Decoding of Stochastically Assembled Nanoarrays[*]

Benjamin Gojman, Eric Rachlin and John E. Savage[†]
Department of Computer Science
Brown University
{bgojman, erachlin, jes}@cs.brown.edu

Abstract

A key challenge that will face nanotechnologies will be controlling the uncertainty introduced by stochastic self-assembly. In this paper we explore architectural and manufacturing strategies to cope with this uncertainty when assembling nanoarrays, crossbars composed of two orthogonal sets of coded parallel nanowires. Because the encodings of nanowires that are assembled into a nanoarray cannot be predicted in advance, a discovery process is needed and specialized decoding circuitry must be employed. We have developed a probabilistic method of analysis so that various design strategies can be evaluated.

1 Introduction to Stochastic Assembly and Nanoarrays

The forty-year exponential growth in the density of integrated circuits, codified as "Moore's Law", is the result of remarkable advances in top-down manufacturing technologies, notably photolithography. Unfortunately, physical constraints and the exponential growth in manufacturing costs are expected to end the exponential growth based on these technologies within about a decade [1].

Anticipating these problems, the physical sciences community is developing new materials and devices with nanometer-sized features as well as new bottom-up (non-lithographic) technologies to manufacture chips with these materials. The new materials include carbon nanotubes (CNT) [9] and semiconducting nanowires (NW) [6, 16] with diameters measured in nanometers as well as thin molecular layers capable of forming non-volatile switches. Sets of NWs can be arranged in parallel with nanometer spacing using a form of directed self-assembly [13, 14].

One of the most promising nanotechnology architectures currently under investigation is the the **nanoarray**, a crossbar [15] (see Figure 1 (a)) formed by placing one set of parallel NWs above another at right angles and implementing switches at the crosspoints of the NWs. Materials to implement these switches have been developed and several methods to place them at crosspoints have been demonstrated [17, 5, 10]. Such crossbars can be used as programmable nanoscale logic arrays [7].

In order to provide a microsized interface to the nanoarrays, several proposals have been put forth. Each involves a parallel set of microsized address wires (AWs) being laid down perpendicular to the NWs, which have been treated with materials that will allow them to act as decoders. This allows for individual nanowires to be activated by subsets of AWs. Kuekes and Williams [18] describe a decoder based on the random deposition of gold nanoparticles over the region in which AWs and NWs intersect to make contacts between them. DeHon, Lincoln and Savage [8] describe a decoder based on modulation-doped NWs (Section 2.1) in which the AWs determine if a NW is conducting or not (see Figure 1 (b)). (Over doped regions, the intersection of a NW and an AW forms a field-effect transistor (FET).)

In the following section we provide a detailed introduction to nanoarray-based memories. In Section 3 we introduce binary reflected codes, present four strategies for exploiting the self-assembly of nanoarray crossbars, and examine tradeoffs between wasted NWs and micro-sized decoding circuitry. These tradeoffs are supported with detailed analysis of relevant probability distributions.

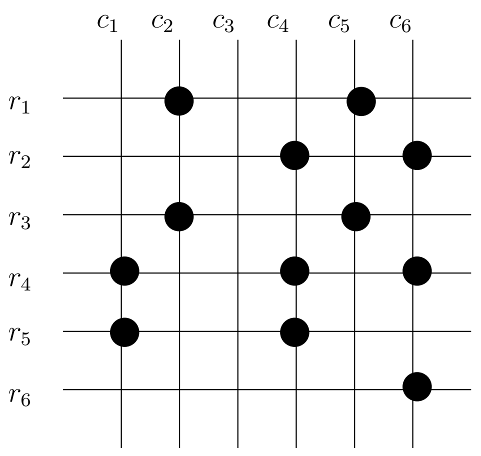
2 Overview of the Crossbar and Modulation-Doped Nanowires

This section describes modulation-doped nanowires and their role in stochastically assembled nanoarrays.

2.1 Modulation-Doped Nanowires

Crystalline semiconducting wires with nanometer dimensions [6, 16] have been grown by a vapor-liquid-solid

[*]This research was funded in part by NSF Grant CCR-0210225. [†] Contact person. Appears in Procs. ISVLSI 2004.



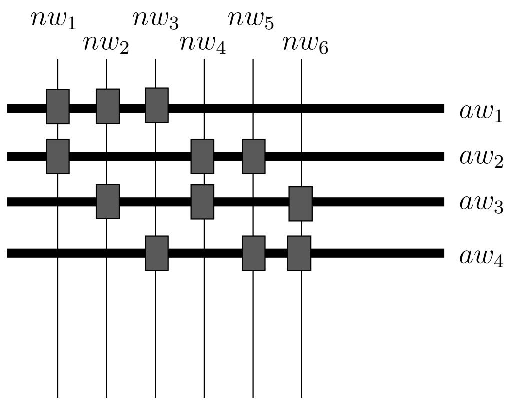


Figure 1. (a) A crossbar in which sites holding 1s are marked by dots and (b) (2,4)-hot addressing of six nanowires $\{nw_1, \ldots, nw_6\}$ with four micro-level address wires $\{aw_1, \ldots, aw_4\}$.

growth process that controls the diameter of the NWs. When the vapor level of reactants in this process is modulated over time, the semiconducting wires can be doped along the axial dimension [12, 19, 2]. The length of a doped region can be precisely controlled by the length of time during which doping reactants are in the vapor (**modulation-doping**); the transition between differently doped regions occurs over an axial length that has been shown to be as small as fractions of a nanometer. (See [8] for a discussion of the importance of this transition length.) The doped regions act as field-effect transistors (FETs). A NW can be gated by a high electric field provided by a microwire at right angles to the NW, as suggested in Figure 1 (b).

A practical method of controlling NWs is $(\boldsymbol{h}, \boldsymbol{b})$**-hot addressing**. Each NW has b addressable regions exactly h of which are doped. Subsets of this set of codes $\mathcal{C}$ may also be used. It follows that a NW is conducting unless one of its doped regions is adjacent to an address wire carrying an electric field of sufficient magnitude to reduce its conductance to near zero. Thus, exactly one NW in the set (h, b)-hot addressable NWs will be conducting if h of the address wires carry a large electric field. In order to create a functioning memory, external binary addresses must be mapped to the internal addresses of the individual nanowires.

2.2 Stochastic Self-Assembly of NWs

Since NWs are too small to be manufactured lithographically and will be too numerous to manipulate at the atomic level, some others means of assembling them is necessary.

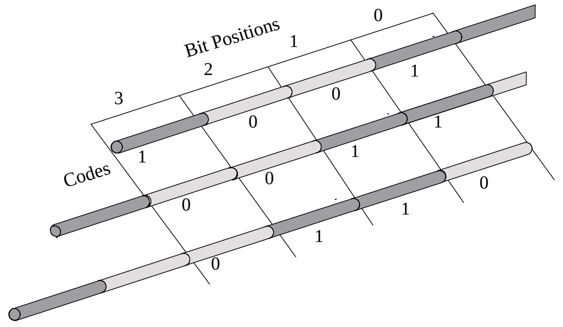


Figure 2. Three shifted copies of a nanowire with doping along its length.

Directed self-assembly of undifferentiated NWs into sets of parallel wires has been demonstrated [13, 14]. The assembly process mixes a set of modulation-doped NWs in a solution and flows the mixture into a trough. This process aligns the NWs in parallel just as twigs line up as they flow downstream in a river [20]. NWs are then transferred to a chip. We assume that NW codes form a subset the (h, b)-hot addresses, and that each address is chosen at random with equal probability.

2.3 Stochastic Alignment of NWs

The self-assembly process cannot guarantee the lengthwise alignment of the doped regions on individual NWs

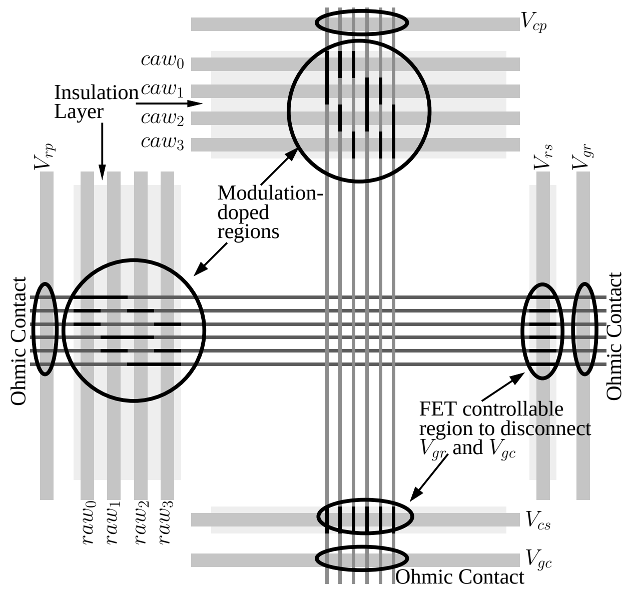


Figure 3. Nanoarray interfaced with address modulation-doped NWs.

with microscale address wires, a fact that must be taken into account when assigning codes to NWs. We cope with this problem in the manner proposed in [8], namely, by assigning more potentially-doped regions to NWs than there are address wires. As a NW shifts relative to the address wires, different coded regions line up with the address wires, as suggested in Figure 2, and generally results in one of many different doping patterns being associated with each NW. For example, if multiple copies of some (h, b)-hot address are placed on a nanowire in a repeating pattern, then when the nanowire is shifted, a new (h, b)-hot address is produced.

2.4 Forming Crossbars

A crossbar is formed by applying the self-assembly process twice, a second set of NWs is placed above and at right angles to a first set (see Figure 3). This two-step process will require trimming of NWs that extend beyond the array, which can be done lithographically.

Switches are defined at the crosspoints of NWs by either depositing a thin film of a supramolecule between the two parallel sets of wires [15, 3] or by coating one set of NWs with such a material (a form of radial doping) [10]. In both cases when a large field is applied across the switcheable material it becomes conducting (represented by 1) or non-conducting (represented by 0) depending on the polarity of the field. The conductance of the material at each crosspoint can be sensed by applying a field large enough to detect current flow but not large enough to change the state of the material.

This memory technology will also support writing blocks of 1s (**stores**) or 0s (**restores**) at the intersection of rows and columns of a crossbar. To execute such an operation, the rows and columns are selected and either a large positive or negative voltage is applied. The complexity of programming nanoarrays has been studied [11].

Observe that two NWs with the same encoding attached to a common ohmic region behave as a single NW with that encoding. When this happens the effective number of NWs will be less than the actual number.

2.5 Crossbar Connection Architectures

To use a standard crossbar, an ohmic contact is needed at both ends of each set of parallel NWs, as shown in Figure 3, so that it is possible for current to flow through an individually addressed NW. It is also necessary to disconnect one of these two ohmic contacts so that current can flow from a NW in one parallel set to an orthogonal NW in the other parallel set and be sensed. This can be achieved by doping the ends of each NW just inside the “ground” ohmic contacts (associated with V_{gc} and V_{gr}) so that each NW can be logically disconnected from its ohmic contact by a FET controllable region.

Hybrid crossbars [8] use multiple ohmic regions, that is, the NWs in each parallel set are divided into sets of equal size and one ohmic region is attached to one end of each set and a common ohmic region to the other ends. (See Figure 4.) To address a single NW, an ohmic region is activated and a NW address for a NW connected to the ohmic region is chosen. As shown in [8] and discussed below, hybrid crossbars provide several ways in which the size of the code space needed to realize memories can be greatly reduced.

3 Designing Nanoarray Memories

A nanoarray-based memory, NanoMem, is an $N \times N$ nanoarray that has an external circuit, LithoMem, to map external binary addresses to internal addresses that are generally unknown in advance. Each nanoarray has two sets of b address wires and two sets of m ohmic region.

NanoMems will have to be constructed by a stochastic process. We must deal with the uncertainty that this process brings. Architectural choices must be made to ensure that with high probability all nanowire configurations result in functioning memories. In the following section we explore tradeoffs between C, m, w, and the size of LithoMem.

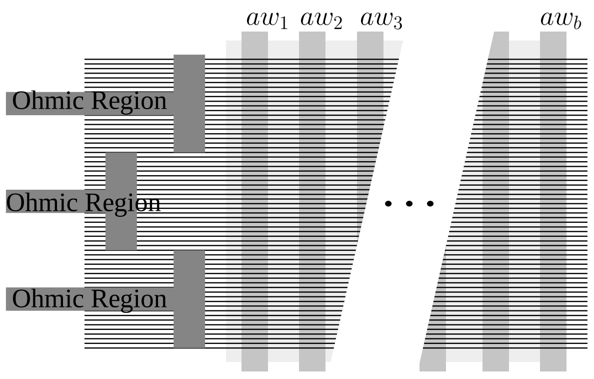


Figure 4. Nanoarray interface using multiple ohmic regions.

3.1 Binary Reflected Code

Section 2 introduces (h, b)-hot encoding. Recall that if an (h, b)-hot encoding is repeated multiple times, then a shift of a NW length-wise produces another valid encoding. Mapping external binary addresses to (h, b)-hot encodings is difficult. For this reason we prefer binary reflected codes defined as follows.

A standard binary n-tuple $\boldsymbol{x} = (x_{k-1}, \ldots, x_1, x_0)$ is encoded in the binary reflected code as $\boldsymbol{z} = \boldsymbol{x}\overline{\boldsymbol{x}} = (x_{k-1}, \ldots, x_1, x_0, \overline{x}_{k-1}, \ldots, \overline{x}_1, \overline{x}_0)$. A 1 in a codeword $\boldsymbol{z}$ corresponds to a doped region in a NW. Codewords generated this way form a subset of the set of $(k, 2k)$-hot encodings. Translating an external binary address to such codewords is trivial.

The set of binary reflected codewords is closed under shifts when an encoding is repeated along the length of a NW. For example, when $k = 6$, multiple cyclic shifts of the codewords 110001001110 and 110011001100 produce other codewords. Under shifts the first codeword maps into 12 distinct codewords whereas the second maps into four distinct codewords. The periods of codewords under cyclic shift are given below. Since each codeword is equally likely to undergo each cyclic shift, any one of the codewords resulting from the cyclic shift of a codeword (its **equivalence class**) can be used as a **seed** to generate all other codewords in the class.

Lemma 3.1 *Let $d_2(n)$ be the largest power of 2 dividing n. The period of $\boldsymbol{z}$, a codeword in the binary reflected code, divides $2k$ but not k. The set $\mathcal{P}(k)$ of periods of codewords satisfies*

$$\mathcal{P}(k) = 2\, d_2(k) * \mathcal{D}(k/d_2(k))$$

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.

There are 2^k codewords of length $2k$. The number of equivalence classes that have period p for $p \in \mathcal{P}(k)$ is given below. Clearly, the smallest period is $2d_2(k)$. One seed is needed for each equivalence class. The number $\nu_k(p)$ of equivalence classes of codewords of period p is given below.

Theorem 3.1 *The number of equivalence classes of codewords of length $2k$ that have period p but no smaller period, $\nu_k(p)$, satisfies the following relation where the sum is over p in $Q(p) = \{q \mid q \in \mathcal{P}(k)$ such that q strictly divides $p\}$.*

$$\nu_k(p) = \begin{cases} 2^{p/2}/p & p = 2d_2(k) \\ \left(2^{p/2} - \sum q\nu_k(q)\right)/p & \textit{otherwise} \end{cases}$$

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

Example 3.1 *Consider codewords $\boldsymbol{z} = \boldsymbol{x}\overline{\boldsymbol{x}}$ of length $2k = 36$. Since $d_2(18) = 2$, $\mathcal{P}(18) = 4 * \mathcal{D}(9) = 4 * \{1, 3, 9\} = \{4, 12, 36\}$. The number of different equivalence classes with these periods is $\nu_{18}(4) = 1$, $\nu_{18}(12) = \left(2^6 - 4\,\nu_{18}(4)\right)/12 = 5$, and $\nu_{18}(36) = \left(2^{18} - 4\,\nu_{18}(4) - 12\,\nu_{18}(12)\right)/36 = 7,280$. 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.*

In general approximately $2^k/2k$ seeds suffice to generate all encodings of length $2k$. This is an important consideration in the manufacturing of encoded NWs.

3.2 External Address Decoding Strategies

From the previous section we see that the codes appearing on the NWs can be thought of as random integers in $\{1, \ldots, C\}$ each selected with equal probability. Listed below are a few strategies for choosing values for C, m, the number of ohmic regions, and w, the number of NWs per ohmic region. (Other cases are also possible.) In all cases, the goal is to have n_A uniquely addressable wires (two wires are uniquely addressable if they have different codes or lie in different ohmic regions). The lower limit on the size of w is determined by the lithographic technology in use.

S_a: Choose w and then choose C to be large enough so that every region is very likely to contain no duplicate NW encodings. With high probability there will be no wasted nanowires and $n_A = N = mw$.

S_b: Choose w and C such that all ohmic regions have at least $d \leq w$ distinct NW encodings with high probability. In this case, $n_A = dm$; $(wm - n_A)$ NWs are wasted. This strategy weakens the requirement that each NW attached to each ohmic region has a distinct encoding, thereby reducing the value of C needed to ensure that $n_A = dm$ NWs can be addressed.

S_c: Choose w and C so that with high probability every NW encoding is present in each ohmic region ($n_A = mC$). This will waste many NWs but makes decoding very simple, especially when binary reflected codes are used. For large mw, the decoding process becomes exceptionally simple. $n_A = mC$ NWs can be addressed by using $\log_2 m$ external bits to choose an ohmic region and $\log_2 C$ bits to activate a single NW address.

S_d: Choose w, C and m so that with high probability each NW encoding appears at least once in at least p ohmic regions. In other words, p complete sets of C encodings can be identified with high probability. This strategy is intermediate between S_c and the other two strategies.

3.3 Bounds on Probability Distributions

Before doing a quantitative analysis of the four design strategies described above, we present bounds on probabilities of events that arise in their use. These bounds have been shown to be very tight. Thus, they can be used with confidence to explore other design strategies. The first two probabilities apply to the case when each NW is drawn with replacement from an ensemble of size C according to the uniform distribution. The third is a generic Chernoff bound [4] on the tail of a binomial distribution.

Lemma 3.2 *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, $w \geq 10$.

Lemma 3.3 *When $d \geq 3$, the probability that fewer than d distinct NW encodings will occur among w NWs satisfies the following bound where $\phi(d, C)$ is the larger of $d(d-1)/2C$ and $(C+1-d)\ln(1-(d-1)/C)+d-1\}$ and $z = w+1-d$.*

$$Q(d, w, C) \leq \begin{cases} e^{-z[\ln(C/d)+\ln z/d]-\phi(d,C)+z} \\ \quad \text{when } w \leq 2d-1 \\ \\ e^{-z(\ln(C/d)+1/d)-\phi(d,C)+d^2/(d-1)} \\ \quad \text{when } w \geq 2d \end{cases} \tag{2}$$

The second bound is valid for all values of w but is stronger when $w \geq 2d$. It is also valid for all values of w when $\phi(d, C)$ is replaced by either $d(d-1)/2C$ or $(C+1-d)\ln(1-(d-1)/C)+d-1$.

Comparison of these bounds with the exact values for the probabilities shows that they provide values of w that are at least 2/3 of the exact values when $20 \leq C \leq 200$, $10 \leq w \leq 100$, and $Q(d, w, C)$ is near .01.

Lemma 3.4 *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.

3.4 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 of $(1-\epsilon)$ or more.

The area A_T of the NanoMem is approximated by the area of the nanoarray, A_{nano}, plus twice the area A_{ext} of LithoMem used along each dimension of the nanoarray. That is, $A_T \approx A_{nano} + 2\,A_{ext}$.

In strategies S_a and S_b a memory is needed to convert external addresses to the internal addresses that are present. The memory requires one word for each of the n_A addressable NW. The number of bits needed per word is equal to $\log_2 C$. In strategies S_c and S_d the entire set of internal NW encodings is known and are not stored in memories but are supplied directly. In this case, if every (h, b)-hot encoding is allowed, a complex circuit is needed to make the translation. However, if binary reflected codes are used, the circuit consists of a set of inverters. The binary reflected codes will require $2\log_2 C$ address wires. We believe that it is better to use the latter codes in all cases. Case S_d is special in that the assignment of external binary addresses to ohmic regions is not known ahead of time and must be stored in a memory.

Let λ_{nano} and λ_{litho} be the pitch of NWs and lithographic wires, respectively, and let σ be the area of a lithographic-level memory cell. Then, if LithoMem consists of just a memory with n_A words, one for each addressable NW in each dimension, each of β bits, then $A_{ext} \approx (n_A\beta)\sigma$. If it also contains a standard decoder for the ohmic regions (as it will except for case S_d), we add area $2\lambda_{litho}^2 m\log_2 m$. The area of the nanoarray itself satisfies $A_{nano} = (2\lambda_{litho}\log_2 C + N\lambda_{nano})^2$. Thus, $A_T \approx A_{nano} + 2\,A_{ext}$ and

$$\begin{aligned} A_T \approx\ & 2(n_A\beta)\sigma + 2\lambda_{litho}^2 m\log_2 m \\ & +(2\lambda_{litho}\log_2 C + \lambda_{nano}N)^2 \end{aligned}$$

3.4.1 Performance of Strategy S_a

In strategy S_a there are no wasted wires, that is, $n_A = N$. The NWs can be addressed with external binary addresses by supplying $\log_2 m$ bits to a standard decoder to activate one ohmic region and by supplying all the bits to a programmable memory to activate individual NWs within ohmic regions. In this case $\beta = \log_2 C$. The area A_T required by strategy S_a is given below.

$$\begin{aligned} A_T \approx\ & 2(n_A \log_2 C)\sigma + 2\lambda_{litho}^2 m \log_2 m \\ & + (2\lambda_{litho} \log_2 C + \lambda_{nano} n_A)^2 \end{aligned}$$

To evaluate this formula we require a bound on C in terms of the number of NWs.

Theorem 3.2 *Strategy S_a succeeds with probability $1 - \epsilon$ when C satisfies the following.*

$$C \geq n_A(w-1)/(-2\ln(1-\epsilon))$$

Proof Under strategy S_a 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 3.2 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 $n_A = mw$ and $N = n_A$, the result follows. ■

3.4.2 Performance of Strategy S_b

In strategy S_b we consider just the case of $d = w/2$, which is indicative of the general requirement on d. Thus, half of the NWs are addressed and $n_A = N/2$. The formula for the size of the external circuit is the same as in strategy S_a although the size of C needed is much smaller, thereby greatly reducing the size of the circuit.

$$\begin{aligned} A_T \approx\ & 2(n_A \log_2 C)\sigma + 2\lambda_{litho}^2 m \log_2 m \\ & + (2\lambda_{litho} \log_2 C + 2\lambda_{nano} n_A)^2 \end{aligned}$$

To evaluate this formula we require a bound on C in terms of the number of NWs.

Theorem 3.3 *Strategy S_b succeeds with probability $1-\epsilon$ when C is chosen to satisfy the following bound where $\ln(1-\delta) = \ln(1-\epsilon)/m$.*

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

Proof With strategy S_b let d, w and C be chosen so that for $(w+1)/2w \leq \alpha < 1$ there are at least $d = \alpha w$ distinct NWs in each 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 $\ln(1-\delta) = \ln(1-\epsilon)/m$. We find conditions on d, w and C that suffice to ensure that strategy S_b works probability at least $1-\epsilon$.

Because $w \leq 2d-1$ we use the first bound in Lemma 3.3 but weaken it by dropping the term $\phi(d, C)$, replacing $\ln(w+1-d)\ \ln(w-d)$, and requiring that the resulting bound be less than δ. This sets conditions on d, w and C that suffice to ensure that strategy S_b works with the stated probability. Substituting $d = \alpha w$ gives

$$\begin{aligned} -\ln(\delta) < ((1-\alpha)w+1)(\ln(C) + \ln((1-\alpha)w+1) \\ - 2\ \ln(\alpha w)) - ((1-\alpha)w+1) \end{aligned}$$

which is equivalent to the following.

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

and is the desired conclusion when $\alpha = (w+1)/2w$. ■

Comparison of the area bound for cases S_b with that of S_a indicates that the requirement on C is substantially reduced. In fact, using the lower bounds in Theorems 3.2 and 3.3, C is reduced from about $50n_A w$ to $(w+1)/2e(100m)^{2/(w+1)}$ for $\epsilon = .01$ since δ is very close to ϵ/m. If $w \geq 10$ and $m \leq 5,000$, $(w+1)/2e(100m)^{2/(w+1)} \leq 3.14(w+1)m^{.182} \leq 17w$. It follows that the size of C is reduced from $50n_A w$ to at most $17w$ when $m \leq 5,000$ (or $n_A \leq 27,500$), a reduction by a factor of about $3n_A$. Thus, S_b is clearly superior to S_a.

In anticipation of the analysis of strategy S_c we identify the dominant terms in the expression for A_T. The first term in $(2\lambda_{litho}\log_2 C + \lambda_{nano} n_A)^2$ is clearly dominated by the second term and is ignored. Comparing the first term, $2(n_A \log_2 C)\sigma$ against $(\lambda_{nano} n_A)^2$, the latter term dominates when $\lambda_{nano}^2 n_A$ is much larger than $2\sigma \log_2 C$.

3.4.3 Performance of Strategy S_c

Strategy S_c uses no external memory. The input bits are split between the ohmic regions and the NWs. The ohmic region bits are supplied to a standard decoder. If (h, b)-hot encodings are used, the NW bits are supplied to a circuit that translates them into (h, b)-addresses. A simpler solution uses the binary reflected code and only requires one inverter for each of the external bits. Since $C = n_A/m$, the area required satisfies the following.

$$\begin{aligned} A_T \approx 2\lambda_{litho}^2 m \log_2 m + (2\lambda_{litho} \log_2(n_A/m) \\ + \lambda_{nano} N)^2 \end{aligned}$$

We compare the area of a nanoarray-based memory, A_T, to that of a lithographic-level memory with the same storage capacity, namely, n_A^2, which uses area $n_A^2\sigma$ and show that the nanoarray-based memory is superior. We use the following result.

Theorem 3.4 *Strategy S_c is successful with probability $1-\epsilon$ when N is chosen to satisfy the following bound where $(1-\delta)^m = 1-\epsilon$.*

$$N \geq n_A(\ln(n_A/m) - \ln\delta + 3.5)$$

Proof Under strategy S_c 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 $\delta \leq -\ln(1-\epsilon)/m$. The goal is to exhibit conditions under which the upper bound to $Q \leq \delta$ for Q given by Lemma 3.3 when $d = C$. Since $w \geq 2d+1$ will be required, we use the second bound. Also, since $d = C$, $\phi(d, C) = -\ln d + d - 1$ and the bound becomes the following when we replace $1/(d-1)$ by .5 when $d = C \geq 3$.

$$e^{-(w+1)/C + \ln C + 3.5} \leq \delta$$

This implies that $w + 1 \geq C(\ln C - \ln\delta + 3.5) \geq C(\ln C - \ln(-\ln(1-\epsilon)/m) + 3.5)$. This result is weakened by replacing $w+1$ by w. Since $N = mw$ and $n_A = mC$, the desired result follows. ■

Since the first term in the expression for A_T is small by comparison with $n_A^2\sigma$, the conclusion follows if the second term $(2\lambda_{litho}\log_2(n_A/m) + \lambda_{nano}N)^2$ is significantly smaller than $n_A^2\sigma$. Since we can expect $\lambda_{litho}/\lambda_{nano} \leq 20$, we can expect that $2\lambda_{litho}\log_2(n_A/m)$ is significantly smaller than $\lambda_{nano}N$.

The nanoarray memory uses less area than a standard memory if $N \leq n_A\sqrt{\sigma/\lambda_{nano}^2}$. Given the above lower bound on N, this is equivalent to $\ln(n_A/m) - \ln\delta + 3.5 \leq \sqrt{\sigma/\lambda_{nano}^2}$. If $\epsilon = .01$, δ is very close to $.01/m$. If our standard memory is a DRAM memory, $\sigma = \lambda_{DRAM}^2$. The 2001 ITRS roadmap [1] predicts $\lambda = 200$ for 2003. Thus, if $\lambda_{nano} = 10$, $\sqrt{\sigma/\lambda_{nano}^2} = 20$ and the condition becomes $n_A \leq 146,507$.

It follows that design strategy S_c will yield a nanoarray memory that uses less area than a standard DRAM memory if the memory has capacity less than about $2\,10^{10}$, which means this method may be practical.

We now compare strategies S_c and S_b. The former uses area approximated by $2\lambda_{litho}^2 m\log_2 m + \lambda_{nano}^2 N^2$ where $N = n_A(\ln(n_A/m) - \ln\delta + 3.5)$. The latter uses area approximated by $2(n_A\log_2 C)\sigma + 2\lambda_{litho}^2 m\log_2 m + 4\lambda_{nano}^2 n_A^2$. The former uses more area if $\lambda_{nano}^2 N^2$ is larger than $2(n_A\log_2 C)\sigma + 4\lambda_{nano}^2 n_A^2$. The first term in this last expression is insignificant by comparison with the second if n_A is large by comparison with $\sigma\log_2 C/(2\lambda_{nano}^2)$, which we assume is 400. As shown for S_b, we can set C to $3.14(w+1)m^{.182}$ when $w \geq 10$. When $m \leq 50,000$, C is at most $25w$. Thus, the second term dominates when n_A is large by comparison with $200\log_2(25w)$, which is $1,600$ when $w = 10$. It follows that S_b uses more area than S_c under this condition.

3.4.4 Performance of Strategy S_d

In strategy S_d the number of wasted wires is substantially less than strategy S_c. However, programmable external decoding circuitry is needed. The external bits are separated into two sets, one set identifying a NW and second identifying a group. The NW bits are used directly to activate a NW. To simplify this addressing problem, we use binary reflected codes. The group bits and the NW bits are used to address the external memory and select the $\log_2 m$ bits needed to activate an ohmic region. (A more efficient design exists but space prevents our presenting it here.) The area A_T required by this strategy is given below.

$$A_T \approx 2(n_A\log_2 m)\sigma + 2\lambda_{litho}^2 m\log_2 m$$
$$+(2\lambda_{litho}\log_2 C + \lambda_{nano}N)^2$$

Theorem 3.5 *Strategy S_d is successful with probability $1-\epsilon$ when C is no larger than bound given below where $.3\,N(1-w/C) \leq n_A \leq .3\,N$.*

$$n_A \geq .89\,C\ln(C/\epsilon)$$

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 3.4, 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 $\theta = .3$, $(1-\theta+\theta\ln\theta) = .3388$. The probability that any of the C codewords fails to occur in at least $\theta\overline{r}$ ohmic regions is at most $Ce^{-.3388\overline{r}}$. Since strategy S_c is successful with probability at least $1-\epsilon$, it follows this condition will be satisfied if w and C satisfy the following bound.

$$Ce^{-.3388\overline{r}} \leq \epsilon$$

When this bound holds, each of the C codewords exists in at least $\theta\overline{r}$ ohmic regions. The number of addressable wires, n_A, satisfies $n_A = \theta\overline{r}C$. We use the above inequality to bound n_A giving the following.

$$n_A \geq .89\,C\ln(C/\epsilon)$$

It follows that an upper limit on C is set once ϵ and n_A are given. Here $n_A = \theta mC(1-(1-1/C)^w)$, which we approximate when C is large. It is easy to show by induction that $w(1-w/C) \leq C(1-(1-1/C)^w) \leq w$. It follows that when C is large relative to w, n_A is close to $\theta N = \theta mw$, that is, a fixed fraction of the NWs are wasted. ■

Comparing this case to strategy S_b, we see that they differ in the area of the nanoarray itself and in the area of the external memory. The last term in A_T above, $\lambda_{nano}N$, is between $3.33\,\lambda_{nano}n_A$ and about $5\,\lambda_{nano}n_A$ if $C \geq 3w$ whereas the same term for strategy S_b is $2n_A$. Thus, if the area of the nanoarray dominates the area of the external memory, strategy S_b is superior. On the other hand, if the area of the external memories dominate, then S_d is superior to S_b if $\log_2 m \leq \log_2 C$. Otherwise S_b is superior.

4 Conclusions

We have analyzed four strategies for the self-assembly of hybrid nanoarray memories. Strategy S_a sets conditions under which it is highly likely that each NW within each ohmic region has a unique encoding. Strategy S_b modifies strategy S_a by asking that at least half of the NWs in each ohmic region have unique encodings. Strategy S_c asks that every one of the C encodings appears in each ohmic region while strategy S_d asks that each encoding appear in at least p regions.

Detailed analysis shows that strategy S_b is superior to S_a; the area occupied by a nanomemory for S_b under reasonable assumptions is expected to be 1/3 that of S_a.

When the area occupied under strategy S_c is compared to that of a standard memory, the nanomemory is superior as long as its storage capacity is limited. The limit depends on the ratio of the area σ of a standard storage cell to that of λ_{nano}^2. When it is on the order of 400, a realistic value, memories designed with S_c are more area efficient than standard memories today. Nanoarray memories constructed using this strategy do not require decoding with an external microsized memory. As a consequence, they may be more readily manufacturable in the immediate future. However, comparison with memories based on strategy S_b shows them to be inferior of use of area.

The area occupied by a nanomemory designed using strategy S_d is compared to that when strategy S_b is used. If the area occupied by the nanoarray dominates that occupied by the external memory, S_b is superior. If not, S_d is superior to S_b if $\log_2 m \leq \log_2 C$. Otherwise S_b is superior.

References

[1] Wolfgang Arden and et al. International technology roadmad for semiconductors, 2001. see http://public.itrs.net.

[2] M. T. Bjork, B. J. Ohlsson, T. Sass, A. I. Persson, C. Thelander, M. H. Magnusson, K. Depper, L. R. Wallenberg, and L. Samuelson. One-dimensional steeplechase for electrons realized. *Nano Letters*, 2(2):87–89, February 2002.

[3] Yong Chen, Gun-Young Jung, Doublas A. A. Ohlberg, Xuema Li, Duncan R. Stewart, Jon O. Jeppeson, Kent A. Nielson, J. Fraser Stoddart, and R. Stanley Williams. Nanoscale molecular-switch crossbar circuits. *Nanotechnology*, 14:462–468, 2003.

[4] H. Chernoff. A measure of asymptotic effi ciency for tests of a hypothesis based on a sum of observations. *Ann. Math. Stat.*, 23:493–507, 1960.

[5] C. P. Collier, E. W. Wong, M. Belohradský, F. M. Raymo, J. F. Stoddart, P. J. Kuekes, R. S. Williams, and J. R. Heath. Electronically confi gurable molecular-based logic gates. *Science*, 285:391–394, 1999.

[6] Y. Cui, L. Lauhon, M. Gudiksen, J. Wang, and C. M. Lieber. Diameter-controlled synthesis of single crystal silicon nanowires. *Applied Physics Letters*, 78(15):2214–2216, 2001.

[7] André DeHon. Array-based architecture for fet-based, nanoscale electronics. *IEEE Transactions on Nanotechnology*, 2(1):23–32, March 2003.

[8] André DeHon, Patrick Lincoln, and John E. Savage. Stochastic assembly of sublithographic nanoscale interfaces. *IEEE Transactions on Nanotechnology*, to appear 2003.

[9] Cees Dekker. Carbon nanotubes as molecular quantum wires. *Physics Today*, pages 22–28, May 1999.

[10] X. Duan, Y. Huang, and C. M. Lieber. Nonvolatile memory and programmable logic from molecule-gated nanowires. *Nano Letters*, 2(5):487–490, 2002.

[11] Lee-Ad J. Gottlieb, John E. Savage, and Arkady Yerukhimovich. Effi cient data storage in large nanoarrays. *submitted for publication*, 2003.

[12] Mark S. Gudiksen, Lincoln J. Lauhon, Jianfang Wang, David C. Smith, and Charles M. Lieber. Growth of nanowire superlattice structures for nanoscale photonics and electronics. *Nature*, 415:617–620, February 7, 2002.

[13] Y. Huang, X. Duan, Q. Wei, and C. M. Lieber. Directed assembly of one-dimensional nanostructures into functional networks. *Science*, 291:630–633, 2001.

[14] Franklin Kim, Serena Kwan, Jennifer Akana, and Peidong Yang. Langmuir-blodgett nanorod assembly. *Journal of the American Chemical Society*, 123(18):4360–4361, 2001.

[15] P. J. Kuekes, R. S. Williams, and J. R. Heath. Molecular wire crossbar memory. Technical Report US Patent Number 6,128,214, US Patent Offi ce, October 3, 2000.

[16] A. M. Morales and C. M. Lieber. A laser ablation method for synthesis of crystalline semiconductor nanowires. *Science*, 279:208–211, 1998.

[17] Thomas Rueckes, Kyoungha Kim, Ernesto Joselevich, Greg Y. Tseng, Chin-Li Cheung, and C. M. Lieber. Carbon nanotube-based nonvolatile random access memory for molecular computing. *Science*, 289:94–97, 2000.

[18] R. S. Williams and P. J. Kuekes. Demultiplexer for a molecular wire crossbar network. Technical Report US Patent Number 6,256,767, US Patent Offi ce, July 3, 2001.

[19] Yiying Wu, Rong Fan, and Peidong Yang. Block-by-block growth of single-crystal si/sige superlattice nanowires. *Nano Letters*, 2(2):83–86, 2002.

[20] Peidong Yang. Wires on water. *Nature*, 425:243–244, 2003.