Variability-Aware Design of Multilevel Logic Decoders for Nanoscale Crossbar Memories

Mohamed Haykel Ben Jamaa, *Student Member, IEEE*, Kirsten Emile Moselund, *Student Member, IEEE*, David Atienza, *Member, IEEE*, Didier Bouvet, Adrian Mihai Ionescu, *Senior Member, IEEE*, Yusuf Leblebici, *Senior Member, IEEE*, and Giovanni De Micheli, *Fellow, IEEE*

***Abstract*—The fabrication of crossbar memories with sublithographic features is expected to be feasible within several emerging technologies; in all of them, the nanowire (NW) decoder is a critical part since it bridges the sublithographic wires to the outer circuitry that is defined on the lithography scale. In this paper, we evaluate the addressing scheme of the decoder circuit for NW crossbar arrays, based on the existing technological solutions for threshold voltage differentiation of NW devices. This is equivalent to using a multivalued logic addressing scheme. With this approach, it is possible to reduce the decoder size and keep it defect tolerant. We formally define two types of multivalued codes (i.e., hot and reflexive codes), and we estimate their yield under high variability conditions. Multivalued hot decoders yield better area saving than n-ary reflexive codes, and under severe conditions, reflexive codes enable a nonvanishing part of the code space to randomly recover. The choice of the optimal combination of decoder type and logic level saves area up to 24%. We also show that the precision of the addressing voltages when a high variability affects the threshold voltages is a crucial parameter for the decoder design and permits large savings in memory area. Moreover, a precise knowledge about the variability level improves the design of memory decoders by giving the right optimal code.**

***Index Terms*—Addressing, crossbar architecture, memory, multivalued logic (MVL), nanotechnology, reliability.**

I. Introduction

THE TREND toward deeply scaled evolutionary and emerging technologies, as predicted by the International Technology Roadmap for Semiconductors [1], and the corresponding increase in variability and defect rates of devices and interconnect justifies the reintroduction of regular architectures. Within this design style, sometimes referred to as *array logic*, switching devices [molecular switches and semiconducting nanowires (NWs)] can be fabricated and deposited into highly dense regular arrays, called *crossbar arrays*. A specific example is given by two perpendicular strips of parallel NWs storing the information or executing the computation at the matrix nodes (i.e., at the NW crosspoints) [5].

Crossbar circuits can be integrated onto CMOS chips, and this *hybrid* approach has to provide ways to bridge the two different dimension scales. Designing the interface between the nano- and mesoscale parts is a challenging issue. The circuit performing this operation is called decoder: It allows the addressing of each single wire in the nanoscale part by the mesoscale part. Many approaches for designing decoders are found in literature; they range from a random attribution of addresses [9], [11] to a deterministic mask-based decoder [3]. The consequence of the variability affecting the placement of NWs is an overhead in the decoder size, which can be up to five times [11].

This paper presents a method to reduce the decoder size by implementing it in a multivalued logic (MVL), while its robustness is enhanced by optimizing the encoding and design parameters. MVL is a possible method for saving area. It was presented, for instance, as a solution to the interconnect bottleneck. While interconnects cannot follow the scaling of transistors, MV wires can carry denser information and reduce the area dedicated to global interconnects [27], [28]. MVL can be also implemented in hardware for computation [7], [21], and many MVL minimizers and input/output encoders exist [10], [23], [25] and have proved their superior performance.

The contribution of this paper is the construction of the MVL encoding and the investigation of its properties and impact on the circuit under high variability conditions. Our underlying technology based on gate-all-around silicon NWs (GAA SiNWs) was demonstrated in [20], and it enables the fabrication of devices with multiple threshold voltages (V_T) that represent a possible technological platform for the MVL addressing of NW arrays. The demonstrated technology yields robust NWs with respect to breakage and placement; however, small radii imply a high V_T variation, which, in turn, causes the MVL code to fluctuate. The detection of code errors depends on the encoding and was investigated for the binary case in [2]. We presented this approach in [4] where we restricted our study to the advantages of MVL addressing within this technology. In this paper, we generalize the implementation of MVL to a wide range of other decoder technologies.

This paper is organized in the following way. In the next section, we review the work related to crossbar memories, focus on the decoder design, and survey the applications of MVL. In Section III, we present the decoder model and the underlying technology. Section IV states the formalism and

Manuscript received January 3, 2008; revised May 16, 2008. Current version published October 22, 2008. This work was supported in part by the Swiss NSF Research under Grant 20021-109450/1 and in part by the Spanish Government Research under Grants TIN2005-5619 and CSD00C-07-20811. This paper was recommended by Associate Editor N. Chang.

M. H. Ben Jamaa, K. E. Moselund, D. Bouvet, A. M. Ionescu, Y. Leblebici, and G. De Micheli are with Ecole Polytechnique Fédérale de Lausanne, 1015 Lausanne, Switzerland (e-mail: haykel.benjamaa@epfl.ch).

D. Atienza is with Complutense University of Madrid, 28040 Madrid, Spain, and also with Ecole Polytechnique Fédérale de Lausanne, 1015 Lausanne, Switzerland.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2008.2006076

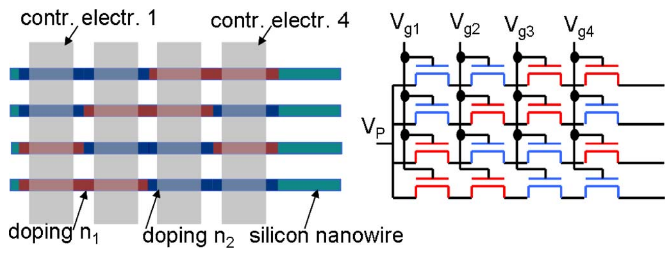

Fig. 1. Decoder layout and circuit.

semantic used for MVL encoding. Then, we construct two types of encoding schemes (Section V). The defects affecting the codes are modeled, and then, their impact on the code space is analyzed in Section VI. In Sections VII and VIII, we investigate multidigit errors. The impact of these errors on the size of the addressable code space is simulated in Section IX. Then, the simulation results for decoders with both encoding schemes and under various reliability and technological assumptions are presented in Section X.

II. Related Work

A. From Technology to Circuits

The fabrication of silicon NWs usually follows one of the two different approaches: bottom–up, where wires are grown, synthesized, or self-assembled onto the silicon substrate [12], and top–down, where NWs are fabricated from the substrate by more traditional means such as lithography, oxidation, and etching [6], [18]. In all these technologies, a controlled connection between the NWs can be implemented with molecular crosspoints, which can switch from a high to low resistance state [16].

Logic gates based on SiNWs and switching crosspoints were demonstrated in [13], and an architectural paradigm for building logic circuits was proposed in [5], which relies upon a full logic out of NOR and OR planes. An architecture with cascaded sequences of OR/NOT planes is also proposed in [8]. Aside from interconnection and logic, SiNW crossbar arrays are highly suitable for storing the information at the crosspoints because of their regular structure and high density. A prototype for this kind of memory was reported in [16].

B. Decoders of Crossbar Circuits

Any crossbar circuit, performing either logic or memory, needs the decoder part. The decoder is shown in a schematic way in Fig. 1, where the aspect ratios are not exact. The current conduction through the horizontal wires (NWs) can be controlled by applying suitable voltages at the vertical wires that are called mesowires (MWs). The cross-areas between NWs and MWs define field effect transistors having two possible V_T's. Depending on the voltage sequence at the MWs, one of the NWs will conduct, while the others will have a high resistance. The vertical wires are called control or addressing wires. Their number depends on the encoding scheme and technology, which, in turn, follows from the fabrication approach.

Differentiated NWs have an axial or a radial doping profile which was defined during the NW growth process. An axial decoder was presented in [9], in which the distribution of the V_T's results from the doping profile and is fully random. Radial decoders [26] rely on a selective etching of NWs with radially doped shells, whose pattern is defined by the shell sequence and the etching order. On the other hand, for undifferentiated NWs, a mask-based decoder reported in [3] uses different oxide types to address the NWs. A random contact decoder was presented in [11]: The pattern of the NWs are defined by the fully random connections between MWs and NWs through randomly deposited impurities.

C. Multivalued Logic

The research areas for MVL can be summarized in three categories: MV algebra, MV semiconductor circuits, and MV network synthesis. A review of the background of MV algebra was presented in [27] and [28]. The implementation of algebraic notions into real circuits was motivated from one side by the exponential growth of interconnects in digital circuits and their limited scaling abilities [28] and, from the other side, by the need for higher density of information storage. The use of the MVL encoding of data reduces the area needed for MVL buses and memories.

Many circuit design techniques were used to implement MVL: Different current-mode MVL circuits were reviewed in [7]; while in [21], a voltage-mode MVL full adder was demonstrated. A charge- and voltage-based approach for MVL Flash memories was described in [15], and MVL SRAM memory and logic blocks fabricated with the same technology were introduced in [17].

Design tools for mapping MVL functions onto field programmable gate array systems were presented in [19]. Optimized design methodologies for MVL Programmable Logic Array (PLA) were introduced in [23] and [24] while considering the MVL encoding problem of inputs and outputs and the benefits of encoding in terms of the number of products in the minimized function. The simplification and minimization of EXOR-sum-of-product expressions for MVL functions were presented in [25] and [29], respectively. The use of MV functional decomposition algorithms based on MV decision diagrams was investigated in [10] for logic synthesis. The efficient minimization of MVL networks with *don't-cares* was presented in [14] as a way to implement MVL hardware and also as an optimization opportunity for binary functions at the MV stage, which cannot be discovered in the binary domain.

III. Circuit Model and Technology

We show the memory architecture considered in this paper in Fig. 2. This architecture has two parts, organized in an identical way and laid out perpendicularly to each other. Each part is a plane of N parallel NWs divided into a set of bundles; the NWs within each group have a common ohmic contact to an MW at either extremity of the bundle. Each group is defined lithographically. We assume that the circuit has a standard binary NW group decoder, which is not investigated in this paper since its design is well established on the lithographic scale. M MWs are used to address the NWs within each group. Then, the NW decoder of each group has the size $N \times M$. The

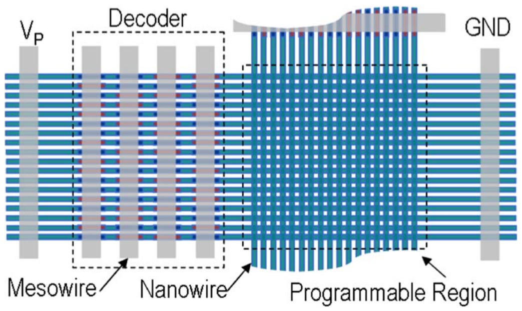

Fig. 2. Crossbar memory architecture.

area sandwiched between the NW arrays is the actual memory, in which the information is stored in bistable switches grafted at the crosspoints.

Each NW within a single group has to be addressed. This operation is performed by the NW decoder which is formed by a set of parallel MWs crossing the NW plane. The part of NWs under the decoder is coated by a dielectric, thus allowing a field-effect control of the NWs by the MWs of the decoder. Any method among those described in Section II-B can be used to fabricate the decoder part.

We fabricated silicon NW arrays to validate the proposed crossbar decoder architectures. The technological details are reported in [20]. The fabricated NWs are parallel and have a length of several tens of micrometers, a cross-sectional circumference ranging from hundreds of nanometers to 5 nm, and a pitch depending on the lithography. The gate wraps the SiNW; thus, the control device represents a gate-all-around field effect transistor (GAA FET). The threshold voltage of the GAA FETs depends on the doping level of the channel. We are experimenting with more than two doping levels, i.e., more than two V_T's, which enables the fabrication of an MVL decoder.

The technology that we demonstrated in [20] is considered as a possible future embodiment of such a decoding scheme, but by no means, the only one: The generalization to other NW technologies will be exposed in this paper. The following sections investigate the types of MVL encodings that can be implemented, their impact on the decoder area and memory density, and the related reliability aspects.

IV. Semantic of MVL Addressing

In this section, we generalize the notion of encoding to multiple-valued bits. Some basic semantic used in encoding can be considered as a generalization of the binary definitions stated in [22] to MVL. In [22], a realistic NW model was presented to reflect their finite resistance. This was called a *real-valued encoding*, because the current through the NWs had real values; however, the applied addresses and the NW patterns were still binary. In this paper, we assume multivalued applied addresses and NW patterns. The issue of the real-valued current through the NWs will be addressed in Section VI, by introducing the parameters q and V_X. In this paper, all algebraic operations are performed as defined in the ring of integers.

Definition 1: A multiple-valued pattern $\mathbf{a}$, or simply a pattern $\mathbf{a}$, is a suite of M digits a_i, in the n-valued base $\mathbb{B}$, i.e., $\mathbf{a} = (a_0, \ldots, a_{M-1}) \in \mathbb{B}^M$, $\mathbb{B} = \{0, \ldots, n-1\}$.

A pattern represents a serial connection of M transistors in the silicon NW core; each digit a_i of the code word represents a threshold voltage $V_{T,i}$, with the convention $a_i < a_j \Leftrightarrow V_{T,i} < V_{T,j}$ $\forall i, j = 0, \ldots, M-1$. An analog equivalence holds for $a_i = a_j$ and, consequently, for $a_i > a_j$. This convention is equivalent to discretizing the n values of V_T and ordering them in an increasing order. Assuming that three V_T's are used (0.2, 0.4, and 0.6 V), the pattern 002120, for instance, represents the V_T sequence (0.2, 0.2, 0.6, 0.4, 0.6, 0.2 V).

Definition 2: A multiple-valued code word $\mathbf{c}$, or simply a code word $\mathbf{c}$, is similarly to a pattern, a suite of M digits c_i, in the n-valued base $\mathbb{B} = \{0, \ldots, n-1\}$, i.e., $\mathbf{c} = (c_0, \ldots, c_{M-1}) \in \mathbb{B}^M$.

A code word represents the suite of applied voltages V_A at the M MWs. These are defined such that every $V_{A,i}$ is slightly higher than $V_{T,i}$ and lower than $V_{T,i+1}$. Hence, a similar convention holds for the order of $V_{A,i}$ with respect to that of c_i: Assuming that three V_A's are used (0.3, 0.5, and 0.7 V), the pattern 202111, for instance, represents the V_A sequence (0.7, 0.3, 0.7, 0.5, 0.5, 0.5 V).

Definition 3: A complement of digit x_i in a code word or pattern $\mathbf{x}$ is defined as follows: $\mathtt{NOT}(x_i) = \overline{x_i} = (n-1) - x_i$. The operator NOT can be generalized to the vector $\mathbf{x}$, acting on each component as defined previously. Notice that $\mathtt{NOT}(\mathtt{NOT}(\mathbf{x})) = \mathbf{x}$.

Definition 4: A pattern $\mathbf{a}$ is covered by a code word $\mathbf{c}$ if and only if the following relation holds: $\forall i = 0, \ldots, M-1$, $c_i \geq a_i$. By using the sigmoid function[1] generalized to vectors $\sigma(\mathbf{x}) = (\sigma(x_0), \ldots, \sigma(x_{M-1}))$, the aforementioned definition becomes $\mathbf{a}$ is covered by $\mathbf{c} \Leftrightarrow \|\sigma(\mathbf{a} - \mathbf{c})\| = 0$. Alternatively, we can use the order relations on vectors $\mathbf{c}$ and $\mathbf{a}$[2]: A pattern $\mathbf{a}$ is covered by a code word $\mathbf{c}$ if and only if $\mathbf{a} \leq \mathbf{c}$. The same definition for covering can be generalized between two patterns or two code words.

Covering a given pattern with a certain code is equivalent to applying a suite of gate voltages making every transistor conductive. Then, the NW is conducting, and we say that it is controlled by the given sequence of gate voltages. In the opposite case, at least one transistor is highly resistive, and the NW is nonconducting.

Definition 5: A pattern $\mathbf{a}$ implies a pattern $\mathbf{b}$ if and only if $\|\sigma(\mathbf{b} - \mathbf{a})\| = 0$, i.e., $\mathbf{b}$ is covered by $\mathbf{a}$. We note this as follows: $\mathbf{a} \Rightarrow \mathbf{b}$. Since a 1-to-1 mapping between the patterns and codes was assumed, we generalize this definition to code words $(\mathbf{c}^a \Rightarrow \mathbf{c}^b) \Leftrightarrow \|\sigma(\mathbf{c}^b - \mathbf{c}^a)\| = 0$, i.e., $\mathbf{c}^b$ is covered by $\mathbf{c}^a$.

This means that, if an NW with the pattern $\mathbf{a}$ corresponding to the code $\mathbf{c}^a$ is covered by a code $\mathbf{c}^*$, then the NW with the pattern $\mathbf{b}$ corresponding to the code $\mathbf{c}^b$ is also covered by the same code $\mathbf{c}^*$. Applying the voltage suite $\mathbf{c}^*$ will result in turning on the NWs with either pattern.

Definition 6: The code words $\mathbf{c}^a$ and $\mathbf{c}^b$ are independently covered if and only if $\mathbf{c}^a$ does not imply $\mathbf{c}^b$ and $\mathbf{c}^b$ does not imply $\mathbf{c}^a$.

[1] $\sigma(x) = 0$ if $x \leq 0$; otherwise, $\sigma(x) = 1$.

[2] $\mathbf{c} < \mathbf{a} \Leftrightarrow \forall i,\ c_i < a_i$, and $\mathbf{c} > \mathbf{a} \Leftrightarrow \forall i,\ c_i > a_i$. The relation becomes relaxed (i.e., $\leq$ or $\geq$) if there exists i such that $c_i = a_i$.

This definition means that there exists a voltage suite that turns on the NW with the pattern $\mathbf{a}$ corresponding to $\mathbf{c}^a$ but not that with the pattern $\mathbf{b}$ corresponding to $\mathbf{c}^b$. Reciprocally, there exists a second voltage pattern that turns on the NW with the pattern $\mathbf{b}$ corresponding to $\mathbf{c}^b$ but not that with the pattern $\mathbf{a}$ corresponding to $\mathbf{c}^a$.

Definition 7: The code word $\mathbf{c}^a$ belonging to the set Ω is addressable if and only if it does not imply any other code in $\Omega \setminus \{\mathbf{c}^a\}$. We define the set Ω to be addressable if and only if every code word in Ω is addressable.

Assuming that there is a 1-to-1 mapping between the code space Ω and the pattern space A, saying that a code $\mathbf{c}^a$ implies no other code in $\Omega \setminus \{\mathbf{c}^a\}$ is equivalent to saying that it covers only the pattern $\mathbf{a}$ but no other pattern in $A \setminus \{\mathbf{a}\}$. Thus, there exists a voltage sequence that activates only the NW with the pattern $\mathbf{a}$ but no other NW having its pattern in $A \setminus \{\mathbf{a}\}$.

Proposition 1: A set Ω of code words is addressable if and only if every code word in Ω is independently covered with respect to any other code word in Ω.

Proof: This follows directly from Definitions 6 and 7. ■

Consequently, an admissible set of applied voltages that uniquely addresses each NW corresponds to the set of code words Ω that independently covers every pattern in A. This set of patterns can be simply taken as Ω itself if Ω is addressable.

V. Code Construction

A. Hot Encoding

In binary logic, the (k, M) hot code space is defined as the set of code words with the length M having k occurrences of the bit "1" and $(M-k)$ occurrences of the bit "0" in every code word $(k \leq M)$. It is also known as the k-out-of-M code, which was first used as a defect tolerant encoding scheme [2]. This definition can be generalized to the n-valued logic. We first define $\mathbf{k}$ as an n-dimensional vector $(k_0, \ldots, k_{n-1})$ such that $\sum_i k_i = M$. Then, the multivalued $(\mathbf{k}, M)$ hot encoding is defined as the set of all code words having the length M such that each k_i represents the occurrence of the digit i, $i = 0, \ldots, n-1$. We consider, for instance, the ternary logic $(n = 3)$, and we set $\mathbf{k} = (4, 3, 1)$ and $M = 8$. Then, every code word in the considered $(\mathbf{k}, M)$ hot space contains four times the digit "0," three times the digit "1," and one time the digit "2." The considered code space includes, for instance, the code words 00001112 and 00210110.

Proposition 2: The code space defined by a multivalued $(\mathbf{k}, M)$ hot encoding is addressable.

Proof: Consider two different code words c^a and c^b in the code space defined by the $(\mathbf{k}, M)$ hot encoding. Both codes are identical except at P different digits lying at the positions $p_0, \ldots, p_{P-1}$. c^b is obtained by a permutation of $\{\mathbf{c}^a_{p_0}, \ldots, \mathbf{c}^a_{p_{P-1}}\}$. Hence, there is at least one position p_i for which $\mathbf{c}^a_{p_i} > \mathbf{c}^b_{p_i}$ holds and at least one position p_j for which $\mathbf{c}^a_{p_j} < \mathbf{c}^b_{p_j}$ holds. This proves that $\|\sigma(\mathbf{c}^b - \mathbf{c}^a)\| \neq 0$ and $\|\sigma(\mathbf{c}^a - \mathbf{c}^b)\| \neq 0$ and that every two code words are independently covered. Then, Proposition 1 states that the whole code space is addressable. ■

Example 1: For instance the code words $\mathbf{a} = 00001112$ and $\mathbf{b} = 00210110$ differ at the third, fourth, fifth, and last digits, in positions $\{2, 3, 4, 7\}$. The third digit of $\mathbf{a}$ is smaller than the third digit of $\mathbf{b}$, which proves that $\mathbf{b} \not\leq \mathbf{a}$, and the fifth digit of $\mathbf{a}$ is bigger than the fifth digit of $\mathbf{b}$, which proves that $\mathbf{a} \not\leq \mathbf{b}$. Thus, both codes are independently covered. ■

Proposition 3: The size of the code space defined by a multivalued $(\mathbf{k}, M)$ hot encoding is maximal for $k_i = M/n$ $\forall i = 0, \ldots, (n-1)$. The size of the maximal-sized space is asymptotically proportional to $\propto n^M / M^{(n-1)/2}$ for a given n.

Proof: The number of code words is given by $M!/(\prod k_i!)$, which can be maximized by using the Gamma function $\Gamma(k_i + 1) = k_i!$. The Stirling formula yields the asymptotic space size for large M's. ■

In the rest of this paper, we implicitly mean the $(\mathbf{k}, M)$ hot code with the maximal-sized space when we simply talk about the $(\mathbf{k}, M)$ hot code.

B. N-ary Reflexive Code

The binary tree code with the length M is a 2-to-2^M encoder representing the 2^M binary numbers $0 \cdots 0$ to $1 \cdots 1$. Similarly, an n-ary tree code with the length M is defined as the set of n^M numbers ranging from $0 \cdots 0$ to $(n-1) \cdots (n-1)$. For instance, the ternary $(n = 3)$ tree code with the length $M = 4$ includes all ternary logic numbers ranging between 0000 and 2222. As one can easily see, some code words imply many others from the same space: For instance, 2222 implies all other codes. It is possible to prevent the inclusive character of the n-ary tree code by attaching the complement of the code word (i.e., 2222 becomes 22220000). The as-constructed code is the N-ary reflexive code (NRC).

Proposition 4: The code space defined by the NRC is addressable.

Proof: The first (nonreflected) halves of any two code words $\mathbf{c}^a$ and $\mathbf{c}^b$ having the total length M (M is even) differ by at least one digit at, for example, position i $[i = 0, \ldots, (M/2 - 1)]$. Let $\mathbf{c}^a$ be the code word such that $\mathbf{c}^a_i < \mathbf{c}^b_i$. The reflection implies that $\mathbf{c}^a_{M/2+i} > \mathbf{c}^b_{M/2+i}$. This proves that $\|\sigma(\mathbf{c}^b - \mathbf{c}^a)\| \neq 0$ and $\|\sigma(\mathbf{c}^a - \mathbf{c}^b)\| \neq 0$ and that $\mathbf{c}^a$ and $\mathbf{c}^b$ are independently covered. Then, Proposition 1 states that the whole code space is addressable. ■

Example 2: We consider, for instance, the ternary $(n = 3)$ reflexive code words with the length $M = 8$: $\mathbf{c}^a = 22220000$ and $\mathbf{c}^b = 00122210$. The codes differ at some positions; we consider, for instance, the first one. The first digit of $\mathbf{c}^a$ ("2") is bigger than the first digit of $\mathbf{c}^b$ ("0"), proving that $\mathbf{c}^a \not\leq \mathbf{c}^b$. The complementary of the first digit is the fifth one (since $M = 8$). Because of the complementarity of the values of the digits, the fifth digit of $\mathbf{c}^a$ ("0") is smaller than the fifth digit of $\mathbf{c}^b$ ("2"), proving that $\mathbf{c}^b \not\leq \mathbf{c}^a$. Consequently, $\mathbf{c}^a$ and $\mathbf{c}^b$ are independently covered. ■

In a similar way, the reflection principle works for any other code (e.g., Hamming code), making the whole code space addressable. However, in return, it doubles the code length. Although the binary hot code is denser than the binary reflexive code, this statement holds for the MVL only if the codes are defect free. This aspect is analyzed in the following sections.

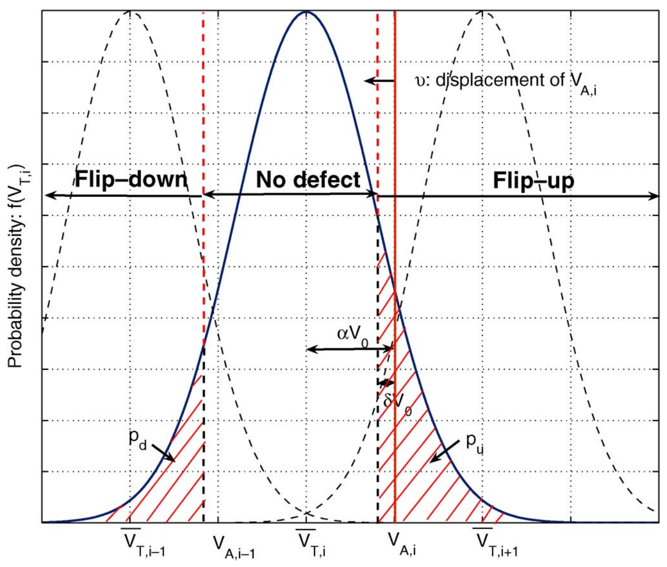

Fig. 3. Coding defects induced by V_T variability.

VI. Defect Models

The control of the silicon NWs fabricated with our technology described in Section III is based on the modulation of the threshold voltage of the controlling transistors. The encoding schemes proposed in Section V impose a distribution of the applied control voltages between the successive threshold voltages. The main issue with the threshold voltage is its variability and process dependence: Several aspects of the insufficient controllability of the technology are expressed on the device level as a random variation of the threshold voltage (e.g., variation of the doping level, oxide thickness, mechanical stress of the NWs, etc). The independent nature of these random effects and their superposition generally justify the assumption of a normal distribution of the threshold voltage (see Fig. 3). The MVL encoding only depends on the value of V_T's; however, other random defects, such as the NW breakage and the NW-to-metal contact quality, impact the array yield (but not the addressable code space size). These random effects were modeled by a statistical factor explained in Section X. Many random errors can affect the molecular switches (bad switching, stack-at-defects, ...); these are beyond the scope of this paper, which focuses on NW defects.

A. Basic Error Model

Fig. 3 shows the main assumptions for basic error models. We assume that the threshold voltages $V_{T,i}$ are equidistant, i.e., $V_{T,i+1} - V_{T,i} = 2\alpha V_0$, with V_0 being a given scaling voltage, and α is given by the technology. The applied voltages $V_{A,i}$ are set between every two successive threshold voltages $V_{T,i}$ and $V_{T,i+1}$, not necessarily in the middle, rather shifted by υV_0 toward $V_{T,i}$, where υ is a design parameter.

If the variability of $V_{T,i}$ is high or the spacing between two successive $V_{T,i}$'s is low due to the large number of doping levels, $V_{T,i}$ may exceed a voltage $V_{X,i}$ given by $V_{A,i} - \delta \cdot V_0$, where δ will be derived in the following. While V_T increases, the sensed current, while a_i is applied on the digit c_i, decreases $(a_i = c_i)$, and the sensed current, while $a_i + 1$ is applied on the same digit, increases. The voltage $V_{X,i}$ is defined as the gate voltage which results in the decrease of the sensed current for a_i by the factor q from its value at $\overline{V}_{T,i}$. The higher q is, the more accurate is the sensing. Thus, q is also considered as a design parameter. Assuming that the transistors are saturated, the current in the saturation region is proportional to $(V_{A,i} - V_T)^2$, where V_T is the actual threshold voltage. Consequently, the following condition on V_X must hold: $(V_{A,i} - \overline{V}_{T,i})^2/(V_{A,i} - V_{X,i})^2 = q$, which gives $\delta = (\alpha + \upsilon)/\sqrt{q}$ for long channel transistors.[3] This fixes the values of $V_{X,i}$; when $V_{T,i}$ exceeds $V_{X,i}$, the digit a_i acts as $a_i + 1$; its address becomes $c_i + 1$, and we call this case the *flip-up defect*.

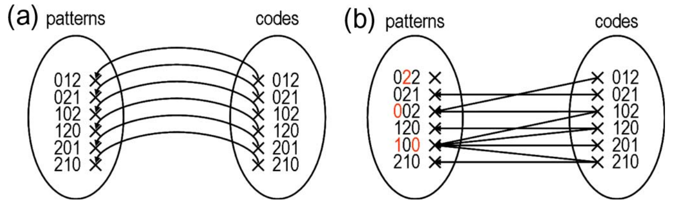

Fig. 4. Mapping of the code space onto the pattern space (a) in the defect-free case and (b) in the case of defects.

Now, consider the case when $V_{T,i}$ falls below $V_{A,i-1} - \delta \cdot V_0 = V_{X,i-1}$; then, the current flowing while $a_i - 1$ is applied is no longer $\sim$0 and always greater than q times the current flowing while $c_i - 2$ is applied. Then, a_i is implied by c_i and $c_i - 1$ but not by $c_i - 2$; its address is $c_i - 1$ which means that a_i acts as $a_i - 1$; this case is called the *flip-down defect*. The probabilities of flip-ups and -downs are given by the following expressions, which are independent of i. Here, f_i is the probability density function of $V_{T,i}$

$$p_u = \int_{V_{X,i}}^{\infty} f_i(x)dx \qquad p_d = \int_{-\infty}^{V_{X,i-1}} f_i(x)dx.$$

When $V_{T,i}$ falls within the range between the threshold values for flip-up and -down defects, the digit is correctly interpreted. We notice that the flip-down error never happens at digits having the smallest value 0, since the corresponding $\overline{V}_{T,i}$ is, by definition, smaller than the smallest $V_{A,i}$ available. For the same reason, the flip-up error never happens at the digits having the biggest value $n - 1$. In order to study the size of the addressable code space, we consider flip-up and -down errors in the code space instead of flip-up and -down defects at the NWs, since both considerations are equivalent.

B. Overall Impact of Variability

If V_T varies within a small range close to its mean value, then the pattern does not change, since the NW still conducts under the same conditions. Then, a 1-to-1 mapping between the code and the pattern space holds, which is shown in Fig. 4 for a ternary hot code with $M = 3$. On the contrary, if the V_T variation is large, then some digits may be shifted up or down, as explained previously. When a pattern has a sequence

[3]If we consider short channel transistors, then the saturation current is proportional to $(V_{A,i} - V_T)$ and $\delta = (\alpha + \upsilon)/q$.

of errors, it can be either covered by one or more codes or it can be uncovered. When we consider the codes, some of them cover one or more patterns, and some cover no pattern under the error assumptions. The following example explains this conjecture.

Example 3: Fig. 4(b) shows the digit shift at some patterns. We notice that the first pattern 022 (which underwent a defect) is no longer covered by any code. Thus, its NW cannot be addressed. All the other patterns are covered at least by one code. Two categories among these covered patterns can be distinguished. On the one hand, the fourth pattern 120 is covered by the fourth code, which, in turn, covers another pattern (the fifth). Thus, by activating the fourth NW, the required control voltages activate either the fourth and fifth NW. Consequently, the fourth NW cannot be addressed uniquely. This case represents the patterns covered only by codes covering more than a single pattern. On the other hand, the complementary case is illustrated by the fifth pattern 100, which is covered by many codes. However, one of these codes (201) covers no other pattern except the considered one. Thus, it is possible to uniquely activate the fifth NW by applying the voltage sequence corresponding to the code 201. ■

The examples shown in Fig. 4 demonstrate that a pattern undergoing defects can be either 1) not covered by any valid code word, in which case the NW cannot be identified as addressable and the pattern is useless, or 2) covered by at least one valid code word. In the second case, if two patterns or more are covered by the same code word, then this code word cannot be used because more than one NW would have the same address. Thus, in the second case, the pattern is only useful if at least one code word covering it covers no other pattern, insuring that the covered pattern can be addressable.

Assuming that, in average, every code word covers ν patterns when errors happen, let p_{I} be the probability that a pattern becomes uncovered and p_U the probability that a code word covers a unique pattern ($\overline{p}_U = 1 - p_U$). Let $|\Omega|$ be the original size of the code space and $|\Omega'|$ the size after errors happen. The set Ω' contains the useful addresses under defect conditions, i.e., those that address unique NWs even though the NWs are undergoing defects. The size of Ω' indicates the number of NWs that remain useful under high variability conditions. Then

$$|\Omega'| = |\Omega| \cdot (1 - p_{\mathrm{I}})\left(1 - \overline{p}_U^{\nu}\right). \tag{1}$$

In order to assess $|\Omega'|$, we model multidigit errors in the following sections. Then, we analytically derive p_{I} and p_U, and we estimate ν as a fit parameter from Monte Carlo simulations.

VII. Errors in the k Hot Code Space

A. Error Types

Ω refers, in the following, to the code space of the maximal-sized multivalued $(\mathbf{k}, M)$ hot encoding in the base $\mathbb{B} = \{0, \ldots, n-1\}$ with $\mathbf{k} = (k, \ldots, k)$ and $M = k \cdot n$. We consider a code word $\mathbf{c}$ in Ω undergoing a series of single digit errors. The multidigit error is described by the following vector $\mathbf{d} = (\mathbf{d}_0, \ldots, \mathbf{d}_{n-1})$, where each $\mathbf{d}_i$ represents a pair of integers (d_i^u, d_i^d) expressing the number of flip-ups and -downs occurring in each digit group having the value i. Since no flip-down occurs at digits with value 0 and no flip-up occurs at digits with value $(n-1)$, we impose $d_0^d = 0$ and $d_{n-1}^u = 0$. For instance, we consider the ternary hot code 010221 undergoing a flip-up defect at the third digit, turning it into 011221. One digit "0" flipped up, while no flip-error happened at the digits "1" and "2." This error is represented as $\mathbf{d} = (\binom{1}{0}, \binom{0}{0}, \binom{0}{0})$. Because the number of digits having the same value is k by definition, it must hold for each d_i: $0 \leq d_i^u + d_i^d \leq k$.

We distinguish two types of multidigit error $\mathbf{d}$ corresponding to uncovered (type I) and covered codes (type II) that we can formally describe in the following way:

1) Type I: $\exists i \in \{0, \ldots, i-2\} / d_i^u > d_{i+1}^d$.
2) Type II: $\forall i \in \{0, \ldots, i-2\}$: $d_i^u \leq d_{i+1}^d$.

This conjecture can be illustrated by the following two examples.

Example 4: We consider the same hot code 010221 turning into the error code 010220 with the defect $\mathbf{d} = (\binom{0}{0}, \binom{0}{1}, \binom{0}{0})$. By comparing the codes before and after the defect happened, we notice that the number of digits "0" has increased, while the number of digits "1" has decreased, and the number of digits "2" has remained the same. Any code coinciding with the error-free code at the digits "2" and "1" assigns "1" and "0" to the positions holding "0" in the error code, for instance, 010221 and 110220. All these codes cover the error code because both "1" and "0" cover "0." This remark can be generalized as follows: Whenever the error induces a decrease of the higher valued digits and an increase in the lower valued digits, it is always possible to find codes covering the error code. ■

Example 5: The opposite case can be illustrated by considering the same code 010221 turning into the error code 010222 with the defect $\mathbf{d} = (\binom{0}{0}, \binom{1}{0}, \binom{0}{0})$. Here, the number of digits "2" has increased, while the number of digits "1" has decreased, and the number of digits "0" has remained the same. Any code that would cover the error code would have at least three digits with the value "2." Such codes do not exist in the considered hot code space. Consequently, the error code cannot be covered by any code in the considered space. ■

B. Error Type I

The proof for the multidigit error of type I is given in the following. Consider the case that a code word in Ω is transformed into a code word $\mathbf{c}^*$ by a multidigit error $\mathbf{d}$ of type I. We denote by i the smallest position in $\mathbf{d}$ at which holds $d_i^u > d_{i+1}^d$. The number of digits in $\mathbf{c}^*$ whose values are $\geq (i+1)$ becomes larger than the permitted number in any code word $\mathbf{c}^b$ in Ω, namely, $k \cdot (n - i - 1)$. No code word would imply each digit of $\mathbf{c}^*$ with a value $\geq (i+1)$. Consequently, the NW with the pattern equal to the code word $\mathbf{c}^*$ is not addressed by any permitted code word. The probability of defect type I is given by p_{I} and uses the recursive Algorithm 1

$$p_{\mathrm{I}} = \sum_{u=0}^{k} \mathtt{ProI}\left(u, 0, 0, \frac{k!}{u! \cdot (k-u)!} \cdot (p_u)^u \cdot (1 - p_u)^{k-u}\right). \tag{2}$$

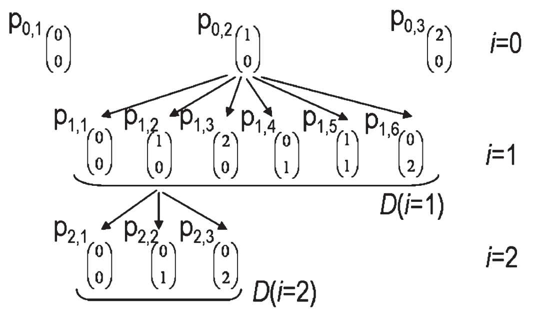

Fig. 5. Partial representation of error subtree for error $\binom{1}{0}$ at $i = 0$.

Here, $u = 0, \ldots, k$ is a variable going through all possible numbers of flip-up errors that can happen at digit "0" (flip-downs cannot occur at digit "0"); p_u is the probability of a flip-up error as explained in Section IV-A, and $\texttt{ProI}()$ gives the probability that a type I error occurs at any higher valued digit and that u flip-ups occur at digit "0" (see Algorithm 1).

```
Algorithm 1 $p_{\mathrm{I}} = \texttt{ProI}(x^u, x^d, i, p)$
1:  $\pi_{\mathrm{sum}} \leftarrow 0$
2:  Construct $D$ = all possible defects in subtree
3:  for all $\mathbf{y} = (y^u, y^d) \in D$ do
4:    if $i < n-1$ then
5:      $p \leftarrow p \cdot k!/(y^d! \cdot y^u! \cdot (k - y^d - y^u)!) \cdot (p_u)^{y^u} \cdot (p_d)^{y^d} \cdot (1 - p_u - p_d)^{k-y^u-y^d}$
6:      if $y^d \geq x^u$ then
7:        $\pi \leftarrow \texttt{ProI}(y^u, y^d, i+1, p)$
8:      else
9:        $\pi \leftarrow p$
10:     end if
11:   else
12:     $p \leftarrow p \cdot k!/(y^d! \cdot (k - y^d)!) \cdot (p_d)^{y^d} \cdot (1 - p_d)^{k-y^d}$
13:     if $y^d \geq x^u$ then
14:       $\pi \leftarrow 0$
15:     else
16:       $\pi \leftarrow p$
17:     end if
18:   end if
19:   $\pi_{\mathrm{sum}} \leftarrow \pi_{\mathrm{sum}} + \pi$
20: end for
21: return $\pi_{\mathrm{sum}}$
```

Algorithm 1 takes as input a defect described by $\binom{x^u}{x^d}$ at the digit level i that can happen with a probability p (describing the history of defects happening at digit levels $\leq i$). It delivers the probability that the assumed defect description can lead to a type I defect. In order to calculate this probability, the algorithm considers all possible error sequences that can happen at digit levels $> i$ and lead to a type I error. The algorithm is explained with the example in Fig. 5.

In Fig. 5, we assumed a ternary hot code with $M = 6$ and $k = 2$ (e.g., 001122, 012012, ...). Three types of errors can happen at digit "0": $\binom{0}{0}$, $\binom{1}{0}$, and $\binom{2}{0}$, because no flip-downs can happen at this digit. When u goes from 0 to $k = 2$ in (2), it describes these three errors. Each error has the probability $k!/(u! \cdot (k-u)!) \cdot (p_u)^u \cdot (1-p_u)^{k-u}$ ($u = 0, 1, 2$), which are denoted by $p_{0,1}$, $p_{0,2}$, and $p_{0,3}$, respectively, in Fig. 5. We assume now that Algorithm 1 is called for the error $\binom{1}{0}$ at the digit "0" [i.e., $u = 1$ in (2)]. The probability that any possible sequence of errors at higher level digits leads to a type I error is initialized at $\pi_{\mathrm{tmp}} = 0$ (line 1 in Algorithm 1). We consider the digit level "1," at which six different errors can happen: $D(i = 1) = \{\binom{0}{0}, \binom{1}{0}, \binom{2}{0}, \binom{0}{1}, \binom{1}{1}, \binom{0}{2}\}$. Generally, for a given level $i \notin \{0, n-1\}$, the elements of D are generated as follows: $D\{\binom{x^u}{x^d}$, s.t. $x^d = 0, \ldots, k$ and $x^u = 0, \ldots, (k - x^d)\}$. For $i = n-1$, since no flip-ups can happen, $D = \{\binom{0}{x^d}$, s.t. $x^d = 0, \ldots, k\}$. Every error in $D(i = 1)$ has a probability designated by $p_{1,1}, \ldots, p_{1,6}$ in Fig. 5 and calculated at line 5. We consider the element $\binom{0}{0}$ in $D(i = 1)$ for which the following holds: The number of flip-ups at level "0" is higher than the number of flip-downs at level "1." This error sequence induces a type I error. The algorithm saves the probability of this event (line 9). On the other hand, if we consider the element $\binom{1}{0}$ in $D(i = 1)$, then no type I error is detected, and the algorithm is called iteratively for the next digit level "2" (line 7). The algorithm constructs the elements of $D(i = 2) = \{\binom{0}{0}, \binom{0}{1}, \binom{0}{2}\}$ as explained previously and goes through them. Only $\binom{0}{0}$ fulfills the condition of a type I error; for which the error probability is saved (line 16). For the two others, 0 is returned (line 14). Then, the updated type I error probability π is returned (line 19 and 21).

C. Error Type II

Now, let the code word $\mathbf{c}^a$ in Ω be transformed into $\mathbf{c}^*$ by a multidigit error $\mathbf{d}$ of type II. The number of digits in $\mathbf{c}^*$ having the value i is always larger than the number of digits having the value $(i + 1)$. It is possible to construct one or more code words $\mathbf{c}^b$ in Ω implying $\mathbf{c}^*$. An intuitive way consists in starting with the smallest digit value 0 and filling the digits of $\mathbf{c}^b$ by 0 with respect to the positions held by the value 0 in the digits of $\mathbf{c}^*$. The procedure is repeated iteratively on the next digit values until all digits of $\mathbf{c}^b$ are allocated (Algorithm 2). In Algorithm 2, we use the following notations: The number of digits having the value i in $\mathbf{c}^*$ is l_i. Their respective positions are $p_0^i, \ldots, p_{l_i-1}^i$. The definition of the defect pattern $\mathbf{d}$ yields $l_i = k - d_i^u + d_{i+1}^d + d_{i-1}^u \ \forall i$, with the conventions $d_{i-1}^u = 0$ for $i = 0$ and $d_{i+1}^d = 0$ for $i = n - 1$.

```
Algorithm 2 $\Sigma = \texttt{CoveredSet}(i, \nu, \Delta, \mathbf{c}^*)$
1:  Construct $S_1$ = set of all positions of digit $i$ in $\mathbf{c}^*$
2:  Construct $\sigma$ = set of subsets of $S_1$ with $\nu$ elements
3:  if $i < n-1$ then
4:    $\Delta_{\mathrm{tmp}} = \varnothing$
5:    for all $S_2 \in \sigma$ do
6:      $\Delta_{\mathrm{rep}} = \Delta$
7:      for all $\mathbf{c}^b \in \Delta_{\mathrm{rep}}$ do
8:        Allocate $i$ to digits of $\mathbf{c}^b$ at positions $S_2$
9:        Allocate $i+1$ to digits of $\mathbf{c}^b$ at positions $S_1 \setminus S_2$
10:     end for
11:     $\Delta_{\mathrm{tmp}} \leftarrow \Delta_{\mathrm{tmp}} \cup \Delta_{\mathrm{rep}}$
12:   end for
13:   $\nu \leftarrow \nu + k - |S_1|$
14:   $i \leftarrow i + 1$
```

```
15:     $\Delta \leftarrow \Delta_{\mathrm{tmp}}$
16:     **return** CoveredSet$(i, \nu, \Delta, \mathbf{c}^*)$
17: **else**
18:     **for all** $\mathbf{c}^b \in \Delta$ **do**
19:         Allocate $n-1$ to digits of $\mathbf{c}^b$ at remaining positions
20:     **end for**
21:     **return** $\Delta$
22: **end if**
```

From the definition of the type II errors, there is at least a digit value i for which $\nu_i < l_i$ holds. Each choice of ν_i elements among l_i possible values (line 2) gives many possible choices for $\mathbf{c}^b$. This proves that it is possible to find more than one code word covering the considered error code type II. The following example explains Algorithm 2 by means of the error code $\mathbf{c}^* = 010220$.

Example 6: In order to find all code words which cover a given error code $\mathbf{c}^*$, Algorithm 2 is called with $i = 0$, $\nu = k$, and Δ a single space-holder code with $M \times$ "$*$," where "$*$" means that the digits are not allocated yet and can take any possible value. For $\mathbf{c}^* = 010220$, Algorithm 2 is called with $i = 0$, $\nu = 2$, and $\Delta = \{******\}$. We start with the digit level "0," whose positions in $\mathbf{c}^*$ are given by $S_1 = \{0, 2, 5\}$ (line 1). Then, $\sigma = \{\{0, 2\}, \{0, 5\}, \{2, 5\}\}$ (line 2). Lines 4–12 set Δ_{tmp} to $\{0*0**1, 0*1**0, 1*0**0\}$. Line 16 calls the algorithm with $i = 1$, $\nu = 1$, and Δ set to Δ_{tmp}. During this call of Algorithm 2, $S_1 = \{1\}$ (line 1) and S_2 can take the only value $\{1\}$. Then, lines 4–12 update Δ_{tmp} to $\{010**1, 011**0, 110**0\}$. Subsequently, the algorithm is called again with $i = 2$, $\nu = 2$, and Δ set to Δ_{tmp} (line 16). During this call, the lines 17–20 are executed and just fill in the remaining space holders with the digit "2." Finally, Δ is set to $\{010221, 011220, 110220\}$ and returned recursively to the top level. To conclude, the error code $\mathbf{c}^*$ can be covered by three possible code words. ■

D. Unique Covering

In order to assess the probability that a code word $\mathbf{c}^a$ uniquely covers an error code, we first need to enumerate the code words $\mathbf{c}^b$ which can undergo a sequence of defects to become covered by $\mathbf{c}^a$ and the probability of each one of these events. Then, we can derive the probability that exactly one of these events happens, which is equivalent to saying that $\mathbf{c}^a$ covers a *unique* error code.

Enumerating these events is performed in two steps: We first define the set S of code words $\mathbf{c}^b$ that can be transformed into $\mathbf{c}^a$ by a sequence of flip-ups and -downs. Then, the considered events consist in making each element $\mathbf{c}^b$ undergo a sequence of defects that turn it not necessarily into $\mathbf{c}^a$ but just make it covered by $\mathbf{c}^a$. We describe the elements $\mathbf{c}^b$ of S in an abstract way, consisting of a transformation matrix $\mathcal{T}$ which describes the flip-ups and -downs that $\mathbf{c}^b$ needs to undergo in order to turn to $\mathbf{c}^a$. Then, each element is assigned the probability of covering under defects.

A code word $\mathbf{c}^b$ is transformed into another code word $\mathbf{c}^a$ by undergoing at each digit level $i = 0, \ldots, n-1$: t_j^i $(j-i)$-order flip-ups (for $j = i+1, \ldots, n-1$) and t_j^i $(i-j)$-order flip-downs (for $j = 0, \ldots, i-1$). We use the following convention: t_i^i designates the number of correct digits at level i. A transformation $\mathcal{T}$ affecting a whole code word is a set of the transformations $\mathbf{t}^i = [t_0^i, \ldots, t_{n-1}^i]^\top$ affecting each digit level $i = 0, \ldots, n-1$. Thus, $\mathcal{T}$ is the matrix $[\mathbf{t}^0, \ldots, \mathbf{t}^{n-1}]$. The $(i+1)$th column of $\mathcal{T}$ describes the transformation happening at digit levels $i = 0, \ldots, n-1$. Since there are k occurrences of the digit level i in each code word, $\mathcal{T}$ must verify: $\forall i \sum_j t_j^i = k$. The $(i+1)$th row indicates the number of digits with the value i in the code obtained after the transformation (i.e., $\mathbf{c}^a$). This number has to be set to k. Then, we derive the second condition on $\mathcal{T}$: $\forall j \sum_i t_j^i = k$.

Example 7: We consider the ternary hot code with $M = 6$ and $k = 2$. The digit level "0" undergoes only one flip-up: $\mathbf{t}^0 = [1, 1, 0]^\top$. The digit level "1" undergoes one flip-up and one flip-down: $\mathbf{t}^1 = [1, 0, 1]^\top$. The digit level "2" undergoes no errors: $\mathbf{t}^2 = [0, 0, 2]^\top$. Thus, the code word 001122 undergoing $\mathcal{T} = [\mathbf{t}^0, \mathbf{t}^1, \mathbf{t}^2]$ can be transformed, for instance, into 010222, which is not an element of the considered code space. This is due to the fact that the second condition is not fulfilled. We suggest the transformation $\widetilde{\mathcal{T}}$ meeting both conditions

$$\mathcal{T} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 2 \end{bmatrix} \qquad \widetilde{\mathcal{T}} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

$\widetilde{\mathcal{T}}$ would transform 001122 into 010212 (for instance), which is in the same code space. ■

Now, we inject defects that make $\mathbf{c}^b$ controlled by $\mathbf{c}^a$. At each digit value i, the t_j^i digits, for which $j > i$ holds, must each undergo a $(j-i)$-order (or higher) flip-down defect, in order to decrease their value down to that of the corresponding digit in $\mathbf{c}^a$. On the contrary, the t_j^i digits, for which $j < i$ holds, already have a value which is smaller than that of the corresponding digit in $\mathbf{c}^a$; therefore, they can undergo any kind of first-order defect or be correct. The remaining t_i^i digits in $\mathbf{c}^b$ have the same values as their counterparts in $\mathbf{c}^a$; therefore, they can undergo a flip-down error or no error. Only first-order flip errors are considered in this paper, because the likelihood of higher order defects is negligible. Thus, we set t_j^i to 0 for $j > i+1$ in order to reduce the computation time. This represents the third condition on $\mathcal{T}$.

These conditions define the eligible set of transformations $\mathcal{T}$. For each transformation, the probability that a sequence of defects transforms the digits of $\mathbf{c}^b$ with the value i into an error code covered by $\mathbf{c}^a$ at the positions corresponding to these digits is $\tilde{p}_i = (p_d)^{u_i} \cdot (1-p_u)^{k-u_i-d_i}$ with $u_i = t_{i+1}^i$ and $d_i = \sum_{j<i} t_j^i$ $\forall i$. The probability that the code word $\mathbf{c}^b$ is transformed into an error code covered by $\mathbf{c}^a$ is identical to the probability that the previous event holds for each digit value i. This gives the event probability $p = \prod_i \tilde{p}_i$.

Given the transformation $\mathbf{t}^i$ affecting the k occurrences of the digit value i, there are μ_i distributions of the t_j^i transformations on the k elements, given by $\tilde{\mu}_i = k!/\prod_j t_j^i!$. Considering the n possible values of i, there are μ code words $\mathbf{c}^b$ transformed into an error code covered by $\mathbf{c}^a$ with the same probability p: $\mu = \prod_i \tilde{\mu}_i$.

Example 8: The code $\mathbf{c}^a = 001122$ in the previous example was transformed by $\widetilde{\mathcal{T}}$ to $\mathbf{c}^b = 010212$. A series of defects transforms $\mathbf{c}^b$ into $\mathbf{c}^*$ that is covered by $\mathbf{c}^a$. Thus, the first digit in $\mathbf{c}^b$ (corresponding to the first occurrence of "0" in $\mathbf{c}^a$) has to remain unchanged in order to be covered by $\mathbf{c}^a$. The second digit in $\mathbf{c}^b$ (corresponding to the second occurrence of "0" in $\mathbf{c}^a$) has to flip down to be covered by $\mathbf{c}^a$. This gives $\tilde{p}_0 = p_d(1-p_u)$. In a similar way, we find $\tilde{p}_1 = p_d$ and $\tilde{p}_2 = 1$. Then, $p = (p_d)^2(1-p_u)$. There are many images of $\mathbf{c}^a$ through $\widetilde{\mathcal{T}}$, for instance, 102012, 102021, etc. Their number is obtained by permuting the flipping digits. For instance, $\widetilde{\mathcal{T}}$ requires that exactly one digit "0" flips up: There are $\widetilde{\mu}_0 = 2!/(1! \cdot 1! \cdot 0!) = 2$ permutations. In a similar way, we find $\widetilde{\mu}_1 = 2$ and $\widetilde{\mu}_2 = 2$ for the digits "1" and "2," respectively. Finally, the number of possible code words $\mathbf{c}^b$ is $\mu = 8$. ■

Algorithm 3 represents a formulation of the method explained previously. It assumes that all eligible transformations $\mathcal{T}$ were calculated, for instance, by selecting in an exhaustive way those meeting the three conditions aforementioned among all matrices in $\{0, \ldots, k\}^{n \times n}$. Then, it defines the set S of all (p, μ), where p is the probability of the code words $\mathbf{c}^b$ represented by the valid transformation to undergo the right error sequence that makes it covered by $\mathbf{c}^a$ and q is the number of its equivalent occurrences.

```
Algorithm 3 $S = \mathtt{UniqueSet}(\mathcal{T})$
 1: Construct $\mathcal{E} = \{$all eligible transformations $\mathcal{T}$ meeting
    the 3 conditions above$\}$
 2: $S \leftarrow \varnothing$
 3: for all $\mathcal{T} \in \mathcal{E}$ do
 4:     $u_i \leftarrow t^i_{i+1} \forall i$
 5:     $d_i \leftarrow \sum_{j<i} t^i_j \forall i$
 6:     $\tilde{p}_i \leftarrow p_d^{u_i} \cdot (1-p_u)^{k-u_i-d_i}$
 7:     $p \leftarrow \prod_i \tilde{p}_i$
 8:     $\tilde{\mu}_i \leftarrow k!/\prod_j t^i_j!$
 9:     $\mu \leftarrow \prod_i \tilde{\mu}_i$
10:     $S \leftarrow S \cup \{(p, \mu)\}$
11: end for
12: return $S$
```

Let S be the set of all eligible events with their respective occurrences; then, the probability that exactly one event happens (i.e., $\mathbf{c}^a$ covers exactly one error code) can be calculated as follows:

$$p_U = \sum_{i=1,\ldots,|S|} \mu_i \cdot p_i/(1-p_i) \times \prod_{i=1,\ldots,|S|} (1-p_i)^{\mu_i}. \quad (3)$$

In summary, this section suggested mathematical methods to estimate the probability of uncovered codes (p_{I}) and the probability of unique covering (p_U). The probability p_{I} was calculated by constructing the whole set of type I errors and estimating the probability of each one of these errors. On the other hand, p_U was estimated from the set of code words that can undergo a sequence of errors and become covered by a given reference code word. Once p_{I} and p_U are known, they can be inserted in (1) in order to estimate the size of the addressable code space under defects, i.e., the number of NWs that can be addressed in the array under high variability conditions.

VIII. Errors in the NRC Space

A. Error Types

In the following, Ω refers to an arbitrary NRC space with length M (M is even) and base size n. We define a flip-up defect at digit c_i in the code word $\mathbf{c}$ to be canceled when a flip-down defect occurs at digit $c_{M/2+i}$. A canceled flip-down defect is defined in a complementary way.

As for the hot codes, we distinguish two types of multidigit errors for reflexive codes, corresponding to the uncovered (type I) and covered codes (type II) that we can describe in the following way.

1) Multidigit errors of type I. The code word undergoes at least one uncanceled flip-up.
2) Multidigit errors of type II. The code word only experiments flip-downs and/or canceled flip-ups.

In order to illustrate this assumption, we suggest the following two examples.

Example 9: We consider the code word $\mathbf{c}^a = 00012221$ in the ternary $(n = 3)$ reflexive code space with the length $M = 8$. $\mathbf{c}^a$ undergoes an uncanceled flip-up error at the fourth digit and turns to the error code $\mathbf{c}^* = 00022221$. Any hypothetical code word that would cover $\mathbf{c}^*$ would have, at the fourth digit, the value 2 and, at the last digit, the value 0 (then, the last digit would not be covered) or it would have, at the last digit, the value 1 or 2 and, at the fourth digit, the value 1 or 0, respectively, (then, in both cases, the fourth digit would not be covered). The uncanceled flip-up defect causes any hypothetical covering code to cover either the first half of the code or its reflected half, but never both of them at the same time. Consequently, the error code $\mathbf{c}^*$ cannot be covered by any code in the considered NRC space. ■

Example 10: On the other hand, if we consider the same code word $\mathbf{c}^a$ undergoing the canceled flip-up defect at the fourth digit, then it turns to $\mathbf{c}^* = 00022220$ which is a code word from the same space, and it is consequently covered by itself. If, in addition to the canceled flip-up defect at the fourth digit, a flip-down defect occurs, for example, at the fifth digit, then the code turns to the error code $\mathbf{c}^* = 00021220$ which is in turn covered by the code words 00022220 and 10021220. ■

B. Error Type I

Here, we explain formally how codes undergoing type I errors are not covered by any code in the considered code space. Let the code word $\mathbf{c}^a$ in Ω be transformed into $\mathbf{c}^*$ by a multidigit error of type I. We denote by i one of the positions in the first half of the code word, at which an uncanceled flip-up error occurs. Then, $c^*_i = c^a_i + 1$, and $c^*_{M/2+i} \geq c^a_{M/2+i}$. Any code word $\mathbf{c}^0$ which would imply the pattern corresponding to $\mathbf{c}^*$ would verify that $c_i^0 \geq c_i^* = c_i^a + 1$. Then, $c_{M/2+i}^0 = \mathtt{NOT}(c_i^0) = n - 1 - c_i^0 \leq n - 1 - c_i^a - 1 = c_{M/2+i} - 1 < c_{M/2+i}^*$ and $\|\sigma(\mathbf{c}^* - \mathbf{c}^0)\| \neq 0$. Thus, there exists no code word in Ω that would cover $\mathbf{c}^*$. The size of the addressable space is reduced by the number of code words undergoing the multidigit error of type I.

Unlike hot codes, we give no recursive form for the probability of type I errors affecting reflexive codes; we rather derive the explicit analytical expression. However, it is much easier to consider the complementary case of the type I error, which is the type II error (probability p_{II}) or no errors (immune codes with the probability p_{im}): $p_{\mathrm{I}} = 1 - (p_{\mathrm{II}} + p_{\mathrm{im}})$. The exact expression of $p_{\mathrm{II}} + p_{\mathrm{im}}$ and, consequently, the one of p_{I} are derived in the following section.

C. Error Type II

Let the code word $\mathbf{c}^a$ in Ω be transformed into $\mathbf{c}^*$ by a multidigit error $\mathbf{d}$ of type II. It is possible to find more than one code word $\mathbf{c}^0$ which covers the pattern corresponding to $\mathbf{c}^*$. We construct first, for example, the left half of $\mathbf{c}^0$; then, the right half is obtained by complementing the left half. The construction rule is the following $(i = 0, \ldots, M/2 - 1)$.

1) If c_i^a undergoes a noncanceled flip-down error, then c_i^0 can be set to either $c_i^a - 1$ or c_i^a.
2) If c_i^a undergoes a canceled flip-down error, then c_i^0 is set to $c_i^a - 1$.
3) If c_i^a undergoes a canceled flip-up error, then c_i^0 is set to $c_i^a + 1$.
4) If c_i^a has no error, then c_i^0 is set to c_i^a.
5) $c_{M/2+i}^0$ is set to $n - 1 - c_i^0$.

It is easy to verify that all patterns corresponding to $\mathbf{c}^0$ constructed this way cover $\mathbf{c}^*$. In order to calculate the probability p_{I} of the multidigit error type I, we observe the complementary event (multidigit error type II or no error in code word $\mathbf{c}^a$).

1) If c_i^a undergoes no error, then $c_{M/2+i}^a$ must undergo a flip-down or no error: $(1 - p_u - p_d)(1 - p_u)$.
2) If c_i^a undergoes a flip-down error, then $c_{M/2+i}^a$ can have any value: p_d.
3) If c_i^a undergoes a flip-up error, then $c_{M/2+i}^a$ must undergo a flip-down error: $p_u p_d$.

This scheme assumes that the digit c_i^a has the value $1, \ldots, n - 2$, i.e., it is not at the range borders. In this case, the probability that the digit c_i^a and its complement in the reflected code half undergo no type I error is $\overline{p}_{\mathrm{I},1} = (1 - p_u - p_d)(1 - p_u) + p_d + p_u p_d = (1 - p_u)^2 + 2 p_u p_d$.

If the considered digit c_i^a has the value 0 or $n - 1$, then its complement $c_{M/2+i}^a$ has the value $n - 1$ or 0, respectively, and we derive, in a similar way, the probability that the digit c_i^a and its complement undergo no type I error in this special case: $\overline{p}_{\mathrm{I},2} = 1 - p_u + p_u p_d$. In order to consider a digit with any value in 0 to $n - 1$, we apply the binomial theorem, which results in a simple averaging of both probabilities: $\overline{p}_{\mathrm{I,avg}} = \overline{p}_{\mathrm{I},1} \cdot (n - 2)/n + \overline{p}_{\mathrm{I},2} \cdot 2/n$.

Since a code word has $M/2$ couples of complementary digits, the probability that no type I error happens, i.e., that no error or only type II errors happen, is given by $p_{\mathrm{II}} + p_{\mathrm{im}} = \overline{p}_{\mathrm{I,avg}}^{M/2}$. The probability of immune codes is not given here, since we are only interested in p_{I}, which is given by $1 - (p_{\mathrm{II}} + p_{\mathrm{im}})$

$$p_{\mathrm{I}} = 1 - \left(\frac{n-2}{n} \overline{p}_{\mathrm{I},1} + \frac{2}{n} \overline{p}_{\mathrm{I},2} \right)^{M/2}. \qquad (4)$$

D. Unique Covering

Now, we would like to calculate p_U. In principle, the same reasoning for hot codes can be applied on reflexive codes too. In order to assess the probability that a code word $\mathbf{c}^a$ uniquely covers an error code, we first enumerate all the code words $\mathbf{c}^b$ that can be transformed into $\mathbf{c}^a$ by a sequence of flip-ups and -downs. Then, we calculate the probability that these code words $\mathbf{c}^b$ undergo errors and become covered by $\mathbf{c}^a$.

For hot codes, we considered all the transformations $\mathcal{T} = (\mathbf{t}^0, \ldots, \mathbf{t}^{n-1})$ that transform $\mathbf{c}^b$ into $\mathbf{c}^a$, and we represented them by matrices. The fact that we considered only first-order defects fixed the limit $t_j^i = 0$ for $j > i + 1$. Because of the reflection principle, the reflected half of the code word fixes the limit $t_j^i = 0$ for $j < i - 1$. Consequently, the transformation becomes much simpler than in the case of hot codes: for each digit i, $(c_i^a - c_i^b) \in \{-1, 0, 1\}$ must hold. We do not need the matrix representation in order to define the set of code words $\mathbf{c}^b$ verifying this condition: Indeed, the set of $\mathbf{c}^b$ represents the neighborhood of $\mathbf{c}^a$ that can be graphically represented in the space $\mathbb{B}^{M/2}$ by a hypercube with the edge length 2 and centered around $\mathbf{c}^a$. If $\mathbf{c}^a$ has a certain number u of digits at the range border (i.e., 0 or $n - 1$), then only $(c_i^a - c_i^b) \in \{-1, 0\}$ or $\{0, 1\}$ holds, respectively, and the volume of hypercube is halved for each one of these digits.

Given this definition of the set of code words $\mathbf{c}^b$, we can now enumerate them, first, by assuming that no digit is at the range borders. We consider only the first half of the code words because it completely defines the whole code word by applying the reflection principle. Let α be the number of digits in a half-word such that $c_i^b - c_i^a = 0$ $(\alpha = 0, \ldots, M/2)$. There are $\tilde{\mu}_\alpha = \binom{M/2}{\alpha} \cdot 2^{M/2 - \alpha}$ possible code words fulfilling this condition. Each one of them can be transformed into an error code $\mathbf{c}^*$ covered by $\mathbf{c}^a$ if the α digits in each half undergo no flip-up error and each one of the $M/2 - \alpha$ digits for which holds $c_i^* - c_i^a = 1$ (in the whole code word) undergoes a flip-down error. The other $M/2 - \alpha$ digits for which holds $c_i^* - c_i^a = -1$ (in the whole code word) can undergo any kind of defect, or they can be correct. The likelihood of each event is $\tilde{p}_\alpha = (1 - p_u)^{2\alpha} \cdot p_d^{M/2 - \alpha} \cdot 1^{M/2 - \alpha}$.

We can now consider the cases in which the first half of $\mathbf{c}^b$ has u digits at the range border such that $c_i^b - c_i^a \neq 0$ and α is still the number of digits in the half-word such that $c_i^b - c_i^a = 0$ $(\alpha = 0, \ldots, M/2$ and $u = 0, \ldots, M/2 - \alpha)$. The number of code words having u digits among $M/2 - \alpha$ such that $c_i^b - c_i^a \neq 0$ is $2^{(M/2 - \alpha) - u} \cdot 1^u$. The average number of occurrences of each one of these words is $\binom{M/2 - \alpha}{u} (2/n)^u ((n - 2)/n)^{M/2 - \alpha - u}$. Then, the binomial theorem yields the average number of code words having α digits such that $c_i^b - c_i^a = 0$

$$\mu_\alpha = \binom{M/2}{\alpha} \left(2 \cdot \frac{n-1}{n} \right)^{M/2 - \alpha}.$$

We consider now the probability of occurrence of the transformation affecting each one of the μ_α code words without discarding the digits at the range borders. The indexes α and u keep the same definitions as before (α being again the number of digits in the half of a code word such that $c_i^b - c_i^a = 0$ and u the number of digits in the rest of the code half having their values at the range borders: $\alpha = 0, \ldots, M/2$ and $u = 0, \ldots, M/2 - \alpha$). In addition, we call v the number of digits in the set of α digits having their value at the range borders ($v = 0, \ldots, \alpha$). First, we fix the value of α. In order for $\mathbf{c}^b$ to undergo errors making it covered by $\mathbf{c}^a$, each one of the α digits must undergo no flip-up error. If the digit is among the ones at the range border (with the probability $2/n$), then having no flip-up error happens with the probability $1^v \cdot (1 - p_u)^v$. If the digit is not at the range border (with the probability $(2 - n)/n$), then having no flip-up error happens with the probability $(1 - p_u)^{2(\alpha - v)}$. The remaining $M/2 - \alpha$ digits are divided into those for which holds $c_i^b - c_i^a = 1$, and which have to undergo a flip-down error, and those for which holds $c_i^b - c_i^a = -1$, and which can undergo any kind of error or be correct. Because of the reflection principle, the probability that the remaining $M/2 - \alpha$ digits undergo the right error sequence is independent of u, and it is equal to $p_d^{M/2-\alpha}$. Consequently, the probability that each one of the μ_α code words $\mathbf{c}^b$ undergoes the right defect sequence and becomes covered by $\mathbf{c}^a$ is given by the binomial theorem

$$p_\alpha = \left(\frac{n-2}{n}(1-p_u)^2 + \frac{2}{n}(1-p_u) \right)^\alpha \cdot p_d^{M/2-\alpha}.$$

Finally, having the set S of all possible events that $\mathbf{c}^b$ becomes covered by $\mathbf{c}^a$ and their respective occurrences, the likelihood that only one event happens, i.e., only one code word in S is transformed into another word covered by $\mathbf{c}^0$, is given by

$$p_U = \sum_{i=0}^{M/2} \mu_i \cdot p_i/(1-p_i) \times \prod_{i=0}^{M/2} (1-p_i)^{\mu_i}. \qquad (5)$$

In summary, this section suggested the analytical expressions of the probability of uncovered codes (p_I) and the probability of unique covering (p_U). For each probability, we considered the simplified case, in which the digits are not at the range border, i.e., 0 and $n-1$, then we derived the general case, where the digits can have any value. The NRC space is easier to describe analytically; thus, the given probabilities are in explicit forms unlike the recursive forms given in the k hot code space. Once p_I and p_U are known, they can be used in (1) in order to analytically estimate the size of the addressable code space under defect conditions.

IX. Simulations of the Addressable Code Space

In order to assess the variation of the addressable code space under variable V_T, we used the following simulation parameters: $q = 100$, $V_0 = V_{\mathbf{DD}} = 1$ V, and $\alpha \sim 1/n$ since the difference between the highest and lowest $V_{T,i}$'s is limited by the constant V_DD. We plotted separately the uncovered part $|\Omega|_\mathrm{un} = p_\mathrm{I} \cdot |\Omega|$, the addressable part $|\Omega'|$, and the immune part

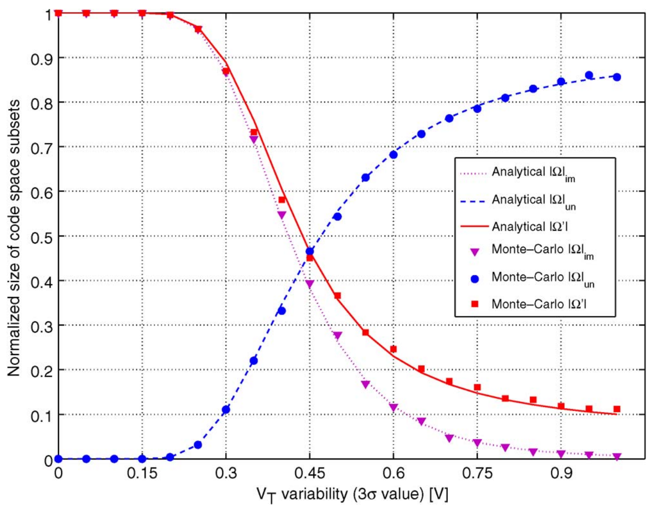

Fig. 6. Dependence of different code space subsets on V_T variability.

in which no defects occur $|\Omega|_\mathrm{im}$. If we neglect the effect of digits at the range border, then $|\Omega|_\mathrm{im} = (1 - p_u - p_d)^M \cdot |\Omega|$. In the simulations, we used the exact expression taking into account the effect of range borders. The fit parameter ν was estimated with Monte Carlo simulations. Fig. 6 shows the sizes of these subspaces for a ternary (3, 14) reflexive code depending on the 3σ value of V_T. The Monte Carlo simulations confirm, in the same figure, the analytical results and gives the value 2.8 for the fit parameter ν. The size of the addressable space $|\Omega'|$ drops quickly when 3σ reaches 0.4 V. At the same time, more patterns become uncovered. Interestingly, there are more addressable than immune patterns, because some defective patterns can be randomly addressed. This tendency increases for unreliable technologies, and around 10% of the original code space size can be randomly addressed under extreme conditions. The simulation of hot codes was not shown, because the result is similar, except for large defect probabilities: Under these conditions, the size of the addressable space goes faster toward 0 because the construction of hot codes imposes more constraints than the NRC.

The fact that the hot code space is under more constraints than the reflexive code space could be illustrated by investigating the value of ν for different codes. It is important to emphasize the fact that ν is a statistical quantity that gives the average number of elements in the code spaces mapped onto the pattern space under variability conditions through the relation "code covers pattern." Its value was estimated by matching the analytical results to the Monte Carlo simulations. For hot codes, ν has a value between 1.3 and 1.4, while for reflexive codes, ν was ranging between 1.8 and 3.0.

We also investigated the impact of the placement of V_A between two successive $V_{T,i}$'s. We postulated, in an intuitive way, that V_A has to be the median of $V_{T,i}$ and $V_{T,i+1}$. While modeling the defects (Section VI), we allowed V_A to translate by a small value v. When v increases (i.e., V_A moves toward $V_{T,i}$), the probability of a flip-up increases and that of a flip-down decreases. The opposite happens when V_A moves toward $V_{T,i+1}$. The normalized number of addressable NWs has been shown in Fig. 7. For unreliable devices with $3\sigma > 0.1$ V, the

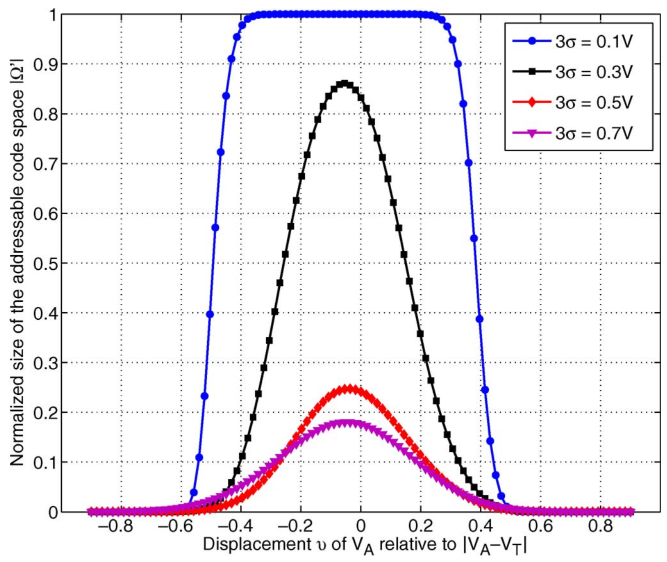

Fig. 7. Impact of the value of V_A on the number of addressable NWs.

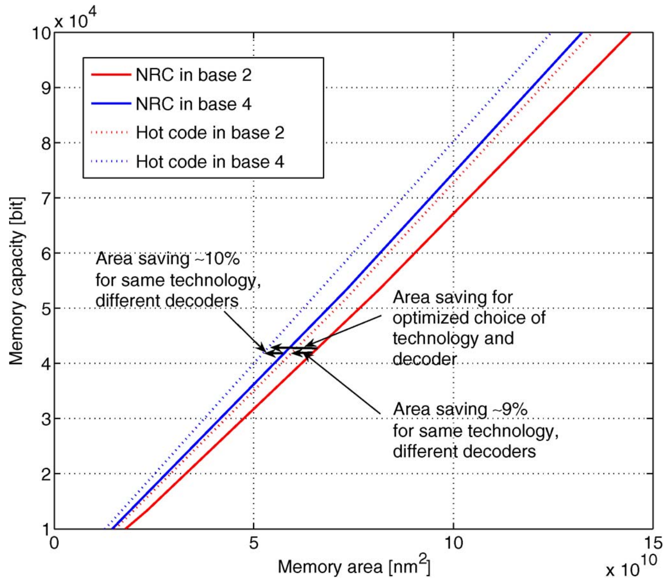

Fig. 8. Memory area/capacity design space for different encoders.

optimal position of V_A is slightly shifted from the middle of $V_{T,i}$ and $V_{T,i+1}$ toward $V_{T,i+1}$ by a few tens of millivolts. While reliable devices show a plateau around the optimal value of υ and necessitate no accurate calibration of V_A, the circuit designer has to calibrate the applied voltages V_A in a precise way when the transistors are not reliable; otherwise, a certain loss in the number of addressable NWs has to be taken into account. This circuit level issue has to be considered for either hot or n-ary reflected codes (both codes showed a similar behavior). In our technology characterized by a 3σ value below 100 mV, no calibration will be needed.

X. Simulations of the Effective Memory Area

The previous section enables the assessment of the number of NWs that the decoder can address uniquely even with variable threshold voltages. Based on this estimate, we can evaluate the effective capacity of the crossbar memory, i.e., the number of addressable bits. An analysis of the defects affecting the switches is beyond the scope of this paper. Thus, we can estimate the effective capacity C_{eff} in an analog way to the memory yield in [9]: $C_{\text{eff}} = (\eta \cdot |\Omega'|)^2$, where η represents the effects of contact and control quality of NWs as explained in [9] and [26]. The area can be estimated from the geometry as shown in Fig. 2. The NW pitch depends on the considered fabrication process, whereas the MWs are defined according to the technology node 45 nm [1].

A. Top–Down Approach: GAA Decoder-Based Memories

We explored the memory effective capacity/area design space, and we performed a simulation of the design space of our technology (Fig. 8). The processes based on two and four V_T's were considered for both hot and n-ary reflected codes. As Fig. 8 shows, the hot code generally reduces the decoder area and, consequently, the memory area, because with the same code length, it is possible to address a larger code space. For instance, by using the same technology with four V_T's to fabricate a 4-kB memory, the NRC has an area overhead of $\sim$10% compared to the hot code. The use of a simpler technology with two V_T's implies an area overhead of $\sim$24% for the same memory size. The area saving can be implemented at either the technology or system level. It is worth to notice that the area saving is more significant for small memory sizes (typically less than 0.1 Mb), because the memory area for large memories is dominated by the area of the programmable array. The programmable array can be split into smaller blocks defined by the size of the ohmic regions (Fig. 2) in order to reach the optimal size.

B. Bottom–Up Approach: Axial Decoder-Based Memories

MVL decoding can be used also within other decoder technologies. As an example of bottom–up approaches, we chose the axial decoders. The yield of axial decoders was investigated under different scenarios, by changing the process variability σ and the code type. The particularity of decoders in this bottom–up approach is that they assign addresses randomly. The size of the ohmic region is set to the smallest value allowed in the considered technology in order to maximize the efficiency of the random address assignment [9].

We considered memory blocks with 32-kB raw density, where the raw density designates the memory density in the error-free case. We first set σ to a very low value (below 10 mV) so that the defects have a negligible impact on the bit area. The results are shown in Fig. 9, using a reflexive code with $n = 2$ and 3. The results for hot codes have a similar qualitative behavior. For $n = 2$, the effective bit area is large for short codes because small code spaces do not insure enough unique codes. The randomness in the bottom–up decoding schemes necessitates a code space that is large enough to insure unique addressing. On the other hand, long codes increase the size of the decoder, and the cost in terms of area cancels the gain in terms of unique codes. Between these two regions, an optimal code length exists for which the effective bit area is

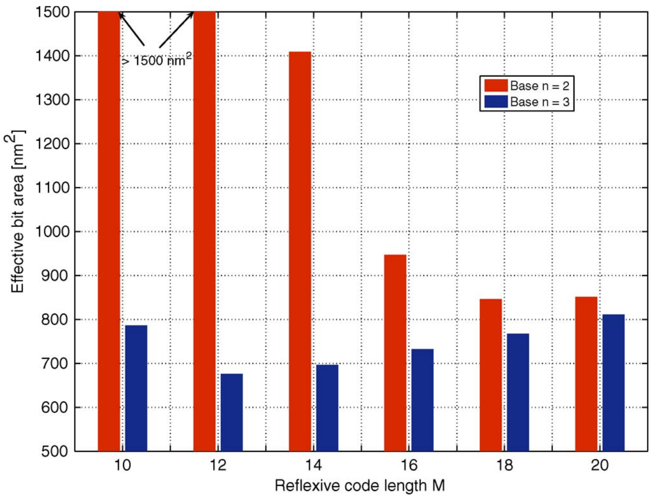

Fig. 9. Effective bit area versus code length for binary and ternary reflexive codes and $C_{\text{raw}} = 32$ kB.

TABLE I
YIELD OF DIFFERENT BOTTOM–UP DECODERS IN TERMS OF AREA PER WORKING BIT (nm^2) AT THE TECHNOLOGY NODE OF 45 nm

Raw Size	Base	Axial Decoder	Mask-Based
8 kB	2	1576	622
8 kB	3	1196	550
8 kB	Δ	24.1%	11.5%
32 kB	2	846	423
32 kB	3	676	373
32 kB	Δ	20.2%	11.8%

minimal. When we use a larger number of logic levels, the code space for short codes is already large enough to insure the unique addressing with random decoders. Thus, the optimal code length becomes smaller for larger n's.

The same simulation described previously was performed with a higher variability. The decrease in effective bit area does not happen in a uniform way for all codes. For instance, the optimal binary reflexive code has $M = 18$ for low σ (30 mV), whereas the optimum is shifted toward $M = 16$ for the same code when σ becomes large (300 mV). Designing the circuit with an overestimated reliability would result in about 10% larger effective bit area. This result shows that optimizing the choice of the decoder strongly depends on the estimated reliability level.

C. Summary

The benefits of using MVL to design bottom–up decoders is summarized in Table I. Among the decoders presented in Section II-B, the radial decoder would need several oxide shell thicknesses, and the random contact decoder would need more than one level of conduction in order to be extended to n-ary logic. These features are not inherent to the decoders as presented in [11] and [26]; thus, they cannot be extended to multilevel logic. On the contrary, it is possible to assume more than two levels of doping for the axial decoder and more than one oxide thickness for the mask-based decoder in order to perform an MVL addressing without altering the underlying decoding paradigm. Consequently, only these two decoders were extended to MVL addressing. The bottom–up approaches promise a high effective density under technological assumptions that are still to be validated. The use of ternary logic in 32-kB raw area memories saves area up to 20.2% for memories with axial decoder and up to 11.8% for memories with mask-based decoder.

TABLE II
YIELD OF DIFFERENT TOP–DOWN DECODERS IN TERMS OF AREA PER WORKING BIT (nm^2) AT THE TECHNOLOGY NODE OF 45 nm

Raw Size	Base	GAA Decoder	DRAM	Flash
8 kB	2	9055	17690	9269
8 kB	3	8651	17690	9269
8 kB	Δ	4.5%	N/A	N/A
32 kB	2	8153	16391	8355
32 kB	3	7870	16391	8355
32 kB	Δ	3.5%	N/A	N/A

On the other hand, memories using GAA decoders have a lithography-dependent pitch; and they consequently compete with the lithography-based memories. We compared the yield of GAA decoder-based crossbar memories with standard DRAM and Flash technologies at the technology node of 45 nm, while we assumed that the variation of V_T is $\sigma \sim 1/A$, where A represents the channel area. For the NW technology, A is given by the NW radius and the transistor length, whereas for DRAM and Flash technologies, it is given by the transistor length and width. We assumed that, for these standard technologies, $\sigma \approx 60$ mV at the technology node of 90 nm.

The crossbar architecture is the main reason for the smaller bit area when compared to DRAM or Flash memories. The larger the raw memory density, the better is the yield of the crossbar memory with GAA decoder (see Table II). For high raw memory density, the ultimate limit is defined by the square of the minimal features (poly pitch) which is almost the same limit for Flash memories. The crossbar memory is denser than DRAM memory for all future technology nodes. Compared to Flash memory, it is denser for all nodes until 22 nm; then, both technologies converge to the ultimate limit defined previously. However, using a ternary logic makes crossbar arrays more competitive even after the 22-nm node.

XI. CONCLUSION

This paper studied the design of decoders based on MVL for crossbar memories. The investigation started from the GAA SiNW technology; then, it was extended to other NWs and decoder technologies. This paper presented a theoretical formalism of multivalued addressing, which enabled the generalization of binary hot and reflexive codes to MVL. We modeled the impact of variability by the digits flipping up and down. Then, we investigated the types of errors that happen when multidigit errors are considered. Depending on the error type, the size of the addressable code space was reduced by a certain fraction that we calculated. Since the size of the addressable code space is equal to the number of NWs that can be used in the memory array, our defect models enabled the estimation of the effective memory density under high variability. The performed simulations showed that hot decoders with multiple V_T yield an area saving up to 24%. For highly unreliable technologies, reflexive codes guarantee that at least 10% of the NWs recover. We proved that the accurate estimation of the variability improves the estimation of the optimal decoder design for bottom–up technologies. Generally, the area saving for all considered technologies was shown to be between 11% and 24% for ternary instead of binary logic.

REFERENCES

[1] *International Technology Roadmap for Semiconductors (ITRS) 2006*, Tech. Rep., 2006. [Online]. Available: www.itrs.net/reports.html

[2] G. Anderson and D. A. Metze, "Design of totally self-checking check circuits for m-out-of-n codes," in *Proc. Int. Symp. Fault-Tolerant Comput.*, Jun. 27–30, 1995, pp. 244–248.

[3] R. Beckman *et al.*, "Bridging dimensions: Demultiplexing ultrahigh-density nanowire circuits," *Science*, vol. 310, no. 5747, pp. 465–468, Oct. 2005.

[4] M. H. Ben Jamaa *et al.*, "Fault-tolerant multi-level logic decoder for nanoscale crossbar memory arrays," in *Proc. ICCAD*, Nov. 4–8, 2007, pp. 765–772.

[5] M. Butts *et al.*, "Molecular electronics: Devices, systems and tools for gigagate, gigabit chips," in *Proc. ICCAD*, 2002, pp. 433–440.

[6] G. F. Cerofolini, "Realistic limits to computation. II. The technological side," *Appl. Phys. A, Solids Surf.*, vol. 86, no. 1, pp. 31–42, Jan. 2007.

[7] K. W. Current, "Current-mode CMOS multiple-valued logic circuits," *IEEE J. Solid-State Circuits*, vol. 29, no. 2, pp. 95–107, Feb. 1994.

[8] A. DeHon, "Design of programmable interconnect for sublithographic programmable logic arrays," in *Proc. Int. Symp. FPGA*, 2005, pp. 127–137.

[9] A. DeHon *et al.*, "Stochastic assembly of sublithographic nanoscale interfaces," *IEEE Trans. Nanotechnol.*, vol. 2, no. 3, pp. 165–174, Sep. 2003.

[10] C. M. Files and M. A. Perkowski, "New multivalued functional decomposition algorithms based on MDDs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 19, no. 9, pp. 1081–1086, Sep. 2000.

[11] T. Hogg *et al.*, "Assembling nanoscale circuits with randomized connections," *IEEE Trans. Nanotechnol.*, vol. 5, no. 2, pp. 110–122, Mar. 2006.

[12] J. D. Holmes, "Control of thickness and orientation of solution-grown silicon nanowires," *Science*, vol. 287, no. 5457, pp. 1471–1473, Feb. 2000.

[13] Y. Huang, "Logic gates and computation from assembled nanowire building blocks," *Science*, vol. 249, no. 5545, pp. 1313–1317, Nov. 2001.

[14] Y. Jiang and R. K. Brayton, "Don't cares and multi-valued logic network minimization," in *Proc. ICCAD*, 2000, pp. 520–525.

[15] D. L. Kencke *et al.*, "A multilevel approach toward quadrupling the density of flash memory," *IEEE Electron Device Lett.*, vol. 19, no. 3, pp. 86–88, Mar. 1998.

[16] Y. Luoer *et al.*, "Two-dimensional molecular electronics circuits," *ChemPhysChem*, vol. 3, no. 6, pp. 519–525, Jun. 2002.

[17] S. Mahapatra and A. M. Ionescu, "Realization of multiple valued logic and memory by hybrid SETMOS architecture," *IEEE Trans. Nanotechnol.*, vol. 4, no. 6, pp. 705–714, Nov. 2005.

[18] N. A. Melosh, "Ultrahigh-density nanowire lattices and circuits," *Science*, vol. 300, no. 5616, pp. 112–115, Apr. 2003.

[19] D. M. Miller, "Multiple-valued logic design tools," in *Proc. Int. Symp. MVL*, May 24–27, 1993, pp. 2–11.

[20] K. E. Moselund, "Cointegration of gate-all-around MOSFETs and local silicon-on-insulator optical waveguides on bulk silicon," *IEEE Trans. Nanotechnol.*, vol. 6, no. 1, pp. 118–125, Jan. 2007.

[21] K. Ogawa *et al.*, "Multiple-input neuron MOS operational amplifier for voltage-mode multivalued full adders," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, vol. 45, no. 9, pp. 1307–1311, Sep. 1998.

[22] E. Rachlin, *Robust Nanowire Decoding*, 2006. [Online]. Available: www.cs.brown.edu/publications/theses/masters/2006/eerac.pdf

[23] R. L. Rudell and A. Sangiovanni-Vincentelli, "Multiple-valued minimization for PLA optimization," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. CAD-6, no. 5, pp. 727–750, Sep. 1987.

[24] T. Sasao, "On the optimal design of multiple-valued PLAs," *IEEE Trans. Comput.*, vol. 38, no. 4, pp. 582–592, Apr. 1989.

[25] T. Sasao, "EXMIN2: A simplification algorithm for exclusive-or-sum-of-products expressions for multiple-valued-input two-valued-output functions," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 12, no. 5, pp. 621–632, May 1993.

[26] J. E. Savage *et al.*, "Radial addressing of nanowires," *ACM J. Emerging Technol. Comput. Syst.*, vol. 2, no. 2, pp. 129–154, Apr. 2006.

[27] K. C. Smith, "A Multiple valued logic: A tutorial and appreciation," *Computer*, vol. 21, no. 4, pp. 17–27, Apr. 1988.

[28] K. C. Smith, "The prospects for multivalued logic: A technology and applications view," *IEEE Trans. Comput.*, vol. C-30, no. 9, pp. 619–634, Sep. 1981.

[29] N. Song and M. A. Perkowski, "Minimization of exclusive sum-of-products expressions for multiple-valued input, incompletely specified functions," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 15, no. 4, pp. 385–395, Apr. 1996.

Mohamed Haykel Ben Jamaa (S'08) received the co-joint B.S. and M.S. degrees in electrical engineering from Technische Universität München, München, Germany, and Ecole Centrale Paris, Châtenay-Malabry, France, and has been working toward the Ph.D. degree at Ecole Polytechnique Fédérale de Lausanne, Lausanne, Switzerland, since 2005.

He is currently working on the design aspects for nanoelectronics with a focus on hybrid systems. His work covers the manufacturing aspects of regular circuits as well as reliable system design and architecture.

Kirsten Emile Moselund (S'02) was born in Denmark, in 1976. She received the M.Sc. degree in engineering from the Technical University of Denmark, Lyngby, Denmark, in 2003 and the Ph.D. degree from Ecole Polytecnique Fédérale de Lausanne, Lausanne, Switzerland, in 2008.

She is currently with IBM Rüschlikon, Switzerland. Her research interests include silicon optoelectronics, advanced transistors, nanowire devices, and microelectronic fabrication technology.

David Atienza (M'05) received the M.Sc. degree in computer science from Complutense University of Madrid (UCM), Madrid, Spain, in 2001 and the Ph.D. degree in computer science from IMEC, Leuven, Belgium, in 2005.

He is currently an Associate Professor with UCM and a Senior Research Associate with Ecole Polytechnique Fédérale de Lausanne, Lausanne, Switzerland. He is the coauthor of more than 90 publications in these areas. He is an Associate Editor of the *Elsevier Integration: The VLSI Journal*. His research interests focus on design methodologies for integrated systems, including thermal management techniques for multiprocessor systems-on-chip, novel nanoscale architectures for logic and memories, dynamic memory and memory hierarchy optimizations, and network-on-a-chip design.

Dr. Atienza is an Associate Editor of IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS. He has been a member of the Executive Committee of IEEE Council on Electronic Design Automation since 2008.

Didier Bouvet was born in Evian, France, in 1968. He received the M.S. degree in microelectronic engineering from the University of Lyon, Lyon, France, in 1991 and the Ph.D. degree in applied physics from the Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland, in 1997.

He was a Process Engineer for the start-up of ATMEL's 8-in fab at Rousset, France, in 1996. Since 1998, he has been with the EPFL, where he was a Research Associate. His present research interests include design, fabrication, and characterization of submicrometer MOS devices and the development of slurries for chemical–mechanical polishing applications.

Adrian Mihai Ionescu (S'91–M'93–SM'06) received the B.S. and M.S. degrees in microelectronics from the University Politehnica of Bucharest, Bucharest, Romania, in 1989, and the Ph.D. degree in physics of semiconductors from the Institut National Polytechnique de Grenoble, Grenoble, France, in 1997.

He has held positions with the Electronics and Information Technology Laboratory, Commissariat à l'Energie Atomique, Grenoble, and with the Centre National de la Recherche Scientifique, France, and he was a Visiting Researcher at the Center of Integrated Systems, Stanford University, Stanford, CA. He is currently an Associate Professor with Ecole Polytechnique Fédérale de Lausanne, Lausanne, Switzerland. His present research interests include the design, modeling, and characterization of submicrometer MOS devices, single electron devices and few electron circuit architectures, SOI novel applications, and RF MEMS. He has (co)authored around 100 research papers.

Yusuf Leblebici (M'90–SM'98) received the Ph.D. degree in electrical and computer engineering from the University of Illinois, Urbana–Champaign (UIUC), in 1990.

He has held positions with the UIUC, Istanbul Technical University, Istanbul, Turkey, and Worcester Polytechnic Institute, Worcester, MA. He was a Director of the New England Center for Analog and Mixed-Signal IC Design. Since 2002, he has been a Chair Professor with Ecole Polytechnique Fédérale de Lausanne, Lausanne, Switzerland. His research interests include the design of high-speed CMOS digital and mixed-signal integrated circuits, computer-aided design of very large scale integration (VLSI) systems, intelligent sensor interfaces, modeling and simulation of semiconductor devices, and VLSI reliability issues.

Giovanni De Micheli (S'79–M'79–SM'80–F'94).

He is a Professor and Director of the Institute of Electrical Engineering and the Integrated Systems Centre, Ecole Polytechnique Fédérale de Lausanne, Lausanne, Switzerland. He also chairs the Scientific Committee, Centre Suisse d'Electronique et de Microtechnique SA, Neuchâtel, Switzerland. His research interests include design technologies for integrated circuits and systems, such as synthesis, hardware/software codesign, low-power design, as well as systems on heterogeneous platforms.

Prof. De Micheli is the recipient of the 2003 IEEE Emanuel Piore Award. He is a Fellow of the Association for Computing Machinery. He was Editor-in-Chief of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS/ICAS from 1987–2001. He is the recipient of the 1987 D. Pederson Award for the best paper on the IEEE TCAD/ICAS and the Golden Jubilee Medal for outstanding contributions to the IEEE CAS Society in 2000. He was the President of the IEEE Circuits and Systems Society in 2003, the Cofounder and President Elect of the IEEE Council on Electronic Design and Automation from 2005–2007, and is currently the Division 1 Director (2008–2009).