On-Access Error Correction in Certain Types of Content-Addressable Memories

Ron M. Roth Giacomo Pedretti

***Abstract*—A summing content-addressable memory (Σ-CAM) is a device which consists of an $\ell \times n$ array of cells, where each cell can be programmed to one of two states, 0 or 1. The input to the array is a binary ℓ-vector, and the output is an integer n-vector whose entries are the Hamming distances between the input vector and the contents of the cells along each of the n columns in the array. An ordinary binary CAM can be seen as a quantized Σ-CAM where the output at each column is just an indicator whether the Hamming distance is 0 (full match) or not. A summing ternary CAM (Σ-TCAM) is a variant of a Σ-CAM where the state/input alphabet is endowed with a third symbol ("don't care") whose Hamming distance from any symbol is defined to be 0. In this work, several coding schemes are presented for on-access correction of errors in both Σ-CAMs and Σ-TCAMs. All schemes entail allocating several columns in the array for redundancy such that, when an input vector is applied to the array, errors in the output vector can be corrected, provided that their number (measured by either the Hamming metric or the L_1-metric) does not exceed a prescribed value. In the case of Σ-CAMs, the operation of the array resembles that of a discrete vector–matrix multiplier; therefore, the presented error-correction schemes for Σ-CAMs build upon schemes that have been proposed for such multipliers. The adaptation of such schemes to Σ-TCAMs, however, is more involved due to the don't care symbol: the schemes in this case make use of a special type of positional binary representation of integer pairs where the representations of the two integers in a pair do not share a 1 in the same position.**

***Index Terms*—Content-addressable memory, Error-correcting codes for the L_1-metric, Accelerators for Hamming distance computation, Vector–matrix multiplier.**

I. Introduction

Hereafter we use the following notation. For $y, z \in \mathbb{Z}$, we denote by $[y : z]$ the integer subset $\{x \in \mathbb{Z} : y \le x \le z\}$ and by $[y : z\rangle$ the set $[y : z-1]$; we will use the shorthand notation $[z\rangle$ for $[0 : z\rangle$ and $\mathbb{B}$ for $[2\rangle$. For an integer vector $\boldsymbol{x}$, we denote its Hamming weight and L_1-norm by $\mathsf{w}(\boldsymbol{x})$ and $\|\boldsymbol{x}\|_1$, respectively. For an $\ell \times n$ matrix $A = (a_{i,j})_{(i,j)\in[\ell\rangle\times[n\rangle}$ (or a row n-vector if $\ell = 1$) and a subset $X \subseteq [n\rangle$, we let $(A)_X$ denote the $\ell \times |X|$ submatrix of A that is formed by the columns that are indexed by X. For $y, z \in \mathbb{Z}$ such that $z > 0$, the notation $y \bmod z$ stands for the remainder in $[z\rangle$ when y is divided by z. The ring of integers modulo q will be denoted by $\mathbb{Z}_q$.

The work of Ron M. Roth was done in part at Hewlett Packard Labs, Milpitas, CA. This work was supported in part by the U.S. Army Research Office under Agreement W911NF2110355, and in part by the Israel Science Foundation under Grants 1713/20 and 2369/24. Parts of this work were presented at the IEEE Int'l Symposium on Information Theory, Athens, Greece (July 2024).

Ron M. Roth is with the Computer Science Department, Technion, Haifa 3200003, Israel. Email: ronny@cs.technion.ac.il.

Giacomo Pedretti is with Hewlett Packard Labs, Milpitas, CA 95035, USA. Email: giacomo.pedretti.90@gmail.com.

A. Summing content-addressable memory (Σ-CAM)

Let $\mathrm{N} : \mathbb{B} \times \mathbb{B} \to \mathbb{B}$ be the inequality-test (XOR) function, which is defined for every $x, a \in \mathbb{B}$ by

$$\mathrm{N}(x, a) = [\![x \ne a]\!] = \begin{cases} 1 & \text{if } x \ne a \\ 0 & \text{otherwise} \end{cases}, \tag{1}$$

where $[\![\,\cdot\,]\!]$ denotes the Iverson bracket (which evaluates to 1 if its argument is true, and to 0 otherwise). In this work, by a *summing content-addressable memory* (Σ-CAM) we refer to a device which consists of an $\ell \times n$ array of *cells*, with each cell $(i, j) \in [\ell\rangle \times [n\rangle$ implementing the function $x \mapsto \mathrm{N}(x, a_{i,j})$, for some *internal state* value $a_{i,j} \in \mathbb{B}$; we represent the internal states as an array (matrix) $A = (a_{i,j})_{(i,j)\in[\ell\rangle\times[n\rangle} \in \mathbb{B}^{\ell\times n}$, with $\boldsymbol{a}_i$ standing for row i and A_j $(= (A)_{\{j\}})$ for column j. The input to the array is a row vector ("search key") $\boldsymbol{x} = (x_i)_{i\in[\ell\rangle} \in \mathbb{B}^\ell$, with x_i serving as the input to all the cells along row i. Each column ("match line") j in the array computes the integer sum of the outputs of the cells along the column,

$$c_j = \sum_{i\in[\ell\rangle} \mathrm{N}(x_i, a_{i,j}) = \mathsf{w}(\boldsymbol{x} - A_j), \tag{2}$$

and these sums form the output row vector, $\boldsymbol{c} = (c_j)_{j\in[n\rangle} \in [0 : \ell]^n$, of the array; namely, a Σ-CAM computes the Hamming distances[1] between the input vector and the contents of the cells along each of the n columns in the array.[2]

Hereafter, we use the notation $\mathsf{S}(\boldsymbol{x}, A)$ for the vector in $[0 : \ell]^n$ whose entries are given by the right-hand side of (2); namely, $\mathsf{S}(\boldsymbol{x}, A)$ is the result $\boldsymbol{c}$ of the computation when an input vector $\boldsymbol{x}$ is applied to a Σ-CAM that is programmed to an array A of internal states. The zero entries in $\boldsymbol{c} = \mathsf{S}(\boldsymbol{x}, A)$ are referred to as "matches," and an ordinary *binary* CAM (BCAM) can be seen as a quantized Σ-CAM whose output is the binary n-vector $([\![c_j = 0]\!])_{j\in[n\rangle}$ (which points at the columns where matches occur).

The Σ-CAM serves as a model for accelerators that have been proposed for computing the Hamming distance in various applications, and there are several designs for their cells (using CMOS as well as resistive technologies) and for the circuitry that computes the Hamming distance along each column [3],[5],[6],[10],[13],[15],[21]. In those designs, a value $\mathrm{N}(x_i, a_{i,j}) = 1$ is typically realized by setting cell (i, j) to some high conductance, whereas when $\mathrm{N}(x_i, a_{i,j}) = 0$,

[1] Admittedly, the term "HD-CAM" would perhaps be a better choice for naming a Σ-CAM; yet this term is already taken in [5] to mean a kind of quantized Σ-CAM.

[2] In the common descriptions of CAMs, the input is supplied to the *columns* while the match lines are the *rows*. The transposition here is due to the connection that we make in the sequel to vector–matrix multiplication.

the cell is virtually open (i.e., has zero conductance). One method for obtaining the Hamming distance is by fixing the voltage level of the match lines and measuring the current that flows through each column; this current is proportional to the number of high-conductance cells along the column. In the applications of interest, the matrix A is modified much less frequently than the input vector $\boldsymbol{x}$.

We next mention several applications of Σ-CAMs. (i) Hyperdimensional computing [13], where Hamming distance is used to compare a query hypervector against stored (learned) hypervectors, enabling the associative memory to identify the closest match for tasks such as language recognition; (ii) Memory Augmented Neural Networks (MANN) [15], where computing the Hamming distance serves as a hardware-efficient approximation of cosine distance for nearest neighbor search in few-shot learning tasks; (iii) Genomic analysis [10], where locality-sensitive hashed signatures of raw genomic signals are compared against stored reference patterns; and (iv) Solution of Boolean satisfiability problems (SAT) [21], where (as part of a heuristic algorithm) a Σ-CAM is used to count how many literals satisfy each clause.

Inaccuracies while programming the cells of the Σ-CAM, manufacturing defects, and noise while reading the output vector are examples of factors that may cause the actually-read row vector, $\boldsymbol{y} = (y_j)_{j\in[n\rangle} \in \mathbb{Z}^n_{\geq 0}$, to differ from the correct vector $\boldsymbol{c} = \mathsf{S}(\boldsymbol{x}, A)$. The error vector is defined as the following vector in $\mathbb{Z}^n$:

$$\boldsymbol{e} = (e_j)_{j\in[n\rangle} = \boldsymbol{y} - \boldsymbol{c}.$$

In a scenario where a faulty cell generates the wrong output, yet that output is still in $\mathbb{B}$, the output at the column that contains this cell will be changed by ± 1; we will refer to such an event as an L_1-*error*. The L_1-norm $\|\boldsymbol{e}\|_1$ then bounds from below the number of L_1-errors, with equality attained if all the faulty cells along the same column err in the same direction. Our focus in this paper will be on this scenario, in which case one of our design parameters will be the largest L_1-norm τ of $\boldsymbol{e}$ that can be tolerated, which—when presuming nothing about the internal states in the array A—serves as a proxy for the largest number τ of faulty cells that can be tolerated. However, we will also consider the scenario where a faulty cell may have a larger effect on the output of the column, in which case τ will stand for the largest Hamming weight of $\boldsymbol{e}$ that can be tolerated. Hereafter, the notation $\|\boldsymbol{e}\|$ stands for either $\|\boldsymbol{e}\|_1$ or $\mathsf{w}(\boldsymbol{e})$, depending on the context.

B. Coding model

The aim of this work is proposing coding schemes for on-access error correction in Σ-CAMs (and in a variant thereof to be presented below). To this end, we adapt the framework proposed in [23] for integer vector–matrix multipliers (VMMs) to Σ-CAMs, since there is a close relationship between the functions of these two devices. Specifically, we see from (2) that

$$\mathsf{S}(\boldsymbol{x}, A) = \sum_{i\,:\,x_i=0} \boldsymbol{a}_i + \sum_{i\,:\,x_i=1} (\mathbf{1}_n - \boldsymbol{a}_i), \tag{3}$$

where $\mathbf{1}_n$ stands for the all-one row n-vector. It follows that

$$\mathsf{S}(\boldsymbol{x}, A) = \underbrace{(\mathbf{1}_\ell - 2\boldsymbol{x})}_{\boldsymbol{u}} A + \mathsf{w}(\boldsymbol{x}) \cdot \mathbf{1}_n, \tag{4}$$

where $\boldsymbol{u} \in \{\pm 1\}^\ell$; namely, up to an additive multiple of the all-one vector, the Σ-CAM performs a multiplication of a vector in $\{\pm 1\}^\ell$ by a matrix in $\mathbb{B}^{\ell\times n}$.

With this in mind, our strategy will be—as in [23]—to program the array A so that the first k $(< n)$ entries in $\boldsymbol{c} = \mathsf{S}(\boldsymbol{x}, A)$ will carry the target computation of the Σ-CAM, while the remaining $n-k$ entries of $\boldsymbol{c}$ will contain redundancy symbols, which can be used to detect or correct computational errors. Thus, the programmed $\ell \times n$ matrix A will have the structure

$$A = (A' \mid A''), \tag{5}$$

where $A' = (A)_{[k\rangle}$ and $A'' = (A)_{[k:n\rangle}$; the computed output row vector for an input vector $\boldsymbol{x} \in \mathbb{B}^\ell$ will then be $\boldsymbol{c} = \mathsf{S}(\boldsymbol{x}, A) = (\boldsymbol{c}' \mid \boldsymbol{c}'')$, where $\boldsymbol{c}' = (\boldsymbol{c})_{[k\rangle} = \mathsf{S}(\boldsymbol{x}, A')$ is the target computation while $\boldsymbol{c}'' = (\boldsymbol{c})_{[k:n\rangle} = \mathsf{S}(\boldsymbol{x}, A'')$ is the redundancy part.

Given positive integers ℓ, n, and $k < n$, a Σ-CAM *coding scheme* is a pair $(\mathcal{E}, \mathcal{D})$, where

- $\mathcal{E} : \mathbb{B}^{\ell\times k} \to \mathbb{B}^{\ell\times n}$ is an *encoding mapping* such that for every $A' \in \mathbb{B}^{\ell\times k}$, the image $A = \mathcal{E}(A')$ has the form (5) for some $A'' \in \mathbb{B}^{\ell\times(n-k)}$, and—
- $\mathcal{D} : [0:\ell]^n \to [0:\ell]^k \cup \{\text{“e”}\}$ is a *decoding mapping* (where “e” designates decoding failure).

The set

$$\mathcal{C} = \mathcal{C}(\mathcal{E}) = \left\{\mathsf{S}(\boldsymbol{x}, \mathcal{E}(A')) \;:\; A' \in \mathbb{B}^{\ell\times k},\ \boldsymbol{x} \in \mathbb{B}^\ell\right\}$$

is the *code induced by* $\mathcal{E}$ and its members are called *codewords*; namely, $\mathcal{C} \subseteq [0:\ell]^n$ and its codewords are all the possible output vectors that can be obtained when A' ranges over all possible $\ell \times k$ matrices over $\mathbb{B}$ and $\boldsymbol{x}$ ranges over all possible 2^ℓ input vectors in $\mathbb{B}^\ell$. We refer to n and k as the length and dimension, respectively, of the coding scheme.[3] For row vectors $\boldsymbol{z} \in \mathbb{Z}^n$ that arise in connection with a coding scheme, we use the notation $\boldsymbol{z}'$ and $\boldsymbol{z}''$ for their k-prefix $(\boldsymbol{z})_{[k\rangle}$ and $(n-k)$-suffix $(\boldsymbol{z})_{[n-k\rangle}$, respectively; this notational convention extends to $\ell \times n$ matrices as well.

Given ℓ, k, and a prescribed number of errors τ (measured by either the L_1-metric or the Hamming metric), the goal is to have a coding scheme $(\mathcal{E}, \mathcal{D})$ with the smallest possible n such that for every $A' \in \mathbb{B}^{\ell\times k}$ and $\boldsymbol{x} \in \mathbb{B}^\ell$, the k-prefix, $\boldsymbol{c}'$, of the respective codeword $\boldsymbol{c} = \mathsf{S}(\boldsymbol{x}, \mathcal{E}(A')) = (\boldsymbol{c}' \mid \boldsymbol{c}'')$ in $\mathcal{C}$ can be recovered correctly in the presence of τ errors or less. Namely, for every read vector $\boldsymbol{y} \in [0:\ell]^n$ such that $\|\boldsymbol{y} - \boldsymbol{c}\| \leq \tau$,

$$\mathcal{D}(\boldsymbol{y}) = \boldsymbol{c}' \left(= \mathsf{S}(\boldsymbol{x}, A')\right)$$

(alternatively, n can be given and k is to be maximized). Note that the decoding mapping $\mathcal{D}$ is *not* required to recover the redundancy part $\boldsymbol{c}''$.

More generally, given nonnegative integers τ and σ, a coding scheme $(\mathcal{E}, \mathcal{D})$ is said to correct τ errors and detect

[3] The use of the term dimension for k captures the fact that the k-prefixes of the codewords in $\mathcal{C}$ range over all the vectors in $[0:\ell]^k$; note, however, that the actual dimension of the linear span of $\mathcal{C}$ over $\mathbb{R}$ can be larger than k.

$\tau+\sigma$ errors (in the L_1-metric or the Hamming metric) if the following conditions hold for every computed output vector $\boldsymbol{c}=\mathsf{S}(\boldsymbol{x},A)\in\mathcal{C}$ and the respective read vector $\boldsymbol{y}\in[0:\ell]^n$.

- If $\|\boldsymbol{y}-\boldsymbol{c}\|\leq\tau$ then $\mathcal{D}(\boldsymbol{y})=\boldsymbol{c}'$.
- Otherwise, if $\|\boldsymbol{y}-\boldsymbol{c}\|\leq\tau+\sigma$ then $\mathcal{D}(\boldsymbol{y})\in\{\boldsymbol{c}',\text{“e”}\}$.

The *modified* minimum distance of $\mathcal{C}$, denoted $\mathsf{d}(\mathcal{C})$, is defined as the smallest L_1/Hamming distance between any two codewords in $\mathcal{C}$ *having distinct k-prefixes*:

$$\mathsf{d}(\mathcal{C})=\min_{\substack{\boldsymbol{c}_1,\boldsymbol{c}_2\in\mathcal{C}:\\ \boldsymbol{c}'_1\neq\boldsymbol{c}'_2}}\|\boldsymbol{c}_1-\boldsymbol{c}_2\|. \tag{6}$$

The next result follows from [23] (see Proposition 1 and Appendix A therein).[4]

Proposition 1. *Let $\mathcal{E}:\mathbb{B}^{\ell\times k}\to\mathbb{B}^{\ell\times n}$ be an encoding mapping and let τ and σ be nonnegative integers such that*

$$2\tau+\sigma<\mathsf{d}(\mathcal{C}(\mathcal{E})).$$

Then there exists a decoding mapping $\mathcal{D}:[0:\ell]^n\to[0:\ell]^k\cup\{\text{“e”}\}$ such that the coding scheme $(\mathcal{E},\mathcal{D})$ can correct τ errors and detect $\tau+\sigma$ errors.

Unless otherwise stated, $\mathcal{E}$ will be *separable*, namely, for each row index $i\in[\ell\rangle$, the contents of row i in $\mathcal{E}(A')$ will only be a function of the contents of row i in A' (and not of the rest of the rows in A'), and that function will be the same for all i; moreover, that function will not depend on ℓ. In the case of separable encoders, we will abuse notation and assume that the domain and range of $\mathcal{E}$ are $\mathbb{B}^k$ and $\mathbb{B}^n$, respectively.

Let $\mathrm{Im}(\mathcal{E})\subseteq\mathbb{B}^n$ be the set of all images of (a separable) $\mathcal{E}$. It follows from (3)–(4) that

$$\mathcal{C}=\Bigg\{\sum_{i\in[\ell\rangle}b_i\cdot\boldsymbol{c}_i+b\cdot\mathbf{1}_n\ :\ \boldsymbol{c}_i\in\mathrm{Im}(\mathcal{E}),\\ b_i\in\{\pm1\},\ b\in[0:\ell],\ \text{ and }\ \sum_{i\in[\ell\rangle}b_i=\ell-2b\Bigg\}.$$

Equivalently,

$$\mathcal{C}=\Bigg\{\sum_{i\in[m\rangle}b_i\cdot\boldsymbol{c}_i+b\cdot\mathbf{1}_n\ :\ m\in[1:\ell],\ \text{distinct }\boldsymbol{c}_i\in\mathrm{Im}(\mathcal{E}),\\ b_i\in\mathbb{Z},\ \sum_{i\in[m\rangle}|b_i|\leq\ell,\ b\in[0:\ell],\ \text{and }\sum_{i\in[m\rangle}b_i=\ell-2b\Bigg\}.$$

In particular, $\ell\cdot\boldsymbol{c}\in\mathcal{C}$ for every $\boldsymbol{c}\in\mathrm{Im}(\mathcal{E})$, and $(\ell/2)\cdot\mathbf{1}_n\in\mathcal{C}$ when ℓ is even. This motivates us to look for coding schemes $(\mathcal{E},\mathcal{D})$ for which $\mathrm{Im}(\mathcal{E})$ is a subset of (the intersection of $\mathbb{B}^n$ with) an ambient module $\mathcal{M}\subseteq\mathbb{Z}^n$ over $\mathbb{Z}$ that contains $\mathbf{1}_n$ and, in addition, satisfies[5] $2\tau+\sigma<\mathsf{d}(\mathcal{M})$, so that *a fortiori* $2\tau+\sigma<\mathsf{d}(\mathcal{C}(\mathcal{E}))$. For simplicity, we will usually assume hereafter that $\sigma=0$.

[4]While $\mathsf{d}(\cdot)$ is called just “minimum distance” in [23], we have added here the adjective “modified” to avoid confusion with the ordinary meaning of this term.

[5]The modified minimum distance $\mathsf{d}(\mathcal{M})$ is defined as in (6), where k is implied by the coding scheme $(\mathcal{E},\mathcal{D})$. Note that $\mathsf{d}(\mathcal{M})$ also equals the minimum L_1-norm/Hamming weight of any vector in $\mathcal{M}$ whose k-prefix is nonzero.

Such coding schemes can be obtained through modifications of the constructions presented in [23] for the special case where the matrix in the VMM is over $\mathbb{B}$. Those modifications may require some mild conditions on the length n, but otherwise incur no penalty in the redundancy.

C. Outline of this work

We will start by describing in Section II the coding scheme which results from a modification of the Hamming-metric construction in [23, §V]. As a general construction, this construction is perhaps the simplest, and it fits both the L_1-metric and the Hamming metric. Better (yet typically more complex) constructions exist for the L_1-metric when τ is sufficiently small compared to n, and we discuss those in Sections V through VII.

In Section III (and continuing in subsequent sections as well), we consider a variant of Σ-CAMs that is based on cells of a *ternary* content-addressable memory (in short, TCAM). TCAMs are extensions of ordinary binary CAMs where the internal state value of a cell and the input to the cell are allowed to take a third “don't care” symbol $*$. Thus, the input and state alphabets of the cells are both $\mathbb{B}_*=\mathbb{B}\cup\{*\}$, and each cell implements the function $\mathrm{T}:\mathbb{B}_*\times\mathbb{B}_*\to\mathbb{B}$ defined in Table I (the restriction of the truth table to its last two rows and columns coincides with the function $\mathrm{N}(\cdot,\cdot)$ in (1)). TCAMs are used in network routers for address classification and packet filtering [18] and, more recently, have demonstrated their capabilities for more applications, such as machine learning [15],[16] and genomic analysis [14].

TABLE I
TRUTH TABLE OF THE FUNCTION $(x,\vartheta)\mapsto\mathrm{T}(x,\vartheta)$.

$x\backslash\vartheta$	$*$	0	1
$*$	0	0	0
0	0	0	1
1	0	1	0

The introduction of the symbol $*$ appears to affect our coding schemes in a nontrivial way. In particular, we will need a special type of positional binary representation of integer pairs where the representations of the two integers in a pair do not share a 1 in the same position. Bases of such representations, which we call *bi-spanners*, may be of independent interest and their properties will be presented in Section IV. Compared to Σ-CAMs, our constructions for Σ-TCAMs incur an increase of the redundancy by roughly 60%.

To the best of our knowledge, our work is the first to study coding schemes for on-access error correction in Σ-CAMs and Σ-TCAMs. With respect to BCAMs and TCAMs, however, several papers have addressed error-correction methods for these devices, as we describe in our final discussion in Section VIII.

II. Construction based on Hamming-metric codes

We present the (separable) construction from [23, §V] with the required modification for Σ-CAMs. For the purpose of this

construction, the set of allowable error patterns is characterized by two parameters: the largest number τ_{c} columns in A that contain faulty cells, and the largest number τ_{i} of L_1-errors per column ("inner errors"), namely, τ_{c} and τ_{i} bound from above—respectively—the Hamming weight and the L_∞-norm of the error vector $\boldsymbol{e}$. Thus, $(\tau_{\mathrm{c}}, \tau_{\mathrm{i}}) = (\tau, 1)$ corresponds to the case where up to τ L_1-errors can occur, yet with at most one such error occurring per column. Taking $(\tau_{\mathrm{c}}, \tau_{\mathrm{i}}) = (\tau, \tau)$ will subsume the case where the overall number of L_1-errors is at most τ. Finally, taking $(\tau_{\mathrm{c}}, \tau_{\mathrm{i}}) = (\tau, \ell)$ will correspond to the Hamming metric, where at most τ columns may be erroneous, without any further assumptions on the number of L_1-errors per column.

Given the number of columns n, an upper bound τ_{c} on the number of erroneous columns, and an upper bound τ_{i} on the number of L_1-errors per column, let $p > 2\tau_{\mathrm{i}}$ be an odd prime and let $m = \lceil \log_2 p \rceil$. We select a respective Hamming-metric linear τ_{c}-error-correcting $[\tilde{n}, k]$ code C over $\mathbb{Z}_p$, which is assumed to satisfy the following three conditions.

a) It contains the all-one codeword.[6]
b) It is systematic, i.e., there is a $\mathbb{Z}_p$-linear mapping $\mathsf{E} : \mathbb{Z}_p^k \to \mathsf{C}$ such that for every $\boldsymbol{u} \in \mathbb{Z}_p^k$, the image $\mathsf{E}(\boldsymbol{u})$ has $\boldsymbol{u}$ as its k-prefix.
c) It has an efficient bounded-distance decoder $\mathsf{D} : \mathbb{Z}_p^{\tilde{n}} \to \mathbb{Z}_p^{\tilde{n}}$: for a received word $\tilde{\boldsymbol{y}} \in \mathbb{Z}_p^{\tilde{n}}$, the decoder returns the true error vector $\tilde{\boldsymbol{e}} \in \mathbb{Z}_p^{\tilde{n}}$, provided that $\mathsf{w}(\tilde{\boldsymbol{e}}) \le \tau_{\mathrm{c}}$.

The parameters n and $\tilde{n}$ are related—by design—by

$$n = k + m(\tilde{n} - k). \tag{7}$$

We next describe the (separable) encoder, $\mathcal{E}_0 : \mathbb{B}^k \to \mathbb{B}^n$, of the proposed coding scheme through its action on a given row $\boldsymbol{a}' \in \mathbb{B}^k$. Regarding $\boldsymbol{a}'$ as a vector in $\mathbb{Z}_p^k$, it is first extended by the systematic encoder for C into a codeword

$$\mathsf{E}(\boldsymbol{a}') = \tilde{\boldsymbol{c}} = (\tilde{c}_v)_{v \in [\tilde{n}\rangle} \tag{8}$$

of C, where $(\tilde{c})_{[k\rangle} = \boldsymbol{a}'$. Now, in the next step of the construction in [23], for each $v \in [\tilde{n} - k\rangle$, the redundancy symbol $\tilde{c}_{k+v}$ $(\in \mathbb{Z}_p)$ was expanded into its base-2 representation $\tilde{\boldsymbol{a}}_v \in \mathbb{B}^m$, i.e.,

$$\tilde{c}_{k+v} = \tilde{\boldsymbol{a}}_v \cdot \boldsymbol{\omega}_m^\top, \tag{9}$$

where

$$\boldsymbol{\omega}_m = \begin{pmatrix} 1 & 2 & 2^2 & \dots & 2^{m-1} \end{pmatrix}.$$

Here, instead of (9), we set the representation $\tilde{\boldsymbol{a}}_v$ to be a vector in $\mathbb{B}^m$ so that

$$((2^m - 2) \cdot \tilde{c}_{k+v}) \equiv \tilde{\boldsymbol{a}}_v \cdot \widehat{\boldsymbol{\omega}}_m^\top \pmod p, \tag{10}$$

where $\widehat{\boldsymbol{\omega}}_m$ is obtained from $\boldsymbol{\omega}_m$ by changing its last entry into $2^{m-1} - 1$. Since $(2^m - 2)/2 < p \ne 2^m - 2$, the multiplier $2^m - 2$ in (10) is invertible modulo p and, since $p \le 2^m - 1$, the expansion (10) is indeed always possible (sometimes with two different representing m-vectors $\tilde{\boldsymbol{a}}_v$); moreover, for $\tilde{c}_{k+v} = 1$ we can take $\tilde{\boldsymbol{a}}_v = \mathbf{1}_m$ (which will be our selection even if another representation is possible).[7]

[6]This condition is essentially equivalent to requiring that C has a codeword of Hamming weight $\tilde{n}$, since we can always re-scale the coordinates of the code.

Input: $\boldsymbol{a}' \in \mathbb{B}^k$.
Output: $(\boldsymbol{a}' \mid \boldsymbol{a}'') \in \mathbb{B}^n$.
Let $\tilde{\boldsymbol{c}} = (\tilde{c}_v)_{v \in [\tilde{n}\rangle} \leftarrow \mathsf{E}(\boldsymbol{a}')$;
For each $v \in [\tilde{n} - k\rangle$ do {
  If $\tilde{c}_{k+v} = 1$ then
    $\tilde{\boldsymbol{a}}_v \leftarrow \mathbf{1}_m$;
  Else
    Set $\tilde{\boldsymbol{a}}_v \in \mathbb{B}^m$ so that (10) holds;
}
Let $\boldsymbol{a}'' \leftarrow (\tilde{\boldsymbol{a}}_0 \mid \tilde{\boldsymbol{a}}_1 \mid \dots \mid \tilde{\boldsymbol{a}}_{\tilde{n}-k-1})$.

Fig. 1. Encoding mapping $\mathcal{E}_0 : \boldsymbol{a}' \mapsto (\boldsymbol{a}' \mid \boldsymbol{a}'')$.

Finally, as in [23], the image of $\boldsymbol{a}'$ under $\mathcal{E}_0$ is defined as:

$$\mathcal{E}_0(\boldsymbol{a}') = \boldsymbol{a} = (\boldsymbol{a}' \mid \boldsymbol{a}''),$$

where

$$\boldsymbol{a}'' = (\boldsymbol{a})_{[k:n\rangle} = (\tilde{\boldsymbol{a}}_0 \mid \tilde{\boldsymbol{a}}_1 \mid \dots \mid \tilde{\boldsymbol{a}}_{\tilde{n}-k-1})$$

(which is of length $m(\tilde{n} - k)$, thereby confirming (7)).

Figure 1 summarizes the encoding process of a typical row in the Σ-CAM (compare with Figure 7 in [23]; for more clarity, we have sorted here the coordinates of $\boldsymbol{a}''$ differently). By conditions (a) and (b) on C it follows that $\mathsf{E}(\mathbf{1}_k) = \mathbf{1}_{\tilde{n}}$ and, so, $\mathcal{E}_0(\mathbf{1}_k) = \mathbf{1}_n$. Also, it can be readily seen that the linear span of $\mathrm{Im}(\mathcal{E}_0)$ over $\mathbb{Z}$ is a module $\mathcal{M} \subseteq \mathbb{Z}^n$ over $\mathbb{Z}$. To see the properties of this module, define $\lambda : \mathbb{Z}^n \to \mathbb{Z}_p^{\tilde{n}}$ to map each vector

$$\boldsymbol{y} = (\boldsymbol{y}' \mid \tilde{\boldsymbol{y}}_0 \mid \tilde{\boldsymbol{y}}_1 \mid \dots \mid \tilde{\boldsymbol{y}}_{\tilde{n}-k-1}) \in \mathbb{Z}^n,$$

where $\boldsymbol{y}' = (\boldsymbol{y})_{[k\rangle}$ and $\tilde{\boldsymbol{y}}_v = (\boldsymbol{y})_{[k+vm:k+(v+1)m\rangle}$ $(\in \mathbb{Z}^m)$, to the following image $\lambda(\boldsymbol{y}) \in \mathbb{Z}_p^{\tilde{n}}$:

$$\begin{aligned} \big(\lambda(\boldsymbol{y})\big)_{[k\rangle} &= \boldsymbol{y}' \bmod p, \\ \big(\lambda(\boldsymbol{y})\big)_{[k:\tilde{n}\rangle} &= (2^m - 2)^{-1} \cdot \big(\tilde{\boldsymbol{y}}_v \cdot \widehat{\boldsymbol{\omega}}_m^\top\big)_{v \in [\tilde{n}-k\rangle} \bmod p \end{aligned}$$

(where the MOD p operation is applied componentwise). Similarly to [23], λ herein is a homomorphism and it maps $\mathcal{M}$ to C. This, in turn, implies that we will be able to decode (efficiently) the vector $\lambda(\boldsymbol{y})$ into the correct codeword in C, whenever the number of erroneous symbols in $\boldsymbol{y}$ (and therefore in $\lambda(\boldsymbol{y})$) does not exceed τ_{c}. Furthermore, if the number of L_1-errors in each entry of $\boldsymbol{y}'$ does not exceed τ_{i}, then, from the inequality $p > 2\tau_{\mathrm{i}}$, we will be able to correct all these errors (see Figure 8 in [23]; note that we might not correct the remaining $n - k$ entries in $\boldsymbol{y}$ uniquely, yet this is allowed in our setting).

Specializing to the case where C is a normalized extended primitive BCH code over $\mathbb{Z}_p$, it satisfies conditions (a)–(c) above, and so do certain shortenings of it (see Remark 1

[7]If we used $\boldsymbol{\omega}_m$ instead of $\widehat{\boldsymbol{\omega}}_m$ in (10), we would need to change the multiplier $2^m - 2$ in (10) into $2^m - 1$. In turn, we would need $2^m - 1$ to be invertible modulo p, which is indeed the case except when p is a Mersenne prime.

below). The redundancy that we then get is bounded from above by

$$n-k \overset{(7)}{=} m(\tilde{n}-k) \le \left(1+\left\lfloor \frac{p-1}{p}\cdot 2\tau_{\mathrm{c}}\right\rfloor \cdot \lceil \log_p \tilde{n}\rceil\right)\cdot\lceil \log_2 p\rceil\,,$$

with equality holding when $\tau_{\mathrm{c}} \ll \tilde{n}$ (which is our regime of interest) [22, p. 260, Problem 8.13]. Since $\lceil \log_p \tilde{n}\rceil < 1+\log_p n$, the redundancy behaves like

$$\frac{\lceil \log_2 p\rceil}{\log_2 p}\cdot\left\lfloor \frac{p-1}{p}\cdot 2\tau_{\mathrm{c}}\right\rfloor \cdot \log_2 n + O(\tau_{\mathrm{c}}) \tag{11}$$

(where the constant multiplier in the $O(\tau_{\mathrm{c}})$ term is proportional to $\log_2 p \approx \log_2 \tau_{\mathrm{i}}$).

Example 1. For the case $\tau_{\mathrm{i}}=1$, where at most one L_1-error is tolerated per column, we take $p=3$, in which case

$$n-k = \frac{2}{\log_2 3}\cdot\left\lfloor \frac{4\tau_{\mathrm{c}}}{3}\right\rfloor \cdot \log_2 n + O(\tau_{\mathrm{c}}) \approx 1.68\cdot\tau_{\mathrm{c}}\cdot\log_2 n.$$

In particular, for $\tau_{\mathrm{c}}=1$ we get $n-k=(2/\log_2 3)\log_2 n + O(1) \approx 1.26\log_2 n$, and twice as much (i.e., $2.52\log_2 n$) when $\tau_{\mathrm{c}}=2$. Yet for $\tau_{\mathrm{c}}=2$ we can do better by replacing the BCH codes with Preparata codes, which are linear over $\mathbb{Z}_4$ [9]; the redundancy thus obtained is only $2\log_2 n$. (We will get a similar redundancy also with the construction of Section VI, with the additional benefit that the two L_1-errors may also occur in the same column of the array.) □

Example 2. For the case $\tau_{\mathrm{i}}=2$ we may take $p=5$, in which case we get the following approximation of the ratio between the redundancy and $\tau_{\mathrm{c}}\cdot\log_2 n$:

$$\frac{n-k}{\tau_{\mathrm{c}}\cdot\log_2 n} \approx \frac{24}{5\log_2 5} \approx 2.07.$$

However, due to the rounding-up in the $\lceil\log_2 p\rceil$ term in (11), we will in fact get a smaller redundancy if we select $p=7$:

$$\frac{n-k}{\tau_{\mathrm{c}}\cdot\log_2 n} \approx \frac{36}{7\log_2 7} \approx 1.83.$$

□

Remark 1. Recall that a normalized extended primitive BCH code C over $\mathbb{Z}_p$, which has length $\tilde{n}=p^h$ for some $h\in\mathbb{Z}^+$, contains the all-one codeword and so it satisfies condition (a). This condition also holds for any code that is obtained by shortening C on the zero entries of any nonzero codeword of C (possibly after re-scaling of the coordinates). For example, for any proper divisor $s > 2\tau_{\mathrm{c}}$ of p^h-1, the code C contains $t=(p^h-1)/s$ codewords $\boldsymbol{c}_i$, $i\in[t\rangle$, each of Hamming weight $s+1$, whose supports overlap only on the coordinate that corresponds to the code locator 0; moreover, all their other nonzero entries are 1. Thus, for any $w\in[0:t]$, the Hamming weight of the codeword $\sum_{i\in[w\rangle}\boldsymbol{c}_i$ of C is either $s\cdot w$ (if p divides w) or $s\cdot w+1$ (otherwise), namely, we can achieve all these code lengths by shortening C while satisfying condition (a). For the practical range of parameters, p^h-1 will have sufficiently many divisors (in particular, it is always divisible by 2 and $p-1$). E.g., when $\tau_{\mathrm{i}}=1$ we can take $p=3$, in which case the values for $h=4,5,6$ are, respectively, $80=2^4\cdot 5$, $242=2\cdot 11^2$, and $728=2^3\cdot 7\cdot 13$. Or when $\tau_{\mathrm{i}}=2$ we can take $p=5$, in which case $5^4-1=624=2^4\cdot 3\cdot 13$. □

For large τ_{i}, the $(p-1)/p$ term in (11) becomes practically 1 and the expression for the redundancy becomes approximately

$$(2\tau_{\mathrm{c}}-1)\cdot\log_2 n. \tag{12}$$

We end this section by comparing this figure with a lower bound on the redundancy of any separable coding scheme $(\mathcal{E},\mathcal{D})$. In such a scheme, the possible contents of the n cells along any given row in the $\ell\times n$ array A must all be codewords of some τ_{c}-error-correcting binary code that contains the all-one codeword $\mathbf{1}_n$. This condition is necessary even under the generous assumption that all the remaining cells in A are error-free and their outputs are fully known to the decoder $\mathcal{D}$. Hence, by the sphere-packing bound, the redundancy of $(\mathcal{E},\mathcal{D})$ is bounded from below by $\log_2\binom{n}{\tau_{\mathrm{c}}} > \tau_{\mathrm{c}}\cdot\log_2(n/\tau_{\mathrm{c}}-1)$, which is approximately half the expression (12) in the regime where $\tau_{\mathrm{c}}\ll n$. (Note, however, that in the degenerate case where $\ell=1$, we can actually approach this lower bound by taking $\mathcal{E}$ to be an encoder of a shortened binary BCH code that contains the codeword $\mathbf{1}_n$: the redundancy of this code is $\tau_{\mathrm{c}}\cdot\lceil\log_2 n\rceil$.)

III. Extension to Σ-TCAMs

In this section, we consider the on-access error-correction problem in Σ-TCAMs, namely, Σ-CAMs that are based on TCAM cells. We first point out that if an entry x_i of the input vector $\boldsymbol{x}=(x_i)_{i\in[\ell\rangle}$ is the "don't care" symbol $*$ then the cells along row i in the array will produce the all-zero vector and therefore will not affect the output vector. Therefore, in the discussion that follows, we can do our analysis assuming that the input vector $\boldsymbol{x}$ is in $\mathbb{B}^\ell$.

For $i\in[\ell\rangle$, we will use the notation $\boldsymbol{\theta}_i=(\vartheta_{i,j})_{j\in[n\rangle}$ for the internal state vector (over $\mathbb{B}_*^n$) in an $\ell\times n$ Σ-TCAM array. For $x\in\mathbb{B}_*$, we let $\boldsymbol{a}_i^{(x)}=(a_{i,j}^{(x)})_{j\in[n\rangle}$ denote the vector in $\mathbb{B}^n$ whose entries are the outputs of the TCAM cells along row i when the input symbol is x; namely,

$$\boldsymbol{a}_i^{(x)} = \left(a_{i,j}^{(x)}\right)_{j\in[n\rangle} = \left(\mathrm{T}(x,\vartheta_{i,j})\right)_{j\in[n\rangle}$$

and, in particular, $\boldsymbol{a}_i^{(*)}\equiv\mathbf{0}$. Eq. (3) then becomes

$$\mathsf{S}(\boldsymbol{x},A) = \sum_{i\in[\ell\rangle}\boldsymbol{a}_i^{(x_i)} = \sum_{i\,:\,x_i=0}\boldsymbol{a}_i^{(0)} + \sum_{i\,:\,x_i=1}\boldsymbol{a}_i^{(1)}. \tag{13}$$

Note that due to the $*$-cells, we generally no longer have the relationship $\boldsymbol{a}_i^{(1)}=\mathbf{1}_n-\boldsymbol{a}_i^{(0)}$.

Eq. (13) suggests that a Σ-TCAM performs the following multiplication of a vector in $\mathbb{B}^{2\ell}$ by a $2\ell\times n$ matrix over $\mathbb{B}$:

$$(\mathbf{1}_\ell-\boldsymbol{x}\mid\boldsymbol{x})\begin{pmatrix}A^{(0)}\\A^{(1)}\end{pmatrix}, \tag{14}$$

where $A^{(0)}$ and $A^{(1)}$ are $\ell\times n$ matrices whose rows are $(\boldsymbol{a}_i^{(0)})_{i\in[\ell\rangle}$ and $(\boldsymbol{a}_i^{(1)})_{i\in[\ell\rangle}$, respectively. The caveat, however, is that the vectors $\boldsymbol{a}_i^{(0)}$ and $\boldsymbol{a}_i^{(1)}$ have to be *non-intersecting*, namely, they cannot have a 1 in the same position. Now, if we just applied a separable encoder $\mathcal{E}$ to (non-intersecting) $(\boldsymbol{a}^{(0)})',(\boldsymbol{a}^{(1)})'\in\mathbb{B}^k$, we could end up with rows

$$\boldsymbol{a}^{(0)}=((\boldsymbol{a}^{(0)})'\mid(\boldsymbol{a}^{(0)})'') \quad\text{and}\quad \boldsymbol{a}^{(1)}=((\boldsymbol{a}^{(1)})'\mid(\boldsymbol{a}^{(1)})'')$$

that are intersecting on their redundancy parts $(\boldsymbol{a}^{(0)})''$ and $(\boldsymbol{a}^{(1)})''$. Thus, if the $\ell \times n$ array of cells is viewed as the $2\ell \times n$ matrix in (14), then—due to the non-intersection requirement also on the redundancy parts—our coding schemes for Σ-TCAMs will not be separable *per se*; however, we will limit the dependence between the rows in the matrix only to pairs of rows—namely, $\boldsymbol{a}_i^{(0)}$ and $\boldsymbol{a}_i^{(1)}$—that correspond to the same row i of cells in the physical array. On the other hand, unlike the setting in Section II, as we no longer have the relationship $\boldsymbol{a}_i^{(1)} = \mathbf{1}_n - \boldsymbol{a}_i^{(0)}$, we can remove the requirement that the all-one vector be a codeword in (the ambient module $\mathcal{M}$ that contains) the induced code.[8]

Our adaptation of the construction of [23] to Σ-TCAMs will therefore differ from what we did in Section II. Specifically, we let C, E, and D be as defined therein, except that m will be selected differently below, and we will no longer need condition (a). Given two vectors $(\boldsymbol{a}^{(0)})', (\boldsymbol{a}^{(1)})' \in \mathbb{B}^k$ that represent the outputs of the first k cells along a given row in the Σ-TCAM, we apply a systematic encoder for C as in (8) to both of them,

$$\mathsf{E}\left((\boldsymbol{a}^{(0)})'\right) = \tilde{\boldsymbol{c}}^{(0)} \quad \text{and} \quad \mathsf{E}\left((\boldsymbol{a}^{(1)})'\right) = \tilde{\boldsymbol{c}}^{(1)},$$

to produce codewords $\tilde{\boldsymbol{c}}^{(0)}$ and $\tilde{\boldsymbol{c}}^{(1)}$ of C. The expansion (9), however, will be changed into

$$\tilde{c}_{k+v}^{(x)} \equiv \tilde{\boldsymbol{a}}_v^{(x)} \cdot \boldsymbol{\rho}^\top \pmod p, \quad x \in \mathbb{B}, \tag{15}$$

where $\tilde{\boldsymbol{a}}_v^{(0)}$ and $\tilde{\boldsymbol{a}}_v^{(1)}$ are non-intersecting in $\mathbb{B}^m$ and $\boldsymbol{\rho}$ is a fixed integer vector in $\mathbb{Z}^m$ selected so that (15) can be satisfied for any arbitrary pair $(\tilde{c}_{k+v}^{(0)}, \tilde{c}_{k+v}^{(1)}) \in \mathbb{Z}_p^2$. One naive choice for such a vector $\boldsymbol{\rho}$ is

$$\boldsymbol{\rho} = (\boldsymbol{\omega}_{m'} \mid \boldsymbol{\omega}_{m'}),$$

where $m' = \lceil \log_2 p \rceil$. Here $m = 2m' = 2\lceil \log_2 p \rceil$, and we can always select $\tilde{\boldsymbol{a}}_v^{(0)}$ (respectively, $\tilde{\boldsymbol{a}}_v^{(1)}$) so that its first (respectively, last) m' entries are all zero. Recall however that the redundancy of the scheme is $n - k = m(\tilde{n} - k)$, which means that such a choice for $\boldsymbol{\rho}$ will double the redundancy compared to the Σ-CAM construction of Section II. Our aim is to do better than this naive solution and, to this end, we introduce the following definition.

Let $\mathcal{G}$ be an Abelian group and let $\mathcal{X}$ be a subset of $\mathcal{G} \times \mathcal{G}$. A *bi-spanner* of $\mathcal{X}$ over $\mathcal{G}$ is a multiset $\mathcal{R} = \{\rho_v\}_{v \in [m\rangle}$ of elements of $\mathcal{G}$ such that for every pair $(\xi^{(0)}, \xi^{(1)}) \in \mathcal{X}$ there exist *disjoint* subsets $\mathcal{I}^{(0)}$ and $\mathcal{I}^{(1)}$ of $[m\rangle$ such that

$$\xi^{(0)} = \sum_{v \in \mathcal{I}^{(0)}} \rho_v \quad \text{and} \quad \xi^{(1)} = \sum_{v \in \mathcal{I}^{(1)}} \rho_v \tag{16}$$

(where a sum over an empty set is defined to be zero). Equivalently, writing $\mathcal{R}$ as a vector $\boldsymbol{\rho} = (\rho_v)_{v \in [m\rangle} \in \mathcal{G}^m$, there exist non-intersecting $\boldsymbol{a}^{(0)}, \boldsymbol{a}^{(1)} \in \mathbb{B}^m$ such that

$$\xi^{(0)} = \boldsymbol{a}^{(0)} \cdot \boldsymbol{\rho}^\top \quad \text{and} \quad \xi^{(1)} = \boldsymbol{a}^{(1)} \cdot \boldsymbol{\rho}^\top. \tag{17}$$

The representation in (15) corresponds to the case $\mathcal{G} = \mathbb{Z}_q$ and $\mathcal{X} = \mathbb{Z}_q^2 = \mathbb{Z}_q \times \mathbb{Z}_q$, where q is a prime p (but for the construction in Section V we will consider values q that are not necessarily prime). For $m \in \mathbb{Z}^+$, let q_m be the largest integer such that $\mathbb{Z}_q^2$ has a bi-spanner of size m over $\mathbb{Z}_q$ for any $q \le q_m$. The rightmost column in Table II lists several values of q_m, which were found by computer search (the quantities ρ_m^+ and z_m^* in the table will be introduced in the next section). For a given $q = p$, we will select m to be the smallest so that $q_m \ge q$, thus allowing us to satisfy (15) for any $(\tilde{c}_{k+v}^{(0)}, \tilde{c}_{k+v}^{(1)}) \in \mathbb{Z}_q^2$. For the range of values of q (> 2) that is covered by the table, we see that such an m will be smaller than the naive one, and, as we show in the next section, this in fact holds in general. We will use bi-spanners later on in other constructions as well (and the values q therein are not necessarily prime).

TABLE II
FIRST FEW ELEMENTS OF ρ_m^+, z_m^*, AND q_m, AND BI-SPANNERS OF $[z_m^*\rangle^2$.

m	ρ_m^+	z_m^*	Bi-spanner of $[z_m^*\rangle^2$ over $\mathbb{Z}$	q_m
1	1	1	$\{1\}$	1
2	2	2	$\{1, 1\}$	2
3	3	3	$\{1, 1, 2\}$	3
4	4	4	$\{1, 1, 2, 2\}$	6
5	6	7	$\{-2, 1, 2, 3, 6\}$	9
6	9	11	$\{-3, 1, 2, 4, 6, 8\}$	16
7	14	18	$\{-6, 1, 2, 4, 5, 8, 16\}$	24
8	21	28	$\{-6, 1, 2, 4, 5, 8, 16, 18\}$	38
9	31	47	$\{-10, 1, 2, 4, 5, 8, 16, 24, 33\}$	
10	47	74	$\{-10, 1, 2, 4, 5, 8, 16, 24, 36, 56\}$	
11	70	115	$\{-10, 1, 2, 4, 5, 8, 16, 24, 36, 56, 84\}$	

Example 3. Referring to the parameters in Example 1, for $p = 3 = q_3$ we can take $m = 3$ (while the naive construction would require taking $m = 2m' = 4$). The vector $\boldsymbol{\rho} = (1\ 1\ 2)$ is a bi-spanner of $\mathbb{Z}_3^2$ over $\mathbb{Z}_3$, and Table III shows respective representations of the elements of $\mathbb{Z}_3^2$ (as in (17)) and the respective TCAM state vectors $\boldsymbol{\theta}$ (due to symmetry, it suffices to list only pairs $(\xi^{(0)}, \xi^{(1)})$ where $\xi^{(0)} \ge \xi^{(1)}$; for the remaining pairs just switch between $\boldsymbol{a}^{(0)}$ and $\boldsymbol{a}^{(1)}$ and, respectively, between 0 and 1 in $\boldsymbol{\theta}$). Compared to the Σ-CAM construction in Example 1, where we have taken $m = \lceil \log_2 3 \rceil = 2$, the

TABLE III
REPRESENTATION OF THE ELEMENTS OF $\mathbb{Z}_3^2$ BY THE BY-SPANNER $(1\ 1\ 2)$.

$(\xi^{(0)}, \xi^{(1)})$	$\boldsymbol{a}^{(0)}$	$\boldsymbol{a}^{(1)}$	$\boldsymbol{\theta}$
$(0, 0)$	000	000	$***$
$(1, 0)$	100	000	$1**$
$(1, 1)$	100	010	$10*$
$(2, 0)$	001	000	$**1$
$(2, 1)$	001	100	$0*1$
$(2, 2)$	110	001	110

[8] Some designs of TCAM cells, such as the one shown in [7, Fig. 2(a)] and [11, Fig. 1], allow the internal state to be in a fourth "reject" state: the output of the cell is then 1 for an input $x \in \mathbb{B}$, and is 0 when $x = *$. In this scenario, $a_{i,j}^{(0)}$ and $a_{i,j}^{(1)}$ can both be 1, i.e., $\boldsymbol{a}_i^{(0)}$ and $\boldsymbol{a}_i^{(1)}$ may intersect, and the machinery in [23, §III] can be applied as is, without any modifications. The requirement that $\boldsymbol{a}^{(0)}$ and $\boldsymbol{a}^{(1)}$ do not intersect fits also the case where the "reject" symbol appears in the input vector (but not as a state value): in this case, the output values along a row will form the (integer) sum $\boldsymbol{a}^{(0)} + \boldsymbol{a}^{(1)}$, implying that our Σ-TCAM coding schemes will work also for this scenario. In contrast, we have no satisfactory solution for the case where both the input and the state values can be "reject," due to the following closure requirement: for any pair $(\boldsymbol{a}^{(0)}, \boldsymbol{a}^{(1)})$ of codewords in the induced code, their bitwise logical-"or," $\boldsymbol{a}^{(0)} \vee \boldsymbol{a}^{(1)}$, should be a codeword too.

Σ-TCAM construction incurs an increase of the redundancy by a factor of $3/2$. □

Example 4. Referring to the parameters in Example 2, if we select $p = 5$ then, since $q_4 = 6$, we can take $m = 4$ (instead of $m = 3$ in that example), in which case the redundancy for Σ-TCAMs will increase only by $4/3$ compared to Σ-CAMs:

$$\frac{n-k}{\tau_{\mathrm{c}} \cdot \log_2 n} \approx \frac{4}{3} \cdot \frac{24}{5 \log_2 5} \approx 2.76.$$

The vector $\boldsymbol{\rho} = (1\ 2\ 3\ 4)$ is a bi-spanner of $\mathbb{Z}_5^2$ over $\mathbb{Z}_5$, and Table IV shows the respective contents of the TCAM cells.

Interestingly, the savings that we gained in Example 2 by selecting $p = 7$ therein does not propagate to Σ-TCAMs: for $p = 7$ we now need to take $m = 5$, resulting in an increase by $5/3$ of the redundancy compared to Σ-CAMs:

$$\frac{n-k}{\tau_{\mathrm{c}} \cdot \log_2 n} \approx \frac{5}{3} \cdot \frac{36}{7 \log_2 7} \approx 3.05.$$

□

TABLE IV
REPRESENTATION OF THE ELEMENTS OF $\mathbb{Z}_5^2$ BY THE BI-SPANNER $(1\ 2\ 3\ 4)$.

$(\xi^{(0)}, \xi^{(1)})$	$\boldsymbol{\theta}$	$(\xi^{(0)}, \xi^{(1)})$	$\boldsymbol{\theta}$	$(\xi^{(0)}, \xi^{(1)})$	$\boldsymbol{\theta}$
$(0,0)$	$****$	$(2,2)$	$*100$	$(4,0)$	$***1$
$(1,0)$	$1***$	$(3,0)$	$**1*$	$(4,1)$	$0**1$
$(1,1)$	$10*0$	$(3,1)$	$0*1*$	$(4,2)$	$*0*1$
$(2,0)$	$*1**$	$(3,2)$	$*01*$	$(4,3)$	$**01$
$(2,1)$	$01**$	$(3,3)$	$001*$	$(4,4)$	$0*01$

IV. PROPERTIES OF BI-SPANNERS

In this section, we present some properties of the sequence $(q_m)_m$ that was defined in Section III; in particular, we show that the sequence $(\log_2 q_m)/m$ converges to a limit which is greater than 0.622. Thus, our strategy of selecting m for a given $q = p$ to be the smallest so that $q_m \geq q$ yields a redundancy for Σ-TCAMs which, for large q, is less than $1/0.622 < 1.607$ times the respective redundancy for Σ-CAMs (while the naive approach doubles the redundancy).

The construction of bi-spanners $\boldsymbol{\rho}$ for the case $\mathcal{G} = \mathbb{Z}$ and

$$\mathcal{X} = \mathcal{X}_r = \left\{ (\xi^{(0)}, \xi^{(1)}) \in \mathbb{Z}_{\geq 0}^2 \ : \ \xi^{(0)} + \xi^{(1)} \leq r \right\},$$

for some positive integer r, has been considered in the literature [19, Seqs. A005428 and A073941], with the following two additional requirements.

B1) The entries of $\boldsymbol{\rho}$ are all positive.
B2) The sum of entries of $\boldsymbol{\rho}$ equals r.

We next summarize the construction in [19, Seq. A005428].

For any $(\xi^{(0)}, \xi^{(1)}) \in \mathcal{X}_r$ let $\xi^{(*)} = r - \xi^{(0)} - \xi^{(1)}$, and for disjoint subsets $\mathcal{I}^{(0)}, \mathcal{I}^{(1)} \subseteq [m\rangle$ let $\mathcal{I}^{(*)} = [m\rangle \setminus (\mathcal{I}^{(0)} \cup \mathcal{I}^{(1)})$. Under the conditions (B1)–(B2), we can rewrite (16) as:

$$\xi^{(x)} = \sum_{v \in \mathcal{I}^{(x)}} \rho_v, \quad \text{for all } x \in \mathbb{B}_*. \tag{18}$$

Henceforth, we find it convenient to append $\xi^{(*)}$ to the pair $(\xi^{(0)}, \xi^{(1)})$ and define $\widehat{\mathcal{X}}_r$ to be the following set of integer triples:

$$\widehat{\mathcal{X}}_r = \left\{ \boldsymbol{\xi} \in \mathbb{Z}_{\geq 0}^3 \ : \ \|\boldsymbol{\xi}\|_1 = r \right\}.$$

Now, if $\boldsymbol{\rho}$ is a bi-spanner of $\mathcal{X}_r$ that satisfies conditions (B1)–(B2) then, from (18), for every triple $\boldsymbol{\xi} \in \widehat{\mathcal{X}}_r$ we must have $\|\boldsymbol{\rho}\|_\infty \leq \|\boldsymbol{\xi}\|_\infty$. In particular,

$$\|\boldsymbol{\rho}\|_\infty \leq \min_{\boldsymbol{\xi} \in \widehat{\mathcal{X}}_r} \|\boldsymbol{\xi}\|_\infty = \lceil r/3 \rceil.$$

Conversely, given $r \in \mathbb{Z}^+$, we can construct a bi-spanner $\boldsymbol{\rho}$ of $\mathcal{X}_r$ by joining the element $\lceil r/3 \rceil$ to a bi-spanner $\boldsymbol{\rho}'$ of $\mathcal{X}_{r'}$ where $r' = r - \lceil r/3 \rceil = \lfloor 2r/3 \rfloor$ (indeed, subtracting $\lceil r/3 \rceil$ from the largest entry in any triple in $\widehat{\mathcal{X}}_r$ yields a triple in $\widehat{\mathcal{X}}_{r'}$). Moreover, $\boldsymbol{\rho}$ satisfies condition (B1) (respectively, (B2)), if and only if so does $\boldsymbol{\rho}'$ (with r replaced by r' in condition (B2)). We can now apply this process recursively.

For $m \in \mathbb{Z}^+$, denote by r_m^+ the largest integer r for which one can obtain in this recursive manner a bi-spanner of $\mathcal{X}_r$ of size m that satisfies conditions (B1)–(B2). From the backward recursion

$$r_{m-1}^+ = \lfloor 2r_m^+/3 \rfloor, \quad m = 2, 3, \cdots, \tag{19}$$

we get the forward recursion

$$r_m^+ = 1 + \lfloor 3r_{m-1}^+/2 \rfloor, \quad m = 2, 3, \cdots, \tag{20}$$

with[9] $r_1^+ = 1$.

Define the infinite sequence $(\rho_m^+)_{m \geq 0}$ by $\rho_0^+ = 1$ and

$$\rho_m^+ = 1 + \lfloor r_m^+/2 \rfloor, \quad m \in \mathbb{Z}^+. \tag{21}$$

We have

$$\rho_{m-1}^+ \overset{(20)+(21)}{=} r_m^+ - r_{m-1}^+ \overset{(19)}{=} \lceil r_m^+/3 \rceil, \quad m \in \mathbb{Z}^+,$$

and, so, by induction on m it readily follows that $\boldsymbol{\rho}^+ = (\rho_v^+)_{v \in [m\rangle}$ is a bi-spanner of $\mathcal{X}_{r_m^+}$ that satisfies conditions (B1)–(B2). In particular,

$$\rho_m^+ \overset{(21)+(B2)}{=} 1 + \left\lfloor \frac{1}{2} \sum_{v \in [m\rangle} \rho_v^+ \right\rfloor, \quad m \in \mathbb{Z}^+. \tag{22}$$

By (20) and (21) it follows that $\rho_m^+ \geq \rho_{m-1}^+$ for every $m \in \mathbb{Z}^+$, and the inequality is strict when $m > 1$. In addition,

$$\lim_{m \to \infty} \frac{\log_2 \rho_m^+}{m} \overset{(21)}{=} \frac{\log_2 r_m^+}{m} \overset{(20)}{=} \log_2(3/2) \approx 0.585 \tag{23}$$

(see [8]). Table II presents the first few values of $(\rho_m^+)_m$.

From (21) we see that $[\rho_m^+\rangle^2 \subseteq \mathcal{X}_{r_m^+}$; therefore, $\boldsymbol{\rho}^+$ is also a bi-spanner of $[q\rangle^2 = [q\rangle \times [q\rangle$ over $\mathbb{Z}$, whenever $q \leq \rho_m^+$. And this holds also when we do the arithmetic modulo q, namely, $\boldsymbol{\rho}^+$ is a bi-spanner of $\mathbb{Z}_q^2$ over $\mathbb{Z}_q$ (which is the motivating case for our study of bi-spanners); hence,

$$q_m \geq \rho_m^+. \tag{24}$$

[9]Note that r_m^+ in (19) is not a multiple of 3; otherwise, this equality would hold also when r_m^+ is increased by 1, contrary to the assumption that r_m is the *largest* r for which the recursive construction yields a bi-spanner of $\mathcal{X}_r$ of size m. Now, by (19), if $r_m^+ \equiv 1 \pmod 3$ then $r_{m-1}^+ = 2(r_m^+ - 1)/3$ (which is even), and if $r_m^+ \equiv 2 \pmod 3$ then $r_{m-1}^+ = 2(r_m^+ - 2)/3 + 1$ (which is odd). In either case, the equality (20) holds.

The next proposition states that the construction $\boldsymbol{\rho}^+$ is optimal for the sets $\mathcal{X}_r$ and $[q\rangle^2$, under the constraint that the entries of $\boldsymbol{\rho}^+$ are all positive. The proof can be found in the Appendix.

Proposition 2. *For every $m \in \mathbb{Z}^+$:*

a) ρ_m^+ is the largest integer q such that $[q\rangle^2$ has an all-positive bi-spanner of size m over $\mathbb{Z}$, and—

b) r_m^+ is the largest integer r for which $\mathcal{X}_r$ has an all-positive bi-spanner of size m over $\mathbb{Z}$.

It turns out that one can do strictly better if the bi-spanner may have negative entries as well. Denoting by r_m the largest integer $r \in \mathbb{Z}^+$ for which $\mathcal{X}_r$ has a bi-spanner $(\rho_v)_{v\in[m\rangle} \in \mathbb{Z}^m$ that satisfies condition (B2), we clearly have $r_m \geq r_m^+$, and the inequality is strict for large enough m (see Eqs. (27)–(28) below); in fact, for $m = 6$ we already have $r_6 = 19$ with the bi-spanner

$$\{-2, 1, 2, 3, 6, 9\},$$

compared to $r_6^+ = 17$ with the bi-spanner

$$\{1, 1, 2, 3, 4, 6\}.$$

Turning to the set $[q\rangle^2$, let z_m denote the largest integer q for which $[q\rangle^2$ has a (not necessarily all-positive) bi-spanner of size m over $\mathbb{Z}$. The sequences $(r_m)_m$ and $(z_m)_m$ are related by

$$1 + \lfloor r_m/2 \rfloor \leq z_m \leq 1 + r_{m+1}. \tag{25}$$

Indeed, any bi-spanner of $\mathcal{X}_{2(q-1)}$ over $\mathbb{Z}$ is also a bi-spanner of $[q\rangle^2$, and any bi-spanner of the latter is also a bi-spanner of $\mathcal{X}_{q-1}$ and can be made to sum to $q-1$ by adding at most one more element.

Now, suppose that $\boldsymbol{\rho}$ and $\tilde{\boldsymbol{\rho}}$ are, respectively, bi-spanners of $[q\rangle^2$ and $[\tilde{q}\rangle^2$ over $\mathbb{Z}$; then it is easy to see that

$$(\boldsymbol{\rho} \mid q \cdot \tilde{\boldsymbol{\rho}}) \quad \text{and} \quad (\tilde{\boldsymbol{\rho}} \mid \tilde{q} \cdot \boldsymbol{\rho})$$

are bi-spanners of $[q \cdot \tilde{q}\rangle^2$ over $\mathbb{Z}$. It follows that the sequence $(z_m)_m$ is super-multiplicative, i.e., for any $m_1, m_2 \in \mathbb{Z}^+$:

$$z_{m_1+m_2} \geq z_{m_1} \cdot z_{m_2}. \tag{26}$$

By Fekete's sub-additivity lemma [4] we then get:

$$\lim_{m\to\infty} \frac{\log_2 r_m}{m} \overset{(25)}{=} \lim_{m\to\infty} \frac{\log_2 z_m}{m} \overset{(26)}{=} \sup_m \frac{\log_2 z_m}{m}. \tag{27}$$

Denote by $\mathfrak{R}$ the limit (27) (this limit is the growth rate of both $(z_m)_m$ and $(r_m)_m$). Lower bounds, z_m^*, on the first few values of z_m, which were found by computer search, are shown in Table II. From (27) we then get:

$$\mathfrak{R} \geq \frac{\log_2 z_{11}^*}{11} = \frac{\log_2 115}{11} > 0.622 > \log_2(3/2), \tag{28}$$

i.e., $\mathfrak{R}$ is strictly larger than the growth rate of both $(\rho_m^+)_m$ and $(r_m^+)_m$ in (23).

Relating now the sequence $(z_m)_m$ to $(q_m)_m$, similarly to (24) we have

$$q_m \geq z_m. \tag{29}$$

The next proposition states that the growth rate of $(q_m)_m$ is in fact the same as that of $(z_m)_m$. The proof can be found in the Appendix.

Proposition 3.

$$\lim_{m\to\infty} \frac{\log_2 q_m}{m} = \mathfrak{R}.$$

We also state the following upper bound on q_m.

Lemma 4. *For every $m \in \mathbb{Z}^+$,*

$$q_m \leq 3^{m/2}.$$

Proof. Write $q = q_m$ and let $\boldsymbol{\rho} = (\rho_v)_{v\in[m\rangle}$ be a bi-spanner of $\mathbb{Z}_q^2$ over $\mathbb{Z}_q$. Every element $(\xi^{(0)}, \xi^{(1)})$ in $\mathbb{Z}_q^2$ corresponds to at least one partition $(\mathcal{I}^{(0)}, \mathcal{I}^{(1)}, \mathcal{I}^{(*)})$ of $[m\rangle$, where $\mathcal{I}^{(0)}$ and $\mathcal{I}^{(1)}$ satisfy (16) and $\mathcal{I}^{(*)} = [m\rangle \setminus (\mathcal{I}^{(0)} \cup \mathcal{I}^{(1)})$. On the other hand, no two distinct elements in $\mathbb{Z}_q^2$ correspond to the same partition. Hence, $q^2 = |\mathbb{Z}_q^2| \leq 3^m$, where the upper bound is the number of partitions of $[m\rangle$ into three subsets. □

From (28), Proposition 3, and Lemma 4 we obtain the following lower and upper bounds on the limit $\mathfrak{R}$:

$$\log_2(3/2) < 0.622 \leq \mathfrak{R} \leq (\log_2 3)/2 \approx 0.792.$$

The exact value of $\mathfrak{R}$ is yet to be found. We mention that for all $m \in [1:9]$ except $m = 8$, the value of z_m^* in Table II is the nearest integer to the power φ^{m-1} of the golden ratio $\varphi = (1+\sqrt{5})/2$ [19, Seq. A169985], but this is true also when φ is replaced by, say, $\sqrt[6]{18}$ [19, Seq. A018028].

V. Single L_1-error correction in Σ-CAMs and Σ-TCAMs

In this section, we consider the case $\tau = 1$ (where there is at most one L_1-error in the whole array). This case corresponds to the parameters $(\tau_{\mathrm{c}}, \tau_{\mathrm{i}}) = (1, 1)$ in Section II. The construction therein for Σ-CAMs yields a redundancy of approximately $1.26 \log_2 n$ (see Example 1), and the adaptation in Section III to Σ-TCAMs results in an increase of that redundancy by a factor of $3/2$ to approximately $1.89 \log_2 n$ (see Example 3).

Here, instead, we build upon [23, §III] and present a construction for Σ-CAMs with a redundancy of $m = \lceil \log_2(2n+1) \rceil$. A respective construction for Σ-TCAMs will have redundancy which is the smallest m such that $q_m \geq 2n+1$; in particular, for sufficiently large n, this redundancy will be less than $1.607 \cdot \log_2(2n+1)$ (see Proposition 3 and Eq. (28)). We start by recalling the construction in [23, §III].

Given a code length n, let $m = \lceil \log_2(2n+1) \rceil$, which will be the redundancy of the construction so that $k = n - m$. Let

$$\boldsymbol{\alpha} = (\alpha_j)_{j\in[n\rangle} = (\boldsymbol{\alpha}' \mid \boldsymbol{\alpha}'')$$

be a vector of code locators in $\mathbb{Z}^n$ (where $\boldsymbol{\alpha}' = (\boldsymbol{\alpha})_{[k\rangle}$ and $\boldsymbol{\alpha}'' = (\boldsymbol{\alpha})_{[k:n\rangle}$) that satisfies the following conditions.[10]

i) The entries of $\boldsymbol{\alpha}$ are nonzero elements in $[2n+1\rangle$.

ii) For any two distinct indexes $i, j \in [n\rangle$ (except when both are in $[k:n\rangle$):

$$\alpha_i \neq \alpha_j \quad \text{and} \quad \alpha_i + \alpha_j \neq 2n+1.$$

iii) $\boldsymbol{\alpha}'' = \boldsymbol{\omega}_m = \begin{pmatrix} 1 & 2 & 2^2 & \dots & 2^{m-1} \end{pmatrix}$.

[10] The conditions herein are slightly weaker than those in [23, §III] in that pairs of code locators that are indexed by $[k:n\rangle$ do not have to satisfy condition (ii) (see Footnote 7 in [23, §III]).

Such vectors $\boldsymbol{\alpha}$ can be constructed for every $n > 4$. In particular, we can select the code locators so that the entries of $\boldsymbol{\alpha}' = (\alpha_j)_{j\in[k\rangle}$ form the set

$$\{\alpha_j\}_{j\in[k\rangle} = \{j+1\}_{j\in[n\rangle} \setminus \big(\{2^j\}_{j\in[m-1\rangle} \cup \{2n{+}1{-}2^{m-1}\}\big) \tag{30}$$

and $\boldsymbol{\alpha}'' = (\alpha_j)_{j\in[k:n\rangle} = \boldsymbol{\omega}_m$ (note that $2^{m-1} > n$, which, by conditions (ii) and (iii), requires $2n{+}1{-}2^{m-1}$ to be excluded from $\boldsymbol{\alpha}'$).[11]

The single-L_1-error-correction (separable) encoder, $\mathcal{E}_1 : \mathbb{B}^k \to \mathbb{B}^n$, maps $\boldsymbol{a}' \in \mathbb{B}^k$ to $\boldsymbol{a} = (\boldsymbol{a}' \mid \boldsymbol{a}'') \in \mathbb{B}^n$ such that

$$-\boldsymbol{a}' \cdot (\boldsymbol{\alpha}')^\top \bmod (2n+1) = \boldsymbol{a}'' \cdot \boldsymbol{\omega}_m^\top \tag{31}$$

(i.e., $\boldsymbol{a}''$ is the base-2 representation of the left-hand side of (31)). Thus, the code that is induced by $\mathcal{E}_1$ is a subset of the following module over $\mathbb{Z}$:

$$\mathcal{M}_1 = \big\{\boldsymbol{c} \in \mathbb{Z}^n \ : \ \boldsymbol{c} \cdot \boldsymbol{\alpha}^\top \bmod (2n+1) = 0\big\}.$$

It follows from [23, §III] that under conditions (i)–(iii), the modified minimum L_1-distance $\mathsf{d}(\mathcal{M}_1)$ is at least 3; i.e., one can always correct any one change of ± 1 occurring in the first k coordinates in any vector in this module.

To adapt this scheme to Σ-CAMs, it suffices to require that $\mathbf{1}_n \in \mathcal{M}_1$. We show that when $n > 9$, this can always be easily achieved by properly selecting the code locators.

We start off by taking $\boldsymbol{\alpha} = (\boldsymbol{\alpha}' \mid \boldsymbol{\alpha}'')$, where the entries of $\boldsymbol{\alpha}'$ are as in (30) sorted in increasing order, and $\boldsymbol{\alpha}'' = \boldsymbol{\omega}_m$; note that $\alpha_{k-1} = n$, except when $n \in \{2^{m-2}, 2^{m-1}-1\}$, in which case $\alpha_{k-1} = n-1$. Denote by μ_1 the first moment (modulo $2n+1$) of the code locators:

$$\mu_1 = \mathbf{1}_n \cdot \boldsymbol{\alpha}^\top \bmod (2n+1). \tag{32}$$

If $\mu_1 = 0$ we are done (yet this rarely happens). Otherwise, we have the following two lemmas.

Lemma 5. *By possibly negating α_{k-1} (namely, replacing it with $2n{+}1{-}\alpha_{k-1}$), the value of μ_1 can be assumed to satisfy*

$$\eta = (\mu_1/2) \bmod (2n+1) \in \big[\lceil n/2\rceil : \lceil 3n/2\rceil{+}1\big]. \tag{33}$$

Proof. The negation changes μ_1 into $(\mu_1{-}2\alpha_{k-1}) \bmod (2n{+}1)$ and, respectively, η becomes $(\eta - \alpha_{k-1}) \bmod (2n+1)$. Recalling that $\alpha_{k-1} \in \{n, n-1\}$, if $0 \le \eta \le \lceil n/2\rceil - 1$ then

$$n+1 \le (\eta - \alpha_{k-1}) \bmod (2n+1) \le \lceil 3n/2\rceil + 1,$$

and if $\lceil 3n/2\rceil + 2 \le \eta \le 2n$ then

$$\lceil n/2\rceil + 2 \le (\eta - \alpha_{k-1}) \bmod (2n+1) \le n+1.$$

□

[11] When $n = 2^{m-2}$ (and only then), we have $2n{+}1{-}2^{m-1} = 1$, i.e., the set on the right-hand side of (30) is of size $n{-}m{+}1 = k{+}1$; hence, (any) one of its elements should be removed. We will hereafter assume that the removed element is not $n{-}1$ (so that $n{-}1$ remains the largest code locator in $\boldsymbol{\alpha}'$). On the other hand, there will be two code locators in $\boldsymbol{\alpha}''$, namely, $\alpha_k = 1$ and $\alpha_{n-1} = 2^{m-1}$, whose sum is zero modulo $2n{+}1$ (condition (ii) allows this). As pointed out in Footnote 7 in [23], when $n = 2^{m-2}$, the dimension k is the same as when the code length is $n{-}1$, namely, the same error-correction capability can be achieved with one less redundancy.

Lemma 6. *Assuming that μ_1 satisfies (33), there exist at least $\lceil n/2\rceil - 2m - 2 = \Omega(k)$ ordered pairs (α_i, α_j) of distinct entries of $\boldsymbol{\alpha}'$ such that $\alpha_i + \alpha_j = \eta$.*

Proof. We distinguish between two ranges of the value of $\eta = (\mu_1/2) \bmod (2n+1)$.

Case 1: $\lceil n/2\rceil \le \eta \le n$. Let $\mathcal{A}$ be the intersection $\{\alpha_j\}_{j\in[k\rangle} \cap [1:\eta\rangle$; note from (30) that

$$|\mathcal{A}| \ge \big|[1:\eta\rangle\big| - m = \eta - 1 - m. \tag{34}$$

Consider the mapping $f : \mathcal{A} \to \mathbb{Z}$ which is given by $f(x) = \eta - x$. This mapping is injective and its set of images, $f(\mathcal{A})$, is contained in $[1:\eta\rangle$; so,

$$\begin{aligned} |f(\mathcal{A}) \cap \mathcal{A}| &= |f(\mathcal{A})| + |\mathcal{A}| - |f(\mathcal{A}) \cup \mathcal{A}| \\ &\ge 2|\mathcal{A}| - \big|[1:\eta\rangle\big| \\ &\overset{(34)}{\ge} \eta - 1 - 2m \\ &\ge \lceil n/2\rceil - 2m - 1. \end{aligned}$$

Moreover f has at most one fixed point; hence, there are at least $\lceil n/2\rceil - 2m - 2$ pairs (α_i, α_j) of distinct entries of $\boldsymbol{\alpha}'$ such that $\alpha_j = f(\alpha_i) = \eta - \alpha_i$.

Case 2: $n{+}1 \le \eta \le \lceil 3n/2\rceil{+}1$. We use the same definition for f except that the set $\mathcal{A}$ is taken to be the intersection $\{\alpha_j\}_{j\in[k\rangle} \cap [\eta{-}n : n]$, which is of size

$$|\mathcal{A}| \ge \big|[\eta{-}n : n]\big| - m = 2n - \eta + 1 - m. \tag{35}$$

Here, $f(\mathcal{A}) \subseteq [\eta{-}n : n]$ and, so,

$$\begin{aligned} |f(\mathcal{A}) \cap \mathcal{A}| &\ge 2|\mathcal{A}| - \big|[\eta{-}n : n]\big| \\ &\overset{(35)}{\ge} 2n - \eta + 1 - 2m \\ &\ge \lfloor n/2\rfloor - 2m \\ &\ge \lceil n/2\rceil - 2m - 1. \end{aligned}$$

The proof now continues as in Case 1. □

The expression $\lceil n/2\rceil - 2m - 2$ in Lemma 6 is positive when $n > 32$. In addition, we have verified numerically that for any smaller $n > 9$, there exist distinct entries α_i and α_j of $\boldsymbol{\alpha}'$ such that $\alpha_i + \alpha_j = \eta$.[12] Letting (α_i, α_j) be such a pair, from $\alpha_i + \alpha_j = \eta = (\mu_1/2) \bmod (2n+1)$ we get that the first moment μ_1 will vanish when both α_i and α_j are negated.

By our analysis at the end of Section II it follows that a redundancy of $n - k = m = \lceil \log_2(2n+1)\rceil$ is within one bit from the smallest possible for any single-L_1-error-correcting coding scheme (see also the discussion at the end of Section III-A in [23]).

Taking n in our construction to be odd and adding an extra parity bit will increase the modified minimum L_1-distance from 3 to 4. Moreover, since $\mathbf{1}_n$ has odd parity, the extended module will contain the vector $\mathbf{1}_{n+1}$. Hence, with an added parity bit, we can also detect two L_1-errors (corresponding to $(\tau, \sigma) = (1, 1)$).

[12] When $n = 19$, there are such entries if we negate α_{k-1}, even though the condition (33) is not met in this case.

Example 5. We consider the case $n = 11$; here $2n+1 = 23$, $m = \lceil \log_2 23 \rceil = 5$, and $k = n - m = 6$. We select our initial vector $\boldsymbol{\alpha}$ based on (30):

$$\boldsymbol{\alpha} = (\boldsymbol{\alpha}' \mid \boldsymbol{\alpha}'') = \big(\underbrace{3 \;\; 5 \;\; 6 \;\; 9 \;\; 10 \;\; 11}_{\boldsymbol{\alpha}'} \mid \underbrace{1 \;\; 2 \;\; 4 \;\; 8 \;\; 16}_{\boldsymbol{\alpha}'' = \boldsymbol{\omega}_5} \big).$$

For this vector we have

$$\mu_1 \overset{(32)}{=} \mathbf{1}_{11} \cdot \boldsymbol{\alpha}^\top \bmod 23 = 6$$

and

$$\eta = (\mu_1/2) \bmod 23 = 3 \notin [6:18],$$

namely, condition (33) is not met. Moreover, no two entries in $\boldsymbol{\alpha}'$ sum to η. We follow our recipe and change $\alpha_5 = 11$ into $23 - 11 = 12$. The value of μ_1 becomes 7, the value of η becomes 15 ($\in [6:18]$), and there are now three (unordered) pairs of entries $\{\alpha_i, \alpha_j\}$ that sum to η, namely,

$$\{\alpha_0, \alpha_5\} = \{3, 12\},\ \{\alpha_1, \alpha_4\} = \{5, 10\},\ \{\alpha_2, \alpha_3\} = \{6, 9\}. \tag{36}$$

Finally, we negate the entries of one of these pairs—say 5 and 10—in $\boldsymbol{\alpha}'$, resulting in the vector

$$\big(3 \;\; 18 \;\; 6 \;\; 9 \;\; 13 \;\; 12 \mid 1 \;\; 2 \;\; 4 \;\; 8 \;\; 16 \big), \tag{37}$$

and, respectively, μ_1 becomes zero. □

The modification of the construction of [23, §III] for Σ-TCAMs is similar in spirit to what we have done in Section II: we change condition (iii) to read:

iii)* $\boldsymbol{\alpha}'' = \boldsymbol{\rho}$, where $\boldsymbol{\rho}$ is a bi-spanner of $\mathbb{Z}_{2n+1}^2$ over $\mathbb{Z}_{2n+1}$.

Accordingly, m should be selected so that $q_m \geq 2n+1$ to guarantee such a bi-spanner.[13] In order to detect two L_1-errors, we need two extra bits to record the two possible values of the parity bit: we extend $\boldsymbol{a}^{(0)}$ by its parity followed by 0, and extend $\boldsymbol{a}^{(1)}$ by 0 followed by its parity. Doing this, both $\boldsymbol{a}^{(0)}$ and $\boldsymbol{a}^{(1)}$ become vectors in modules whose modified minimum L_1-distance is (at least) 4.

Example 6. Suppose that a single-L_1-error-correcting coding scheme is sought for a Σ-CAM with $k = 100$. We select the redundancy m to be the smallest so that

$$2^m \geq 2n + 1 = 2(k+m) + 1 = 201 + 2m.$$

This results in $m = 8$.

For Σ-TCAMs, we will select m to be the smallest so that

$$q_m \geq 2n + 1 = 2(k+m) + 1 = 201 + 2m.$$

Our Table II stops at smaller values of q_m, but we can use the general recurrence (22) for $(\rho_m^+)_m$ to obtain a feasible m and an explicit expression for $\boldsymbol{\rho}$. Specifically, for $m = 12, 13, 14$ we get $\rho_m^+ = 105, 158, 237$, respectively (see [19, Seqs. A005428 and A073941]), and, so, we can take $m = 14$ since

$$q_{14} \overset{(24)}{\geq} \rho_{14}^+ = 237 \geq 229 = 201 + 2 \cdot 14.$$

[13] Note that $\boldsymbol{\rho}$ may contain repeated elements or elements that sum to $2n+1$, which is indeed allowed by our formulation of condition (ii).

Alternatively, we can use the super-multiplicativity (26): we see from Table II that

$$q_{13} \overset{(29)}{\geq} z_{13} \geq z_2 \cdot z_{11} \geq 2 \cdot 115 = 230 \geq 227 = 201 + 2 \cdot 13,$$

which means that we can take $m = 13$. □

VI. Double L_1-error correction

In this section, we consider the case $\tau = 2$ (i.e., at most two L_1-errors in the whole array); part of the discussion herein will be useful also in Section VII, where we consider the correction of an arbitrary number of L_1-errors. Note that the case $\tau = 2$ subsumes the scenario $(\tau_{\mathrm{c}}, \tau_{\mathrm{i}}) = (2, 1)$ in the construction of Section II (now allowing two errors to occur also in the same column); as pointed out in Example 1, that construction for Σ-CAMs has a redundancy of approximately $2.52 \log_2 n$, whereas the construction herein, which is based on [23, §IV-B], will have a redundancy of $2 \log_2 n + O(1)$.

We briefly recall the construction from [23]. Let $p > 3$ be a prime and define $n_1 = (p-1)/2$, $m = \lceil \log_2 p \rceil$, $k = n_1 - m$, $n_2 = n_1 + m$, and $n = n_2 + 1$. The (separable) coding scheme has dimension k, length n and, therefore, redundancy $n - k = 2m + 1$. For the construction we use a vector of code locators $\boldsymbol{\alpha} = (\alpha_j)_{j \in [n_1\rangle}$ that satisfies conditions (i)–(iii) in Section V, and we define $\boldsymbol{\alpha}^{[3]} = (\alpha_j^3)_{j \in [n_1\rangle}$. The encoding mapping, $\mathcal{E}_2 : \mathbb{B}^k \to \mathbb{B}^n$, maps a vector $\boldsymbol{a}' \in \mathbb{B}^k$ to a vector $\boldsymbol{a} \in \mathbb{B}^n$ so that the following four conditions hold.

$$(\boldsymbol{a})_{[k\rangle} = \boldsymbol{a}' \tag{38}$$

$$(\boldsymbol{a})_{[n_1\rangle} \cdot \boldsymbol{\alpha}^\top \equiv 0 \pmod p \tag{39}$$

$$(\boldsymbol{a})_{[n_1\rangle} \cdot (\boldsymbol{\alpha}^{[3]})^\top \equiv (\boldsymbol{a})_{[n_1:n_2\rangle} \cdot \boldsymbol{\omega}_m^\top \pmod p \tag{40}$$

$$(\boldsymbol{a})_{[n_1:n\rangle} \cdot \mathbf{1}_{m+1}^\top \equiv 0 \pmod 2. \tag{41}$$

Eq. (39) is achieved by applying to $\boldsymbol{a}'$ the encoder $\mathcal{E}_1$ in (31) (with n therein replaced by $n_1 = (p-1)/2$). Eq. (40) means that $(\boldsymbol{a})_{[n_1:n_2\rangle}$ forms the base-2 representation of $(\boldsymbol{a})_{[n_1\rangle} \cdot (\boldsymbol{\alpha}^{[3]})^\top \bmod p$. And (41) means that the last entry in $\boldsymbol{a}$ is the parity bit of $(\boldsymbol{a})_{[n_1:n_2\rangle}$. The induced code of $\mathcal{E}_2$ is a subset of the module, $\mathcal{M}_2$, over $\mathbb{Z}$ which consists of all vectors $\boldsymbol{c} \in \mathbb{Z}^n$ that satisfy conditions (39)–(41) (when formulated with $\boldsymbol{c}$ instead of $\boldsymbol{a}$). It is shown in [23, §IV-B] that $\mathsf{d}(\mathcal{M}_2) \geq 5$, i.e., one can correct any pattern of up to two changes of ± 1 occurring within the first k coordinates (possibly in the same coordinate) in any vector in $\mathcal{M}_2$.

As was our strategy in Section V, we adapt this coding scheme to Σ-CAMs by guaranteeing that $\mathbf{1}_n$ belongs to (a modified version of) $\mathcal{M}_2$. Repeating what we have done in that section, by Lemma 6 we can assume that $\boldsymbol{\alpha}$ is such that there exist $\Omega(k)$ pairs (α_i, α_j) of distinct entries of $\boldsymbol{\alpha}'$ such that $\alpha_i + \alpha_j = (\mu_1/2) \bmod p$, where μ_1 is the first moment of the code locators, as defined in (32). Denoting the third moment of the code locators by

$$\mu_3 = \mathbf{1}_{n_1} \cdot (\boldsymbol{\alpha}^{[3]})^\top \bmod p,$$

we will need it to be *nonzero* modulo p (along with a zero first moment). This will be easy to achieve, based on the following lemma.

Lemma 7. *Let (α_i, α_j) and $(\alpha_{i'}, \alpha_{j'})$ be two of the $\Omega(k)$ pairs in Lemma 6, which satisfy*[14]

$$\alpha_i + \alpha_j \equiv \alpha_{i'} + \alpha_{j'} \equiv \mu_1/2 \pmod{p}. \tag{42}$$

Then

$$\alpha_i^3 + \alpha_j^3 \not\equiv \alpha_{i'}^3 + \alpha_{j'}^3 \pmod{p}.$$

Proof. Suppose to the contrary that (42) held simultaneously with

$$\alpha_i^3 + \alpha_j^3 \equiv \alpha_{i'}^3 + \alpha_{j'}^3 \pmod{p}.$$

This would mean that there are two distinct vectors in $\mathbb{Z}^{n_1}$ with L_1-norm 2 that are in the same coset of the module $\mathcal{M}_2$: one that has 1s at positions i and j (and is 0 otherwise) and the other that has 1s at positions i' and j'. This, however, would imply that $\mathsf{d}(\mathcal{M}_2) \leq 4$, which is a contradiction. □

It follows from Lemma 7 that for at least one of the pairs therein, say (α_i, α_j), the value of μ_1 will become zero once we negate α_i and α_j, while μ_3 will be nonzero.

Example 7. For $p = 23$ we have $n_1 = 11$, coinciding with the parameters of Example 5. Recall that in the final step therein, we elected to negate the second among the three pairs in (36) to obtain the vector (37), for which $\mu_1 = 0$. It turns out that for this vector, we also get $\mu_3 = 0$. Lemma 7 guarantees that μ_3 must be nonzero when electing any of the other two pairs; indeed, for these pairs we will get $\mu_3 = 5, 15$. □

At this point, we assume that the first and third moments of the code locators satisfy

$$\begin{array}{rclcrcl} \mu_1 & = & \mathbf{1}_{n_1} \cdot \boldsymbol{\alpha}^\top & \text{MOD } p & = & 0, \\ \mu_3 & = & \mathbf{1}_{n_1} \cdot (\boldsymbol{\alpha}^{[3]})^\top & \text{MOD } p & \neq & 0. \end{array} \tag{43}$$

We next define a module $\widehat{\mathcal{M}}_2$ that is obtained by modifying (40) in two ways. First, we replace the vector $\boldsymbol{\omega}_m$ in the right-hand side of (40) by the vector $\widehat{\boldsymbol{\omega}}_m$ which is obtained by changing the last entry of $\boldsymbol{\omega}_m$ into $2^{m-1} - 1$ (as we did in (10)). Secondly, we multiply the right-hand side of (40) by the following constant

$$\frac{\mu_3}{2^m - 2} \text{ MOD } p,$$

which is (nonzero and) well defined, since $p > 2^{m-1}$ and therefore p does not divide $2^m - 2$. Thus, (40) becomes

$$(2^m - 2) \cdot (\boldsymbol{a})_{[n_1\rangle} \cdot (\boldsymbol{\alpha}^{[3]})^\top \equiv \mu_3 \cdot (\boldsymbol{a})_{[n_1:n_2\rangle} \cdot \widehat{\boldsymbol{\omega}}_m^\top \pmod{p} \tag{44}$$

(compare with (10)). From the analysis in [23, §IV-B] it follows that the module $\widehat{\mathcal{M}}_2$, which is defined by the four conditions (38), (39), (41), and (44), is (still) 2-L_1-error-correcting. Moreover, when m is odd, these four conditions are satisfied by $\boldsymbol{a} = \mathbf{1}_n$, thereby implying that $\mathbf{1}_n \in \widehat{\mathcal{M}}_2$.

[14]It is assumed that the $\Omega(k)$-term in that lemma is greater than 2, which turns out to be the case when $n_1 > 36$. We have verified numerically that for any smaller $n_1 > 9$, the recipe that was used in Section V works here is well: we can always negate two code locators in $\boldsymbol{\alpha}'$ to get both $\mu_1 = 0$ and $\mu_3 \neq 0$.

(When m is even, we can further modify the construction to have one more redundancy bit (i.e., set $n = n_2 + 2$) and replace (41) by, say,

$$\left\{ \begin{array}{rcl} a_{n_2} & = & a_{n_1} \\ (\boldsymbol{a})_{[n_1:n\rangle} \cdot \mathbf{1}_{m+2}^\top & \equiv & 0 \pmod 2 \end{array} \right. ,$$

so that $\mathbf{1}_n \in \widehat{\mathcal{M}}_2$.)

In the case of Σ-TCAMs, our modification to the scheme of [23, §IV-B] is similar to what we did in Sections III and V. Specifically, we select m to be the smallest so that $q_m \geq 2n + 1$, we set $n_1 = (p-1)/2$, $k = n_1 - m$, $n_2 = n_1 + m$, and $n = n_2 + 2$, and we take $\boldsymbol{\alpha}$ so that $(\boldsymbol{\alpha})_{[k:n_1\rangle}$ is a bi-spanner, $\boldsymbol{\rho}$, of $\mathbb{Z}_p^2$ over $\mathbb{Z}_p$ (see condition (iii)* in Section V). Eqs. (38)–(41) now become, for each $x \in \mathbb{B}$:

$$\begin{array}{rcll} (\boldsymbol{a}^{(x)})_{[k\rangle} & = & (\boldsymbol{a}^{(x)})' & \\ (\boldsymbol{a}^{(x)})_{[n_1\rangle} \cdot \boldsymbol{\alpha}^\top & \equiv & 0 & \pmod p \\ (\boldsymbol{a}^{(x)})_{[n_1\rangle} \cdot (\boldsymbol{\alpha}^{[3]})^\top & \equiv & (\boldsymbol{a}^{(x)})_{[n_1:n_2\rangle} \cdot \boldsymbol{\rho}^\top & \pmod p \\ (\boldsymbol{a}^{(x)})_{[n_1:n\rangle} \cdot \mathbf{1}_{m+2}^\top & \equiv & 0 & \pmod 2, \end{array}$$

where $\boldsymbol{a}^{(0)}$ and $\boldsymbol{a}^{(1)}$ are non-intersecting (in particular, note that two bits are allocated for the parity in the last equation).

VII. Multiple L_1-error correction

In this section, we consider the correction of an arbitrary number τ of L_1-errors. We first recall (a variant of) the respective construction in [23, §IV–C].

Given the designed number of correctable L_1-errors τ, let $p > 2\tau$ be a prime and define $n_1 = (p-1)/2$ and $m = \lceil \log_2 p \rceil$. Also, let $\boldsymbol{\alpha}$ be an integer vector in $\mathbb{Z}^{n_1}$ that satisfies conditions (i)–(iii) in Section V. For $i \in \mathbb{Z}^+$, we denote by $\boldsymbol{\alpha}^{[i]}$ the integer vector $(\alpha_j^i)_{j \in [n_1\rangle}$, and we let $H_{\text{Ber}} = H_{\text{Ber}}(\boldsymbol{\alpha}, \tau)$ denote the $\tau \times n_1$ integer matrix whose rows are $\boldsymbol{\alpha}^{[2i+1]}$, $i \in [\tau\rangle$; namely, H_{Ber}, when seen as a matrix over $\mathbb{Z}_p$, is a parity-check matrix of a Berlekamp code C_{Ber} over $\mathbb{Z}_p$ (see [1, Ch. 9] and [22, §10.6]). Let P be any $\tau \times \tau$ integer matrix such that $\det(P) \not\equiv 0 \pmod p$ and define the following module over $\mathbb{Z}$:

$$\begin{array}{rcl} \mathcal{M}_{\text{Ber}} & = & \{\boldsymbol{c} \in \mathbb{Z}^{n_1} \ : \ \boldsymbol{c} \text{ MOD } p \in \mathsf{C}_{\text{Ber}}\} \\ & = & \left\{\boldsymbol{c} \in \mathbb{Z}^{n_1} \ : \ \boldsymbol{c} H_{\text{Ber}}^\top P \text{ MOD } p = \mathbf{0}\right\}. \end{array}$$

When $p > 2\tau$, the *minimum Lee distance* of C_{Ber} is known to be at least $2\tau + 1$ and, so,[15]

$$\mathsf{d}(\mathcal{M}_{\text{Ber}}) \geq 2\tau + 1. \tag{45}$$

Clearly, that holds also for any coset of $\mathcal{M}_{\text{Ber}}$ within $\mathbb{Z}^{n_1}$.

Let $\mathcal{E}_1 : \mathbb{B}^k \to \mathbb{B}^{n_1}$ be the encoder in Section V (with n therein replaced by n_1) and let $\boldsymbol{a} = (\boldsymbol{a}' \mid \boldsymbol{a}'') \in \text{Im}(\mathcal{E}_1)$ (where $\boldsymbol{a}' = (\boldsymbol{a})_{[k\rangle}$). We can compute the following syndrome vector $\tilde{\boldsymbol{s}} \in [p\rangle^\tau$ of $\boldsymbol{a}$:

$$\tilde{\boldsymbol{s}} = (\tilde{s}_v)_{v \in [\tau\rangle} = \boldsymbol{a} H_{\text{Ber}}^\top P \text{ MOD } p.$$

If, in addition, we select P so that its first column is the standard unit vector $(1\ 0\ 0\ \ldots\ 0)^\top$, we get (from $\boldsymbol{a} \in \text{Im}(\mathcal{E}_1)$)

[15]The inequality $p > 2\tau$ provides the excuse to ignore the distinction between Lee weights over $\mathbb{Z}_p$ and L_1-norms over $\mathbb{Z}$.

that $\tilde{s}_0 = 0$. We expand each entry $\tilde{s}_v$ in $\tilde{\boldsymbol{s}}$ to its base-2 representation $\boldsymbol{s}_v \in \mathbb{B}^m$:

$$\tilde{s}_v = \boldsymbol{s}_v \cdot \boldsymbol{\omega}_m^\top. \tag{46}$$

Consider now an encoding mapping $\mathcal{E} : \mathbb{B}^{n_1} \to \mathbb{B}^{n_1+(\tau-1)m}$ defined by

$$\mathcal{E} : \boldsymbol{a} \mapsto (\boldsymbol{a} \mid \boldsymbol{s}), \tag{47}$$

where

$$\boldsymbol{s} = (\boldsymbol{s}_1 \mid \boldsymbol{s}_2 \mid \cdots \mid \boldsymbol{s}_{\tau-1}) \in \mathbb{B}^{(\tau-1)m}.$$

If $\boldsymbol{y} = \mathcal{E}(\boldsymbol{a}) + \boldsymbol{e}$ where $\|\boldsymbol{e}\|_1 \leq \tau$, then, from (45) and Proposition 1, we will be able to recover $\boldsymbol{a}'$ from $\boldsymbol{y}$ under the assumption that the $(\tau-1)m$-suffix of $\boldsymbol{y}$ is error-free. This assumption can be guaranteed by applying to $\boldsymbol{s}$ a (second) encoder of a linear τ-L_1-error-correcting code[16] of dimension $(\tau-1)m$. If we just do $(2\tau+1)$-fold repetition in the second step we end up with a total redundancy of

$$\tau\lceil \log_2(2n+1)\rceil + O\left(\tau^2 \log_2(\tau \log_2 n)\right), \tag{48}$$

where n is the ultimate code length. Hence, for n large compared to τ, most of the redundancy is due to the first encoding step. By our discussion at the end of Section II it follows that the redundancy (48) is optimal, up to the $O(\cdot)$ term.

To adapt this scheme to Σ-CAMs, we select $\boldsymbol{\alpha}$ as in Section VI and, as was done therein, we replace the vector $\boldsymbol{\omega}_m$ in (46) by $\widehat{\boldsymbol{\omega}}_m$. Consider the syndrome of the all-one vector in $\mathbb{Z}^{n_1}$:

$$\tilde{\boldsymbol{s}} = \mathbf{1}_{n_1}(H_{\mathrm{Ber}})^\top P \bmod p,$$

where the first column in P is the standard unit vector. Recalling from (43) that

$$\begin{array}{lll} \mathbf{1}_{n_1} \cdot \boldsymbol{\alpha}^\top & \equiv 0 \pmod p, & \text{yet} \\ \mathbf{1}_{n_1} \cdot (\boldsymbol{\alpha}^{[3]})^\top & \not\equiv 0 \pmod p, & \end{array}$$

it follows that $\tilde{s}_0 = 0$, yet $(\tilde{\boldsymbol{s}})_{[1:\tau\rangle} \neq \mathbf{0}$. By properly selecting P, we can in fact have

$$(\tilde{\boldsymbol{s}})_{[1:\tau\rangle} = (2^m - 2) \cdot \mathbf{1}_{\tau-1},$$

in which case we get that the image of $\mathbf{1}_{n_1}$ under $\mathcal{E}$ in (47) is the all-one vector $\mathbf{1}_{n_1+(\tau-1)m}$. This property can be made to hold also for the second encoding step[17] and, in particular, it holds when that step is just repetition. Thus, we have shown that under the choice of the parameters as in Section VI, the all-one vector is guaranteed to be a codeword of the induced code.

Comparing the redundancy (48) with the redundancy of the construction in Section II, if we substitute $\tau_{\mathrm{c}} = \tau$ in (11), we get values that are close to twice those in (48) when $\tau = o\big((\log n)/\log\log n\big)$. On the other hand, there are many cases (even when $\tau_{\mathrm{i}} > 1$) where (11) is smaller than (48).

Finally, for Σ-TCAMs, we replace $\boldsymbol{\omega}_m$ in (46) by a bi-spanner of $\mathbb{Z}_p^2$ over $\mathbb{Z}_p$.

VIII. Ordinary BCAMs and TCAMs

While this work deals with error-correction schemes for Σ-CAMs and Σ-TCAMs, such schemes are needed also in the much more ubiquitous BCAMs and TCAMs (which can be seen as Σ-CAMs/Σ-TCAMs where the *integer sum* of the outputs of the cells along each column is replaced by the complement of their *logical-"or"*). Clearly, by replicating each column $2\tau+1$ times we will be able to recover from any τ errors[18] in the array, yet a with prohibitively large redundancy.

The error-correction problem for BCAMs has been addressed in several papers; see [17],[18],[20], and the survey [6]. The prevailing proposed solution involves hardware modification of the sense amplifiers along each match line: a "match" is redefined to mean that the (integer) sum of the outputs of the cells along a column does not exceed a prescribed threshold t (thus, the modified device is in effect a quantized Σ-CAM; ordinary BCAMs correspond to $t = 0$).[19] In such a device, we can encode the contents of each *column*, and respectively encode also the input vector, using a t-error-correcting binary code. The reading will then be the same as that of an error-free BCAM, provided that the number of errors per column does not exceed t. As for TCAMs, it has been shown in [12] that this approach would require no less redundancy than a simple $(2t+1)$-fold repetition of the contents of each column (resulting in a prohibitive large redundancy).[20] An off-line error detection protocol for TCAMs is described in [2].

The question remains how much better than replication of rows one can do without resorting to changing the functionality of the BCAM/TCAM.

Appendix

Proof of Proposition 2. Starting with the proof of part (a), we show by induction on m that $[q\rangle^2$ has an all-positive bi-spanner of size m, only if $q \leq \rho_m^+$. The induction base $m = 0$ is obvious. Turning to the induction step, suppose to the contrary that for some $q > \rho_m^+$, the set $[q\rangle^2$ has an all-positive bi-spanner $\boldsymbol{\rho}' = (\rho'_v)_{v\in[m\rangle}$, where we assume that the entries of $\boldsymbol{\rho}'$ are non-decreasing. Since we should be able to express $(\xi^{(0)}, \xi^{(1)}) = (q-1, q-1)$ in the form (17), we must have

$$\|\boldsymbol{\rho}'\|_1 \geq 2(q-1).$$

On the other hand, for the bi-spanner $\boldsymbol{\rho}^+ = (\rho_v^+)_{v\in[m\rangle}$ of $\mathcal{X}_{r_m^+}$ (and of $[\rho_m^+\rangle^2$),

$$\|\boldsymbol{\rho}^+\|_1 \overset{\text{(B2)}}{=} r_m^+ \overset{(21)}{<} 2\rho_m^+ \leq 2(q-1).$$

[16]The second encoding step can be recursive. Alternatively, if P is selected to be upper-triangular, it suffices that the second encoder satisfies the following property: it maps a nonzero vector $\boldsymbol{s} = (\boldsymbol{s}_v)_{v\in[1:\tau\rangle}$ in $\mathbb{B}^{(\tau-1)m}$ to a codeword whose L_1-norm is at least $2(\tau-h)$, where h is the smallest index v such that $\boldsymbol{s}_v \neq \mathbf{0}$. This follows from the fact that the first h rows in $P^\top H_{\mathrm{Ber}}$ form a parity-check matrix of a code with (ordinary, as well as modified) minimum L_1-distance $\geq 2h+1$. The special case $\tau = 2$ coincides with the construction in Section VI.

[17]By (43), the second entry of $\mathbf{1}_{n_1}(H_{\mathrm{Ber}})^\top \bmod p$ is nonzero and, so, the matrix P herein can be selected to be upper-triangular (see Footnote 16).

[18]In the context of BCAMs and TCAMs, it is assumed here that the output of a faulty cell is still in $\mathbb{B}$.

[19]A similar modification is also described in [5], yet for the purpose of performing approximate matches; see also [6].

[20]A special case of this approach is used in [24] for detecting a single error per column by duplicating the contents of the TCAM.

The last two equations imply that $\|\boldsymbol{\rho}'\|_1 > \|\boldsymbol{\rho}^+\|_1$, which means that there is an index $s \in [m\rangle$ for which $\rho'_s \geq \rho^+_s + 1$ (since ρ'_0 is necessarily 1, this index must in fact be positive).

Consider now the elements of $[q\rangle^2$ in the subset $\mathcal{Y} = [\rho^+_s{+}1\rangle^2$ of $[q\rangle^2$. The expansion (17) of each such element can only use entries of $\boldsymbol{\rho}'$ that are indexed by $[s\rangle$, namely, $\mathcal{Y}$ has an all-positive bi-spanner of size s over $\mathbb{Z}$; yet this contradicts our induction hypothesis on $s\ (< m)$.

The proof of part (b) builds upon part (a). Suppose to the contrary that $\mathcal{X}_r$, for some $r > r^+_m$, has a bi-spanner $\boldsymbol{\rho}' = (\rho'_v)_{v\in[m\rangle}$ whose entries are positive and non-decreasing. Expressing $(\xi^{(0)}, \xi^{(1)}) = (0, r)$ in the form (17), we must have $\|\boldsymbol{\rho}'\|_1 \geq r > r^+_m$; yet, on the other hand, $\|\boldsymbol{\rho}^+\|_1 = r^+_m$, which means that there is an index $s \in [m\rangle$ for which $\rho'_s \geq \rho^+_s + 1$.

Consider again the set $\mathcal{Y} = [\rho^+_s{+}1\rangle^2$. Recalling that ($1 \leq s < m$ and therefore) $\rho^+_s < \rho^+_m$, we have

$$2\rho^+_s \leq 2(\rho^+_m - 1) \overset{(21)}{\leq} r^+_m < r,$$

namely, $\mathcal{Y} \subset \mathcal{X}_r$. But the expansion (17) of the elements of $\mathcal{Y}$ can only use entries of $\boldsymbol{\rho}'$ that are indexed by $[s\rangle$, namely, $\mathcal{Y}$ has an all-positive bi-spanner of size s. This, however, contradicts part (a). □

Proof of Proposition 3. By (29) we have

$$\varliminf_{m\to\infty} \frac{\log_2 q_m}{m} \geq \lim_{m\to\infty} \frac{\log_2 z_m}{m} = \mathfrak{R}. \tag{49}$$

Conversely, given $m \in \mathbb{Z}^+$, let $q = q_m$ and suppose that $\boldsymbol{\rho}_1$ is a bi-spanner of $\mathbb{Z}_q^2$ of size m over $\mathbb{Z}_q$; we will regard $\boldsymbol{\rho}_1$ as an integer vector in $[q\rangle^m$. Let $\boldsymbol{\rho}_2$ be a smallest bi-spanner of $[m\rangle^2$ over $\mathbb{Z}$ and let ν_m denote its size; note that $\nu_m \leq 2\lceil\log_2 m\rceil$ (where the upper bound corresponds to the naive construction). We claim that

$$\boldsymbol{\rho} = (\boldsymbol{\rho}_1 \mid -q \cdot \boldsymbol{\rho}_2) \tag{50}$$

is a bi-spanner of $[q\rangle^2$ over $\mathbb{Z}$. Indeed, given any $(\xi^{(0)}, \xi^{(1)}) \in [q\rangle^2$, there exist $\boldsymbol{a}^{(0)}, \boldsymbol{a}^{(1)} \in \mathbb{B}^m$ such that

$$\xi^{(x)} = \boldsymbol{a}^{(x)} \cdot \boldsymbol{\rho}_1^\top \bmod q, \quad x \in \mathbb{B},$$

which means that there exist $\beta^{(0)}, \beta^{(1)} \in [m\rangle$ such that

$$\xi^{(x)} = \boldsymbol{a}^{(x)} \cdot \boldsymbol{\rho}_1^\top - q \cdot \beta^{(x)}, \quad x \in \mathbb{B}.$$

Since $\boldsymbol{\rho}_2$ is a bi-spanner of $[m\rangle^2$, there exist $\boldsymbol{b}^{(0)}, \boldsymbol{b}^{(1)} \in \mathbb{B}^{\nu_m}$ such that

$$\beta^{(x)} = \boldsymbol{b}^{(x)} \cdot \boldsymbol{\rho}_2^\top, \quad x \in \mathbb{B}.$$

The last two equations imply that $\boldsymbol{\rho}$ in (50) is a bi-spanner of $[q\rangle^2$ of size $m + \nu_m$ over $\mathbb{Z}$. Hence, $q_m = q \leq z_{m+\nu_m}$ and, so,

$$\varlimsup_{m\to\infty} \frac{\log_2 q_m}{m} = \varlimsup_{m\to\infty} \frac{\log_2 q_m}{m+\nu_m} \leq \lim_{m\to\infty} \frac{\log_2 z_{m+\nu_m}}{m+\nu_m} = \mathfrak{R}. \tag{51}$$

The result follows from (49) and (51). □

References

[1] E.R. Berlekamp, *Algebraic Coding Theory,* Revised Edition, Aegean Park Press, Laguna Hills, California, 1984.

[2] 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.

[3] T.T. Bui, T. Shibata, "A low-power associative processor with the R-th nearest-match Hamming-distance search engine employing time-domain techniques," *Proc. Fifth IEEE Int'l Symp. on Electronic Design, Test and Applications (DELTA 2010),* Ho Chi Minh City, Vietnam (2010), 54–57.

[4] M. Fekete, "Über die Verteilung der Wurzeln bei gewissen algebraischen Gleichungen mit ganzzahligen Koeffizienten," *Math. Z.,* 17 (1923), 228–249.

[5] E. Garzón, R. Golman, Z. Jahshan, R. Hanhan, N. Vinshtok-Melnik, M. Lanuzza, A. Teman, L. Yavits, "Hamming distance tolerant content-addressable memory (HD-CAM) for DNA classification," *IEEE Access,* 10 (2022), 28080–28093.

[6] E. Garzón, L. Yavits, A. Teman, M. Lanuzza, "Approximate content-addressable memories: A review," *Chips,* 2 (2023), 70–82.

[7] R. Govindaraj, S. Ghosh, "Design and analysis of STTRAM-based ternary content addressable memory cell," *ACM J. Emerg. Technol. Comput. Syst.,* 13 (2017), Article No. 52, 1–22.

[8] L. Halbeisen, N. Hungerbühler, "The Josephus problem," *J. de Théor. des Nr. de Bordeaux,* 9 (1997), 303–318.

[9] A.R. Hammons, P.V. Kumar, A.R. Calderbank, N.J.A. Sloane, P. Solé, "The $\mathbb{Z}_4$ linearity of the Kerdock, Preparata, Goethals, and related codes," *IEEE Trans. Inf. Theory,* 40 (1994), 301–319.

[10] P. He, S. Wang, R. Mao, M. Jiang, S. Siegel, G. Pedretti, J. Ignowski, J.P. Strachan, R. Luo, C. Li, "Real-time raw signal genomic analysis using fully integrated memristor hardware," *Nat. Comput. Sci.,* 5 (2025), 940–951.

[11] G. Kasai, Y. Takarabe, K. Furumi, M. Yoneda, "200MH/200MSPS 3.2W at 1.5V Vdd, 9.4Mbits ternary CAM with new charge injection match detect circuits and bank selection scheme," *Proc. IEEE 2003 Custom Integrated Circuits Conf.,* San Jose, CA (2003), 387–380.

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

[13] M. Imani, A. Rahimi, D. Kong, T. Rosing, J. M. Rabaey, "Exploring hyperdimensional associative memory," *Proc. 23rd IEEE Int'l Symp. on High Performance Computer Architecture (HPCA 2017),* Austin, TX (2017), 445–556.

[14] A.F. Laguna, H. Gamaarachchi, X. Yin, M. Niemier, S. Parameswaran, X.S. Hu, "Seed-and-vote based in-memory accelerator for DNA read mapping," *Proc. 2020 IEEE/ACM Int'l Conf. on Computer Aided Design (ICCAD),* virtual event (2020), Article No. 56, 1–9.

[15] R. Mao, B. Wen, A. Kazemi, Y. Zhao, A.F. Laguna, R. Lin, N. Wong, M. Niemier, X.S. Hu, X. Sheng, C.E. Graves, J.P. Strachan, C. Li, "Experimentally validated memristive memory augmented neural network with efficient hashing and similarity search," *Nat. Commun.,* 13 (2022), Article No. 6284, 1–13.

[16] K. Ni, X. Yin, A.F. Laguna, S. Joshi, S. Dünkel, M. Trentzsch, J. Müller, S. Beyer, M. Niemier, X.S. Hu, S. Datta, "Ferroelectric ternary content-addressable memory for one-shot learning," *Nat. Electron.,* 2 (2019), 521–529.

[17] 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.

[18] 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.

[19] OEIS Foundation Inc., *The On-Line Encyclopedia of Integer Sequences,* published electronically at https://oeis.org (2003).

[20] K. Pagiamtzis, N. Azizi, F.N. Najm, "A soft-error tolerant content-addressable memory (CAM) using an error-correcting-match scheme," *Proc. IEEE 2006 Custom Integrated Circuits Conf. (CICC),* San Jose, CA (2006), 301–304

[21] G. Pedretti, F. Böhm, T. Bhattacharya, A. Heittmann, X. Zhang, M. Hizzani, G. Hutchinson, D. Kwon, J. Moon, E. Valiante, I. Rozada, C.E. Graces, J. Ignowski, M. Mohseni, J.P. Strachan, D. Strukov, R. Beausoleil, T. Van Vaerenbergh, "Solving Boolean satisfiability problems with resistive content addressable memories," *npj Unconv. Comput.,* 2, 7 (2025).

[22] R.M. Roth, *Introduction to Coding Theory,* Cambridge University Press, Cambridge, UK, 2006.
[23] R.M. Roth, "Fault-tolerant dot-product engines," *IEEE Trans. Inf. Theory,* 65 (2019), 2046–2057.
[24] 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.