Evaluation of Design Strategies for Stochastically Assembled Nanoarray Memories

BENJAMIN GOJMAN, ERIC RACHLIN, and JOHN E. SAVAGE
Brown University

A key challenge facing nanotechnologies is learning to control uncertainty introduced by stochastic self-assembly. In this article, we explore architectural and manufacturing strategies to cope with this uncertainty when assembling nanoarrays, crossbars composed of two orthogonal sets of parallel nanowires (NWs) that are differentiated at their time of manufacture. NW deposition is a stochastic process and the NW encodings present in an array cannot be known in advance. We explore the reliable construction of memories from stochastically assembled arrays. This is accomplished by describing several families of NW encodings and developing strategies to map external binary addresses onto internal NW encodings using programmable circuitry. We explore a variety of different mapping strategies and develop probabilistic methods of analysis. This is the first article that makes clear the wide range of choices that are available.

Categories and Subject Descriptors: B.3.m [**Memory Structures**]: Miscellaneous; C.5.m [**Computer Systems Implementation**]: Miscellaneous

General Terms: Design, Performance

Additional Key Words and Phrases: Addressing schemes, coupon collector problem, nanotechnology, nanowire crossbars

1. INTRODUCTION

Anticipating the end of Moore's Law for photolithography, scientists have sought new methods of constructing memories and logic circuits. They have developed methods for growing nanometer-sized nanowires (NWs) and carbon nanotubes [Dekker 1999; Cui et al. 2001; Morales and Lieber 1998; Melosh et al. 2003; Johnston-Halperin et al. 2004]. They have also exhibited methods to assemble NWs into *nanoarrays*, crossbars consisting of two orthogonal

This research was funded in part by National Science Foundation Grants CCR-0210225 and CF-0403674. An abbreviated version of this article appears in the Proceedings of the 2004 International Symposium on VLSI.
Authors' address: Department of Computer Science, Brown University, 115 Waterman Street, Providence, RI 02912-1910; email: jes@cs.brown.edu.

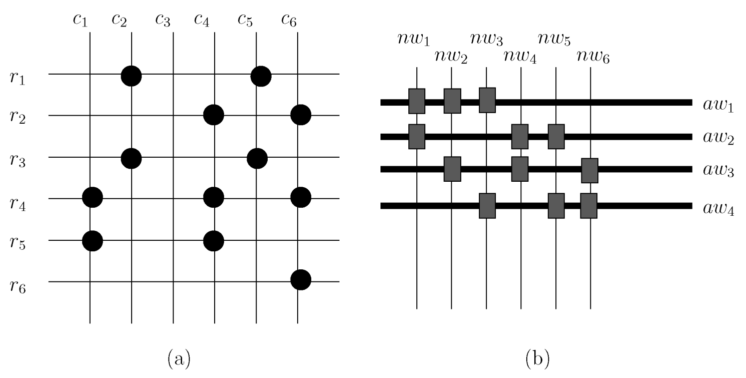

Fig. 1. (a) A crossbar has two sets of orthogonal wires. Binary data is stored at crosspoints defined by pairs of intersecting wires. (Dots are associated with 1s.) (b) (2,4)-hot addressing of six NWs $\{nw_1, \ldots, nw_6\}$ with four mesoscale address wires $\{mw_1, \ldots, mw_4\}$. In practice, the MWs would be much thicker than the NWs. Dark rectangles identify a lightly doped (controllable) region of a vertical NW. Other regions of NWs are uncontrollable. MWs either carry a high electric field or no field. A NW is nonconducting only if the MWs carrying high fields lie across its uncontrollable regions. If exactly two of the four MWs carry a high electric field, exactly one of the six NWs is conducting.

sets of parallel wires separated by a molecular layer [Kuekes et al. 2000; Chen et al. 2003; Melosh et al. 2003; Whang et al. 2003; Zhong et al. 2003; Huang et al. 2001; Kim et al. 2001; Wu et al. 2002]. (See Figure 1(a).) Within this layer, molecules at the crosspoints of pairs of orthogonal NWs can switch their conductivity under the application of large positive and negative electric fields [Collier et al. 2000; Collier et al. 1999; Rueckes et al. 2000; Duan et al. 2002]. The conductivity at crosspoints can be sensed without changing it by the application of smaller electric fields.

It is proposed that nanocrossbars be used as very high density memories and programmed logic arrays (PLAs) [DeHon 2003; DeHon et al. 2003; Gojman et al. 2004; Rachlin et al. 2005; DeHon et al. 2005]. A prototype 8×8 crossbar with a density of 6.4 Gbits/cm^2 has been announced that is based on these technologies [Chen et al. 2003] and a memory with storage capacity of 10 Gbits based on cross-bars of NTs is promised (see http://www.nantero.com). DeHon et al. [2005] estimate that a memory density exceeding 10^{11} bits/cm^2 is possible.

Two types of NW have been proposed for use in crossbars, differentiated (or encoded) NWs and undifferentiated (or uniform) NWs. In this article, we consider crossbars constructed with differentiated NWs. NWs are so small that the process of quickly assembling large numbers of them into an array is stochastic. As a consequence, it isn't possible to predict with certainty which NW types will fall onto a chip or whether the number of distinct addresses available will meet stated minimal requirements. Instead, design parameters are chosen so that these requirements are met with a certain minimal probability.

In this article, we present a broad study of the stochastic assembly of nanoarray memories constructed with differentiated NWs. Unlike DeHon et al. [2003, 2005], we don't examine just the area occupied by the crosspoints and address wires, we also examine the area of translation memories used to map external binary addresses onto internal NW encodings. Also, we don't restrict attention to just one strategy to perform this mapping but consider four representative strategies. We do this because the total area occupied by a nanoarray is affected by the address mapping strategy.

The "all different" address mapping strategy examined in DeHon et al. [2003, 2005] requires that, with high probability, all NWs have different addresses. We show that relaxing this requirement so that half of them are different (the "most different" strategy) dramatically reduces the number of different NW types needed to ensure success with high probability and uses less total chip area. We also introduce the "all present" strategy in which every NW type is present and show that it doesn't require a translation memory and has the potential to use less area than DRAMs. However, it is inferior to the "most different" strategy. Finally, we present the "repeated codeword" strategy in which each codeword appears many times and show that it can succeed using a very small number of NW types. This will be an important consideration in the manufacture of nanoarrays.

We also introduce a new type of NW code called the "binary reflected codes," examine the use of address wildcards in storing data and codeword discovery, and give nearly optimal algorithms for the latter problem.

Even though the assembly of nanoarrays is stochastic, there is a wealth of design opportunities to be considered. This is the first work that attempts to explore the wide range of design possibilities for individual nanoarrays in a comprehensive fashion.

1.1 Crossbar Nanoarrays

In nanoarrays, data is stored using the state of conductivity of the molecules at crosspoints. The standard method of reading and writing data in nanoarrays involves turning off (reducing the conductance of) all but one NW in each dimension through the application of electric fields to mesoscale wires (MWs) that are orthogonal to the NWs. There are many ways to construct NWs so they will operate in this way. One way is illustrated in Figure 1(b) by NWs that have two lightly doped (controllable) regions under two of four MWs. When an electric field is applied by a MW to a controllable region of a NW, the NW conductance is greatly reduced, implementing a field effect transistor (FET). When electric fields are applied to two of the four MWs in Figure 1(b), exactly one NW remains conducting.

To write data at a crosspoint, a high positive or negative electric field is applied across it by applying fields across the two NWs defining the crosspoint. (See Figure 2.) This is accomplished by activating the subset of the MWs that make only the two NWs conducting, grounding one end of both NWs, and applying voltages at their other ends. The application of a large voltage across a crosspoint causes the molecules at the crosspoint to become conducting or nonconducting. That is, it sets the state of a switch.

To read data at a crosspoint, the NWs defining the crosspoint are again made conducting, their grounded ends are disconnected (see Figure 2), and a voltage

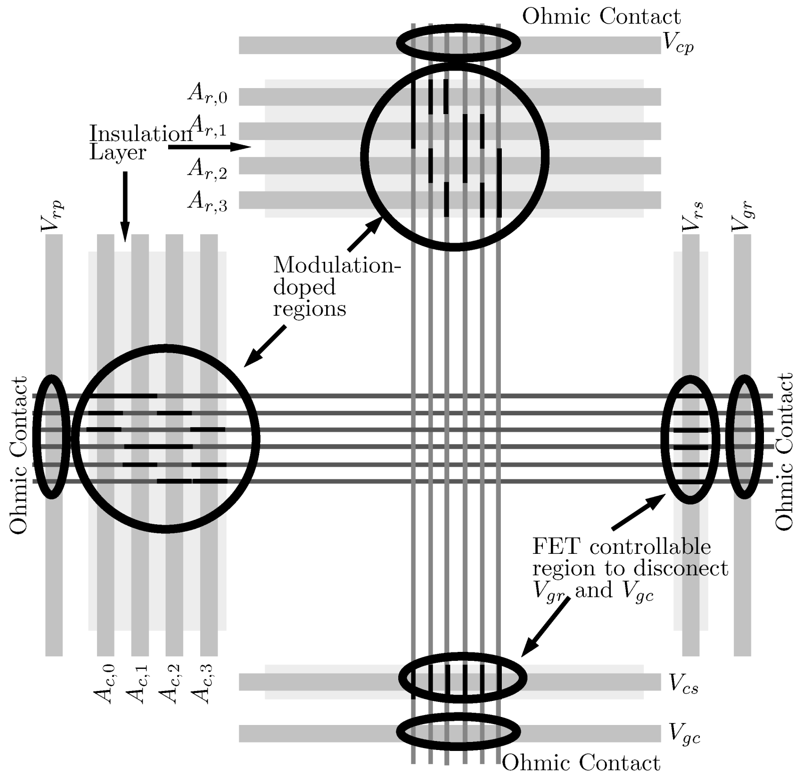

Fig. 2. A nanoarray in which horizontal and vertical NWs are each controlled by a set of mesoscale address wires (MWs), labeled $A_{r,i}$ and $A_{c,j}$, that are separated by an insulation layer from NWs. The intersection of lightly doped (dark) NW regions with MWs defines FETs. Ohmic contacts are made with both ends of each set of NWs. A lightly doped region at one end of each set of NWs allows voltages V_{gr} and V_{gc} to disconnect that end from its ohmic contact. Data is stored in the state of conductivity of a molecular layer for reading of the status of crosspoint switches.

is applied to one NW while current is sensed on the other. Current will flow from one NW to the other only if the switch between the two NWs is closed.

The crossbar shown in Figure 2 has one ohmic contact at each end of each set of NWs. In practice, multiple ohmic regions will be used at one end of each set of NWs. This reduces the number of different NW types that need to be manufactured because the same codeword can be activated independently in different ohmic regions.

1.2 Nanowire Decoders

A *decoder* is an arrangement of MWs and NWs such that activation of a subset of the MWs causes one NW (or a small number of NWs) to remain conducting. An example of a decoder is given in Figure 1(b). Three methods of realizing decoders have been proposed. The first two assume that NWs are undifferentiated. That is, when grown, the NWs are all the same. The third method, which is the subject of this article, assumes that NWs are differentiated during their manufacture.

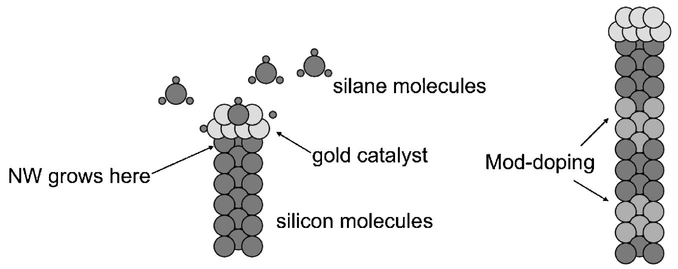

Fig. 3. (a) NWs grown through a vapor-liquid-solid process. (b) NWs are doped as they grow.

Although a decoder could be realized by attaching a single MW to each NW, to do so would lose the advantage of nanometer spacing. For that reason, each of the decoders being proposed for nanoarrays utilizes a few MWs to control many NWs.

Williams and Kuekes [2001] describe the *randomized contact decoder* based on the random deposition of gold nanoparticles between MWs and undifferentiated NWs that results in contacts at about half of the intersections. Heath et al. [2005] and Heath and Ratner [2003] describe the *mask-based decoder* in which a collection of high-K and low-K dielectric regions are interposed between NWs and MWs. Because the size of such regions is determined by the low precision of lithography, many of the regions are randomly positioned to achieve NW differentiation with high probability. This decoder has been analyzed by Rachlin et al. [2005]. Finally, DeHon et al. [2003] describe and analyze one type of *differentiated NW decoder* for axially-encoded NWs (see Section 2). (The decoder shown in Figure 1(b) is of this type.) A similar type of decoder has been developed for radially encoded NWs [Savage et al. 2005] in which NWs are encoded using shells of multiple types [Lauhon et al. 2002]. Decoders for differentiated NWs are examined in Section 5.1.

1.3 Nanowire Differentiation

Two methods have been devised to differentiate NWs. The first method is known as *modulation doping* when referring to the doping process and *axial doping* when referring to the result. Modulation doping encodes NWs along their axial dimension [Huang et al. 2001; Kim et al. 2001; Wu et al. 2002].

Axially-encoded NWs can be grown from seed catalysts through a vapor-liquid-solid (VLS) process depicted in Figure 3 [Gudiksen et al. 2002; Wu et al. 2002; Björk et al. 2002]. In the example, silane molecules (SiH_4) fall onto gold clusters, precipitating out Si atoms that solidify into crystalline silicon NWs. These NWs can be differentiated by adding dopant molecules to the gaseous mixture as they grow. NWs can be heavily and lightly doped over lengths that are determined by exposure time.

Under modulation doping, each NW is given a pattern of controllable and uncontrollable regions, each approximately the same length. (A controllable region is generally a few nanometers longer than an uncontrollable region [DeHon et al. 2003].) For example, two of four regions could be made controllable as suggested in Figure 1(b) where all six different doping patterns are shown.

In Section 2, two types of axial code are described, the *(h,b)-hot codes* [DeHon 2003] and *binary reflected codes (BRCs)*, a new type of code introduced here. Because BRCs are based on binary sequences and their complements, they simplify the decoding process and permit operations that use binary sequences.

The second method of differentiating NWs is called *radial encoding* [Savage et al. 2005]. NWs are encoded by giving them multiple layers of shells of different types. Core-shell NWs [Lauhon et al. 2002] are produced by first growing a NW core that is lightly doped and then depositing shells on the core epitaxially. The shells are made of materials that can be separately etched [Savage et al. 2005]. *Hybrid encoding* of NWs is also possible. In this case, NW cores are given an axial encoding and shells have a radial encoding. Unfortunately, hybrid codes are inferior in the use of area to both axial and radial codes [Savage et al. 2005].

1.4 Nanowire Placement

NWs are too small and numerous to place individually on a chip. Instead, several methods have been proposed to place many NWs on a chip simultaneously.

Undifferentiated NWs have been grown outside a chip and then deposited in groups [Melosh et al. 2003]. They have also been imprinted directly onto a chip using stamps with nanometer dimensions [Chou et al. 1996; Xia and Whitesides 1998; Chen et al. 2003]. These methods of placing NWs are largely deterministic, although the methods of differentiating them are stochastic.

This article is concerned with differentiated NWs and explores the consequences of the stochastic assembly process used to place them on a chip. Many NWs with a given codeword are grown at one time and mixed in a solution. The solution contains many copies of each type of NW codeword. A random subset of them is deposited on a chip by a process that aligns the NWs along an axis [Whang et al. 2003; Huang et al. 2001]. It is not possible to control which NWs fall onto a chip nor can their locations be predicted in advance. Furthermore, because NWs can shift along their axial dimension, their controllable and uncontrollable regions may not be perfectly aligned with MWs.

We assume that the NW mixture has so many copies of each NW that drawing a small number of NWs doesn't change the distribution of NW types. This is a very safe assumption since withdrawals made without replacement only increase the likelihood that the NWs selected have different types. Thus, we assume that a) NWs are drawn with replacement, and b) the probability of drawing a NW is proportional to its representation in the population.

To cope with the axial displacement that NWs experience in the fluidic assembly process, an axial doping pattern is repeated along the length of each NW as proposed in DeHon et al. [2003]. The question then arises as to whether a shifted version of one NW encoding (the shift is assumed to be by multiples of a MW pitch) generates another useable NW encoding. This problem is examined in Section 5.1.

When a NW has an axial offset of its controllable and uncontrollable NW regions with respect to MWs, the offset will be either by the pitch of MWs, or by such an amount plus a fraction of a NW pitch. (See Figure 4.) If the offset

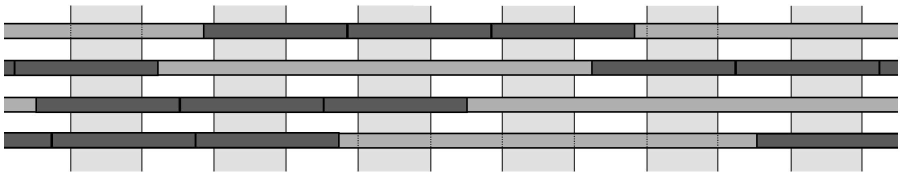

Fig. 4. An example of a NW encoding producing multiple codewords under shifts. All the shifts are close to a multiple of a MW pitch.

is close to a multiple of the NW pitch, the NW will operate as if it the offset is an exact multiple of a NW pitch. In this article, we only consider this case. The other case is examined in DeHon et al. [2003]. The analysis given there applies here as well.

1.5 Nanowire Discovery

Because the nanoarray manufacturing process for differentiated NWs is stochastic, it is not possible to know ahead of time which NWs will be present in the array. As a consequence, a discovery process is needed after chip manufacture to determine which NW types are on the chip. An auxiliary memory must then be programmed to map external addresses to the NW types that are present. This topic is examined in Section 5.1.

To help with the discovery process, it may be useful to employ wildcarding to access NW encodings. In a binary address, a wildcard bit means that both addresses specified by using both values of the bit are accessed. If multiple wildcard bits are used, all addresses that are consistent with the non-wildcard bits are accessed. This topic is explored in Section 8.

1.6 Article Outline

In Section 2, we introduce binary reflected codes, a new type of axial code that is an alternative to (h, b)-hot codes [DeHon et al. 2003]. It is constructed from binary sequences and their complements. In Section 3, we examine the periodicity of codewords and explain the importance of periodicity for efficiently manufacturing NWs.

In Section 4, we summarize bounds on the tails of probability distributions derived in the appendix so that this information is available for analysis in later sections. In Section 5, we describe four addressing strategies, that is, ways of mapping binary sequences to NWs. The first of the four addressing strategies is the all different strategy studied in DeHon et al. [2003]. The three remaining strategies are original to this article. One of these, the repeated codeword strategy, is particularly noteworthy in that it requires only a very small number of NW encodings. We also suggest additional strategies for further study.

In Section 7, we compare the four strategies based on the area requirements described in Section 6. We find that the most different and repeated codeword strategies are superior to the other two. We also show that if NW pitches of about 10nm are possible, the all present strategy provides a higher area density than is expected from DRAMs in the near future.

The use of wildcards is explored in Section 8. Wildcards can be useful in storing data in arrays as well as in codeword discovery. We describe an efficient hardware implementation of wildcarding.

For most design strategies, only certain codewords are present in each ohmic region. Section 9 gives efficient algorithms for codeword discovery that apply to all all addressing strategies. Conclusions are presented in Section 10.

2. ENCODED NANOWIRES

There are two ways to differentiate NWs during their manufacture. Their cores can be given a nonuniform encoding (axial codes), or shells of various types can be added to a uniform core (radial codes), [Savage et al. 2005]. Hybrid axial/radial codes are also possible but they offer no advantages over either pure axial or radial encodings [Savage et al. 2005].

Following we describe two types of axial codes, (h, b)-hot and binary reflected codes. Binary reflected codes are introduced here.

When differentiated NWs are placed on a chip using fluidic methods, NWs are equally likely to be in any position relative to MWs. If such NWs don't make contact with ohmic regions at each end, they are disabled. If they do make contact, the codewords that they represent depend on their alignment with respect to MWs.

In Section 3, we ask how to generate ensembles of NWs with axial encoding so that, when NWs are drawn with equal probability and replacement and placed on a chip subject to random axial displacement by multiples of a MW pitch, all codewords in the code are equally likely to appear. This condition offers the best chance of having d distinct codewords for any d.

2.1 $(\boldsymbol{h}, \boldsymbol{b})$-Hot Codes

The (h, b)-hot code [DeHon 2003] is one in which h of b regions is lightly doped and controllable and the $(b - h)$ remaining regions are heavily doped and uncontrollable. There are $C = \binom{b}{h}$ such doping patterns or codewords. A NW is nonconducting if a MW carrying a high electric field is adjacent to one of its controllable regions. A NW is conducting when each MW carrying a high electric field is adjacent to one of its $b - h$ uncontrollable regions. $M = b$ MWs are necessary to control (h, b)-hot codewords.

It is easy to show (see Lemma B.1) that $C = \binom{b}{h}$ implies that $b \geq (\log_2 C)/H(h/b)$. Here $H(x) = -x \log_2 x - (1 - x) \log_2(1 - x)$ is the binary entropy function. Thus, $M = b$ is proportional to $\log_2 C$ when h is a fixed fraction of b.

To cope with NW axial displacements, a (h, b)-hot codeword is repeated along its axial dimension. As a consequence, a NW axial displacement by a multiple of a MW pitch is equivalent to a cyclic shift of the NW codeword. An (h, b)-hot codeword is mapped to another such codeword when it undergoes a cyclic shift since the number of 1s is unchanged.

2.2 Binary Nested Codes

(h, b)-hot codeword are closed under cyclic shift. Binary nested codes are examples of codes that are not closed. In such codes, each codeword is formed from a length-n binary number by replacing each 0 by 01, and 1 by 10, and then letting 1 denote a lightly doped region and 0 a heavily doped region.

To see that these codes are not closed under shifting, consider the mapping of the binary number 101, namely, 100110. When shifted cyclically right by one place, the codeword becomes 010011 which is not a valid codeword.

2.3 Binary Reflected Code

The *binary reflected code (BRC)* consists of codewords of the form $\boldsymbol{z} = \boldsymbol{x}\bar{\boldsymbol{x}}$ where $\boldsymbol{x} = (x_{k-1}, \ldots, x_1, x_0)$ is a binary tuple, $\bar{\boldsymbol{x}}$ is its Boolean complement, and 1 and 0 denote lightly and heavily doped regions, respectively. The number of codewords in a BRC is $C = 2^k$. The number M of MWs needed to address BRCs is $M = 2k = 2\log_2 C$.

Codewords in the BRC represent a permutation of codewords in a binary nested code that results in their being closed under shifting. To see this, observe that codewords in BRCs have the property that bits that are separated by k positions are Boolean complements of one another. If a BRC codeword is shifted left by one place, the first k bits are $x_{k-2}, \ldots, x_1, x_0, \bar{x}_{k-1}$, and the second k bits are $\bar{x}_{k-2}, \ldots, \bar{x}_1, \bar{x}_0, x_{k-1}$ for which the property clearly holds. Since the code space is preserved by one left cyclic shift, it is preserved by arbitrarily many such shifts. Since the results of right cyclic shifts are included among the left cyclic shifts, the property holds.

3. GENERATING RANDOM ENSEMBLES OF AXIAL CODEWORDS

The codeword that appears on a particular NW depends on its encoding and its axial displacement when placed on a chip. The goal is to create an ensemble of axially-encoded NWs such that each codeword is equally likely to appear in the nanoarray. Axial displacement causes encodings to shift cyclically, allowing a single encoding to produce multiple codewords. This observation allows for a reduction in the number of encodings which must be manufactured. In order to correctly determine how many NWs to produce with each encoding, the periodicity of the encoding must be taken into account. The details of this procedure are discussed in the following.

3.1 Codeword Equivalence Classes

Consider a codeword, $\boldsymbol{z}$, which has length l. Let $R^t(\boldsymbol{z})$ denote the cyclic shift of $\boldsymbol{z}$ by t places, and let $H(\boldsymbol{z}) = \{\boldsymbol{z}, R(\boldsymbol{z}), \ldots, R^l(\boldsymbol{z})\}$ be the set of codewords that result from all possible cyclic shifts of $\boldsymbol{z}$. Let $p(\boldsymbol{z}) = |H(\boldsymbol{z})|$ be the number of distinct codewords in $H(\boldsymbol{z})$. Observe that $p(\boldsymbol{z})$ is the *period* of $\boldsymbol{z}$.

Let $\boldsymbol{z}$ and $\boldsymbol{w}$ be two codewords. Then either $H(\boldsymbol{z}) = H(\boldsymbol{w})$ or $H(\boldsymbol{z}) \cap H(\boldsymbol{w}) = \emptyset$, the empty set. We call $H(\boldsymbol{z})$ the *equivalence class containing* $\boldsymbol{z}$. Any member of an equivalence class is called a *seed* because, under cyclic shifts, it generates each member of the class. If one seed can generate all members of a large set, this greatly reduces manufacturing costs.

3.2 Codeword Probability

Given a code, suppose that a single seed is selected for each equivalence class, and that K copies of each seed are present in the ensemble from which NWs are selected. If $p(\boldsymbol{z}) = l$ for all codewords, then each seed will generate exactly l codewords and each codeword will be equally likely.

On the other hand, suppose some seeds have period $\frac{l}{2}$. These seeds will generate only $\frac{l}{2}$ codewords, and, as a result, these codewords will be generated twice as often as those generated from seeds with period l. If the ensemble contains K copies of each seed with period l, it should only contain $\frac{K}{2}$ copies of each seed with period $\frac{l}{2}$.

More generally, if codewords have length l, a seed with period p will generate p codewords and should have multiplicity $\frac{Kp}{l}$, where K is some large constant. This condition ensures that each codeword is equally likely to appear in the nanoarray.

3.3 Binary Reflected Code

We now consider the relationship between p and l for binary reflected codewords.

Example 3.1. When $l = 12$, both *110001001110* and *110011001100* are valid codewords. Though $p(110001001110) = 12$, $p(110011001100) = 4$.

For a given l, the following lemma states which periods are possible.

LEMMA 3.1. *Let $d_2(l)$ be the largest power of 2 dividing l. The period of a binary reflected codeword must divide l, the length of the codeword, but not $\frac{l}{2}$. The set $\mathcal{P}(l)$ of periods of binary reflected codewords of length l satisfies*

$$\mathcal{P}(l) = d_2(l) * \mathcal{D}\left(\frac{2l}{d_2(l)}\right),$$

where $\mathcal{D}(n)$ is the set of the divisors of n, and $*$ denotes multiplication by an integer of every element of a set of integers.

PROOF. Let $\boldsymbol{z} = \boldsymbol{x}\overline{\boldsymbol{x}}$ be a binary reflected codeword of length $l = 2k$. Because $R^l(\boldsymbol{z}) = \boldsymbol{z}$, $p(\boldsymbol{z})$ divides $2k$. If $p(\boldsymbol{z}) = k$, then $\boldsymbol{z} = \boldsymbol{y}\boldsymbol{y}$ for some sequence $\boldsymbol{y}$, but this violates the condition that $\boldsymbol{z} = \boldsymbol{x}\overline{\boldsymbol{x}}$ for some sequence $\boldsymbol{x}$. It follows that $p(\boldsymbol{z})$ divides $2k$, but not k. In other words, $\mathcal{P}(l) \subseteq \mathcal{D}(2k) - \mathcal{D}(k)$.

To see that $\mathcal{P}(l) \supseteq \mathcal{D}(2k) - \mathcal{D}(k)$, observe that if $p \in \mathcal{D}(2k) - \mathcal{D}(k)$, p cannot divide k and thus must go into $2k$ an odd number of times. If $\boldsymbol{w} = \boldsymbol{v}\overline{\boldsymbol{v}}$ is a binary reflected codeword of length p, then $\boldsymbol{w}$ repeated an odd number of times will also be a binary reflected codeword. Repeating $\boldsymbol{w}$ $(2k/p)$ times will result in a codeword of length $2k$ with period p. Thus, $\mathcal{P}(l) = \mathcal{D}(2k) - \mathcal{D}(k)$. Since $\mathcal{D}(2k) = 2 * \mathcal{D}(k) \cup \mathcal{D}(k)$, it follows that $\mathcal{P}(l) = 2 * \mathcal{D}(k) - \mathcal{D}(k)$.

Finally, let $n \in \mathcal{D}(k)$, and thus $2n \in 2 * \mathcal{D}(k)$. If k is odd, $2n \in 2 * \mathcal{D}(k) - \mathcal{D}(k)$. If k is even and the largest power of 2 dividing n is less than $d_2(k)$, $2n \in \mathcal{D}(k)$ and $2n \notin 2 * \mathcal{D}(k) - \mathcal{D}(k)$. Thus $2n \in 2 * \mathcal{D}(k) - \mathcal{D}(k)$ if and only if n is divisible by $d_2(k)$, so $\mathcal{P}(l) = 2\,d_2(k) * \mathcal{D}(k/d_2(k)) = d_2(l) * \mathcal{D}(2l/d_2(l))$ since $2\,d_2(k) = d_2(l)$. □

Example 3.2. Consider codewords $\boldsymbol{z} = \boldsymbol{x}\overline{\boldsymbol{x}}$ of length $l = 36$. Since $d_2(36) = 4$, $\mathcal{P}(18) = 4 * \mathcal{D}(9) = 4 * \{1, 3, 9\} = \{4, 12, 36\}$.

We now determine the number of seeds that have period p for $p \in \mathcal{P}(l)$. Clearly, the smallest period is $d_2(l)$. We show that our method of generating reflected binary codewords significantly reduces the number of seeds needed to generate all codewords.

THEOREM 3.1. *The number of equivalence classes of codewords of length l that have period p but no smaller period, $\nu_k(p)$, satisfies the following relation where $Q(p) = \{q \mid q \in \mathcal{P}(l)$ where q strictly divides $p\}$.*

$$\nu_k(p) = \left(2^{p/2} - \sum_{q \in Q(p)} q\,\nu_k(q)\right)/p$$

The number of equivalence classes is $\sum_{p \in \mathcal{P}(k)} \nu_k(p)$.

PROOF. Since codeword $\boldsymbol{z}$ has period $p(\boldsymbol{z})$, we can write it as $\boldsymbol{z} = \boldsymbol{w}^m$, where $p(\boldsymbol{z}) = |\boldsymbol{w}| = l/m$. If m is even, $\boldsymbol{z} = \boldsymbol{xx}$ for $\boldsymbol{x} = \boldsymbol{w}^{m/2}$, which is of improper form. It follows that m is odd. Thus, $|\boldsymbol{w}| = l/m$ is even. Furthermore, the second half of the middle instance of $\boldsymbol{w}$ falls in the first $|\boldsymbol{w}|/2$ positions of the second half of $\boldsymbol{z}$. Consequently, the second half of $\boldsymbol{w}$ is the complement of its first half. That is, we can write $\boldsymbol{w} = \boldsymbol{y}\bar{\boldsymbol{y}}$, where $|\boldsymbol{y}| = p/2$, thus, $\boldsymbol{w}$ is a BRC codeword of length p.

We now derive an expression for $p\nu_k(p)$ where $\nu_k(p)$ is the number of equivalence classes formed by codewords of period exactly p.

Consider codewords $\boldsymbol{z}$ that have the smallest period $p = d_2(l)$. Then $\boldsymbol{z} = (\boldsymbol{y}\bar{\boldsymbol{y}})^{l/p}$, where $|\boldsymbol{y}| = d_2(l)/2$. There are $2^{d_2(l)/2}$ sequences $\boldsymbol{y}$. They fall into exactly $2^{d_2(l)/2}/d_2(l)$ equivalence classes since $p = d_2(l)$ is the smallest possible period. Thus, $p\nu_k(p) = 2^{p/2}$ for $p = d_2(l)$.

Now let $\boldsymbol{z}$ have period $p > d_2(l)$. If q divides p, every sequence of period q also has period p. Conversely, if a sequence has period p and also a period less than p, then it has a period q such that q strictly divides p. The set $Q(p)$ contains the periods less than p of BRC codewords of length p.

Since there are $q\,\nu_k(q)$ BRC codewords of period exactly q, it follows that there are $\sum_{q \in Q(p)} q\,\nu_k(q)$ codewords of period p and q such that $q < p$. The remaining codewords of period p have period exactly p. Because there are $2^{p/2}$ codewords with period $q \in \mathcal{D}(p)$, the result follows. □

Example 3.3. The periods of the codewords in Example 3.2 are $\{4, 12, 36\}$. The number of different equivalence classes with these periods is $\nu_{18}(4) = 1$, $\nu_{18}(12) = (26 - 4\,\nu_{18}(4))/12 = 5$, and $\nu_{18}(36) = (2^{18} - 4\nu_{18}(4) - 12\nu_{18}(12))/36 = 7,280$. It follows that an ensemble of $\nu_{18}(4) + \nu_{18}(12) + \nu_{18}(36) = 7,286$ codeword seeds will generate all 262, 144 codewords, a reduction by a factor of almost 36 in the number of doped NWs.

3.4 (h, b)-Hot Codes

We can investigate the relationship between p and b for (h, b)- hot codes using a similar approach. The period of a seed in an (h, b)-hot code must divide b, the codeword length. A codeword with period p will have the same sequence of p bits repeated (b/p) times. This implies that h, the number of ones, must be divisible by (b/p). Since $h = q(b/p)$ for some integer q, $p = q(b/h)$. In other words, p is a multiple of (b/h). The requirement on p in terms of h and b is summarized in the following lemma:

LEMMA 3.2. *The set $\mathcal{P}(h, b)$ of periods of $(\boldsymbol{h}, \boldsymbol{b})$-hot codes satisfies*

$$\mathcal{P}(h, b) = \mathcal{D}(b) \cap \mathcal{M}(b/h),$$

where $\mathcal{D}(n)$ is the set of the divisors of n, and $\mathcal{M}(n)$ is the set of multiples of n.

As an application of this lemma, observe that when b and h are relatively prime, $\mathcal{M}(b/h)$ will not include any integers less than b. As a result, $\mathcal{D}(b) \cap \mathcal{M}(b/h) = b$, and all seeds will have period b.

THEOREM 3.2. *The number of equivalence classes of (h, b)-hot codewords that have period p but no smaller period, $\nu_k(p)$, satisfies the following relation where $Q(p) = \{q \mid q \in \mathcal{P}(h, b)$ where q strictly divides $p\}$.*

$$\nu_k(p) = \left(\binom{p}{hp/b} - \sum_{q \in Q(p)} q\,\nu_k(q)\right)/p$$

The number of equivalence classes is $\sum_{p \in \mathcal{P}(k)} \nu_k(p)$.

PROOF. An (h, b)-hot codeword with period p contains (b/p) copies of the same sequence. As noted previously, this sequence must contains (hp/b) ones, thus there are exactly $\binom{p}{hp/b}$ codewords with period p.

Once again, the set $Q(p)$ contains the periods less than p of codewords of length p. Just as with BRCs, there are $q\,\nu_k(q)$ sequences with period exactly q and $\sum_{q \in Q(p)} q\,\nu_k(q)$ codewords of period q strictly less than p.

There are thus $\binom{p}{hp/b} - \sum_{q \in Q(p)} q\,\nu_k(q)$ code words with period p but no smaller period. Each codeword belongs to an equivalence class of size p, so the result follows. □

4. BOUNDS ON PROBABILITY DISTRIBUTIONS

Before doing a quantitative analysis of the four design strategies, we present bounds on probabilities of events that arise in their use. They are derived in the appendix.

LEMMA 4.1. *The probability that each of the w NWs in an ohmic region has a distinct encoding satisfies the following bound.*

$$P_{distinct}(w, C) \leq e^{-w(w-1)/2C} \tag{1}$$

Consequently, $P_{distinct}(w, C) \geq 1 - \delta$ when $C \geq w(w-1)/(-2\ln(1-\delta))$.

This bound is very tight even for the smallest values of w that we consider, that is, $w \geq 10$, as shown in the appendix.

LEMMA 4.2. *When $w \leq 2d$ and $d \geq 4$, the number of different NW encodings, C, needed to have at least d distinct NW encodings occur among w NWs with probability at least $1 - \delta$ must satisfy the following bound.*

$$C \geq (d-1)e^{((d-1)-\ln\delta)/(w-d+1)} \tag{2}$$

Comparison of these bounds with the exact values for the probabilities shows that they provide values of w that are at least 80% of the exact values when $20 \le C \le 200$, $10 \le w \le 100$, and $Q(d, w, C)$ is near .01. (See Figure 7.)

LEMMA 4.3. *The number of NWs in an ohmic region w needed to ensure with probability at least* $1 - \epsilon$ *that each of the C NW encodings occurs in the region at least once* ($d = C$) *satisfies the bound* $w \ge C \ln(C/\epsilon)$.

LEMMA 4.4. *Let x be the sum of n* 0-1 *valued variables with value 1 occurring with probability p. Then,*

$$P_r[x \le \theta np] \le e^{-np(1-\theta+\theta \ln \theta)},$$

where $0 \le \theta < 1$ *and np is the mean value of the sum.*

5. ADDRESSING NANOARRAYS

A nanoarray is useful as a memory or PLA only if its NWs can be addressed using standard binary addresses. Because the NW codewords that fall onto a chip cannot be predicted in advance, a separate programmable *translation memory* is needed to map binary addresses to the internal addresses represented by the MW fields used to activate individual NWs. When a nanoarray is addressed from the mesoscale level, the current situation under investigation, this programmable memory is a mesoscale memory, which we call *MesoMem*, although, in principle, it may also be implemented at a lower level.

To fix ideas and evaluate addressing strategies, we assume that a nanoarray is as a memory, denoted *NanoMem*, and that it has crosspoints, two decoders and two translation memories. We also assume that, in each dimension, NWs are connected to m ohmic regions each containing w NWs. We let M and N denote the number of MWs and NWs in each dimension of the nanoarray. Clearly, $N = mw$. Because not all addressing strategies require that each of the N NWs in each dimension have different internal addresses, we let N_{aw} denote the number of unique addresses in each dimension. Obviously, the NanoMem stores N_{aw}^2 bits. Finally, we let A_{meso} denote the area of a NanoMem. We compare addressing strategies based on the value A_{meso} for a given storage capacity N_{aw}^2.

In the following section, we describe four addressing strategies. These strategies can be used to address either axially-or radially-encoded NWs, that is, NWs that are differentiated before deposition on a chip.

To evaluate a strategy, it is necessary to know the relationship between M, the number of MWs, and the size C of the code space. As shown in Section 2 for (h, b)-hot codes, $M \ge C/H(\eta)$, where $\eta = h/b$, for BRCs, $M = 2\log_2 C$, and for radial codes with τ types in each shell, $M = (\tau/\log_2 \tau)\log_2 C$ [Savage et al. 2005]. Since the value of M is similar for each of these codes, for analytical convenience, we evaluate strategies for the BRCs. That is, we let $M = 2\log_2 C$.

5.1 Addressing Strategies

In DeHon et al. [2003, 2005], it is assumed that the set of axial codes is chosen so that, with high probability, all NWs that fall in an ohmic region have distinct addresses. As mentioned, in DeHon et al. [2003], it was estimated that this would require $C > 100w^2$ different code types to ensure that all w NWs in one ohmic region are different with probability .99. When $w = 1,000$, $C > 10^8$, a totally impractical number. For this reason, we subdivided the ohmic region in each dimension into m ohmic regions for which w is much smaller, say on the order of 10 to 30.

We examine four NW addressing strategies. The first three assume that some of the external binary address bits in each dimension of a nanoarray are used to select one of m ohmic regions with a standard decoder. The remaining bits, which are stored in a translation memory, are used to activate one NW type in an ohmic region. These three strategies are called *all different* (S_{ad}), *most different* (S_{md}), and *all present* (S_{ap}) *codeword strategies*. The fourth strategy, the *repeated codeword strategy* (S_{rc}), uses the low order bits to address NWs using a trivial decoder when BRCs are used but requires a translation memory to select the ohmic region to be activated. It is used to illustrate the point that other nonobvious addressing strategies may exist that may lead to superior performance to the four that are examined here.

5.1.1 *All Different Codeword Strategy.* The all different codeword strategy S_{ad} is the one used in DeHon et al. [2003] and for differentiated NWs in DeHon et al. [2005]. It requires that, with probability at least $1-\epsilon$ within each of the m ohmic regions all w NWs have different doping patterns when the NWs types are drawn with equal probability from an ensemble of C different types.

This strategy has no wasted NWs. The number of addressable NWs is $N_{aw} = mw$ with probability at least $1-\epsilon$. We show in the following that the value of C required for this strategy is at least $N_{aw}(w-1)/(-2\ln(1-\epsilon))$, which is better by a factor of 2 than the bound given in DeHon et al. [2003]. It requires that the chip be examined to discover which NW types are present.

THEOREM 5.1. *Strategy S_{ad} succeeds with probability $1-\epsilon$ when C satisfies the following bound.*

$$C \geq N_{aw}(w-1)/(-2\ln(1-\epsilon))$$

PROOF. Under strategy S_{ad}, the values of w and C are chosen so that, with probability at least $1-\delta$, each of the w NWs in each ohmic region is distinct. The probability that this condition holds in all m ohmic regions is at least $(1-\delta)^m = 1-\epsilon$. Thus, $\ln(1-\delta) = \ln(1-\epsilon)/m$. By Lemma 4.1, to achieve probability $1-\delta$ in each region requires that that $C \geq mw(w-1)/(-2\ln(1-\epsilon))$. Since the number of addressable NWs is $N_{aw} = N = mw$, the result follows. □

When ϵ is small, say $\epsilon \leq 0.01$, $\ln(1-\epsilon)$ is closely approximated by $-\epsilon$. In this case, the lower bound becomes $C \geq N_{aw}(w-1)/2\epsilon$ which is too large to be practical.

5.1.2 *Most Different Codeword Strategy.* The most different codeword strategy S_{md} requires that, with probability at least $1-\epsilon$ within each of the m ohmic regions, at least $(w+1)/2$ of the w NWs have different doping patterns when the NWs are drawn with equal probability from an ensemble of C different types.

This strategy has $N_{aw} \geq m(w+1)/2$ NWs with distinct addresses with probability at least $1-\epsilon$. The remaining NWs are duplicates of NW types. Following, we show that the value of C required for this strategy is at least $e^{(w-1)/(w+1)-2\ln\delta/(w+1)}(w-1)/2$, where δ satisfies $\ln(1-\delta) = \ln(1-\epsilon)/m$ (or δ is approximately ϵ/m). This quantity is much smaller than the lower bound on C for strategy S_{ad}. That is, a factor of about two reduction in the number of addressable NWs results in a very large decrease in the number of NW types that need to be manufactured. The strategy requires that the chip be examined to discover which NW types are present.

THEOREM 5.2. *Strategy S_{md} succeeds with probability $1-\epsilon$ when C is chosen to satisfy the following bound where* $\ln(1-\delta) = \ln(1-\epsilon)/m$.

$$C \geq e^{(w-1-2\ln\delta)/(w+1)}(w-1)/2$$

PROOF. With strategy S_{md}, let w and C be chosen so that there are at least $(w+1)/2$ distinct NWs in an ohmic region with probability at least $1-\delta$. The probability that this is true for all ohmic regions is at least $(1-\delta)^m = 1-\epsilon$ for $\ln(1-\delta) = \ln(1-\epsilon)/m$. The result follows directly from Lemma 4.2. □

When ϵ is small, say, $\epsilon \leq 0.01$, $\ln(1-\delta) = \ln(1-\epsilon)/m$ yields $\delta = \epsilon/m$ to very good approximation. When $\epsilon = .01$, the lower bound on C ranges from 15 to 30 for $20 \leq m \leq 500$ and $10 \leq w \leq 20$, very practical ranges for these parameters. When $\epsilon = .001$ and the same range of values for m and w, the lower bound on C ranges from 23 to 45. These values of C make this a practical addressing method.

5.1.3 *All Present Codeword Strategy.* The all present codeword strategy S_{ap} requires that with probability at least $1-\epsilon$, within each of the m ohmic regions every one of the C NW types appears among the w NWs when the NWs are drawn with equal probability from an ensemble of C different types.

This strategy results in many duplicates of each NW type within an ohmic region. In the following, we show that it succeeds with probability $1-\epsilon$ when $N \geq mC\ln(C/\delta)$, where $(1-\delta)^m = 1-\epsilon$ (or δ is approximately ϵ/m). The number of addressable NWs satisfies $N_{aw} = mC$ with probability $1-\epsilon$. Thus, the number of NWs in each dimension, N, is larger than the number of addressable NWs by a factor of approximately $\ln(mC/\epsilon)$. The advantage of this strategy is that a standard decoder can be used to address ohmic regions and a trivial decoder can be used to address NWs if BRCs are used. That is, the decoding process is very simple and no translation memory is needed. The strategy does not require that the chip be examined to discover which NW types are present if ϵ is small enough.

THEOREM 5.3. *Strategy S_{ap} succeeds with probability $1-\epsilon$ when N is chosen to satisfy the following bound where* $(1-\delta)^m = 1-\epsilon$.

$$N \geq mC\ln(C/\delta)$$

Here $N_{aw} = mC$ is the number of addressable NWs.

PROOF. Under strategy S_{ap}, the values of w and C are chosen so that with probability at least $1-\delta$, each of the C different codewords appears at least once in each ohmic region. The probability that this condition holds in all m ohmic regions is at least $(1-\delta)^m = 1-\epsilon$. This implies that $\delta \leq -\ln(1-\epsilon)/m$. From Lemma 4.3 it follows that $w \geq C\ln(C/\delta)$ for one ohmic region. Since $N = mw$, the result follows. □

When $\epsilon \leq 0.01$, $\delta \approx \epsilon/m$ and the lower bound on N becomes $N \geq mC\ln(mC/\epsilon)$. Since $N_{aw} = mC$, this is equivalent to $N \geq N_{aw}\ln(N_{aw}/\epsilon)$.

5.1.4 *Repeated Codeword Strategy.* The repeated codeword strategy (S_{rc}) requires that with probability at least $1-\epsilon$, each of the C codeword types appears at least once in at least p different ohmic regions where $p = 0.3 * \overline{r}$, where $\overline{r} = m(1-(1-1/C)^w$ is the average number of ohmic regions in which each NW falls.

Since every possible codeword appears multiple times, if the axial code is a BRC, the low-order external binary bits can be used to select a NW by supplying the bits and their complements. A translation memory is needed to choose the ohmic region to activate based on the value of the remaining external bits. The strategy does require that the chip be examined to discover which NW types are present in which ohmic regions.

We now show that this strategy succeeds with a very small number of NW types, C. In particular, if $C = 2w$, $N_{aw} \geq .15N$ (or $N \leq yN_{aw}$) NWs are addressable with probability at least $1-\epsilon$, when $m \geq 11.8\ln(2w/\epsilon)$. (These bounds hold when $\nu = 1/2$.) Here N is the number of NWs in each dimension. When the size of the code space is an issue, this may be the best strategy to use.

THEOREM 5.4. *Strategy S_{rc} succeeds with probability $1-\epsilon$ when the number of ohmic regions, m, satisfies*

$$m \geq 2.952(\nu(1-\nu))^{-1}\ln(w/(\nu\epsilon)),$$

where $\nu = w/C$. The number of addressable NWs satisfies $N_{aw} \geq 0.3(1-\nu)N$, where $N = mw$ is the number of NWs in each dimension.

PROOF. The ith codeword occurs in a given ohmic region with probability $1-(1-1/C)^w$, and it occurs in $\overline{r} = m(1-(1-1/C)^w)$ regions on average. As shown in Lemma 4.4, the probability that it occurs in at most $\theta\overline{r}$ regions, $0 \leq \theta < 1$, is at most $e^{-\overline{r}(1-\theta+\theta\ln\theta)}$. When $\theta = 0.3$, $(1-\theta+\theta\ln\theta) = 0.3388$. The probability that any of the C codewords fails to occur in at least $\theta\overline{r}$ ohmic regions is at most $Ce^{-0.3388\overline{r}}$. Since the goal for strategy S_{rc} is that it be successful with probability at least $1-\epsilon$, it follows that this condition will be satisfied if w and C satisfy the following bound.

$$Ce^{-0.3388\overline{r}} \leq \epsilon \tag{3}$$

When this bound holds, which is equivalent to $\overline{r} \geq \rho\ln(C/\epsilon)$ for $\theta = 0.3$ and $\rho = 2.952$, each of the C codewords exists in at least $\theta\overline{r}$ ohmic regions with probability at least $1-\epsilon$.

Let $f(w,C) = (1-(1-1/C)^w)$. Then $\overline{r} = mf(w,C)$. It is easy to show by induction that $f(w,C) \geq (w/C)(1-w/C)$. This implies that $\overline{r} \geq m(w/C)(1-w/C)$. In particular, if $m(w/C)(1-w/C) \geq \rho\ln(C/\epsilon)$, then (3) holds. Under these conditions, each of the C codewords will appear at least $0.3\overline{r}$ times and the number of addressable wires satisfies $N_{aw} \geq 0.3\bar{r}C$, all with probability at least $1-\epsilon$. Given the lower bound on $f(w, C)$, this implies that $N_{aw} \geq 0.3mw(1-w/C)$.

If w is fixed at $w = \nu C$, $0 < \nu < 1$, the condition $m(w/C)(1-w/C) \geq \rho \ln(C/\epsilon)$ holds when $m \geq (\nu(1-\nu))^{-1}\rho \ln(w/(\nu\epsilon))$. It follows that C can be a small multiple of w and m a slowly growing function of w and ϵ, and the number of addressable NWs will be $N_{aw} \geq 0.3(1-\nu)mw$. $\square$

5.1.5 *Other Addressing Strategies.* The four strategies described are representative. Many other types are possible. For example, instead of requiring that all or half of the NWs in every ohmic region be different as in the all different and most different strategies, we could ask that these conditions apply to some fraction of the ohmic regions. This would reduce the code space required but increase the burden on the translation memory. Similarly, in the repeated codeword strategy, we could use a more sophisticated translation memory to utilize codewords that appear more than p times.

Another very different approach is to encode NWs so that groups of them can be formed with the property that all NWs in each group can be activated simultaneously without activating NWs in other groups. Also, the numbers of codewords should be chosen so that with high probability, each ohmic region contains at least one NW from each group. Then, the high-order bits of an external address could be used to choose an ohmic region and low-order bits used to activate all NWs in one group.

6. AREA ESTIMATES FOR NANOARRAYS

A nanoarray-based memory, NanoMem, is an $N \times N$ nanoarray that has a set of crosspoints, two sets of address decoders, and two copies of a translation memory, MesoMem, to map external binary addresses to internal addresses. The internal addresses generally must be discovered and their identities programmed into each copy of MesoMem. Thus, each nanoarray has two sets of M address wires, two sets of m ohmic regions, each with w NWs, and two MesoMems. In each dimension, the number of NWs is $N = mw$ and the number of addressable NWs is $N_{aw} \leq N$. The pitch of NWs and MWs are λ_{nano} and λ_{meso}, respectively.

We compare strategies for addressing nanoarrays by the area, A_T, needed to implement them. This area accounts for the area used by array crosspoints, decoders, and the translation memories. The area is compared to nanoarrays that have the same number of addressable crosspoints.

To accurately calculate the area of a nanoarray would require that a full layout be done. Since this isn't possible, we estimate the area by adding the areas of the crosspoints, decoders, and translational memories.

The area A_{nano} of the crosspoints and decoders satisfies $A_{nano} \approx (\lambda_{meso}M + \lambda_{nano}N)^2$ because one side of the nanoarray has length approximately $\lambda_{meso}M + \lambda_{nano}N$.

Let A_{meso} be the area of a standard decoder and a translation memo MesoMem. We approximate the area of a standard decoder by $\lambda_{meso}^2 m \log_2 m$ and the area of MesoMem by $\chi N_{aw}\beta$, where β is the of number of bits associated with each addressable NW and χ is the area used to store one mesoscale bit. Thus, $A_{meso} \approx \chi N_{aw}\beta + \lambda_{meso}^2 m \log_2 m$.

The area A_T of the NanoMem is approximated by $2A_{meso} + A_{nano}$ because it uses two decoders and two copies of MesoMem. Thus,

$$A_T \approx 2\chi N_{aw}\beta + 2\lambda_{meso}^2 m \log_2 m + (\lambda_{meso} M + \lambda_{nano} N)^2,$$

where χ is the area of a mesoscale memory cell, λ_{meso} and λ_{nano} are the pitch of MWs and NWs, N_{aw} is the number of addressable NWs, N is the total number of NWs, M is the number of address wires, m is the number of ohmic regions, w is the number of NWs per ohmic region, and $N = mw$. As mentioned at the beginning of Section 5, we assume that $M = 2\log_2 C$.

7. QUANTITATIVE ASSESSMENT OF DESIGN STRATEGIES

We now examine the performance of the four strategies under the assumption that each strategy achieves its objective with probability at least $(1 - \epsilon)$. With each strategy, the goal is to minimize, A_T, the area of a nanoarray for a given number of addressable NWs, N_{aw}.

We find that S_{md} is always superior to S_{ad} and S_{ap}. Also, S_{rc} is superior to S_{ad}. The analysis isn't strong enough to determine which of S_{md} or S_{rc} is superior. However, S_{rc} can be implemented with a $C = 2w$ codewords, a very small number. This alone favors its further study.

When the pitch of NWs is 10nm, strategy S_{ap} uses considerably less area than that forecast over the next couple of years by the ITRS Roadmap [2001] for standard DRAMs. Because S_{ap} doesn't require a translation memory, it may be a practicable first type of nanoarray memory to implement.

7.1 Performance of Strategy S_{ad}

In strategy S_{ad}, there are no wasted wires, that is, $N_{aw} = N = mw$. The NWs can be addressed with external binary addresses by supplying $\log_2 m$ of the $\log_2 N_{aw} = \log_2 m + \log_2 w$ bits to a standard decoder to activate one ohmic region and by supplying all the bits to a translation memory to provide the $M = 2\log_2 C$ bits to select individual NWs within ohmic regions.

The area A_T required is given as follows. Here $N = N_{aw}$, and $C \geq N_{aw}(w - 1)/2\epsilon$ was derived in Section 5.1.1.

$$A_T \approx 2\chi N_{aw} \log_2 C + 2\lambda_{meso}^2 m \log_2 m + (2\lambda_{meso} \log_2 C + \lambda_{nano} N_{aw})^2 \tag{4}$$

7.2 Performance of Strategy S_{md}

In strategy S_{md}, we guarantee that at least $d = (w + 1)/2$ NWs will be addressable. Thus, $N_{aw} > N/2$ or $N < 2N_{aw}$. At most half the NWs are wasted. The use of external binary addresses is the same as in strategy S_{ad}.

The area A_T required by this strategy is given as follows where $C \geq e^{(w-1-2\ln\delta)/(w+1)}(w-1)/2$, derived in Section 5.1.2 where $\delta \approx \epsilon/m$ when $\epsilon \leq 0.01$.

$$A_T \approx 2\chi N_{aw} \log_2 C + 2\lambda_{meso}^2 m \log_2 m + (2\lambda_{meso} \log_2 C + 2\lambda_{nano} N_{aw})^2 \tag{5}$$

Table I.

Year	2005	2006	2007
$\lambda_{nano} \backslash \lambda_{DRAM}$	160	140	130
10	88,862	12,206	4,424
20	30	11	7

7.2.1 *Comparison of S_{ad} and S_{md}.* The formulas for the area used by strategies S_{ad} and S_{md} are the same except that the last term for S_{md} is twice that for S_{ad}. However, the value of C needed for S_{ad} is much larger than that for S_{md} when N_{aw} is fixed, as we show.

The lower bound on C in S_{ad} is $C_{ad} = N_{aw}(w-1)/2\epsilon$ whereas in S_{md}, it is $C_{md} = e^{(w-1-2\ln\delta)/(w+1)}(w-1)/2$ for $\delta = \epsilon/m$.

When $\epsilon = .01$, C_{ad} is about $50N_{aw}w$ whereas C_{md} is about $e^{(w-1)/(w+1)}$ $(100m)^{2/(w+1)}(w-1)/2$. If $w \geq 10$ and $m \leq 5,000$, C_{md} is no more than $3.14(w-1)m^{0.182} \leq 15w$. In other words, C_{md} is smaller than C_{ap} by a factor of $3.4N_{aw}$. Thus, S_{md} is clearly superior to S_{ap}.

7.3 Performance of Strategy S_{ap}

Strategy S_{ap} uses no translation memory. The input bits are split between the ohmic regions and the NWs. The ohmic region bits are supplied to a standard decoder. Since $C = N_{aw}/m$ for this strategy, from Section 5.1.3 the area required satisfies the following bound, where $N \geq N_{aw}\ln(N_{aw}/\epsilon)$ is required for the strategy to succeed with probability at least $1-\epsilon$.

$$A_T \approx 2\lambda_{meso}^2 m \log_2 m + (2\lambda_{meso}\log_2(N_{aw}/m) + \lambda_{nano}N)^2 \qquad (6)$$

7.3.1 *Comparison of S_{ap} and Standard RAM.* A nanoarray-based memory designed using strategy S_{ap} is superior in the use of area to a standard RAM using current or projected technology parameters when λ_{nano} is small, as we show. We model the area of a RAM holding N_{aw}^2 bits by χN_{aw}^2 where χ is the area per RAM bit.

Since the first term in the expression A_T for S_{ap} is small by comparison with χN_{aw}^2, the conclusion follows if the second term $(2\lambda_{meso}\log_2(N_{aw}/m) + \lambda_{nano}N)^2$ is significantly smaller than χN_{aw}^2. Since we can expect $\lambda_{meso}/\lambda_{nano} \leq 20$, we can expect that $2\lambda_{meso}\log_2(N_{aw}/m)$ will be significantly smaller than $\lambda_{nano}N$.

If $N \leq N_{aw}\sqrt{\chi/\lambda_{nano}^2}$, the nanoarray memory NanoMem uses less area than a DRAM. Given the lower bound on $N \geq N_{aw}\ln(N_{aw}/\epsilon)$, this is equivalent to $N_{aw} \leq \epsilon e^{\sqrt{\chi/\lambda_{nano}^2}}$. Because a cell in standard memory DRAM memory uses area $\chi = \lambda_{DRAM}^2$, the condition becomes $N_{aw} \leq \epsilon e^{\lambda_{DRAM}/\lambda_{nano}}$.

The ITRS Roadmap [2001] predicts $\lambda_{DRAM} =$ 160nm, 140nm, and 130nm for 2005, 2006, and 2007. Shown in Table I are the values of N_{aw} below which a NanoMem is predicted to be superior to a DRAM in terms of areal density when $\epsilon = 0.01$. The storage capacity of memories using N_{aw} NWs in each dimension is N_{aw}^2.

It follows that design strategy S_{ap} will yield a nanoarray memory that uses less area than a standard DRAM memory when $\epsilon = 0.01$ if the pitch of NWs can be kept as small as 10nm because a NanoMem with $N_{aw} \geq 2,000$ is thought to be impractical due to the likelihood of faults [DeHon et al. 2005]. When the pitch of NWs is 20nm, the analysis, which is approximate, suggests that it is not likely that strategy S_{ap} will prevail over DRAMs.

7.3.2 *Comparison of S_{md} and S_{ap}.* The area used by S_{ap}, A_T^{ad}, given in (6) is approximated by $2\lambda_{meso}^2 m \log_2 m + \lambda_{nano}^2 N_{ap}^2$ because $2\lambda_{meso} \log_2(N_{aw}/m)$ is dominated by $\lambda_{nano} N_{ap}$. (N_{ap} is the number of NWs used by S_{ap}.) This follows because for S_{ap} to be successful with probability at least $1 - \epsilon$ requires that $N_{ap} = mw \geq N_{aw} \ln(N_{aw}/\epsilon)$.

In the area estimate for S_{md}, A_T^{md}, given in (5), the term $\lambda_{meso} \log_2 C_{md}$ is dominated by $\lambda_{nano} N_{aw}$ because C_{md} is small. (C_{md} is the number of codewords used by S_{md}.) Thus, A_T^{md} is approximated by $2\chi N_{aw} \log_2 C_{md} + 2\lambda_{meso}^2 m \log_2 m + 4\lambda_{nano}^2 N_{aw}^2$.

S_{ap} uses more area than S_{md} if $N_{ap}^2 \geq 2(\chi/\lambda_{nano}^2) N_{aw} \log_2 C_{md} + 4N_{aw}^2$, where $N_{ap} \geq N_{aw} \ln(N_{aw}/\epsilon)$, and $\chi/\lambda_{nano}^2 = (\lambda_{DRAM}/\lambda_{nano})^2$. Combining the first two inequalities, $N_{aw}(\ln^2(N_{aw}/\epsilon) - 4) \geq 2(\lambda_{DRAM}/\lambda_{nano})^2 \log_2 C_{md}$, S_{ap} uses more area than S_{md}. As shown at the end of Section 7.2, under reasonable conditions, C_{md} is no more than $15w$ when $\epsilon = .01$. Also, for the data given, $\lambda_{DRAM}/\lambda_{nano} \leq 16$. Under these assumptions, it is easy to show that the condition holds if $N_{aw} \geq 100$ and $\epsilon = 0.01$. We conclude that strategy S_{md} is superior to strategy S_{ap}.

7.4 Performance of Strategy S_{rc}

In strategy S_{rc}, the number of wasted wires is substantially less than strategy S_{ap}. However, a translation memory is needed. The external bits are separated into two sets, one set identifying a NW and a second identifying a group. The first set of $2 \log_2 C$ bits are used directly to activate a NW. Both sets of bits are used to address the translation memory and select the $\log_2 m$ bits needed to activate an ohmic region. Thus, each translation memory has N_{aw} words of length $\log_2 m$. The area A_T required by this strategy is given as:

$$A_T \approx 2\chi N_{aw} \log_2 m + 2\lambda_{meso}^2 m \log_2 m + (2\lambda_{meso} \log_2 C + \lambda_{nano} N)^2.$$

7.4.1 *Comparison of S_{md}, S_{ap} and S_{rc}.* If in strategy S_{rc}, we require that each NW encoding occur in m ohmic regions, it would be equivalent to S_{ap}. Thus, S_{ap} is inferior to S_{rc}.

Strategies S_{md} and S_{rc} use the areas shown here.

$$\begin{aligned} A_T^{md} &\approx 2\chi N_{aw} \log_2 C_{md} + 2\lambda_{meso}^2 m \log_2 m + (2\lambda_{meso} \log_2 C_{md} + 2\lambda_{nano} N_{aw})^2 \\ A_T^{rc} &\approx 2\chi N_{aw} \log_2 m + 2\lambda_{meso}^2 m \log_2 m + (2\lambda_{meso} \log_2 C_{rc} + \lambda_{nano} N_{rc})^2 \end{aligned}$$

C_{md} and C_{rc} are the sizes of the code spaces for S_{md} and S_{rc}. N_{rc} is the number of NWs for S_{rc}. As shown in Section 5.1.4, $N_{rc} \leq 7N_{aw}$ when $C_{rc} = 2w$. Also, as shown in Section 7.2.1, $C_{md} \leq 15w$ when $\epsilon = .01$, $w \geq 10$ and $m \leq 5,000$.

If the area of the nanoarray (the squared term on the right) dominates the area of the translation memory, strategy S_{md} is superior in the use of area because $\log_2 C_{md}$ is small, and N_{aw} is smaller than N_{rc}. When the area of the translation memories dominate, it again appears that S_{md} is superior to S_{md}. However, the area bounds are too close to make a determination based on the analysis that we have available.

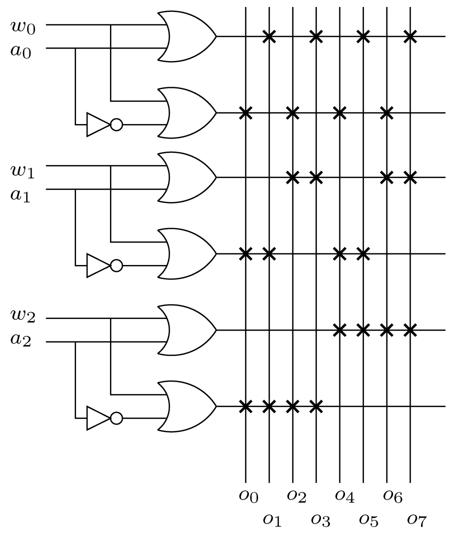

Fig. 5. Standard decoder implemented with wildcards. The address word $\mathbf{a} = (a_0, a_1, ..., a_{b-1})$ and its complement is ORed with the wildcard word $\mathbf{w} = (w_0, w_1, ..., w_{b-1})$, and then all 2^b combinations of the b inputs and their complements are ANDed to form the output word $\boldsymbol{O} = (o_0, o_1, ..., o_{2^b})$. The x's on a vertical wire, labeled o_j, represent ANDs.

It should be noted again that strategy S_{rc} can be used with C as small as $2w$. This very small value of C, which reduces the cost of manufacturing NWs, may make S_{rc} a superior overall strategy.

8. WILDCARDING

Because nanoarray-based memories may be very large, writing or even reading them, one bit at a time, may be expensive. To cope with this problem, we examine the use of wildcarding in which the memory address input is augmented by a binary wildcard word of the same length in which a 1 indicates that the corresponding address bit is allowed to assume both values. It follows that a wildcard word containing k bits defines 2^k addresses when combined with an address word.

Wildcarding may prove invaluable in some applications, including codeword discovery (see Section 9) and fast ROM programming. We investigate the architectural support needed for wildcarding for each of the four designs previously visited. Wildcarding can be done by a standard decoding circuit augmented with ORs on the input, as shown in Figure 5. A standard decoder with b inputs can be implemented by forming the AND of all 2^b combinations of the b inputs and their complements. A wildcard decoder can be implemented by ORing the bits of the wildcard word $\mathbf{w} = (w_0, w_1, \ldots, w_{b-1})$ with the bits of the address word $\mathbf{a} = (a_0, a_1, \ldots, a_{b-1})$ and their complements.

8.1 Wildcarding on Strategy S_{ad} and S_{md}

In these two cases, each ohmic region has a stochastically chosen subset of all the codes. As a result, wildcarding is not possible within an ohmic region.

Wildcarding, however, is possible over the portion of the address that corresponds to the ohmic regions. This is useful if it is desirable to activate all the nanowires in the ohmic regions that are addressed and for code discovery.

8.2 Wildcarding on Strategy S_{ap}

For this strategy, codewords and ohmic regions are specified independently by sub-sequences of the external binary address. Thus, if binary reflected codes are used, wildcarding can be done on both codewords and ohmic regions simultaneously. Moreover, the decoding circuitry remains simple since we only replace the two standard decoders for the ohmic regions and codes with wildcard decoders.

8.3 Wildcarding on Strategy S_{rc}

In this strategy, it is assumed that each codeword appears in at least p ohmic regions, p about one third of the average number of ohmic regions in which a given NW type appears. These codewords are organized into p groups each containing C different codewords. If a codeword appears in different groups, it occurs in different ohmic regions.

Although many types of wildcarding, such as over ohmic regions and codewords, cannot be implemented efficiently, there is one type of wildcarding that is suitable, namely, when a single codeword is activated in a set of groups defined by wildcards. We describe several new designs that support this type of wildcarding.

The first type of wildcarding uses a C-word memory with m-bits per word that, when given the address for the jth codeword, produces an m-tuple $\boldsymbol{O}$ of 1s that identifies the ohmic regions containing the jth codeword. Let the kth instance of the jth codeword $\boldsymbol{z}$ be associated with the kth group. Using a decoder of the type shown in Figure 5, specify a set of groups using wildcards with a 0-1 p-tuple $\boldsymbol{P}$ that specifies which of the p groups is to be activated. A circuit is then designed to match up the entries in $\boldsymbol{P}$ with the 1s in $\boldsymbol{O}$ to identify those ohmic regions containing the codeword $\boldsymbol{z}$ associated with the groups specified by $\boldsymbol{P}$.

To illustrate this operation, suppose that the NWs are subdivided into four groups and that the output of the memory is $\boldsymbol{O} = (0, 1, 0, 1, 1, 1)$, meaning that the instances of the chosen codeword occur in four ohmic regions, namely, the second, fourth, fifth, and sixth regions. Suppose also that the output of the wildcard decoder is $\boldsymbol{P} = (1, 1, 0, 0)$, meaning that the first and second groups are selected. If we align the bits in $\boldsymbol{P}$ with the 1s in $\boldsymbol{O}$ and AND them together, the resulting 1s identify the ohmic regions that need to be activated to activate the specified NW in the groups identified by wildcarding. In this example, the resulting word is $(0, 1 \wedge 1, 0, 1 \wedge 1, 1 \wedge 0, 1 \wedge 0) = (0, 1, 0, 1, 0, 0)$ which means that the second and fourth ohmic regions are activated.

We now describe a circuit to match up the components of a 0-1 p-tuple $\boldsymbol{P}$ with the p 1s in a 0-1 m-tuple $\boldsymbol{O} = (o_1, o_2, \ldots, o_m)$. Such a circuit shifts the ith component of $\boldsymbol{P}$ by the number of 0s before the first i 1s in $\boldsymbol{O}$. The number of 0s before the ith 1 in $\boldsymbol{O}$ can be obtained by forming the prefix sum [Savage 1998, p. 55] on the Boolean complement of $\boldsymbol{O}$. (The prefix sum on an m-tuple $\boldsymbol{w}$

is an m-tuple $\boldsymbol{S} = (s_1, s_2, \ldots, s_m)$, where s_j is the sum of the first j components of $\boldsymbol{w}$. For example, if $\boldsymbol{O} = (0, 1, 0, 1, 1, 1)$, its complement is $(1, 0, 1, 0, 0, 0)$ and its prefix sum is $(1, 1, 2, 2, 2, 2)$.) Next use the prefix sum, say $(1, 1, 2, 2, 2, 2)$, to produce m shifts of $\boldsymbol{P} = (P(s_1), P(s_2), \ldots, P(s_m))$ by the m values in $\boldsymbol{S}$. Form an m-tuple, $\boldsymbol{R} = (r_1, r_2, \ldots, r_m)$, containing in the ith position the bit in the ith position of $\boldsymbol{P}(s_i)$, the shift of $\boldsymbol{P}$ by s_i places. The positions in $\boldsymbol{R}$ that correspond to 1s in $\boldsymbol{O}$ determine which ohmic regions are activated. Thus, combine these two m-tuples by ANDing them together componentwise.

While computing $\boldsymbol{P}(s_1), \boldsymbol{P}(s_2), \ldots, \boldsymbol{P}(s_m)$ can be done in parallel as just described, it can also be done serially. $\boldsymbol{P}(s_1)$ can be obtained by shifting or not shifting $\boldsymbol{P}$ right by one place, depending upon whether o_1 is 1 or 0. Similarly, we can obtain $\boldsymbol{P}(s_i)$ by shifting or not shifting $\boldsymbol{P}(s_{i-1})$ right by one place. We may now form $\boldsymbol{R}$ as suggested previously where r_i is the bit in the ith position of $\boldsymbol{P}(s_i)$. This design uses m two-bit multiplexers to select the correct bit from $\boldsymbol{P}(s_{i-1})$ to create $\boldsymbol{P}(s_i)$ and does this p times, for a total of pm two-bit multiplexers. However, it takes more time than the design that uses prefix sums and shifters by multiple positions.

The designs previously mentioned compute an m-tuple based on shifts of $\boldsymbol{P}$, the p-tuple computed by the group wildcard decoder. However, the amount of shift is independent of the wildcard pattern selected and only depends on which ohmic regions contain the jth codeword. Furthermore, the computation can be seen as selecting one of the p locations in $\boldsymbol{P}$ for each of the m locations of $\boldsymbol{R}$. We can form the m-tuple $\boldsymbol{R}^* = (r_1^*, r_2^*, \ldots, r_m^*)$ where r_i^* is 0 if o_i is 0, and otherwise it is j if o_i is the jth 1 in $\boldsymbol{O}$. For example, if $\boldsymbol{O} = (0, 0, 1, 0, 1, 1)$, then $\boldsymbol{R}^* = (0, 0, 1, 0, 2, 3)$ which identifies for each of the m locations the pth location to select from the wildcard decoder to create $\boldsymbol{R}$. Finally, we can AND the components of $\boldsymbol{R}$ with $\boldsymbol{O}$ to generate which ohmic regions to activate given the groups and code selected. It is possible to precompute $\boldsymbol{R}^*$ and store it along with $\boldsymbol{O}$ in a memory with word length $m \log_2 p + m$. With this information, using a multiplexer for each ohmic region, we can compute the desired output. This design uses a larger memory but is faster then the previous two.

Finally, it is also possible to reduce the size of the memory by reducing the length of words in the memory from m to $p \log_2 m$. This can be done by encoding the 0s in $\boldsymbol{O}$. Such an encoding can consist of p integers having values in the range $[0, \ldots, m]$ and represented by $\log_2 m$ bits. If such a representation is used, then a circuit is needed to expand this representation into the m-tuple with 1s representing the ohmic regions containing the jth codeword.

9. CODEWORD DISCOVERY

In this section, we describe efficient codeword discovery algorithms for the four address mapping strategies previously presented.

9.1 Exhaustive Search

The assembly of nanoarrays is stochastic. In order to determine which NW code-words are present in the various regions, testing must occur. Recall that a NW address is determined by its ohmic region and codeword. Data is stored at the intersection of two NWs. The existence of a NW address in one dimension of the memory can be tested by activating a wire (or all the wires in an ohmic region) in the other dimension, writing a 1 and then reading from that address. If a 1 is returned, the NW address exists.

Addresses do not need to be tested one at a time. Wildcarding allows for a 1 or a 0 to be written to multiple addresses at once. Similarly, it allows for reading from multiple addresses simultaneously. A multiple-read operation outputs the OR of the values stored at the multiple addresses. A 1 is returned if any of the addresses contains a 1. By writing to and then reading from multiple addresses at once, one can verify that at least one of the addresses exists. If many possible addresses are missing (as in strategies S_{ad}, S_{md}, and S_{rc}), multiple address testing can dramatically reduce the time to test for the existence of NWs. For these cases, we describe how wildcarding along one dimension of the array can be employed to determine addresses efficiently.

It should be noted that, in this section, we assume that all address wires and ohmic regions are accessible externally. This will allow multiple ohmic regions and address to be activated via wildcarding. Our final search strategy, described in Section 9.3, only requires wildcarding over ohmic regions.

9.1.1 *Exhaustive Search for S_{ap}.* Exhaustive search for S_{ap} is quite efficient. In the case where all codewords are expected to be present in each ohmic region. Along each dimension, every possible address is expected, and thus they must be tested one at a time. This could be done with N_{aw} read/write operations. A better approach, however, would be to first write 1s to all NWs in one ohmic region of the array at a time, and then read from the addresses one at a time. If all addresses are present, each read operation will yield a 1. This algorithm requires $O(N_{aw})$ steps.

9.2 Searching Across the Code Space

In case S_{ad}, because all addresses that are present are very likely unique, it is natural to search across the code space within each ohmic region.

In S_{ad}, the number of codewords used is roughly $C = 50N_{aw}w = 50mw^2$ (when $\epsilon = 0.01$). If a search is done for each of the C codewords one at a time in each ohmic region, about $mC = 50N_{aw}^2$ read operations will be required. Fortunately, wildcarding provides a significant improvement.

Our improved method is based on the search for a single codeword in a single ohmic region. Write 1's to every codeword in the region and use wildcarding to read from all codewords with a first bit of 1. If a 1 is returned, a codeword is present whose first bit is 1; otherwise, all codewords present have a first bit of 0. Once the first bit of a codeword has been determined, fix its value and use wildcarding to test for the value of the second bit. If the codewords contain b-bits, b read operations will be required to determine the value of a single codeword. With binary reflected codes, $b = \log_2 C$.

To search for multiple addresses, use the same procedure. First all bits are set to 1. Next a series of $\log_2 C$ read operations are used to locate a single codeword. Once the codeword is found, it is set to 0. The search procedure can then be repeated to find a new codeword. Once w codewords have been found, or all codewords present have been written with 0's, the search ends. All codewords will be identified in at most $w \log_2 C + w$ steps.

Using this binary style search, each ohmic region will require $w \log_2 C + w$ steps. When done over all ohmic regions, at most $mw \log_2 C + mw = N_{aw} \log_2(2C) \approx N_{aw} \log_2(100wN_{aw})$ when $\epsilon = 0.01$. This represents a substantial improvement over the $50N_{aw}^2$ operations required by a sequential search algorithm.

9.3 Searching Across Ohmic Regions

Instead of searching for codewords within ohmic regions, we can search for code-words across multiple ohmic regions. This strategy is better suited to S_{md}, although it can also be used with S_{ad}.

In case S_{ad}, the number of codewords C needed far exceeds the total number of addressable wires. In case S_{md}, C is much smaller which means that a different type of binary search can be used. We now describe a procedure that works for either case but which is more efficient for strategy S_{md}.

The procedure searches through all C codewords. It writes value 1 to each NW codeword in turn (this writes 1 to all NWs in all ohmic regions containing that codeword), and then reads from all regions at once to see if that codeword is present. If a codeword is absent from all regions, one test suffices to determine this. Let C' be the number of NW codewords that are not present. Thus, C' read tests suffice for the C' codewords that are absent.

If the ith codeword is present in k_i regions, we show that at most $k_i \lceil \log_2 m \rceil$ read operations suffice to identify the ohmic regions containing it. To see this, search for codeword i in each half of the ohmic regions. If both halves contain such a codeword, search the lexicographically first half. Otherwise, search the half containing codeword i. Applying this procedure recursively uses $\lceil \log_2 m \rceil$ tests to find a region containing the ith codeword. Write a 0 to ith in the discovered ohmic region. Repeat the procedure on the remaining $k_i - 1$ instances of codeword i. When done, $\sum_i k_i \lceil \log_2 m \rceil = N_{aw} \lceil \log_2 m \rceil$ read operations are used to determine the ohmic regions containing codewords that are present.

It follows that $N_{aw} \log_2 m + C'$ read operations are required to identify the N_{aw} addresses that are present, and the C' codewords that are absent.

In case S_{ad}, C' is a very large fraction of C because there are N_{aw} addressable NWs with probability $1 - \epsilon$ and $C \geq 50N_{aw}$. Thus, it is not an effective code discovery algorithm for case S_{ad}.

In case S_{md}, under reasonable conditions, C is not more than $15w$ and we can expect all codewords to be present in some region with high probability. In other words, C' is close to zero with high probability. As a result, approximately $N_{aw} \log_2 m$ read operations will be required to determine which codewords are present. Note that $N_{aw} \geq m(w + 1)/2$. Although more than N_{aw} addresses are likely to be present, searching for these addresses can be avoided if every time $(w + 1)/2$ codewords are discovered within a particular ohmic region, the region is overwritten with 0's. If $m < C$, this algorithm provides a savings over the one used in S_{ad} since the latter uses $N_{aw} \log_2(2C)$ operations. However, the new algorithm can be made even more efficient.

The algorithm's effectiveness hinges on the fact that C' remains small (preferably 0). To improve upon it, note that if N_{aw} is much larger than C, the ohmic regions can be divided into smaller groups such that within each group all C codewords are very likely to occur, and C' is small. A value of n for which this condition can be met approximately satisfies $n(w + 1)/2 = C$. Applying the search algorithm to each group of n regions will take at most approximately $n(w + 1)(\log_2 n)/2 + C$ steps (since $C' < C$). When the search is done over m/n groups of ohmic regions, we see that at most $m(w + 1)(\log_2 n)/2 + mC/n = N_{aw} \log_2(4C/(w + 1))$ tests suffice since $mC/n = 2N_{aw}$. The factor $\log_2(4C/(w + 1))$ is an improvement over $\log_2 m$. For Strategy S_{md}, we have a more efficient algorithm than the one used in Strategy S_{ad}.

Because this search strategy involves wildcarding over the ohmic regions and not the code space, it applies to both (h, b)-hot and binary reflected codes.

9.3.1 *Lower Bounds.* We show the effectiveness of our algorithms by providing a lower bound on the number of read operations required to identify the codewords of N_{aw} addressable wires. Each ohmic region consists of w wires. Let αw be a lower bound on the number of unique codewords in each ohmic region. It follows that the NW codewords can be chosen in at least $\binom{C}{\alpha w}$ ways in one ohmic region and at least $\binom{C}{\alpha w}^m$ ways in m ohmic regions. This is also the number of distinct outputs of the decoding algorithm. Each read operation produces a binary output. Thus, to choose between $\binom{C}{\alpha w}^m$ outputs, at least $\log_2(\binom{C}{\alpha w}^m) = m \log_2 \binom{C}{\alpha w}$ reads will be required.

To see that $\binom{C}{\alpha w}$ is lower bounded by $(C/(\alpha w))^{\alpha w}$ observe that $\binom{C}{n} = (C/n) * ((C - 1)/(n - 1)) * ... * ((C - n + 1)/1)$. If $C > n$, $C/n < (C - 1)/(n - 1)$, so $\binom{C}{n} > (C/n)^n$. From this, we have $\log_2 \binom{C}{\alpha w} > \alpha w \log_2(C/\alpha w)$. The code search will require more than $m * \alpha w \log_2(C/\alpha w) = N_{aw} \log_2(C/\alpha w)$ steps.

Our improved algorithm for strategy S_{md} requires approximately $N_{aw} \log_2(4C/(w + 1))$ reads which is very close to the lower bound (consider $\alpha = 2$). Thus, this algorithm is quite efficient considering that in case S_{md}, C is proportional to w and thus $\log_2(4C/(w + 1))$ effectively becomes a small constant.

9.3.2 *Strategy S_{rc}.* Case S_{rc} is essentially analogous to case S_{md}. By definition, it must have many instances of each codeword and have many more than C addressable wires. The ohmic regions can be divided up into groups just as in case S_{md}. The only difference is that, in this case, the number of addressable wires per ohmic region is not fixed, thus there is no need to overwrite entire ohmic regions with 0's. The algorithm simply finds as many addresses as possible until the desired N_{aw} addresses have been found.

10. CONCLUSIONS

The nanoarray is one of the most promising new nanotechnologies to emerge from recent research. It offers the potential for large storage capacities at the expense of having to cope with stochastic self-assembly. Nanoarrays are assembled from randomly selected subsets of encoded NWs. As a result, it is not possible to guarantee their exact storage capacity. Storage capacities are instead assured with some probability determined by address decoding circuitry and the size of the code space.

One characteristic of stochastically assembled nanoarrays is that end-to-end registration of NWs cannot be controlled. To cope with this issue, we have designed binary reflected codes which are closed under shifting and have the attractive property that they represent binary numbers. We have also demonstrated how binary reflected codes and h-hot codes can be efficiently manufactured using a reduced number of encodings which shift to produce all codewords.

Because the stochastically assembled nanoarrays vary greatly, a common interface must be provided in the form of a programmable address decoder. To program this interface, a codeword discovery process is required to determine which NW encodings are present in each ohmic region of the array. We have given an efficient binary search algorithm for codeword discovery and shown that as array size increases, its runtime approaches the lower bound.

Given the parameters of the nanoarray and the size of the code space, several decoding strategies can be employed to provide a common interface to the nanoarray. We have chosen to examine four of them. The first, the all different strategy, assumes that w, the number of NWs per ohmic region, and C, the size of the code space, are such that each NW in each ohmic region has a unique encoding with probability at least $1-\delta$. The second, the most different strategy, chooses the same parameters so that about half of the NWs in each ohmic region have a unique encoding with the same probability. The third, the all present strategy, assumes that w and C are chosen so that, with the same probability, each NW encoding occurs at least once in each ohmic region. Finally, the fourth, the repeated code-word strategy, assumes that the parameters are set so that, with the same high probability, each NW encoding appears at least once in at least p ohmic regions, where p is a parameter of the design.

To analyze each of the four strategies, we have developed bounds on relevant probabilities. Each probability is a variant of the classic coupon collector problem which asks how many trials are needed to ensure with probability at least $1 - \delta$ that each of the C different coupons will be collected when each of the C coupons is equally likely to be drawn on each of trial. Two of the variants we consider involve the probability that all and half of the coupons drawn in T trials are unique.

Analytical bounds on these probabilities allow us to compare the size of the decoder memory required for each of the four strategies. Based on this, we conclude that the most different strategy is preferable to the all different strategy because the same storage capacity can be provided with a much smaller code space. This reduces decoder size and the number of NW encodings that must be manufactured.

The all present strategy, while inferior to the most different strategy, is expected to use less area than conventional DRAM memories. Since it requires minimal address decoding, it might serve as a bridging technology between the fully-realized nanoarrays and conventional memories.

The repeated codeword strategy may be superior or inferior to the second, depending on the parameters of the nanoarray. Regardless, it has the distinct advantage that it succeeds with a very small number of codewords.

This is the first systematic study of nanoarrays that explores a variety of address mapping strategies and takes into account the area required for translation memories. We have demonstrated that a wide range of mapping strategies are possible and have considered four strategies in detail. Our investigation offers concrete examples of the type of modeling and analysis that is likely to be needed to understand other nanoscale computing devices.

APPENDIX

A. BOUNDS ON THE TAIL OF THE COUPON COLLECTOR'S DISTRIBUTION

The Coupon Collector problem [Feller 1968] is to determine the number of trials, T, that are needed to ensure that with probability at least $1-\epsilon$ all C coupons are drawn at least once when trials are statistically independent and coupons are drawn according to the uniform distribution. Here we derive a tight lower bound on T as well as tight upper bounds on (a) the probability $P(d)$ that d different coupons are drawn in $T = d$, trials, and (b) the probability $Q(d, T, C)$ that more than T trials are needed to generate d different coupons when T is a fraction of at least twice the number of trials. In both cases, coupons are drawn on each trial with statistical independence according to the uniform distribution.

A.1 A Bound on the Probability that All Outcomes are Distinct

Consider the probability $P(d)$ that d distinct coupons are drawn in $T = d$ trials with statistical independence according to uniform distribution. If $(j-1)$ different coupons have been drawn in the first $(j-1)$ trials, the probability that the coupon drawn on the jth trial is new is $(1-(j-1)/C)$. It follows that $P(d)$ satisfies the following expression.

$$P(d) = \Pi_{j=2}^{d}(1-(j-1)/C) \tag{7}$$

We derive two bounds to this probability by bounding the sum $\ln P(d)$. The first bound is obtained by applying the inequality $\ln(1-x) \leq -x$ which holds for all real values of x. The second is obtained by overbounding a sum by an integral, as illustrated in Figure 6.

LEMMA A.1. *The probability $P(d)$ that a new coupon is produced in each trial satisfies the following bounds.*

$$P(d) \leq e^{-\phi(d,C)}, \tag{8}$$

where $\phi(d, C) = \max\{d(d-1)/(2C), q(d, C)\}$, and $q(d, C) = C[(1-(d-1)/C)$ $\ln(1-(d-1)/C) - (d-1)/C]$.

PROOF. The bound is the smaller of two bounds, namely, the first which uses the blackinequality $(1-x) \leq e^{-x}$ and $\sum_{i=1}^{d-1} i = d(d-1)/2$. The second inequality bounds an integer sum by an integral, as illustrated in Figure 6(b).

$$\sum_{j=2}^{d} \ln(1-(j-1)/C) \leq \int_{1}^{d} \ln(1-(x-1)/C)\,dx = C\int_{1-(d-1)/C}^{1} \ln y\,dy$$

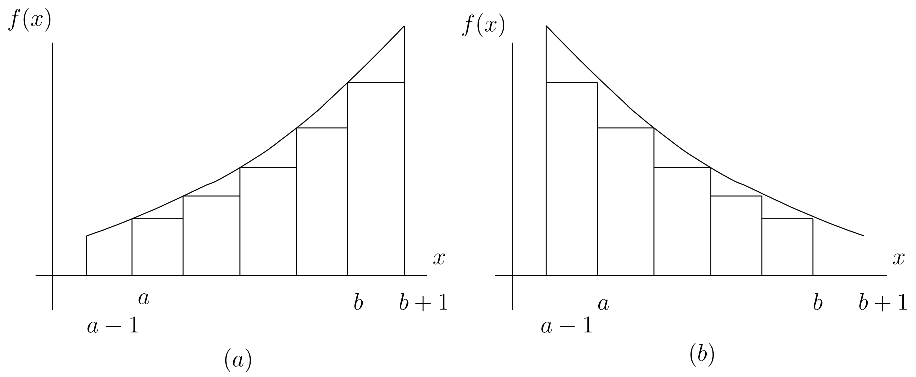

Fig. 6. When $f(x)$ is increasing, $\sum_a^b f(x) \le \int_a^{b+1} f(x)dx$, as suggested in (a). When $f(x)$ is decreasing $\sum_a^b f(x) \le \int_{a-1}^b f(x)dx$, as suggested in (b).

Here, we substituted $y = 1-(x-1)/C$. Since $\int_a^b \ln y dy = (y \ln y - y)|_a^b$, it follows that

$$\sum_{j=2}^{d} \ln(1-(j-1)/C) \le q(d,C) = -C[(1-(d-1)/C)\ln(1-(d-1)/C)+(d-1)/C].$$

The smaller of the two bounds is the value of $\phi(d,C)$. $\square$

Using the first of the two bounds, we see that $P(d) \ge 1-\epsilon$ when $C \ge d(d-1)/(-2\ln(1-\epsilon))$.

A.2 Bounds on $Q(d,T,C)$

$Q(d,T,C)$ is the probability that more than T trials are needed to produce d different coupons. This is equivalent to the probability that in T trials at least $C-d+1$ different coupons are missing.

We derive an upper bound on $Q(d,T,C)$ by relating it to the probability that a specific set of $C-d+1$ coupons is missing which is easy to compute. We also derive a lower bound using the principle of inclusion and exclusion. The bounds are tight when d is close to C.

Although we can't guarantee that the upper bound remains tight when d is a small fraction of C, our empirical data, obtained by comparing the analytical upper bound with the exact value of $Q(d,T,C)$, demonstrates that it gives very good results for the range of parameters likely to be in use in nanoarray-based memories.

THEOREM A.1. *The probability $Q(d,T,C)$ that more than T trials are needed to produce d different coupons when coupons are selected independently and uniformly satisfies the following bounds.*

$$\binom{C}{d-1}\left(\frac{d-1}{C}\right)^T \left(1-\frac{1}{2}\binom{C}{d-1}\left(1-\frac{1}{d-1}\right)^T\right) \le Q(d,T,C)$$

$$Q(d,T,C) \le \binom{C}{d-1}\left(\frac{d-1}{C}\right)^T$$

PROOF. $Q(d, T, C)$ is the probability that at least $C - d + 1$ different coupons are missing in T trials. Let $P(j, T)$ be the probability that exactly j coupons are missing in T trials. Thus, $Q(d, T, C) = \sum_{j=C-d+1}^{C} P(j, T)$.

Let $\boldsymbol{c} = \{c_1, c_2, \ldots, c_k\}$ denote a set of k coupons, and let $F_{\boldsymbol{c}}$ be the event that coupons $\{c_1, c_2, \ldots, c_k\}$ are missing in T trials where $k = C - d + 1$. It follows that $Q(d, T, C)$ is the probability of the union of the events $F_{\boldsymbol{c}}$ for all $\binom{C}{k}$ choices of $\boldsymbol{c}$. In other words,

$$Q(d, T, C) = P_r\left(\bigcup_{\boldsymbol{c}} F_{\boldsymbol{c}}\right).$$

The sum of the probabilities of the events $F_{\boldsymbol{c}}$ provides an upper bound for $Q(d, T, C)$. A lower bound is obtained using the principle of inclusion and exclusion. These bounds are shown here.

$$\sum_{\boldsymbol{c}} P_r\,(F_{\boldsymbol{c}}) - \frac{1}{2}\sum_{\boldsymbol{c}\neq\boldsymbol{c}'} P_r(F_{\boldsymbol{c}} \cap F_{\boldsymbol{c}'}) \leq Q(d, T, C) \leq \sum_{\boldsymbol{c}} P_r\,(F_{\boldsymbol{c}})$$

The probability that a specific set of l coupons is missing in T trials is obviously $(1 - l/C)^T$. Thus, $P_r\,(F_{\boldsymbol{c}}) = (1 - k/C)^T$. Let pairs $(\boldsymbol{c}, \boldsymbol{c}')$ have s coupons in common. Thus, they have $2k - s$ distinct coupons between them. The probability that coupons in both $\boldsymbol{c}$ and $\boldsymbol{c}'$ are missing, $P_r\,(F_{\boldsymbol{c}} \cap F_{\boldsymbol{c}'})$, is $(1 - (2k - s)/C)^T$.

The number of pairs with s coupons in common is $\binom{C}{k}\binom{k}{s}\binom{C-k}{k-s}$ because there are $\binom{C}{k}$ ways to choose $\boldsymbol{c}$, $\binom{k}{s}$ ways to choose the s common coupons from $\boldsymbol{c}$ to form $\boldsymbol{c}'$, and $\binom{C-k}{k-s}$ ways to choose the $k - s$ remaining coupons that are in $\boldsymbol{c}'$ but not in $\boldsymbol{c}$.

Combining these observations, we have the following bounds on $Q(d, T, C)$ where $k = C - d + 1$.

$$Q(d, T, C) \leq \binom{C}{k}\left(1 - \frac{k}{C}\right)^T$$

$$\binom{C}{k}\left(\left(1 - \frac{k}{C}\right)^T - \frac{1}{2}\sum_{s=0}^{k-1}\binom{k}{s}\binom{C-k}{k-s}\left(1 - \frac{(2k-s)}{C}\right)^T\right) \leq Q(d, T, C) \quad (9)$$

We derive a lower bound to $Q(d, T, C)$ by deriving an upper bound to the term L defined as:

$$L = \sum_{s=0}^{k-1}\binom{k}{s}\binom{C-k}{k-s}\left(1 - \frac{(2k-s)}{C}\right)^T. \quad (10)$$

This bound is derived by observing that $(1 - \frac{(2k-s)}{C})^T$ is largest when $s = k - 1$, as shown.

$$L \leq \left(1 - \frac{(k+1)}{C}\right)^T \sum_{s=0}^{k-1}\binom{k}{s}\binom{C-k}{k-s} = \binom{C}{k}\left(1 - \frac{(k+1)}{C}\right)^T \quad (11)$$

	T									
C	10	20	30	40	50	60	70	80	90	100
20	**5**/6	**9**/10	**12**/12	**14**/14	**16**/16	**17**/17	**17**/17	**18**/18	**18**/18	**19**/19
40	**6**/7	**11**/12	**16**/17	**19**/21	**22**/24	**25**/27	**27**/29	**29**/30	**31**/32	**32**/33
60	**7**/7	**12**/14	**17**/19	**22**/24	**26**/29	**29**/33	**33**/36	**36**/39	**38**/41	**40**/43
80	**7**/8	**13**/15	**18**/21	**23**/27	**28**/32	**32**/36	**36**/41	**39**/44	**43**/48	**46**/51
100	**7**/8	**13**/15	**19**/22	**24**/28	**29**/34	**34**/39	**38**/44	**42**/48	**46**/52	**49**/56
120	**7**/8	**14**/16	**20**/23	**25**/29	**30**/36	**35**/41	**40**/47	**44**/52	**48**/56	**52**/60
140	**7**/8	**14**/16	**20**/23	**26**/30	**31**/37	**36**/43	**41**/49	**46**/54	**50**/59	**54**/64
160	**7**/8	**14**/16	**20**/24	**26**/31	**32**/38	**37**/44	**42**/50	**47**/56	**52**/62	**56**/67
180	**8**/8	**14**/16	**21**/24	**27**/32	**33**/39	**38**/45	**43**/52	**48**/58	**53**/64	**58**/69
200	**8**/8	**15**/17	**21**/25	**27**/32	**33**/39	**39**/46	**44**/53	**49**/59	**54**/65	**59**/71

Fig. 7. Each pair of entries $\mathbf{n}/m$ is a value of d, the number of distinct codewords that occur with probability .99 when T trials are run over an ensemble of C codewords, when the probability of success is computed with the upper bound to $Q(d, T, C)$ given in Theorem A.1, the value of $\mathbf{n}$, and the exact formula, the value of m. In this table, $\mathbf{n}/m \geq .8$.

Substituting this result in (9) and simplifying, we have the following bound on $Q(d, T, C)$.

$$\binom{C}{k}\left(1-\frac{k}{C}\right)^T\left(1-\frac{1}{2}\binom{C}{k}\left(1-\frac{1}{C-k}\right)^T\right) \leq Q(d, T, C) \tag{12}$$

When the term $\binom{C}{k}(1-1/(C-k))^T$ is small, the upper and lower bounds match. The result follows from the substitution of $k = C - d + 1$. $\square$

The bounds are tight when $\binom{C}{d-1}(1 - 1/(d - 1))^T$ is less than 1. This is the case when d is close to C, if T is on the order of $C \ln C/\epsilon$.

Comparisons have been made between the exact value of $Q(d, T, C)$ and the approximation given by the upper bound in Theorem A.1. (See Figure 7.) This has been done by determining the smallest value of d that ensures that $Q(d, T, C) \leq .01$ for a variety of values of T and C. The exact value of $Q(d, T, C)$ has been computed from a recurrence for $P(d, n)$ [DeHon and Wilson 2004] which is the probability that exactly d distinct NWs will occur in n trials when C codes are used.

$$\begin{aligned}
P(1,1) &= 1\\
P(d,1) &= 0 \text{ (for } d > 1)\\
P(1,n) &= (1/C) * P(1, n-1)\\
P(d,n) &= (d/C)P(d, n-1) + (1-(d-1)/C)P(d-1, n-1)
\end{aligned}$$

As the data indicate, the upper bound gives very good results. We use it in our analysis because of the simplicity that it offers.

Corollary A.1. *The number of trials T_C needed to ensure that the probability $Q(C, T, C)$ of failing to produce all $C \geq 2$ coupons is at most ϵ satisfies the following bounds.*

$$\frac{C}{(1+1/C)}\ln(C/Q(d, T, C)) \leq T \leq C\ln(C/Q(d, T, C))$$

If the number of trials T satisfies $T \geq C \ln(C/\epsilon)$, all C coupons will be drawn with probability at least $1 - \epsilon$.

Proof. The bounds given on $Q(d, T, C)$ given in Theorem A.1 are specialized to $d = C$. Using the well known inequality $(1 - x) \leq e^{-x}$, the upper bound becomes $Q(C, T, C) \leq Ce^{-T/C}$. Using this inequality and the inequality $(1-x) \geq e^{-x(1+x)}$ which holds for $0 \leq x \leq .65$, the lower bound becomes the following.

$$Q(C, T, C) \geq C\left(1 - \frac{1}{C}\right)^T \left(1 - \frac{C}{2}\left(1 - \frac{1}{C-1}\right)^T\right)$$
$$\geq Ce^{-(T/C)(1+1/C)}\left(1 - \frac{C}{2}e^{-T/(C-1)}\right)$$

If $T \geq (C-1)\ln C$, the term in parentheses is at least 1/2. In this case, the lower bound becomes $Q(C, T, C) \geq (C/2)e^{-(T/C)(1+1/C)}$ from which the lower bound on T follows. It remains to show that, when T satisfies this lower bound, T also satisfies $T \geq (C - 1)\ln C$. This only requires that $Q(C, T, C)$ be at most 1/2.

Set this upper bound to be at most ϵ when $d = C$ from which the desired blackconclusion follows. □

Lemma A.2. *The probability $Q(d, T, C)$ that more than T trials are needed to produce d different coupons satisfies the following bound when $C \geq 1.54(d - 1)$.*

$$Q(d, T, C) \leq \binom{C}{d-1}\left(\frac{d-1}{C}\right)^T \leq e^{-(T-d+1)\ln(C/(d-1))+(d-1)(1-\left((d-1)/C)^2\right)} \quad (13)$$

The bound of Lemma A.2 is very tight. As shown in Figure 8, it gives an estimate on the number of distinct codewords to appear in T trials with probability .99, that is, 0, 1, 2, or 3 smaller than the estimate given in Lemma A.1 which is already tight.

Proof. It is well known that $\binom{C}{k} \leq e^{CH(k/C)}$, where $H(x) = -x\ln x - (1 - x)\ln(1 - x)$. To see this, observe that when $\rho \geq 0$, the following bounds apply.

$$\binom{C}{k} \leq \sum_{j=k}^{C}\binom{C}{j} \leq e^{-\rho k}e^{\rho k}\sum_{j=k}^{C}\binom{C}{j} \leq e^{-\rho k}\sum_{j=1}^{C}\binom{C}{j}e^{\rho j} = e^{-\rho k}(e^{\rho} + 1)^C$$

The result follows by setting $\rho = \ln k/(C - k)$.

Combining this bound with Theorem A.1, we have $Q(d, T, C) \leq e^{E(d,T,C)}$, where

$$E(d, T, C) = (T - (d - 1))\ln\left(\frac{d-1}{C}\right) - C\left(1 - \frac{d-1}{C}\right)\ln\left(1 - \frac{d-1}{C}\right). \quad (14)$$

It is easy to show that $-\ln(1 - x) \leq x(1 + x)$ when $x \leq .65$ from which the following holds when $C \geq 1.54(d - 1)$.

$$-C\left(1 - \frac{d-1}{C}\right)\ln\left(1 - \frac{d-1}{C}\right) \leq C\left(1 - \frac{d-1}{C}\right)\frac{d-1}{C}\left(1 + \frac{d-1}{C}\right) \quad (15)$$
$$\leq (d - 1)\left(1 - \left(\frac{d-1}{C}\right)^2\right) \quad (16)$$

Combining the above bounds we have the desired result. □

	T									
C	10	20	30	40	50	60	70	80	90	100
20	0	1	1	1	1	1	0	0	0	0
40	0	1	2	1	1	1	1	1	1	0
60	1	0	1	2	2	1	2	2	2	1
80	1	1	1	1	2	2	2	2	2	2
100	0	0	1	1	1	2	2	2	2	2
120	0	1	1	1	1	2	2	2	2	2
140	0	1	1	1	1	1	2	2	2	2
160	0	0	0	1	2	1	2	2	3	2
180	1	0	1	1	2	2	2	2	2	3
200	1	1	1	1	1	2	2	2	2	3

Fig. 8. The difference between the value of $\boldsymbol{n}$ shown in Figure 7 and that computed from the approximation $Q(d,T,C) \leq e^{-(T-d+1)\ln(C/(d-1))+(d-1)(1-((d-1)/C)^2)}$ when d, the number of distinct codewords, is chosen so that d distinct coupons occur in T trials with probability at least .99.

The bound on $E(d,T,C)$ given in (13) can be further simplified, as shown in the following. As this result indicates, as T is decreased from $2d$ to guarantee that $Q(d,T,C) \leq \epsilon$, the value of C needs to grow rapidly. The lower bound on C, given as follows, requires that $C \geq ed, e = 2.718$. This bound is weaker than that implied by (14) because the term $-(d-1)^3/C^2$ has been dropped. The error introduced by this change is largest in the range $d \leq T \leq 2d$ when $T = 2d$.

Corollary A.2. *When $T \leq 2d$ and $d \geq 4$, if the number of coupons, C, satisfies the following bound, at least d distinct coupons will occur in T trials with probability at least $1-\epsilon$.*

$$C \geq (d-1)e^{((d-1)-\ln(\epsilon))/(T-d+1)}$$

Proof. Let $\overline{E(d,T,C)}$ be the exponent in the bound of Lemma A.2. Clearly,

$$\begin{aligned}\overline{E(d,T,C)} &\leq (T-(d-1))\ln\left(\frac{d-1}{C}\right)+(d-1)\left(1-\left(\frac{d-1}{C}\right)^2\right)\\ &\leq (T-(d-1))\ln\left(\frac{d-1}{C}\right)+(d-1).\end{aligned}$$

The bound of the corollary follows when $\overline{E(d,T,C)} \leq \ln(\epsilon)$ and $C \geq 1.54(d-1)$. Examination of the lower bound on C demonstrates that $C \geq 1.54(d-1)$, when $d \geq 3$ for all $\epsilon \leq 1$. □

A.3 The Chernoff Bound

Because the probability in Case (b) is harder to compute, we use the Chernoff bound [Chernoff 1960]. This bound uses the moment generating function $g(\rho) = \overline{e^{\rho x}} = \sum_x P(x)e^{\rho x}$ and is sketched here.

$$g(\rho) = \sum_x P(x)e^{\rho x} \geq \sum_{x \geq x_0} P(x)e^{\rho x} \geq e^{\rho x_0}\sum_{x\geq x_0} P(x),$$

when $\rho \geq 0$. This implies that

$$P_r[x \geq x_0] \leq e^{-\rho x_0}g(\rho) = e^{E(\rho)}, \tag{17}$$

where $E(\rho) = -\rho x_0 + \ln g(\rho)$. Since $\rho \geq 0$ is arbitrary, it is chosen to minimize $E(\rho)$, a convex function[1] whose minimum occurs at ρ for which $\frac{dE(\rho)}{d\rho} = 0$.

A.4 Bound on the Tail of the Binomial Distribution

LEMMA A.3. *Let x be the sum n 0-1 valued variables with value 1 occurring with probability p. Then,*

$$P_r[x \leq \theta np] \leq e^{-np(1-\theta+\theta\ln\theta)},$$

where $0 \leq \theta < 1$ and np is the mean value of the sum.

PROOF. We apply the Chernoff bound of (17) to the binomial distribution $P(x) = \binom{n}{x} p^x (1-p)^{n-x}$ that models the sum of n independent, identically-distributed random variables that have value 1 with probability p, and 0 with probability $1-p$. In this case, $g(\rho) = (e^\rho p + (1-p))^n$ and $E(\rho)/n = -\rho\alpha + \ln(e^\rho p + (1-p))$ where $\alpha = x_0/n$. Without loss of generality, let $\alpha = \theta p$ or $x_0 = \theta np$. Clearly, the minimum value of $E(\rho)$ occurs when ρ satisfies

$$e^\rho p(1-\alpha) = \alpha(1-p).$$

It follows that $E(\rho)/n = (1-\alpha)\ln[(1-p)/(1-\alpha)] + \alpha \ln p/\alpha$.

We show that $E(\rho)/n \leq -p(1-\theta+\theta\ln\theta)$ when $\alpha = \theta p$, $0 \leq \theta \leq 1$. Since $E(\rho)/n = (1-\theta p)(\ln(1-p) - \ln(1-\theta p)) - \theta p \ln\theta$, to show the desired result, it suffices to show that $f(\theta) = (1-\theta p)(\ln(1-p) - \ln(1-\theta p)) + p(1-\theta) \leq 0$. Since $\frac{\partial f(\theta)}{\partial\theta} = -p\ln[(1-p)/(1-\theta p)] > 0$ and $f(1) = 0$, the result follows directly. □

B. A BOUND ON A BINOMIAL COEFFICIENT

LEMMA B.1. *Let $H(x) = -x\log_2 x - (1-x)\log_2(1-x)$, the binary entropy function. Then,*

$$\binom{b}{h} \leq 2^{bH(h/b)}.$$

PROOF. The following inequality follows from the binomial theorem when $x, y \geq 0$.

$$\binom{b}{h} x^h y^{b-h} \leq (x+y)^b$$

Rewriting this inequality as shown.

$$\binom{b}{h} \leq \left(\frac{x}{x+y}\right)^{-h} \left(\frac{y}{x+y}\right)^{-(b-h)},$$

and letting $x = h$ and $y = b-h$, we have the desired conclusion. □

REFERENCES

BJÖRK, M. T., OHLSSON, B. J., SASS, T., PERSSON, A. I., THELANDER, C., MAGNUSSON, M. H., DEPPERT, K., WALLENBERG, L. R., AND SAMUELSON, L. 2002. One-dimensional steeplechase for electrons realized. *Nano Letters 2*, 2, 87–89.

[1]This follows because $e^{-\rho x_0} g(\rho) = \overline{e^{\rho(x-x_0)}}$ whose second derivative is positive.

CHEN, Y., JUNG, G.-Y., OHLBERG, D. A. A., LI, X., STEWART, D. R., JEPPESON, J. O., NIELSON, K. A., STODDART, J. F., AND WILLIAMS, R. S. 2003. Nanoscale molecular-switch crossbar circuits. *Nanotechnology 14*, 462–468.

CHEN, Y., OHLBERG, D. A. A., LI, X., STEWART, D. R., WILLIAMS, R. S., JEPPESEN, J. O., NIELSEN, K. A., STODDART, J. F., OLYNICK, D. L., AND ANDERSON, E. 2003. Nanoscale molecular-switch devices fabricated by imprint lithography. *Applied Physics Letters 82,* 10, 1610–1612.

CHERNOFF, H. 1960. A measure of asymptotic efficiency for tests of a hypothesis based on a sum of observations. *Ann. Math. Stat. 23*, 493–507.

CHOU, S. Y., KRAUS, P. R., AND RENSTROM, P. J. 1996. Imprint lithography with 25-nanometer resolution. *Science 272,* 5258, 85–87.

COLLIER, C. P., MATTERSTEIG, G., WONG, E. W., LUO, Y., BEVERLY, K., SAMPAIO, J., RAYMO, F., STODDART, J. F., AND HEATH, J. R. 2000. A [2]catenate-based solid state electronically reconfigurable switch. *Science 290*, 1172–1175.

COLLIER, C. P., WONG, E. W., BELOHRADSKÝ, M., RAYMO, F. M., STODDART, J. F., KUEKES, P. J., WILLIAMS, R. S., AND HEATH, J. R. 1999. Electronically configurable molecular-based logic gates. *Science 285*, 391–394.

CUI, Y., LAUHON, L., GUDIKSEN, M., WANG, J., AND LIEBER, C. M. 2001. Diameter-controlled synthesis of single crystal silicon nanowires. *Applied Physics Letters 78,* 15, 2214–2216.

DEHON, A. 2003. Array-based architecture for FET-based, nanoscale electronics. *IEEE Trans. Nanotech. 2*, 1 (Mar.), 23–32.

DEHON, A., GOLDSTEIN, S. C., KUEKES, P., AND LINCOLN, P. 2005. Nonphotolithographic nanoscale memory density prospects. *IEEE Trans. Nanotech. 4*, 2, 215–228.

DEHON, A., LINCOLN, P., AND SAVAGE, J. E. 2003. Stochastic assembly of sublithographic nanoscale interfaces. *IEEE Trans. Nanotech. 2*, 3, 165–174.

DEHON, A. AND WILSON, M. J. 2004. Nanowire-based sublithographic programmable logic arrays. In *Proceedings of the International Symposium on Field-Programmable Gate Arrays*. 22–24.

DEKKER, C. 1999. Carbon nanotubes as molecular quantum wires. *Physics Today*, 22–28.

DUAN, X., HUANG, Y., AND LIEBER, C. M. 2002. Nonvolatile memory and programmable logic from molecule-gated nanowires. *Nano Letters 2,* 5, 487–490.

FELLER, W. 1968. *An Introduction to Probability Theory and Its Applications*. Vol. 1, 3rd Ed. John Wiley & Sons, Inc., New York, NY.

GOJMAN, B., RACHLIN, E., AND SAVAGE, J. E. 2004. Decoding of stochastically assembled nanoarrays. In *Proceedings of the International Symposium on VLSI*. Lafayette, LA.

GUDIKSEN, M. S., LAUHON, L. J., WANG, J., SMITH, D. C., AND LIEBER, C. M. 2002. Growth of nanowire superlattice structures for nanoscale photonics and electronics. *Nature 415*, 617–620.

HEATH, J. R., LUO, Y., AND BECKMAN, R. 2005. System and method based on field-effect transistors for addressing nanometer-scale devices. US Patent Application 20050006671.

HEATH, J. R. AND RATNER, M. A. 2003. Molecular electronics. *Physics Today 56*, 5, 43–49.

HUANG, Y., DUAN, X., WEI, Q., AND LIEBER, C. M. 2001. Directed assembly of one-dimensional nanostructures into functional networks. *Science 291*, 630–633.

ITRS. 2001. International technology roadmap for semiconductors. Available at http://public.itrs.net.

JOHNSTON-HALPERIN, E., BECKMAN, R., LUO, Y., MELOSH, N., GREEN, J., AND HEATH, J. 2004. Fabrication of conducting silicon nanowire arrays. *Applied Physics Letters 96,* 10, 5921–5923.

KIM, F., KWAN, S., AKANA, J., AND YANG, P. 2001. Langmuir-Blodgett nanorod assembly. *J. Amer. Chem. Soc. 123*, 18, 4360–4361.

KUEKES, P. J., WILLIAMS, R. S., AND HEATH, J. R. 2000. Molecular wire crossbar memory. US Patent Number 6, 128, 214.

LAUHON, L. J., GUDIKSEN, M. S., WANG, D., AND LIEBER, C. M. 2002. Epitaxial core-shell and core-multishell nanowire heterostructures. *Nature 420*, 57–61.

MELOSH, N. A., BOUKAI, A., DIANA, F., GERARDOT, B., BADOLATO, A., PETROFF, P. M., AND HEATH, J. R. 2003. Ultrahigh-density nanowire lattices and circuits. *Science 300*, 112–115.

MORALES, A. M. AND LIEBER, C. M. 1998. A laser ablation method for synthesis of crystalline semiconductor nanowires. *Science 279*, 208–211.

RACHLIN, E., SAVAGE, J. E., AND GOJMAN, B. 2005. Analysis of a mask-based nanowire decoder. In *Proceedings of the International Symposium on VLSI*. Tampa, FL.

RUECKES, T., KIM, K., JOSELEVICH, E., TSENG, G. Y., CHEUNG, C.-L., AND LIEBER, C. M. 2000. Carbon nanotube-based nonvolatile random access memory for molecular computing. *Science 289*, 94–97.

SAVAGE, J. E. 1998. *Models of Computation: Exploring the Power of Computing*. Addison Wesley.

SAVAGE, J. E., RACHLIN, E., DEHON, A., LIEBER, C. M., AND WU, Y. 2005. Radial addressing of nanowires. Submitted for publication.

WHANG, D., JIN, S., AND LIEBER, C. M. 2003. Nanolithography using hierarchically assembled nanowire masks. *Nano Letters 3*, 7, 951–954.

WILLIAMS, R. S. AND KUEKES, P. J. 2001. Demultiplexer for a molecular wire crossbar network. US Patent Number 6,256,767.

WU, Y., FAN, R., AND YANG, P. 2002. Block-by-block growth of single-crystal Si/SiGe superlattice nanowires. *Nano Letters 2,* 2, 83–86.

XIA, Y. AND WHITESIDES, G. M. 1998. Soft lithography. *Annu. Rev. Math. Sci. 28*, 153–184.

ZHONG, Z., WANG, D., CUI, Y., BOCKRATH, M. W., AND LIEBER, C. M. 2003. Nanowire crossbar arrays as address decoders for integrated nanosystems. *Science 302*, 1377–1379.

Received November 2004; revised March 2005, July 2005; accepted July 2005