parallel—to all rows (in particular, the latency of the process will not depend on the number of rows). We will first describe the schemes assuming the internal-redundancy approach, where r (out of n) columns in the a-CAM are set aside for storing the redundancy symbols as thresholds; we refer to r as the *column redundancy*. The external-redundancy approach will be discussed in Section III-C.

Since both the encoding and detection processes are uniform across rows, we use the notation $\boldsymbol{\vartheta} = (\vartheta_j)_{j\in[n\rangle}$ to stand for the vector of thresholds along a generic row in the array, omitting the index i of the row.

A. Construction A: Bit-interleaving scheme

The coding scheme to be presented in this section—referred to as the bit interleaving scheme or Construction A—applies to cases where the alphabet size q is a power of 2. Assuming that τ is much smaller than n, we will take $\mathcal{C}$ to be an $[n, k{=}n{-}r, d{=}\tau{+}1]$ binary alternant code (in particular, a BCH code) whose redundancy is given by

$$\mathsf{r}(\tau, n) = \begin{cases} \dfrac{\tau}{2} \cdot \lceil \log_2 (n+1) \rceil & \text{if } \tau \text{ is even} \\ \dfrac{\tau-1}{2} \cdot \lceil \log_2 n \rceil + 1 & \text{if } \tau \text{ is odd} \end{cases} \tag{4}$$

(see [25, Ch. 5 and Problem 8.12]). Writing $b = \log_2 q$ (which is a positive integer), we regard the row vector $\boldsymbol{\vartheta}$ as a $b \times n$ binary array

$$\Phi = \Phi(\boldsymbol{\vartheta}) = (\phi_{s,j})_{(s,j)\in[b\rangle\times[n\rangle}, \tag{5}$$

with column j in Φ being the b-vector in $\{0,1\}^b$ representing (the integer) ϑ_j to base 2:

$$\vartheta_j = \sum_{s\in[b\rangle} 2^s \cdot \phi_{s,j}, \quad j \in [n\rangle. \tag{6}$$

The encoding is carried out so that for each $s \in [b\rangle$, row s in Φ, namely, $\boldsymbol{\phi}_s = (\phi_{s,j})_{j\in[n\rangle}$, is a codeword of $\mathcal{C}$ (hence the name bit interleaving). Thus, for $r = \mathsf{r}(\tau, n)$ as in (4), the first $k = n - r$ entries in $\boldsymbol{\vartheta}$ are freely selected (according to the computation task), while the remaining r entries are determined by the encoder.

We turn to describing the detection process. The underlying principle is as follows: given an $r \times n$ binary parity check matrix $H = (H_{\ell,j})_{(\ell,j)\in[r\rangle\times[n\rangle}$ of $\mathcal{C}$, error detection will be achieved through testing, in each detection cycle, that the following equality holds for every $s \in [b\rangle$:

$$H\boldsymbol{\phi}_s^\top = \mathbf{0} \tag{7}$$

(where the equality is over $\mathbb{Z}_2$). For this test, we will need a modulo-2 counter (toggle bit) to be added to each match line: it is synchronized with the clock of the input n-vectors to the a-CAM and flips its value each time its input is a 1. This added hardware is shown in red in Figure 1, with REG therein standing for a one-bit memory and the circled "+" standing for addition modulo 2.

As part of the test (7), we will employ these counters to extract the values $\phi_{s,j}$ for each $s \in [b\rangle$ and $j \in [n\rangle$: this will be carried out by applying the $q/2^s - 1$ test vectors

$$a \cdot 2^s \cdot \boldsymbol{e}_j, \quad a \in [1 : q/2^s\rangle,$$

where $\boldsymbol{e}_j$ denotes the standard unit vector that contains its (only) 1 at position j. It is easy to see that the number of matches will then equal $\lfloor \vartheta_j/2^s \rfloor$ which, when taken modulo 2, equals $\phi_{s,j}$ (see (6)). Based on this simple observation, we next define the set of input test vectors, $\mathcal{X} \subseteq [q\rangle^n$, which is applied to the a-CAM during each detection cycle.

For each $\ell \in [r\rangle$, denote by J_ℓ the support of row ℓ in H:

$$J_\ell = \big\{ j \in [n\rangle \,:\, H_{\ell,j} \neq 0 \big\}.$$

Also, for $(\ell, s) \in [r\rangle \times [b\rangle$, let $\mathcal{X}_{\ell,s}$ be the following subset of $[q\rangle^n$:

$$\mathcal{X}_{\ell,s} = \big\{ a \cdot 2^s \cdot \boldsymbol{e}_j \,:\, a \in [1 : q/2^s\rangle,\ j \in J_\ell \big\}. \tag{8}$$

The set $\mathcal{X}$ is now defined as the union

$$\mathcal{X} = \bigcup_{(\ell,s)\in[r\rangle\times[b\rangle} \mathcal{X}_{\ell,s}.$$

Figure 2 presents the detection cycle. In that figure, ML_i and MODC_i stand for the (binary) contents of the match line and the modulo-2 counter, respectively, of row i. In each iteration over (ℓ, s), the test vectors in $\mathcal{X}_{\ell,s}$ compute entry ℓ in the vector $H\boldsymbol{\phi}_s^\top$; if it is nonzero at some row, an error is flagged at that row (referring to Figure 1, during the detection cycle, we actually switch the output v_i of row i to be MODC_i instead of ML_i).

Observe that as long as row i has not been flagged for error, we will have $\text{MODC}_i = 0$ at the beginning of each iteration of the loop over s. Hence, for a given (ℓ, s), we can reuse computations made earlier for $(\ell, s' > s)$. This, in turn, allows us to reduce each subset $\mathcal{X}_{\ell,s}$ so that a in (8) ranges just over the odd values in $[1 : q/2^s\rangle$, namely,

$$\mathcal{X}_{\ell,s} = \big\{ (2\alpha+1) \cdot 2^s \cdot \boldsymbol{e}_j \,:\, \alpha \in [q/2^{s+1}\rangle,\ j \in J_\ell \big\}.$$

This set is of size $(q/2^{s+1}) \cdot |J_\ell|$ and so we have:

$$|\mathcal{X}| = \Big(\sum_{s\in[b\rangle} \frac{q}{2^{s+1}} \Big) \cdot \sum_{\ell\in[r\rangle} |J_\ell| = (q-1) \cdot \|H\|, \tag{9}$$

where $\|H\|$ denotes the total number of 1's in the matrix H. This number is at most $r \cdot n$, and for $r > 1$ it is typically close to $r \cdot n/2$.

```
for each i ∈ [m⟩ do                                              1
    MODC_i ← 0;                                                  2
for each ℓ ∈ [r⟩ do                                              3
    for s ← b−1, b−2, ..., 0 do                                  4
        for each x ∈ X_{ℓ,s} do                                  5
            apply x to the a-CAM;                                6
            for each i ∈ [m⟩ do                                  7
                MODC_i ← (MODC_i + ML_i) mod 2;                  8
        for each i ∈ [m⟩ do                                      9
            if MODC_i ≠ 0 then flag error in row i.              10
```

Fig. 2. Detection cycle of Construction A.

In summary, the detection process consists of applying at most $(q-1)\cdot r\cdot n$ test vectors to the a-CAM (and typically half that number). The required column redundancy is determined by (4).

Remark 2. The index $s = b-1$ corresponds to checking the most-significant bit of the base-2 representations of the thresholds in $\boldsymbol{\vartheta}$, and the size of $\mathcal{X}_{\ell,s}$ in this case is $|J_\ell|$. That size doubles as we decrease s until reaching the size $(q/2)|J_\ell|$ for the index $s = 0$, which corresponds to the least-significant bit (l.s.b.) of the thresholds. Thus, in terms of time complexity, the l.s.b.'s are the costliest to check. Now suppose that for some $\lambda \in [b\rangle$, we elect to skip the detection of errors in the λ l.s.b.'s (say, because the computation task is insensitive to changes smaller than $\pm 2^\lambda$ in the threshold values). Under this scenario, we can stop the iteration over s at λ which, in terms of complexity, is equivalent to decreasing q by a factor of 2^λ. □

Example 1. Consider an a-CAM with $m = 512$ rows and $k = 50$ columns, which is susceptible to errors. A naive approach of detecting errors would be reading periodically the thresholds in the whole a-CAM directly: that would entail $m \cdot k = 25,600$ reads. In comparison, Table I presents the column redundancy r and the number of test vectors when we use Construction A, for $q = 8, 16$ and $\tau = 1, 2, 3$.[5] We have used (9) to compute $\mathcal{X}$. For $\tau = 1$, the matrix H is the 1×51 all-one vector $(1\ 1\ \ldots 1)$. For $\tau = 2$ it is a 6×56 parity-check matrix of a shortened binary Hamming code, with the columns ranging over the lightest (in terms of Hamming weight) nonzero binary 6-vectors.[6] And for $\tau = 3$ it is a 7×57 parity-check matrix of a shortened extended binary Hamming code, with the columns ranging over the lightest binary 7-vectors with odd Hamming weight. □

Construction A can be generalized by changing the radix 2 therein to any radix $\rho \geq 2$ with $q = \rho^b$ for some positive integer b. Respectively, the counters in each row will now be modulo ρ and the code $\mathcal{C}$ will be over $\mathbb{Z}_\rho$. There is one significant caveat though: we will need $\mathcal{C}$ to have a 0–1 parity-check matrix, namely, a matrix whose entries are constrained to the subset $\{0, 1\}$ of $\mathbb{Z}_q$. Otherwise, an implementation of the check of the equality (7) will require a different (and potentially much larger) set $\mathcal{X}$ of test vectors.

[5]The determination of the number τ of errors per row to be detected can be made either through simulations or through analysis that is based on some probabilistic model of failure of the devices in the a-CAM.

[6]A redundancy of 6 is the smallest possible for any binary linear $[n, k{=}50, d{=}3]$ code. By increasing the column redundancy r, we may sometimes reduce $\|H\|$—and, thus, $|\mathcal{X}|$. For example, for $(k, \tau) = (50, 2)$ and $r = 7, 8, 9, 10, 11$ we get $\|H\| = 136, 130, 123, 115, 111$, respectively.

TABLE I
PARAMETERS OF CONSTRUCTION A FOR AN A-CAM WITH $k = 50$ TASK-DRIVEN COLUMNS.

τ	r	n	$\|H\|$	$\|\mathcal{X}\|$	
				$q = 8$	$q = 16$
1	1	51	51	357	765
2	6	56	156	1,092	2,340
3	7	57	187	1,309	2,805

We discuss such codes in Section VI and present constructions for $q = 2^b$ as well as for $\tau \in [1 : 4]$ (and any q). As these constructions are based on (binary) alternant codes, their redundancy equals $\mathsf{r}(\tau, n)$ as in (4), except for the case $\tau = 4$ and q that is a multiple of 3, in which case the redundancy is (at most) $\mathsf{r}(5, n)$.

In this more general setting, the sets $\mathcal{X}_{\ell,s}$ are defined by

$$\mathcal{X}_{\ell,s} = \left\{a \cdot \rho^s \cdot \boldsymbol{e}_j \,:\, a \in [1 : q/\rho^s\rangle,\ \rho \nmid a,\ j \in J_\ell\right\},$$

and the expression $(q-1)\cdot\|H\|$ for $|\mathcal{X}|$ (as in (9)) still holds.

A special (interesting) case of this generalization is $q = \rho$ (in which case $b = 1$; admittedly, the term "bit interleaving" becomes a misnomer in this particular case). For the parameters of Example 1, the figures in Table I apply also to this case; yet recall that we will need the counters to be modulo q rather than modulo 2. Nevertheless, this case will turn out to be useful in the scheme that we present in Section V.

B. Construction B: Shift-and-count scheme

The construction to be presented in this section—referred to as the shift-and-count scheme or Construction B—will have a smaller column redundancy than Construction A, at the price of using more complex counters at the a-CAM rows. The construction will require that q be a prime, namely, that $\mathbb{Z}_q$ is the finite field $\mathrm{GF}(q)$; to make this clear, we will use in this section the notation p instead of q. In contrast to Construction A, where we looked at the base-2 representation of the threshold vector $\boldsymbol{\vartheta}$, here we look at the base-2 representation of the entries of the parity-check matrix of the underlying code that defines the scheme. This approach was also used in [7] to construct linear codes over $\mathbb{Z}_p$ with 0–1 parity-check matrices (see Construction 1 in Section VI), although the resulting construction herein will be different; in fact, the threshold vectors will be codewords of $\mathcal{C}$.

Let $\mathcal{C}$ be a linear $[n, k{=}n{-}r, d{=}\tau{+}1]$ code over $\mathbb{Z}_p$ and let $H = (H_{\ell,j})_{(\ell,j)\in[r\rangle\times[n\rangle}$ be an $r \times n$ parity-check matrix of $\mathcal{C}$ over $\mathbb{Z}_p$. Write $b = \lceil \log_2 p \rceil$ and, for each $(\ell, j) \in [r\rangle \times [n\rangle$,

let $\boldsymbol{h}_{\ell,j} = (h_{s,\ell,j})_{s\in[b\rangle} \in \{0,1\}^b$ be the base-2 representation of $H_{\ell,j}$, when regarded as an integer in $[p\rangle$:

$$H_{\ell,j} = \sum_{s\in[b\rangle} 2^s \cdot h_{s,\ell,j}.$$

Then, for any $\boldsymbol{\vartheta} = (\vartheta_j)_{j\in[n\rangle} \in \mathbb{Z}_p^n$ we have

$$H\boldsymbol{\vartheta}^\top = \mathbf{0} \tag{10}$$

(over $\mathbb{Z}_p$), if and only if the following equality holds for every $\ell \in [r\rangle$:

$$\sum_{s\in[b\rangle} 2^s \sum_{j\in[n\rangle} h_{\ell,s,j} \cdot \vartheta_j = 0 \tag{11}$$

(over $\mathbb{Z}_p$). The shift-and-count scheme consists of an encoding stage, which sets the redundancy entries so that (10) holds, and a detection cycle, which verifies (11) by applying to the a-CAM (as before) a set of test vectors $\mathcal{X}^* \subseteq [p\rangle^n$, yet requires counters modulo p which can also multiply by 2 (modulo p).

We now describe the set $\mathcal{X}^*$. To this end, we find it convenient to define an $(rb) \times n$ 0–1 matrix H^* which is obtained from H by replacing each entry, $H_{\ell,j}$, by the column b-vector $\boldsymbol{h}_{\ell,j} \in \{0,1\}^b$. Specifically, the rb rows of H^* are indexed by pairs $(\ell, s) \in [r\rangle \times [b\rangle$ and its n columns are indexed by $j \in [n\rangle$, and $(H^*)_{(\ell,s),j} = (h_{\ell,s,j})$. Denoting by $J^*_{\ell,s}$ the support of row (ℓ, s) in H^*, we define the set $\mathcal{X}^*$ as the union

$$\mathcal{X}^* = \bigcup_{\ell\in[r\rangle} \bigcup_{s\in[b\rangle} \mathcal{X}^*_{\ell,s},$$

where, for each $\ell \in [r\rangle$ and $s \in [b\rangle$,

$$\mathcal{X}^*_{\ell,s} = \big\{a \cdot \boldsymbol{e}_j \,:\, a \in [1:q\rangle,\ j \in J^*_{\ell,s}\big\}.$$

We have

$$|\mathcal{X}^*| = (q-1) \cdot \sum_{\ell\in[r\rangle} \sum_{s\in[b\rangle} |J^*_{\ell,s}| = (q-1) \cdot \|H^*\|. \tag{12}$$

Figure 3 presents the detection cycle, which verifies the equality (11) for every $\ell \in [r\rangle$ and $s \in [b\rangle$. The notation MODSC_i stands for a counter which can also multiply by 2 modulo p. In each iteration over (ℓ, s), the number of matches seen in line 10 (namely, the number of vectors $\boldsymbol{x} \in \mathcal{X}^*_{\ell,s}$ for which $\text{ML}_i = 1$) equals

$$\sum_{j\in J^*_{\ell,s}} \vartheta_j = \sum_{j\in[n\rangle} h_{\ell,s,j} \cdot \vartheta_j,$$

where the sums here are over $\mathbb{Z}$. Therefore, lines 6 and 10 in effect compute into MODSC_i the left-hand side of (11) (over $\mathbb{Z}_p$).

Example 2. Revisiting the a-CAM of Example 1, Table II presents the parameters for Construction B, where we have changed the values $q = 8, 16$ into the primes $p = 11, 17$, respectively. For $\tau = 1$, the matrix H is again the all-one vector. For $\tau = 2$ it is a 3×53 parity-check matrix of a shortened Hamming code over $\mathbb{Z}_p$ (while selecting columns so as to minimize $\|H^*\|$), and for $\tau = 3$ it is a 4×54 parity-check matrix of (a shortening of) the code presented in Problem 3.44 in [25], while minimizing $\|H^*\|$ over all possible choices for the irreducible polynomial $f(\cdot,\cdot)$ therein and for the entries in the second and third rows, and adding to the fourth row all possible linear combinations of the first three rows. □

```
for each i ∈ [m⟩ do                                              1
    MODSC_i ← 0;                                                 2
for each ℓ ∈ [r⟩ do                                              3
    for s ← b−1, b−2, ..., 0 do                                  4
        for each i ∈ [m⟩ do                                      5
            MODSC_i ← (2 · MODSC_i) mod p;                       6
        for each x ∈ X*_{ℓ,s} do                                 7
            apply x to the a-CAM;                                8
            for each i ∈ [m⟩ do                                  9
                MODSC_i ← (MODSC_i + ML_i) mod p;                10
    for each i ∈ [m⟩ do                                          11
        if MODSC_i ≠ 0 then flag error in row i.                 12
```

Fig. 3. Detection cycle of Construction B.

TABLE II
PARAMETERS OF CONSTRUCTION B FOR AN A-CAM WITH $k = 50$ TASK-DRIVEN COLUMNS.

τ	r	n	$\|H^*\|$		$\|\mathcal{X}^*\|$	
			$p = 11$	$p = 17$	$p = 11$	$p = 17$
1	1	51	51	51	510	816
2	3	53	148	138	1,480	2,208
3	4	54	218	209	2,180	3,344

If we take $\mathcal{C}$ to be a normalized alternant code over $\mathbb{Z}_p$ then (under the assumption that $\tau \le \sqrt{n}$),

$$r = 1 + \left\lceil \frac{p-1}{p} \cdot (\tau - 1) \right\rceil \cdot \lceil \log_p n \rceil \tag{13}$$

(see [25, Ch. 5 and Problem 8.12]). For large p, this expression is roughly $(1/2)\log_2 p$ smaller than the column redundancy, $\mathsf{r}(t, n)$ (in (4)), of Construction A. The number of rows in H^* is

$$r \cdot b = r \cdot \lceil \log_2 p \rceil \overset{(13)}{\approx} \frac{p-1}{p} \cdot \tau \log_2 n$$

and, so, if we estimate $\|H^*\|$ to be $r \cdot b \cdot n/2$ and, respectively, estimate $\|H\|$ to be $\mathsf{r}(t, n) \cdot n/2$ in (9), then the size of $\mathcal{X}^*$ in (12) is twice the size of $\mathcal{X}$ in Construction A. Note that in the case of Example 2, our estimates are quite pessimistic as we have found matrices H^* that are considerably sparser: e.g., for $\tau = 2$, the size of $\mathcal{X}^*$ for $p = 17$ in Table II is in fact smaller than the respective size of $\mathcal{X}$ for $q = 16$ in Table I.

Table III summarizes the (approximate) values of the column redundancy and of the number of test vectors for Constructions A and B when $\tau > 1$ (notice the different base of logarithms). When computing the number of test vectors, we have taken in (9) and (12) the (conservative) estimate that half of the entries in H and H^* are 1. For $\tau = 1$, the column redundancy in both schemes is 1 and the number of test vectors is $n(q-1)$.

TABLE III
APPROXIMATE COLUMN REDUNDANCY AND NUMBER OF TEST VECTORS FOR CONSTRUCTIONS A AND B ($\tau > 1$).

	Scheme	Column redundancy	Number of test vectors
A.	Bit interleaving	$\frac{1}{2} \cdot \tau \log_{\mathbf{2}} n$	$\frac{1}{4} \cdot (q-1) \cdot n \cdot \tau \log_{\mathbf{2}} n$
B.	Shift and count	$\frac{q-1}{q} \cdot \tau \log_{\boldsymbol{q}} n$	$\frac{1}{2} \cdot \frac{(q-1)^2}{q} \cdot n \cdot \tau \log_{\mathbf{2}} n$

C. External redundancy

Constructions A and B, as presented in Sections III-A–III-B, use some of the columns in the a-CAM array in order to store the redundancy symbols in each row; those symbols are programmed as thresholds, similarly to the task-driven thresholds.

In the alternate approach that we consider in this section, the redundancy symbols are stored in a separate external memory, instead of in the a-CAM; we will refer to this memory as a RAM (without committing to any specific technology).[7] This approach can be applied to both Constructions A and B; for the sake of conciseness, we will demonstrate it for Construction A only.

The conceptually simplest way to utilize such an external memory would probably be pre-computing into the memory—during the encoding stage—the contribution of the redundancy symbols to the left-hand side of (7), thereby eliminating the need for the redundancy columns in the a-CAM. Such an external-redundancy scheme, however, assumes that the redundancy symbols are arranged in columns (similarly to the a-CAM) and that they are susceptible to errors as any a-CAM cell. Yet these assumptions can be relaxed, since we are free to arrange the redundancy symbols in the RAM as we see fit, and we can use standard error correction and detection techniques to protect those symbols against errors. Thus, for the purpose of error detection of the a-CAM, we can assume that the redundancy symbols are error-free.

Doing so (with the same parity-check matrix H), we can use the full $m \times n$ a-CAM array (instead of just an $m \times k$ array) for the task-driven thresholds, with each row filled (freely) with a threshold vector $\boldsymbol{\vartheta} \in \mathbb{Z}_q^n$. Instead of requiring the equality (7), the encoding stage computes for each $s \in [b\rangle$ the syndrome vector of $\boldsymbol{\phi}_s$ with respect to the $r \times n$ parity-check matrix H:

$$\boldsymbol{\sigma}_s = (\sigma_{\ell,s})_{\ell \in [r\rangle} = H \boldsymbol{\phi}_s^\top . \tag{14}$$

Each syndrome vector $\boldsymbol{\sigma}_s$ is a ρ-ary r-vector (in particular, a binary r-vector when the radix ρ is 2) and so is each difference

$$\boldsymbol{\Delta}_{s,i} = (\Delta_{\ell,s,i})_{\ell \in [r\rangle} = \boldsymbol{\sigma}_{s+1} - \boldsymbol{\sigma}_s, \quad s \in [b\rangle,$$

where we define $\boldsymbol{\sigma}_b = \mathbf{0}$ and make explicit the dependence of $\boldsymbol{\Delta}_{s,i}$ on the row index i in the a-CAM. These differences, which amount to br symbols, are stored in the RAM. The role of the detection cycle will be now to verify (14) (instead of (7)). This is achieved by changing the initialization of the modulo-ρ counters MODC_i in lines 1–2 in Figure 2 into

$$\text{MODC}_i \leftarrow \Delta_{\ell,s,i}$$

and placing it after line 4.

Using a RAM for the redundancy has several advantages. First, the length of the codes that we use does not need to go beyond the number of columns that is dictated by the computation task. This, in turn, reduces the number of test vectors, as there are no redundancy columns to test; moreover, as we show in Example 3 below, affording a shorter code sometimes implies a smaller redundancy. Secondly, using a RAM allows the alphabet of the redundancy to be larger than the native alphabet of the a-CAM. For example, if the native alphabet is 8 and we wish to use Construction B, then we need to take $p = 11$. The task-driven thresholds, which are freely selected, can still be constrained to $[8\rangle$, yet the redundancy (or syndrome) symbols may also take values in $[8 : 11\rangle$, namely, outside the range of thresholds that can be programmed into the a-CAM.

When using Construction A, a RAM offers an additional advantage in allowing unequal error protection to lower versus higher significant bits. E.g., lower bits might need less (or no) protection if they do not affect the result; or they may need higher protection since they are more susceptible to drifts in the values of the programmed thresholds. This flexibility can be achieved by selecting different codes $\mathcal{C}_s$ (all of length n), with respective (binary) parity-check matrices H_s and redundancies r_s, for different indexes $s \in [b\rangle$ of the radix-ρ representation of the thresholds. With such varying codes, we may be able to reduce the overall number of redundancy/syndrome symbols (we omit further details). Yet even if we stick to equal error protection (or use Construction B), storing the (error-free) syndrome in a RAM may also allow us to use codes $\mathcal{C}$ with (the same minimum distance yet with) smaller redundancy. We demonstrate this in the next example.

Example 3. We revisit the a-CAM of Example 1 yet with alphabet size $q = 7$ and consider Construction B with $\tau = 3$. With internal redundancy, we would need $\mathcal{C}$ to be a linear $[n, k{=}50, d{=}4]$ code over $\mathbb{Z}_7$, and the shortest such code currently known has length $n = 55$, corresponding to redundancy $r = 5$ [14]. On the other hand, with external syndrome, we can have redundancy $r = 4$ by taking H to be the 4×50 parity-check matrix of the linear $[n{=}50, k{=}46, d{=}4]$ code over $\mathbb{Z}_7$ presented in Problem 3.44 in [25]. □

IV. CODING SCHEMES FOR THE LEE METRIC

While the previous section dealt with coding schemes for the Hamming metric, in this section we introduce two schemes for the L_1-metric. We will in fact employ known codes for the slightly more conservative Lee metric over $\mathbb{Z}_q$, which we next recall for completeness.

For an integer $q \geq 2$, we consider the elements of the alphabet $[q\rangle$ as elements of the ring $\mathbb{Z}_q$ and view

$$\lfloor q/2 \rfloor + 1, \lfloor q/2 \rfloor + 2, \ldots, q-2, q-1$$

[7] In addition to random-access memory, the acronym "RAM" can stand here for *redundancy auxiliary memory*.

as the "negative" elements in $\mathbb{Z}_q$. The absolute value of an element $z \in \mathbb{Z}_q$ is defined as the integer

$$|z|_q = \begin{cases} z & \text{if } z \text{ is "nonnegative"} \\ q - z & \text{otherwise} \end{cases}$$

(where, for computing the right-hand size, z is regarded as an integer). The Lee distance between any two vectors $\boldsymbol{y} = (y_j)_{j\in[n\rangle}$ and $\boldsymbol{y}' = (y'_j)_{j\in[n\rangle}$ in $\mathbb{Z}_q^n$ is defined by

$$\mathsf{d}_{\mathcal{L}}(\boldsymbol{y}, \boldsymbol{y}') = \sum_{j\in[n\rangle} |y_j - y'_j|_q.$$

It is easy to see that $(\boldsymbol{y}, \boldsymbol{y}') \mapsto \mathsf{d}_{\mathcal{L}}(\boldsymbol{y}, \boldsymbol{y}')$ is a metric and it never exceeds the L_1-distance between $\boldsymbol{y}$ and $\boldsymbol{y}'$ (when viewed as integer vectors).

For simplicity of the exposition, we will present the coding schemes under the framework of internal redundancy. As the codes will all be linear, the discussion in Section III-C will apply also to the constructions herein.

A. Construction C: Gray conversion scheme

The scheme to be presented here—referred to as the Gray conversion scheme or Construction C—uses the Lee-metric codes presented in [24] and suits the cases where $q = 2^b$. Those codes are based on Gray-code mappings from (the set of binary presentations of) $\mathbb{Z}_q$ to $\mathbb{Z}_2^b$; in such mappings, elements of $\mathbb{Z}_q$ that are at Lee distance 1 apart, map to vectors in $\mathbb{Z}_2^b$ that are at Hamming distance 1 apart. In particular, under such a mapping, a Lee-metric error becomes a Hamming error; hence, we can use binary codes for the Hamming distance in order to handle Lee-metric errors. Moreover, since there are Gray-code mappings that are linear over $\mathbb{Z}_2$, we can in fact obtain in this way codes for the Lee metric that are linear over $\mathbb{Z}_2$. For binary (and, more generally, ρ-ary) Gray codes, see [29] and references therein.

We now get to the details of Construction C. Given $q = 2^b$, n, and τ, let $\mathcal{C}$ be a linear $[bn, k, d{=}\tau{+}1]$ code over $\mathbb{Z}_2$ and let H be an $r \times bn$ parity-check matrix of $\mathcal{C}$, where $r = bn - k$. We will write

$$H = \left(\begin{array}{c|c|c|c} H_0 & H_1 & \dots & H_{b-1} \end{array} \right), \tag{15}$$

where H_s is an $r \times n$ submatrix of H, for each $s \in [b\rangle$.

Let $M = (M_{t,s})_{(t,s)\in[b\rangle\times[b\rangle}$ denote the following $b \times b$ matrix[8] over $\mathbb{Z}_2$:

$$M = \begin{pmatrix} 1 & -1 & & & & \\ & 1 & -1 & & & \\ & & \ddots & \ddots & & \\ & & & & 1 & -1 \\ & & & & & 1 \end{pmatrix}. \tag{16}$$

One well-known construction for a Gray code (see [29]) is the mapping $\mathbb{Z}_q \to \mathbb{Z}_2^b$ which is defined for every $w \in \mathbb{Z}_q$ by

$$w \mapsto M\boldsymbol{w},$$

where $\boldsymbol{w}$ is a binary column b-vector whose entries form the base-2 representation of (the integer) w, namely,

$$w = (1\ 2\ 4\ \dots\ 2^{b-1}) \cdot \boldsymbol{w}.$$

For $s \in [b\rangle$, define the $r \times n$ matrix $\widehat{H}_s$ by

$$\widehat{H}_s = \sum_{t\in[b\rangle} M_{t,s} H_t = H_s - H_{s-1}, \tag{17}$$

where we define H_{-1} to be the all-zero $r \times n$ matrix. Similarly to what we have done in Construction A, we regard the vector $\boldsymbol{\vartheta} = (\vartheta_j)_{j\in[n\rangle}$ of thresholds in a given row as a $b \times n$ binary array Φ in which each column j is the representation of ϑ_j to base 2 (see (5)–(6)). The encoding is carried out so that the b rows of Φ, namely, $\boldsymbol{\phi}_s = (\phi_{s,j})_{j\in[n\rangle}$ for $s \in [b\rangle$, satisfy the equality

$$\sum_{s\in[b\rangle} \widehat{H}_s \boldsymbol{\phi}_s^\top = \boldsymbol{0} \tag{18}$$

(over $\mathbb{Z}_2$); equivalently, the row bn-vector

$$\boldsymbol{\phi} = \left(\begin{array}{c|c|c|c} \boldsymbol{\phi}_0 & \boldsymbol{\phi}_1 & \dots & \boldsymbol{\phi}_{b-1} \end{array} \right)$$

is a codeword of the linear code of length bn over $\mathbb{Z}_2$ with an $r \times bn$ parity-check matrix

$$\widehat{H} = \left(\begin{array}{c|c|c|c} \widehat{H}_0 & \widehat{H}_1 & \dots & \widehat{H}_{b-1} \end{array} \right). \tag{19}$$

From (17) and (18) we get

$$\sum_{t\in[b\rangle} H_t \sum_{s\in[b\rangle} M_{t,s} \cdot \boldsymbol{\phi}_s^\top = \boldsymbol{0}.$$

Thus, the b rows of $M\Phi$, which we denote by $\boldsymbol{c}_t$, $t \in [b\rangle$, satisfy the equality

$$\sum_{t\in[b\rangle} H_t \boldsymbol{c}_t^\top = \boldsymbol{0}. \tag{20}$$

It follows from (15) and (20) that the row bn-vector

$$\boldsymbol{c} = \left(\begin{array}{c|c|c|c} \boldsymbol{c}_0 & \boldsymbol{c}_1 & \dots & \boldsymbol{c}_{b-1} \end{array} \right)$$

is a codeword of $\mathcal{C}$. Hence, we can detect any pattern of up to τ Hamming-metric errors in $\boldsymbol{c}$. Recalling that the n columns of $M\Phi$ are the Gray-code mappings of the n entries of $\boldsymbol{\vartheta}$, we conclude that we are able to detect any pattern of up to τ Lee-metric errors in $\boldsymbol{\vartheta}$.

The detection cycle is identical to that of Construction A in Figure 2, except for the selection of the test vectors (since for a given pair (ℓ, s), we cannot reuse computations made earlier for $(\ell, s' > s)$). Specifically, for $(\ell, s) \in [r\rangle \times [b\rangle$ we let

$$\widehat{\mathcal{X}}_{\ell,s} = \left\{ a \cdot 2^s \cdot \boldsymbol{e}_j \,:\, a \in [1 : q/2^s\rangle,\ j \in \widehat{J}_{\ell,s} \right\}, \tag{21}$$

where $\widehat{J}_{\ell,s}$ is the support of row ℓ in $\widehat{H}_s = H_s - H_{s-1}$. The set of test vectors is defined as the union

$$\widehat{\mathcal{X}} = \bigcup_{(\ell,s)\in[r\rangle\times[b\rangle} \widehat{\mathcal{X}}_{\ell,s} \tag{22}$$

and we have

$$|\widehat{\mathcal{X}}| = \sum_{s\in[b\rangle} \left(\frac{q}{2^s} - 1 \right) \cdot \|\widehat{H}_s\|. \tag{23}$$

[8]The signs are unnecessary over $\mathbb{Z}_2$, yet we have included them for reference in Remark 3 below.

Taking $\mathcal{C}$ as a binary alternant code, the column redundancy of this scheme is

$$\left\lceil \frac{\mathsf{r}(\tau, bn)}{b} \right\rceil \approx \frac{1}{2} \cdot \tau \log_q (bn) \approx \frac{1}{2} \cdot \tau \log_q n \qquad (24)$$

(see (4); the second approximation in (24) neglects the term $(\tau/2) \cdot \log_q(\log_2 q) = (\tau/2) \cdot (\log \log_2 q)/\log q$). Comparing with Table III, this column redundancy is smaller by almost a factor of b than that of Construction A, and is also smaller than that of Construction B. From (23) and (24) we get that the number of test vectors is at most

$$\mathsf{r}(\tau, bn) \cdot n \cdot (q-1)$$

and typically half that number, namely:

$$|\widehat{\mathcal{X}}| \approx \frac{1}{4} \cdot (q-1) \cdot n \cdot \tau \log_2 n. \qquad (25)$$

For small τ (specifically, when $\tau < q/2$), we can reduce the size of $\widehat{\mathcal{X}}$ (yet typically by less than a factor of 2), by observing that if the number of errors is τ or less, then Lee errors that occur in any entry of $\boldsymbol{\vartheta}$ will necessarily affect the λ l.s.b.'s of that entry, where $\lambda = \lceil \log_2 (\tau+1) \rceil$; moreover, the effect of the errors on the λ l.s.b.'s is the same as if those λ bits are seen as the binary representation of an element of $\mathbb{Z}_{2^\lambda}$. Hence, for the purpose of error detection, it suffices to protect only the λ l.s.b.'s of the task-driven entries in $\boldsymbol{\vartheta}$, as shown in Figure 4. This approach is useful also for handling alphabet sizes q that are not powers of 2, as long as $(\tau <)\ 2^\lambda \leq q$; in this case, we will constrain the elements in the redundancy columns to belong to $[2^b\rangle$ where $b = \lfloor \log_2 q \rfloor$ $(\geq \lambda)$, while the task-driven thresholds will be represented by $\widehat{b} = \lceil \log_2 q \rceil$ bits (of which only the λ l.s.b.'s will partake in the error-detection scheme).

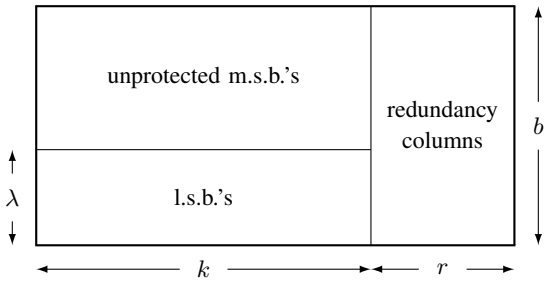

Fig. 4. Organization of the entries in Φ.

Getting into the details, denote $J = [0:k\rangle$ and $J' = [k:n\rangle$, which are, respectively, the index sets of the k task-driven columns and of the $r = n-k$ redundancy columns (where we determine r below). The respective $b \times k$ and $b \times r$ submatrices of Φ will be denoted by $(\Phi)_J$ and $(\Phi)_{J'}$. During the encoding stage, we compute br redundancy bits into $(\Phi)_{J'}$ as if the $\widehat{b} - \lambda$ m.s.b.'s in all the columns in $(\Phi)_J$ are zero, namely, as if $(\boldsymbol{\phi}_s)_J = \mathbf{0}$ for all $s \in [\lambda : \widehat{b}\rangle$. Correspondingly, we assume that $(\widehat{H}_s)_J = 0$ for $s \in [\lambda : \widehat{b}\rangle$ when computing the set $\widehat{\mathcal{X}}$ in (21)–(22). This, in turn, reduces (23) to

$$|\widehat{\mathcal{X}}| = \sum_{s \in [\lambda\rangle} \left(\frac{q}{2^s} - 1\right) \cdot \|\widehat{H}_s\| + \sum_{s \in [\lambda : b\rangle} \left(\frac{q}{2^s} - 1\right) \cdot \|(\widehat{H}_s)_{J'}\|. \qquad (26)$$

TABLE IV
PARAMETERS OF CONSTRUCTION C FOR AN A-CAM WITH $k = 50$ TASK-DRIVEN COLUMNS.

(a) $q = 8$

τ	r	n	$\|\widehat{H}_0\|$	$\|\widehat{H}_1\|$	$\|(\widehat{H}_2)_{J'}\|$	$\|\widehat{\mathcal{X}}\|$
1	1	51	51	0	0	357
2	3	53	124	87	3	1,132
3	3	53	143	106	6	1,325

(b) $q = 16$

τ	r	n	$\|\widehat{H}_0\|$	$\|\widehat{H}_1\|$	$\|(\widehat{H}_2)_{J'}\|$	$\|(\widehat{H}_3)_{J'}\|$	$\|\widehat{\mathcal{X}}\|$
1	1	51	51	0	0	0	765
2	2	52	124	84	2	2	2,411
3	2	52	140	104	4	4	2,844

Moreover, taking $(\widehat{H}_s)_J$ to be zero for $s \in [\lambda : \widehat{b}\rangle$ means that the detection capabilities of the scheme is determined only by the following $r \times \widehat{n}$ submatrix of H:

$$\big(\ H_0 \,\big|\, H_1 \,\big|\, \ldots \,\big|\, H_{\lambda-1} \,\big|\, (H_\lambda)_{J'} \,\big|\, (H_{\lambda+1})_{J'} \,\big|\, \ldots \,\big|\, (H_{b-1})_{J'}\ \big), \qquad (27)$$

where $\widehat{n} = \lambda n + (b-\lambda)r = \lambda k + br$. Thus, r can be taken to be the smallest integer satisfying

$$br \geq \mathsf{r}(\tau, \lambda k + br). \qquad (28)$$

By properly selecting the order of columns in H (thus affecting the definitions of the submatrices H_s, for $s \in [b\rangle$), we can optimize over the right-hand side of (26).

Example 4. Revisiting the a-CAM of Example 1, Table IV(a) presents the parameters for Construction C with $q = 8$. For $\tau = 1$, the numbers are the same as in Table I. For $\tau = 2, 3$ we have $\lambda = 2$ and $b = 3$ and, thus, from (28), the column redundancy is $r = |J'| = 3$. For $\tau = 2$, the matrix in (27) was selected to be a parity-check matrix of a shortened binary Hamming code of length $\widehat{n} = 109$ and redundancy 7. Thus, H_0 and H_1 are 7×53 matrices and $(H_2)_{J'}$ is a 7×3 matrix. The (distinct nonzero) columns of each matrix were selected greedily so as to minimize the expression in (26). Specifically, the 53 lightest nonzero columns of $\mathbb{Z}_2^7$ were selected into H_0; then 53 new columns were selected into H_1 so as to minimize $\|H_1 - H_0\|$; and, finally, three new columns were selected into H_2 so as to minimize $\|(H_2 - H_1)_{J'}\|$. A similar strategy was applied for $\tau = 3$, now using as columns odd-weight vectors in $\mathbb{Z}_2^8$. Table IV(b) presents the respective numbers for $q = 16$: here $b = 4$ and, for $\tau = 2, 3$, we have $\lambda = 2$ and $r = 2$. □

Remark 3. One could consider generalizing Construction C to any radix $\rho \geq 2$, as we did in Construction A. Indeed, the $b \times b$ matrix M in (16), when viewed as a matrix over $\mathbb{Z}_\rho$, defines a Gray-code mapping $\mathbb{Z}_q \to \mathbb{Z}_\rho^b$ for any $\rho \geq 2$. However, we will run into the challenge of finding linear codes $\mathcal{C}$ over $\mathbb{Z}_\rho$ for which the matrix $\widehat{H}$ in (19) (rather than the parity-check matrix H of $\mathcal{C}$) is a 0–1 matrix. In addition, with respect to the column redundancy, the advantage of the Lee-to-Hamming conversion technique diminishes as ρ becomes

TABLE V
PARAMETERS OF CONSTRUCTION D FOR AN A-CAM WITH $k = 50$ TASK-DRIVEN COLUMNS.

τ	r	n	$\|H^*\|$		$\|\mathcal{X}^*\|$	
			$p=11$	$p=17$	$p=11$	$p=17$
1	1	51	51	51	510	816
2	2	52	135	114	1,350	1,824
3	3	53	167	152	1,670	2,432

larger (corresponding to a smaller b). In fact, when $b = 1$, we reduce to Construction A. □

B. Construction D: Lee-metric shift and count

The construction that we consider in this section, referred to as Construction D, is essentially the same as Construction B in Section III-B, except that the code $\mathcal{C}$ is selected to have minimum Lee distance $\tau+1$. When specifying the parameters of a linear code, we will write $[n, k, d]_{\mathcal{L}}$ to indicate that d is the minimum Lee distance.

There are several known constructions for codes for the Lee metric, and we will concentrate here on two families of linear $[n, k{=}n{-}r, d{=}\tau{+}1]_{\mathcal{L}}$ codes over prime fields $\mathbb{Z}_p$, where $\tau < p$. These two families are the (possibly shortened) Berlekamp codes (for τ even) and normalized alternant codes (for τ odd) [25, Ch. 10], and their redundancies are given by:

$$\mathsf{r}_{\mathcal{L}}(\tau, n, p) = \begin{cases} \dfrac{\tau}{2} \cdot \lceil \log_p (2n+1) \rceil & \text{if } \tau \text{ is even} \\ \dfrac{\tau-1}{2} \cdot \lceil \log_p n \rceil + 1 & \text{if } \tau \text{ is odd} \end{cases} . \tag{29}$$

Example 5. Revisiting the a-CAM of Example 1, Table V presents the parameters for Construction D for alphabet sizes $p = 11, 17$. □

Remark 4. When the alphabet size q is not a prime, we can still use Construction D over any prime p in the range $\tau < p < q$ (the overhead in redundancy will be minimized if p is selected to be the largest in this range). While task-driven thresholds may take any value in $[q\rangle$, their effective values for encoding and detection purposes will be their remainders modulo p (being values in $[p\rangle$). This indeed works since the L_1-distance between any element in $[p\rangle$ and its remainder in $[p\rangle$ is at least p ($> \tau$) and, thus, error events cannot substitute one for the other. □

The approximations (24) and (25) for the column redundancy and the number of test vectors apply also to Construction D.

V. CODING SCHEME BASED ON THE READ CIRCUITRY

The detection schemes that were presented in Sections III and IV take advantage of the inherent parallelism of the a-CAM operation in that all rows produce their outputs on their match lines simultaneously for each input test vector; as such, these schemes do not depend on the number of rows m in the a-CAM. On the other hand, in all these schemes, the number of test vectors scales like $q \cdot \tau \cdot n \log n$ and, in particular, it grows linearly with the alphabet size q.

In this section, we present another detection strategy, which may suit large alphabets. Instead of using the match lines for output, we will use a circuitry which allows to read the (integer) sum of the thresholds on selected cells along a selected row. That circuitry exists on board the a-CAM as it is utilized during the programming stage, when each cell is read after it has been written. Specifically, this circuitry allows to select a row i and cells (i, j) along that row, with j ranging over a prescribed subset $J \subseteq [n\rangle$, by setting the input vector $\boldsymbol{x} = (x_j)_{j\in[n\rangle}$ so that $x_j = 0$ for $j \in J$ and $x_j = \infty$ otherwise (where "∞" means any value greater than $q-1$, namely, a value for which $\mathrm{L}(x, \vartheta) = 0$ for any $\vartheta \in [q\rangle$). With the same voltage applied across each memristive device along row i, a dedicated "word line" conductor, WL_i, then collects the sum of currents that flow through the selected cells; this sum, in turn, is proportional to the integer sum $\sum_{j\in J} \vartheta_{i,j}$ and, in fact, can be read as an integer through an analog-to-digital converter (ADC). Thus, the a-CAM circuitry supports instantaneous analog computation of sums of subsets of thresholds along a row (yet this computation is done separately for each row); in other words, we trade the parallelism across rows (per a particular column) for parallelism across columns (per a particular row).[9]

Our coding scheme herein, which we refer to as Construction E, uses this machinery in conjunction with Construction A (from Section III-A) when applied to the special case $b = 1$ (where the radix ρ equals q). We select the underlying code to be a linear $[n, k{=}n{-}r, \tau{+}1]$ code $\mathcal{C}$ over $\mathbb{Z}_q$ with an $r \times n$ 0–1 parity-check matrix H over $\mathbb{Z}_q$. Such codes will be discussed in Section VI; in particular, we present such codes when q is a power of 2 or when $\tau \in [1 : 4]$. In either case, the redundancy r of these codes equals $\mathsf{r}(\tau, n)$ as in (4), except when $\tau = 4$ and q that is a multiple of 3, in which case it is (at most) $\mathsf{r}(5, n)$.

The set of test vectors, $\mathcal{X}$, consists of r vectors $\boldsymbol{x}_\ell = (x_{\ell,j})_{j\in[n\rangle}$, $\ell \in [r\rangle$, which are defined as follows. For $\ell \in [r\rangle$, let J_ℓ be the support of row ℓ of H. Then,

$$x_{\ell,j} = \begin{cases} 0 & \text{if } j \in J_\ell \\ \infty & \text{otherwise} \end{cases} .$$

Denoting by $\boldsymbol{\vartheta}_i$ the vector of thresholds along row i, upon selecting row i and applying (sequentially) the r test vectors of $\mathcal{X}$ to the a-CAM, we will read in WL_i (sequentially) the r entries of the *integer* vector

$$\boldsymbol{\sigma} = H\boldsymbol{\vartheta}_i^{\top} .$$

This vector is the syndrome of $\boldsymbol{\vartheta}_i$ with respect to the parity-check matrix H, and each entry in $\boldsymbol{\sigma}$ is an integer in $[0 : n(q{-}1)]$.

τ-error detection can be achieved in one of several methods.

- Using external memory to store the entries of $\boldsymbol{\sigma}$, thus requiring $r\lceil \log_2(n(q{-}1){+}1) \rceil$ bits per row, and comparing the computed syndrome with the stored one.

[9] In principle, more than one row could be selected at a time; however, this would require multiple ADC units, which are rather expensive.

- Doing the same, yet storing only the remainders of the entries of $\boldsymbol{\sigma}$ modulo q; this requires only $r\lceil \log_2 q \rceil$ bits per row but also requires a circuit to compute the remainders upon each read from WL_i.
- Using internal redundancy and encoding the r redundancy columns so that $H\boldsymbol{\vartheta}_i^\top \equiv \mathbf{0} \pmod q$. This method, too, requires a remaindering circuit at the output of WL_i.

When q is a power of 2, the remaindering in the latter two methods is immediate, as it amounts to reading the $\log_2 q$ l.s.b.'s of the ADC. Hence, for such alphabets, these two methods are preferable over the first.

In each detection cycle, the set of test vectors, $\mathcal{X}$, is applied m times, once for each row in the a-CAM, thereby totaling to $r \cdot m$ test vectors. The naive detection scheme—where each cell in the array is read individually—corresponds to the (trivial) case where H is the $n \times n$ identity matrix (and, therefore, $r = n$). In the cases where r equals (4) (e.g., when q is a power of 2), the column redundancies of Constructions A and E are the same and, with respect to the number of test vectors, Construction E improves on Construction A when $(q-1) \cdot \|H\| > r \cdot m$, which (for $\tau > 1$) occurs when $q \gtrsim 2m/n$.

Example 6. Revisiting the a-CAM of Example 1, Table VI presents the parameters for Construction E (the values of r and n are the same as those in Table I). □

TABLE VI
PARAMETERS OF CONSTRUCTION E FOR AN A-CAM WITH $m = 512$ ROWS AND $k = 50$ TASK-DRIVEN COLUMNS.

τ	r	n	$\lvert\mathcal{X}\rvert \cdot m$
1	1	51	512
2	6	56	3,072
3	7	57	3,584

Remark 5. The strategy that was presented in this section can also be applied to Constructions B and D, yet we would then need to add to the remaindering circuits the shift operation (i.e., multiplication by 2). The number of test vectors to be applied to each row would then be the number of rows in the matrix H^*, namely, roughly $r \cdot b = r \cdot \lceil \log_2 p \rceil$. Yet with r as in (4) or (29), we do not get an improvement over Construction A. □

VI. 0–1 PARITY-CHECK MATRICES

The problem of constructing linear $[n, k{=}n{-}r, d]$ codes that have 0–1 parity-check matrices has been studied in several papers, for various applications [2],[7],[8],[11],[12]. In those papers, the underlying alphabet is either $\mathbb{Z}$ or the finite field $\mathbb{Z}_p$. For our purposes, we consider here the more general setting of systematic linear codes over $\mathbb{Z}_q$ where q is not necessarily a prime.

Writing $t = \lfloor (d-1)/2 \rfloor$, by the ordinary sphere-packing bound we have, for any code of size q^{n-r} in $\mathbb{Z}_q^n$,

$$V_q(n,t) \le q^r,$$

where $V_q(n,t)$ is the volume of a Hamming sphere of radius t in $\mathbb{Z}_q^n$, namely

$$V_q(n,t) = \sum_{i \in [0:t]} \binom{n}{i} (q-1)^i$$

(see, e.g., [25, §4.2]). Equivalently,

$$r \ge \frac{\log V_q(n,t)}{\log q} \ge t \cdot \frac{\log(q-1) + \log(n/t)}{\log q}. \tag{30}$$

For the special case where the code is linear and systematic over $\mathbb{Z}_q$ with a 0–1 parity-check matrix H, we also have

$$V_{w+1}(n,t) \le (tw+1)^r, \tag{31}$$

for any positive integer $w < q/t$: this follows from enumerating over all linear combinations of at most t columns of H where the coefficients are taken from $[0:w]$ and observing that all the resulting vectors are distinct in $[0:tw]^r$ (see [7, Theorem 3.1]). It follows from (31) that for any positive $w < q/t$:

$$r \ge \frac{\log V_{w+1}(n,t)}{\log(tw+1)} \ge t \cdot \frac{\log w + \log(n/t)}{\log(tw+1)}. \tag{32}$$

In the range where $t \lesssim \sqrt{n}$, the right-hand side of (32) is maximized for $w = 1$, in which case (32) becomes

$$r \ge \frac{\log V_2(n,t)}{\log(t+1)} \ge \frac{t \cdot \log(n/t)}{\log(t+1)}. \tag{33}$$

Combining (30) and (33) and noting that $V_q(n,t) \ge V_2(n,t)$, we get

$$\begin{aligned} r &\ge \max\left\{ \frac{\log V_q(n,t)}{\log q}, \frac{\log V_2(n,t)}{\log(t+1)} \right\} \\ &\ge \frac{t \cdot \log(n/t)}{\log(\min\{q, t+1\})}. \end{aligned} \tag{34}$$

As we are mainly interested in the regime where $t \ll n$, we use (34) as a reference when evaluating code constructions.

The case of prime q was considered in [7], where it was shown that, up to a constant factor, the lower bound (34) can be attained; the proof, however, is nonconstructive. A construction is given in Appendix B in [7] with a value of r which is greater than (33) (or (34)) by roughly $2((q-1)/q) \cdot \log_2 d$. For completeness, we describe the construction next.

Construction 1. Given a prime p, let $\tilde{\mathcal{C}}$ be a linear $[n, n{-}\tilde{r}, d]$ code over $\mathbb{Z}_p$ and let $\tilde{H} = (\tilde{H}_{\ell,j})_{\ell,j}$ be an $\tilde{r} \times n$ parity-check matrix of $\tilde{\mathcal{C}}$ over $\mathbb{Z}_p$. Writing $b = \lceil \log_2 p \rceil$ and $r = \tilde{r} \cdot b$, we construct an $r \times n$ 0–1 matrix H by replacing each entry, $\tilde{H}_{\ell,j}$, by the column b-vector $\boldsymbol{h}_{\ell,j} \in \{0,1\}^b$ whose entries form the representation of (the integer) $\tilde{H}_{\ell,j}$ to base 2:

$$\tilde{H}_{\ell,j} = (1\ 2\ 4\ \ldots\ 2^{b-1}) \cdot \boldsymbol{h}_{\ell,j}.$$

(Notice that the relationship between $\tilde{H}$ and H herein is the same as the relationship between the matrices H and H^* in Construction B in Section III-B. In both places, H stands for the parity-check matrix of the code of interest, yet here the 0–1 matrix is H while in Construction B it is H^*.) It is shown in [7] that H inherits from $\tilde{H}$ the property that every $d-1$ of its columns are linearly independent over $\mathbb{Z}_p$ (in fact, H is

a parity-check of a code that is a subcode of $\tilde{\mathcal{C}}$). If $\tilde{\mathcal{C}}$ is taken as a normalized alternant code over $\mathbb{Z}_p$ then

$$r = \tilde{r} \cdot b = \tilde{r} \cdot \lceil \log_2 p \rceil \overset{(13)}{\approx} \frac{p-1}{p} \cdot d \cdot \log_2 n,$$

which is greater than (33) (or (34)) by a factor of (approximately) $2((p-1)/p) \cdot \log_2 d$ (recall that $t = \lfloor (d-1)/2 \rfloor$. □

In the next construction, we reduce this factor to only $\log_2 d$. The construction applies to (not necessarily prime) q that is sufficiently large (relative to d, in a precise sense to be stated).

Construction 2. Let H be an $r \times n$ parity-check matrix of a linear $[n, n-r, d]$ code $\mathcal{C}$ over $\mathbb{Z}_2$, e.g., $\mathcal{C}$ is a binary alternant code, in which case $r = \mathsf{r}(d-1, n) \approx (d/2)\log_2 n$ (as in (4)). Since every $d-1$ columns in H are linearly independent over $\mathbb{Z}_2$, they are also so when H is regarded as a matrix over $\mathbb{Z}$; namely, every $d-1$ columns in (the integer 0–1 matrix) H contain a $(d-1) \times (d-1)$ submatrix whose (integer) determinant is odd (and, therefore, nonzero). In addition, H contains an $r \times r$ submatrix whose determinant is odd. It readily follows that (the integer matrix) H is a parity-check matrix of a systematic linear code over $\mathbb{Z}$; moreover, if q is a power of 2 then H (when now regarded as a matrix over $\mathbb{Z}_q$) is a parity-check matrix of a systematic linear code over $\mathbb{Z}_q$. In what follows we consider the case where q has an odd divisor.

Denote by $T(m)$ the largest absolute value of the determinant of any $m \times m$ integer 0–1 matrix. From the properties of H as an integer matrix we can conclude that every $d-1$ columns in H are linearly independent over $\mathbb{Z}_q$, for any q that is not divisible by any of the odd primes in $[3 : T(d-1)]$. To guarantee the systematic property over $\mathbb{Z}_q$, we will select H so that it has an $r \times r$ submatrix which is unimodular over $\mathbb{Z}$ (i.e., its determinant is ± 1); e.g., we can take H such that it contains I_r as a submatrix (which is always possible). □

Now, it is known that

$$T(m) \le 2^{-m} \cdot (m+1)^{(m+1)/2},$$

with equality holding if and only if an $(m+1) \times (m+1)$ Hadamard matrix exists (see [5],[13, Problem 523]). This number grows rapidly with m, yet we will be interested in small values of m. The first few values of $T(m)$ (taken from `https://oeis.org/A003432`) are listed in Table VII.

TABLE VII
LARGEST ABSOLUTE VALUE OF THE DETERMINANT OF ANY $m \times m$ INTEGER 0–1 MATRIX, FOR $m \in [1 : 6]$.

m	1	2	3	4	5	6
$T(m)$	1	1	2	3	5	9

Remark 6. We note that $T(m)$ is also the largest absolute value of the determinant of any $(m+1) \times (m+1)$ integer 0–1 matrix that contains the all-one row vector $\mathbf{1}_{m+1} = (1\ 1\ \ldots\ 1)$ as one of its rows (say, the first row). Indeed, if B is such a matrix, we can subtract the first row from any other row that starts with a 1. The resulting matrix takes the form

$$\left(\begin{array}{c|c} 1 & \mathbf{1}_m \\ \hline \mathbf{0}_m & \tilde{B} \end{array}\right)$$

where $\tilde{B}$ is an $m \times m$ matrix over $\{0, -1\}$. Hence, $|\det(B)| = |\det(\tilde{B})| \le T(m)$. □

We next consider the performance of Construction 2 for few small values of d.

$d = 2$. We take the $1 \times n$ matrix $H = \mathbf{1}_n$, which is the parity-check of the binary $[n, n-1, 2]$ parity code. Since $T(d-1) = T(1) = 1$, Construction 2 applies to any $q \ge 2$ and its redundancy, 1, is the smallest possible for minimum distance 2.

$d = 3$. We let $r = \lceil \log_2(n+1) \rceil$, which is the smallest integer that satisfies (33), and take H to be an $r \times n$ 0–1 matrix whose columns are all nonzero and distinct, with r of the columns forming the submatrix I_r. This choice corresponds to taking $\mathcal{C}$ as a binary $[n, n-r, \ge 3]$ (possibly shortened) Hamming code (which is a special case of an alternant code). Since $T(d-1) = T(2) = 1$, Construction 2 applies to any $q \ge 2$ and, in view of (34), it has the smallest redundancy possible.

$d = 4$. We let $r = \lceil \log_2 n \rceil + 1$ and take H to be an $r \times n$ 0–1 matrix whose first row is $\mathbf{1}_n$ and whose columns are all distinct, with r of the columns forming the following $r \times r$ submatrix:

$$\left(\begin{array}{c|c} 1 & \mathbf{1}_{r-1} \\ \hline \mathbf{0}_{r-1} & I_{r-1} \end{array}\right). \tag{35}$$

This choice corresponds to $\mathcal{C}$ being a binary $[n, n-r, \ge 4]$ (possibly shortened) extended Hamming code. Let J be any subset of size 3 of the coordinate set of $\mathcal{C}$ and let $(H)_J$ be the $r \times 3$ submatrix of H that is formed by the columns that are indexed by J. The matrix $(H)_J$ has full rank (of 3) over $\mathbb{Z}_2$ and, so, it has full rank also over $\mathbb{Z}$. In particular, we can find a 3×3 submatrix B in $(H)_J$ which is nonsingular over $\mathbb{Z}$; moreover, we can always assume that the first row in B is $(1\ 1\ 1)$. Yet by Remark 6 we also have $|\det(B)| \le T(2) = 1$, thereby implying that $|\det(B)| = 1$. This means that $(H)_J$ has full rank also over $\mathbb{Z}_q$, for any $q \ge 2$. In summary, compared to the case $d = 3$, we have increased the redundancy by at most 1. Now, among all binary linear codes of minimum distance 4 with a given redundancy r, the extended Hamming code is the longest possible. It readily follows that (at least) when q is even, the code $\mathcal{C}$ is the longest possible among all linear $[n, n-r, 4]$ codes over $\mathbb{Z}_q$ with a 0–1 parity-check matrix.

$d = 5$. We let $r = 2\lceil \log_2(n+1) \rceil$ and take H to be an $r \times n$ parity-check matrix of a binary $[n, n-r, \ge 5]$ alternant code which contains I_r as a submatrix. Here $T(d-1) = T(4) = 3$, so Construction 2 applies to $q = 2^b$ and to any odd q that is not divisible by 3.

For odd q that is divisible by 3, we let $r = 2\lceil \log_2 n \rceil + 1$ and take H to be an $r \times n$ parity-check matrix of a binary $[n, n-r, \ge 6]$ alternant code, with the first row being $\mathbf{1}_n$ and with r of the columns forming the $r \times r$ submatrix (35) (such a selection is always possible). Similarly to what we have done in the case $d = 3$, we show that every four columns in H are linearly independent over $\mathbb{Z}_q$. Specifically, given any subset J

of size 4 of the coordinate set of the code, the $r \times 4$ submatrix of $(H)_J$ has rank 4 over $\mathbb{Z}$ and, so, it contains a nonsingular 4×4 submatrix B whose first row is $\mathbf{1}$. By Remark 6 we get that $|\det(B)| \leq T(3) = 2$ (in $\mathbb{Z}$), i.e., $(H)_J$ has full rank also over $\mathbb{Z}_q$, for any odd q.

Finally, we consider the case where q is even with an odd divisor. Writing $q = 2^b \cdot q'$ where $q' \geq 3$ is odd, it follows from the Chinese remainder theorem that whenever the construction works over $\mathbb{Z}_{q'}$, then it also works over $\mathbb{Z}_q$ [15, Section 3.4].

Comparing the redundancy r with the lower bound (34), we see that it is optimal (up to an additive constant) when $q = 2$; however, for larger q, the main term of the lower bound is

$$\frac{2}{\log_2 3} \cdot \log_2 n \approx 1.26 \log_2 n,$$

so r is larger than the bound by a factor of (approximately) $\log_2 3 \approx 1.585$. It turns out, however, that the lower bound (34) is not tight for $d = 5$ and can be improved to

$$\sim 1.73 \log_2 n, \tag{36}$$

thus making r sub-optimal only by a factor of (roughly) 1.15. The lower bound (36) was proved in [10] (improving earlier results in [18] and [19]) and applies, in fact, to the (looser) setting of constructing the largest possible set of integer vectors in $\{0,1\}^r$ such that the sums $\boldsymbol{h}_1 + \boldsymbol{h}_2$ are different for all distinct pairs $\{\boldsymbol{h}_1, \boldsymbol{h}_2\}$ of vectors in the set. Such a set is called a *Sidon set* or a B_2*-sequence* (in $\{0,1\}^r$) [1],[9],[22] (for an application, see [6]). Clearly, the columns of any 0–1 parity-check matrix of a linear $[n, n-r, \geq 5]$ code over $\mathbb{Z}$ necessarily form such a Sidon set. In fact, the following almost-converse also holds.

Proposition 1. *Let H be an $r \times n$ integer 0–1 matrix whose columns form a Sidon set in $\{0,1\}^r$ and let H' be obtained by adding an all-one row to H. Then H' is a parity-check matrix of a linear $[n, \geq n-r-1, \geq 5]$ code over $\mathbb{Z}$.*

Proof. We assume that the rows of H' are linearly independent; otherwise, we can remove rows (while keeping the first row) and obtain a parity-check matrix for the same code. Since the columns of H' are distinct and start with a 1, any three of them are linearly independent over $\mathbb{Z}_2$ and, thus, also over $\mathbb{Z}$. Hence, the minimum distance of the code is guaranteed to be at least 4. We next show that any four columns in H' are linearly independent as well. Let J denote the index set of any four columns and suppose, to the contrary, that

$$(H')_J \cdot \boldsymbol{a} = \mathbf{0}_{r+1}$$

(over $\mathbb{Z}$) for some nonzero column vector $\boldsymbol{a} = (a_j)_{j=1}^4 \in \mathbb{Z}^4$; note that $\boldsymbol{a}$ may not contain any zero entries. Now, the rank of the $(r+1) \times 4$ submatrix $(H')_J$ is at least 3 and, so, it contains a 3×4 submatrix B of full rank 3 whose first row is the all-one vector $\mathbf{1}$ and

$$B\boldsymbol{a} = \mathbf{0}; \tag{37}$$

in particular,

$$\mathbf{1} \cdot \boldsymbol{a} = \mathbf{0}. \tag{38}$$

For $j \in [1:4]$, let B_j denote the 3×3 submatrix of B that is obtained by removing the jth column. By Cramer's rule for solving linear equations we get that, up to scaling, the solution of (37) for $\boldsymbol{a}$ is given by

$$a_j = (-1)^j \cdot \det(B_j), \quad j \in [1:4].$$

By Remark 6 it follows that $\det(B_j) = \pm 1$ for all j, where we rule out the case $\det(B_j) = 0$ since $\boldsymbol{a}$ must be all-nonzero. From (38) we then get that—up to scaling—two of the entries of $\boldsymbol{a}$ equal 1 and the remaining two equal -1. This, in turn, implies that we can find in H two distinct pairs of columns, $\{\boldsymbol{h}_1, \boldsymbol{h}_2\}$ and $\{\boldsymbol{h}_3, \boldsymbol{h}_4\}$, such that $\boldsymbol{h}_1 + \boldsymbol{h}_2 = \boldsymbol{h}_3 + \boldsymbol{h}_4$. This, however, contradicts the assumption that the columns of H form a Sidon set. □

TABLE VIII
SUMMARY OF THE (APPROXIMATE) PARAMETERS OF CONSTRUCTIONS A–E ($\tau > 1$).

	Scheme	Column redundancy	Number of test vectors
A.	Bit interleaving	$\frac{1}{2} \cdot \tau \log_{\mathbf{2}} n$	$\frac{1}{4} \cdot (q-1) \cdot n \cdot \tau \log_{\mathbf{2}} n$
B.	Shift and count	$\frac{q-1}{q} \cdot \tau \log_{\boldsymbol{q}} n$	$\frac{1}{2} \cdot \frac{(q-1)^2}{q} \cdot n \cdot \tau \log_{\mathbf{2}} n$
C, D	Lee metric	$\frac{1}{2} \cdot \tau \log_{\boldsymbol{q}} n$	$\frac{1}{4} \cdot (q-1) \cdot n \cdot \tau \log_{\mathbf{2}} n$
E	Read circuitry	$\frac{1}{2} \cdot \tau \log_{\mathbf{2}} n$	$\frac{1}{2} \cdot \boldsymbol{m} \cdot \tau \log_{\mathbf{2}} n$

VII. SUMMARY

Table VIII summarizes the (approximate) parameter values of the coding schemes that were presented in this work, when used in an $m \times n$ array with alphabet size q for detecting up to τ errors per row (the upper half of the table coincides with Table III). Constructions A and C suit cases where the alphabet size q is a power of 2 and use modulo 2 counters at the match lines, whereas Constructions B and D suit cases where q is a prime and use counters modulo q that can also multiply by 2. In all these constructions, the number of test vectors scales linearly with $(q \cdot n) \cdot \tau \log_2 n$ and does not depend on m. In contrast, in Construction E, the number of test vectors scales linearly with $m \cdot \tau \log_2 n$ and does not depend on q; the rows in this scheme are tested one at time by using an ADC, possibly with a remaindering circuit modulo q.

ACKNOWLEDGMENT

I would like to thank Cat Graves and Giacomo Pedretti from Hewlett Packard Labs for many helpful discussions.

REFERENCES

[1] L. Babai, V.T. Sós, "Sidon sets in groups and induced subgraphs of Cayley graphs," *Europ. J. Comb.,* 6 (1985), 101–114.

[2] R. Berinde, A.C. Gilbert, P. Indyk, H. Karloff, M.J. Strauss, "Combining geometry and combinatorics: A unified approach to sparse signal recovery," *Proc. 46th Annual Allerton Conf. on Communication, Control, and Computing,* Monticello, Illinois, September 2008, pp. 798–805.

[3] A. Bremler–Barr, D. Hay, D. Hendler, R.M. Roth, "PEDS: A Parallel error detection scheme for TCAM devices," *IEEE/ACM Trans. Netw.,* 18 (2010), 1665–1675.

[4] A. Bremler–Barr, D. Hay, D. Hendler, R.M. Roth, "Efficient detection of errors in associative memory," US Patent 8,887,026, 2014.

[5] J. Brenner, L. Cummings, "The Hadamard Maximum determinant problem," *Am. Math. Mon.,* 79 (1972), 626–630.

[6] N.H. Bshouty, "Optimal algorithms for the coin weighing problem with a spring scale," *Proc. 22nd Conf. on Learning Theory (COLT 2009),* Montreal, Quebec, June 2009, #4.

[7] N.H. Bshouty, H. Mazzawi, "On parity-check $(0,1)$-matrix over $\mathbb{Z}_p$," *Proc. 22nd ACM–SIAM Symp. on Discrete Algorithms (SODA'11),* San Francisco, California, January 2011, pp. 1383–1394.

[8] M. Cheraghchi, J. Ribeiro, "Simple codes and sparse recovery with fast decoding," *Proc. 2019 IEEE Int'l Symp. on Information Theory (ISIT),* Paris, France, July 2019, pp. 156–160.

[9] J. Cilleruelo, "Combinatorial problems in finite fields and Sidon sets," *Combinatorica,* 5 (2012), 497–511.

[10] G. Cohen, S. Litsyn, G. Zémor, "B_2-sequences: A new upper bound," *J. Comb. Theory Ser. A,* 94 (2001), 152–155.

[11] A. G. Dyachkov, V.V. Rykov, "On a coding model for a multiple-access adder channel," *Probl. Peredeachi Inf.,* 17:2 (1981), 26–38.

[12] E. Egorova, M. Fernandez, G. Kabatiansky, M.H. Lee, "Signature codes for weighted noisy adder channel, multimedia fingerprinting and compressed sensing," *Des. Codes Cryptogr.,* 87 (2019), 455–462.

[13] D. Faddeev, L. Sominsky, *Problems in Higher Algebra,* translated from Russian, Mir Publishers, Moscow, 1968.

[14] M. Grassl, "Bounds on the minimum distance of linear codes and quantum codes," Available online at http://www.codetables.de (accessed on January 24, 2023).

[15] K. Ireland, M. Rosen, *A Classical Introduction to Modern Number Theory,* (2nd ed.), Springer, New York, 1990.

[16] S.C. Krishnan, R. Panigrahy, S. Parthasarathy, "Error-correcting codes for ternary addressable memories," *IEEE Trans. Comput.,* 58 (2009), 275–279.

[17] C. Li, C.E. Graves, X. Sheng, D. Miller, M. Foltin, G. Pedretti, J.P. Strachan, "Analog content-addressable memories with memristors," *Nat. Commun.,* 11 (2020), #1638.

[18] B. Lindström, "Determination of two vectors from the sum," *J. Comb. Theory,* 6 (1969), 402–407.

[19] B. Lindström, "On B_2-sequences of vectors," *J. Number Theory,* 4 (1972), 261–265.

[20] H. Noda, K. Dosaka, H.J. Mattausch, T. Koide, F. Morishita, K. Arimoto, "A reliability-enhanced TCAM architecture with associated embedded DRAM and ECC," *IEICE Trans. Electron.,* E89–C (2006), 1612–1619.

[21] H. Noda, K. Inoue, M. Kuroiwa, F. Igaue, K. Yamamoto, H.J. Mattausch, T. Koide, A. Amo, A. Hachisuka, S. Soeda, I. Hayashi, F. Morishita, K. Dosaka, K. Arimoto, K. Fujishima, K. Anami, T. Yoshihara, "A cost-efficient high-performance dynamic TCAM with pipelined hierarchical searching and shift redundancy architecture," *IEEE J. Solid-State Circuits,* 40 (2005), 245–253.

[22] K. O'Bryant, "A complete annotated bibliography of work related to Sidon sequences," *Electron. J. Comb.,* Dynamic Survey No. 11 (2004).

[23] G. Pedretti, C.E. Graves, C. Li, S. Serebryakov, X. Sheng, M. Foltin, R. Mao, J.P. Strachan, "Tree-based machine learning performed in-memory with memristive analog CAM," *Nat. Commun.,* 12 (2021), #5806.

[24] A. Orlitsky, "Interactive communication of balanced distributions and of correlated files," *SIAM J. Discrete Math.,* 6 (1993), 548–564.

[25] R.M. Roth, *Introduction to Coding Theory,* Cambridge University Press, Cambridge, UK, 2006.

[26] R.M. Roth, "Fault-tolerant dot-product engines," *IEEE Trans. Inf. Theory,* 65 (2019), 2046–2057.

[27] R.M. Roth, "Analog error-correcting codes," *IEEE Trans. Inf. Theory,* 66 (2020), 4075–4088.

[28] M.Z. Shafiq, C. Meiners, Z. Qin, K. Shen, A.X. Liu, "TCAMChecker: A software based approach to the error detection and correction of TCAM-based networking systems," *J. Netw. Syst. Manage.,* 21 (2013), 335–352.

[29] B.D. Sharma, R.K. Khanna, "On m-ary Gray codes," *Inform. Sci.,* 15 (1978), 31–43.

[30] T. Song, X. Chen, X. Zhang, Y. Han, "BRAHMS: Beyond conventional RRAM-based neural network accelerators using hybrid analog memory system," *Proc. 58th ACM/IEEE Design Automation Conf. (DAC 2021),* (2021, virtual), 1033–1038.

[31] R. Wisbauer, *Foundations of Module and Ring Theory,* Gordon and Breach Science Publishers, Philadelphia, 1991.

[32] L. Zhao, L. Buoanno, R.M. Roth, S. Serebryakov, A. Gajjar, J. Moon, I. Ignowski, G. Pedretti, "RACE-IT: A reconfigurable analog CAM-crossbar engine for in-memory transformer acceleration," https://arxiv.org/abs/2312.06532.

[33] H. Zhu, K. Zhu, J. Gu, H. Jin, R.T. Chen, J.A. Incorvia, D.Z. Pan, "Fuse and mix MACAM-enabled analog activation for energy-efficient neural acceleration," *Proc. 41st IEEE/ACM Int'l Conf. on Computer-Aided Design (ICCAD 2022),* San Diego, CA (2022), #37, 1–9.