Stochastic Assembly of Sublithographic Nanoscale Interfaces

André DeHon, *Member, IEEE*, Patrick Lincoln, and John E. Savage, *Life Fellow, IEEE*

***Abstract*—We describe a technique for addressing individual nanoscale wires with microscale control wires without using lithographic-scale processing to define nanoscale dimensions. Such a scheme is necessary to exploit sublithographic nanoscale storage and computational devices. Our technique uses modulation doping to address individual nanowires and self-assembly to organize them into nanoscale-pitch decoder arrays. We show that if coded nanowires are chosen at random from a sufficiently large population, we can ensure that a large fraction of the selected nanowires have unique addresses. For example, we show that N lines can be uniquely addressesd over 99% of the time using no more than $\lceil 2.2 \log_2(N) \rceil + 11$ address wires. We further show a hybrid decoder scheme that only needs to address $N = O(W_{\text{litho\_pitch}}/W_{\text{nano\_pitch}})$ wires at a time through this stochastic scheme; as a result, the number of unique codes required for the nanowires does not grow with decoder size. We give an $O(N^2)$ procedure to discover the addresses which are present. We also demonstrate schemes that tolerate the misalignment of nanowires which can occur during the self-assembly process.**

***Index Terms*—Bootstrapping, electronic nanotechnology, molecular electronics, nanoscale interfacing, stochastic assembly.**

I. Introduction

RECENT developments demonstrate that we can build carbon nanotubes (CNT) [1] and semiconducting nanowires (NW) [2], [3] that are just a few nanometers in diameter. Furthermore, it has been shown that self-assembly techniques can be used to produce sets of parallel NWs with nanometer spacing. One set can then be placed above another at right angles [4], [5]. The crosspoints in these arrays can act as nonvolatile switching elements [6], [7], allowing us to control and differentiate the behavior of the assembled arrays at the nanoscale. Technology of this kind may form the basis for nanoscale memory devices and even programmable nanoscale logic arrays [8].

Remarkably, the dimensions of these nanoarrays (diameter of the wires, spacing between wires) are controlled to nanometer dimensions without using direct lithographic patterning [9]. Molecular seed catalysts control the diameter and physical forces between wires control spacing.

Manuscript received April 28, 2003; revised June 1, 2003. This work was supported by the Defense Advanced Research Projects Agency Moletronics Program under Grant ONR N00014–01-0651 and by the National Science Foundation under Grant CCR-0210225.

A. DeHon is with the Department of Computer Science, California Institute of Technology, Pasadena, CA 91109 USA (e-mail: andre@acm.org).

P. Lincoln is with the Computer Science Laboratory, SRI International, Menlo Park, CA 94025 USA.

J. E. Savage is with the Department of Computer Science, Brown University, Providence, RI 02903 USA.

Digital Object Identifier 10.1109/TNANO.2003.816658

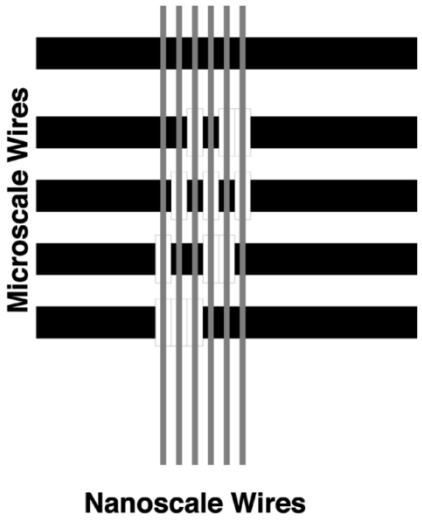

Fig. 1. Decoder bridging between microscale and nanoscale wires (not shown to typical scale); the decoder arrangement allows a small number of microscale wires to address any single nanoscale wire in a large array.

This leaves a critical weak link in our path to the construction of fully nanoscale memory and logic arrays: constructing the interface that allows us to individually address these nanoscale wires from our conventional, microscale wires. We must be able to control single NWs individually so that individual crosspoints can be programmed and addressed.

In this paper, we propose an address decoder that uses a small number of microscale control wires to selectively activate one of a large number of NWs as suggested in Fig. 1. Differently coded modulation-doped NWs (Section III) provide the independent NW addressability. Our address decoder can be assembled without relying on lithographic patterning at nanoscale dimensions by randomly mixing differently coded NWs and enabling them to self-assemble (Section V) into a parallel array at right angles to a pre-existing array of microwires using previously demonstrated flow and Langmuir–Blodgett techniques. This approach realizes a microscale-to-nanoscale interface, bridging the gap from top-down lithographic processing to bottom-up self-assembly. The differently coded modulation-doped NW-based address decoder is robust: It overcomes misalignment of NWs (Section VI), allows the customization of nanoscale programmable computing arrays to personalize behavior and tolerate faults, and directly enables reliable nanoscale memory devices (Section VII). We can discover the codes present in such a decoder with reasonable efficiency (Section VIII).

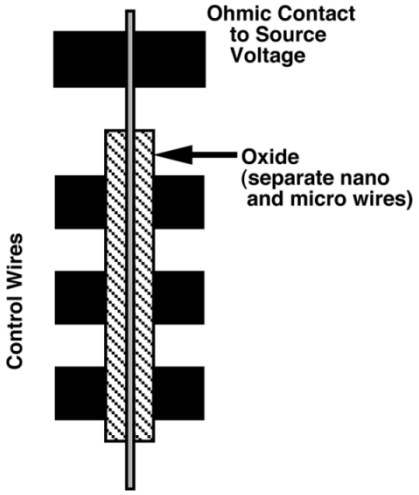

Fig. 2. NW FETs with multiple gated wire crossings serve as an AND, allowing signal flow only when all control wires have suitable voltages.

II. Prior Work

To date, only one other scheme has been proposed to address this microscale–nanoscale interface problem [10]. Kuekes and Williams describe a scheme for bridging the microscale–nanoscale gap with a decoder based on randomly deposited gold nanoparticles. The gold particles must be deposited over the region in which control and address wires intersect. The approach relies on close control of the density of deposited particles, ideally targeting half of the points of intersection. Additionally, the approach relies on strongly quantized connection values for each intersection, while imprecisely localized gold nanoparticles could lead to intermediate values that complicate the discovery approach. Consequently, the Kuekes and Williams approach comes with its own set of manufacturing challenges.

Our addressing scheme offers tighter address encoding, requires fewer novel processes, and uses standard semiconductor industry materials and dopants.

III. Modulation-Doped Coded NWs

Doped NWs act as field-effect transistors (FETs) [11], that is, conduction along the length of an NW can be controlled by an applied voltage field. For the depletion-mode p-type devices demonstrated to date, a low voltage (or no applied voltage) will allow good conduction, whereas a high applied voltage will evacuate carriers from the doped semiconductor, preventing conduction along the NW length. This allows us to build a combining logic when several conductors cross a doped NW—if all the inputs are low, there is a conduction path from one side of the crossed wires to the other; if any of the inputs are high, there will be no conduction path (see Fig. 2).

Gudiksen [12] and others [13], [14] have recently demonstrated that it is possible to control the doping profile or material composition along the axial dimension of an NW. By controlling the doping profile, we can effectively control the threshold voltage for the FET. That is, with high doping, it becomes very hard to deplete the carriers from the channel and stop conduction through the wire; consequently, the threshold voltage is high. With low doping, there are fewer carriers, allowing a low voltage to deplete the channel and stop conduction. This allows us to construct wires which are gateable in some regions but not gateable in others (see Fig. 3).

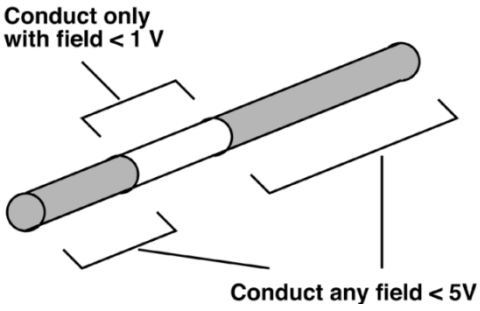

Fig. 3. Modulation doping places selective gateable regions in an NW.

The growth along the length of the NW is controlled by time. The NW crystal grows by incorporating new atoms into its lattice at one end. To control the dopant profile, we simply control the dopant concentration in the NW's growth environment over time. Consequently, we can precisely control the width of each doping region by controlling the rate of the growth reaction and the introduction of dopants into the growth atmosphere at the appropriate times. The dimensions of the doping regions are thus defined completely without lithographic processing [12].

We note from the Gudiksen experiments [12] that the transition between materials occurs over a 20-nm-length scale, while the Björk experiments [14] show subnanometer transitions between materials. Gudiksen notes that sharper transitions (< 5 nm) are likely in smaller diameter NWs. For our usage, we only need to transition between a strongly doped (conducting) region and a weakly doped region of the same semiconducting material, which should be even easier than these demonstrations. We are ultimately interested in using NWs that are just a few nanometers in diameter [2], while the lithographic scale wires will be tens of nanometers in width (e.g., 90 nm). Consequently, we expect the transition region to be small compared to the lithographic microwire pitch. Fig. 4 shows a rough band diagram.

IV. NW Coding

With the ability to modulation dope NWs, code words can be assigned to NWs. Each NW is segmented into regions that are doped as either FET-controllable or noncontrollable (see Fig. 5). When a coded NW is aligned across a set of microwires, the flow of current through the NW can be controlled. If we apply a suitably low field on all the FET-controlled regions, the NW will conduct. If we apply a high field on any of the FET-controlled regions, the NW will not conduct. Applying a high field on the non-FET controlled regions will not affect conduction. The controlling voltages are provided by control microwires, which are at right angles to the addressed NWs (see Fig. 6).

We employ binary coding schemes for NWs in which 0's correspond to FET-controllable regions and 1's to non-FET-controllable regions. There are many coding schemes that could be used. A natural coding scheme is the k-hot scheme in which

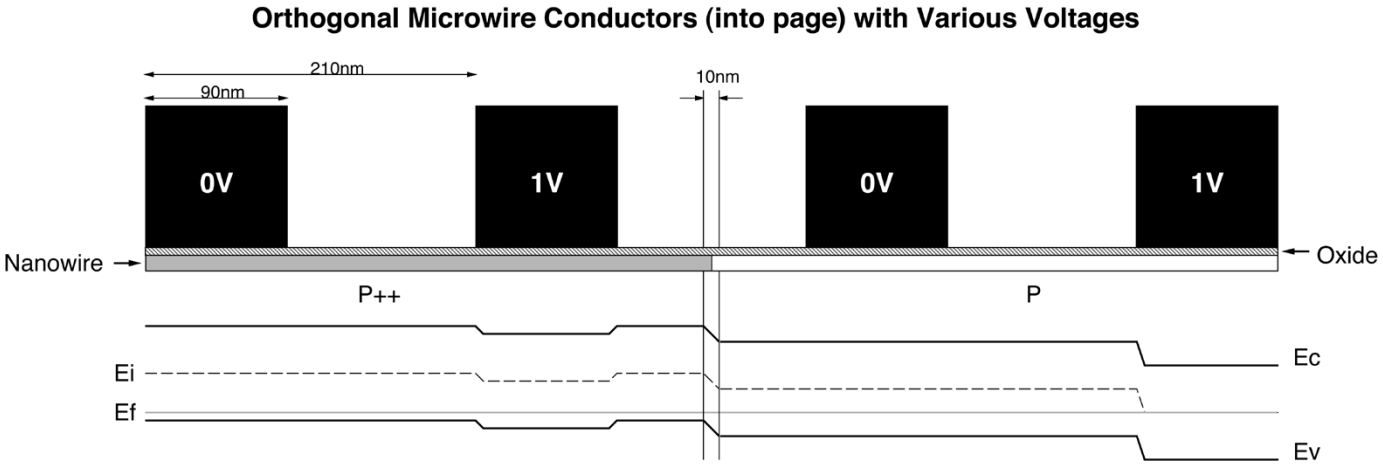

Fig. 4. Band bending diagram for modulation-doped NW controlled by microscale wires.

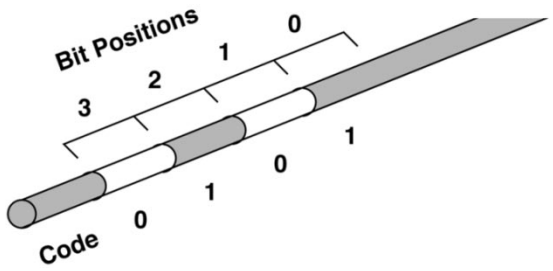

Fig. 5. NWs coded with address.

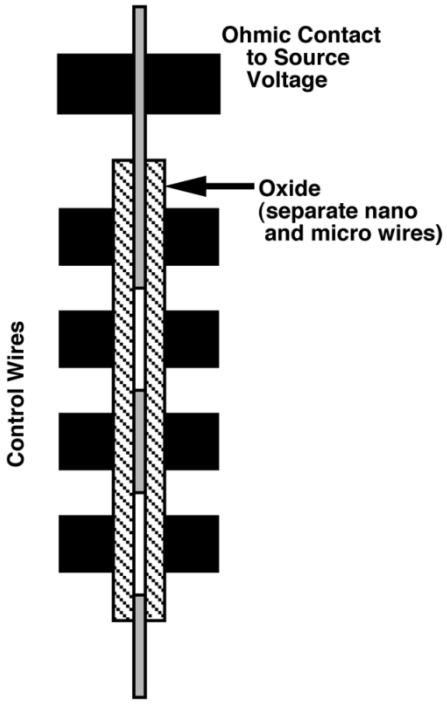

Fig. 6. Single coded NW and control wires.

each NW has n potentially controllable regions, exactly k of which are controllable (they must be "hot" to be controlled). This scheme allows for $\binom{n}{k}$ distinct codes.

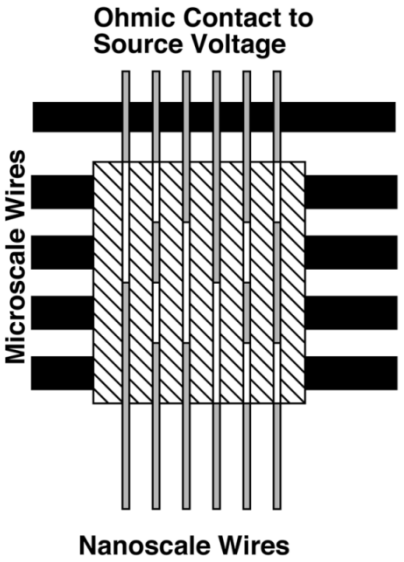

Fig. 7. Decoder constructed from addressable NWs.

We consider here the $(n/2)$-hot scheme. If we place low voltages on the $n/2$ control lines that correspond to the 0's in a code word for an NW and high voltages on the rest, then this NW is the only one that can conduct. All other NWs will have a FET-controlled region where the code word has a 1 and will be disabled as a result. If we could assemble exactly one of each type of coded NW into an array, we would have an address decoder (see Fig. 7) with $\mathrm{codes}(n) = \binom{n}{n/2}$ distinct codes for addressable NWs. Simple calculations show that the number of codes in the $n/2$-hot encoding scheme satisfies the following relationships:

$$\mathrm{codes}(n) = \binom{n}{\frac{n}{2}} = \frac{n!}{\left(\frac{n}{2}\right)!\left(\frac{n}{2}\right)!} \tag{1}$$

$$\frac{\mathrm{codes}(n)}{\mathrm{codes}(n-2)} = \frac{n\cdot(n-1)}{\left(\frac{n}{2}\right)\left(\frac{n}{2}\right)} = 4\left(\frac{n-1}{n}\right). \tag{2}$$

As the second calculation demonstrates, for large n, because $\mathrm{codes}(n)/\mathrm{codes}(n-2)$ approaches 4, the asymptotic growth

of codes(n) approaches 2^n. Inverting this, to uniquely address N wires, we need no more than $1.1\log_2(N)+3$ address bits. Consequently, for large enough arrays, the overhead associated with control lines, even if built out of microscale wires, becomes small compared to the size of the nanoscale logic or memory core which it addresses. The overhead remains modest even if k-hot addressing is used with k much smaller than $n/2$.

V. Stochastic Assembly

Since NWs are too small to be selected individually for inclusion into a nanoarray, some other method of selection is necessary. Techniques for assembling undifferentiated NWs into orthogonal sets of parallel wires have been demonstrated [5]. We show that stochastic selection of coded NWs from a sufficiently large ensemble of such NWs (the code space) ensures that all or almost all codes are unique. For the uses that we make of nanoarrays, it is not necessary that all codes be represented among the NWs.

For the sake of intuition, consider that we have a large code space (e.g., 10^6 codes) and a very large number of wires of each code type (e.g., 10^6 of each, or 10^{12} total wires), and our goal is to build a small array with ten wires in it. If we selected each wire randomly from the 10^{12} total wires, we have a very high likelihood that all ten wires are unique (in fact, over a 99.995% chance). There is an even higher likelihood that we get at least nine unique wires. From this example, it should be clear that we can randomly select the coded wires and obtain the independent nanoscale addressability that we desire.

It should be feasible to mix together a large number of NWs in solution in order to achieve random code mixing. Common techniques for aligning NWs are generally based on flow alignment in solution [4], [5], so an additional mixing step should be easy to accommodate.

Of course, we do not want to use a gratuitously large code space as this does cost us additional control wires. Consequently, the question becomes: How large does the size of the code space C need to be compared to the number N of NWs in an array in order to ensure that a large number of NWs have unique addresses?

We can obtain a lower bound on the probability $P(C,N,u)$ that we have u unique codes in an array of N wires randomly selected from a code space of size C by counting set sizes. We model the problem of code selection by assuming that each of the C codes appears equally frequently in the set of codes and that there are sufficiently many instances of each code that removing one does not change the probability $1/C$ of choosing a particular code.

Thus, there are C^N ways to select the N wires. One way to ensure that at least u NWs have unique addresses is to let the first u addresses be unique, which can be done in $C(C-1)(C-2)\ldots(C-u+1)=C!/(C-u)!$ ways, and select the remaining $N-u$ NWs from the set of remaining $C-u$ addresses in all possible ways, which can be done in $(C-u)^{(N-u)}$ ways. It follows that $P(C,N,u)$ satisfies the following inequality:

$$P(C,N,u) > \frac{\left(\frac{C!}{(C-u)!}\right)(C-u)^{(N-u)}}{C^N}. \tag{3}$$

TABLE I
Probability That All N Wires in a Set Are Unique When Selected From Codes of Size C

Array Size ($N=u$)	Code Size (C)	Probability Estimates (Eq. 4)	(Eq. 3)
10	100	0.35	0.62
10	1,000	0.90	0.96
10	10,000	0.990	0.996
100	10,000	0.37	0.61
100	100,000	0.90	0.95
100	1,000,000	0.990	0.995
1000	1,000,000	0.37	0.61
1000	10,000,000	0.90	0.95
1000	100,000,000	0.990	0.995

A weaker and simpler bound is

$$P(C,N,u) \geq \left(\frac{C-u}{C}\right)^N = (1-x)^N. \tag{4}$$

Here, $x=u/C$. It is straightforward to show that $(1-x)^N \geq (1-Nx)$ by induction where the base case is $N=2$. It follows that if $Nu/C \ll 1$, $P(C,N,u)$ is close to 1. In fact, if $Nu/C \leq 10^{-2}$, then $P(C,N,u) \geq 0.99$.

Table I shows sample calculated lower bound probabilities for achieving unique sets of coded wires for 10, 100, and 1000 NW arrays using various size code spaces. This data confirms that $C=100\times N^2$ is sufficient to yield almost all unique codes and $C=10\times N^2$ provides at most a 5% chance of not achieving unique codes.

For the even-weight codes described above, it would take a dense code with 14 bits to uniquely address over 1000 coded NWs. A code with 30 bits will support 155 117 520 unique codes, exceeding the $C=100\times N^2$ bound for $N=1000$. In other words, this scheme requires a little over twice the number of control lines we would need if we could perfectly select and place coded NWs. This is true asymptotically since $\log(100N^2)=\log(100)+2\log(N)$.

VI. Alignment

In practice, NWs will not be perfectly aligned to the control lines. We can divide any misalignment into: 1) misalignment by multiples of the width of control wires (the control bit pitch); and 2) misalignment by fractions of the bit pitch.

A. Multiples of Bit Pitch

We can tolerate misalignments by multiple bit pitches by repeating the code multiple times on the wire. For an $n/2$-hot code, if we concatenate multiple copies of the code [see Fig. 8(a)], any contiguous group of n coded bits will only be a rotation of the original code and, hence, will also be a valid code in this code space. Since we are selecting codes randomly, random misalignment does not change the random code selection. In some applications (e.g., memories when each nanowire layer does not see the field of the orthogonal nanowire layer), we can simply repeat the code along the entire

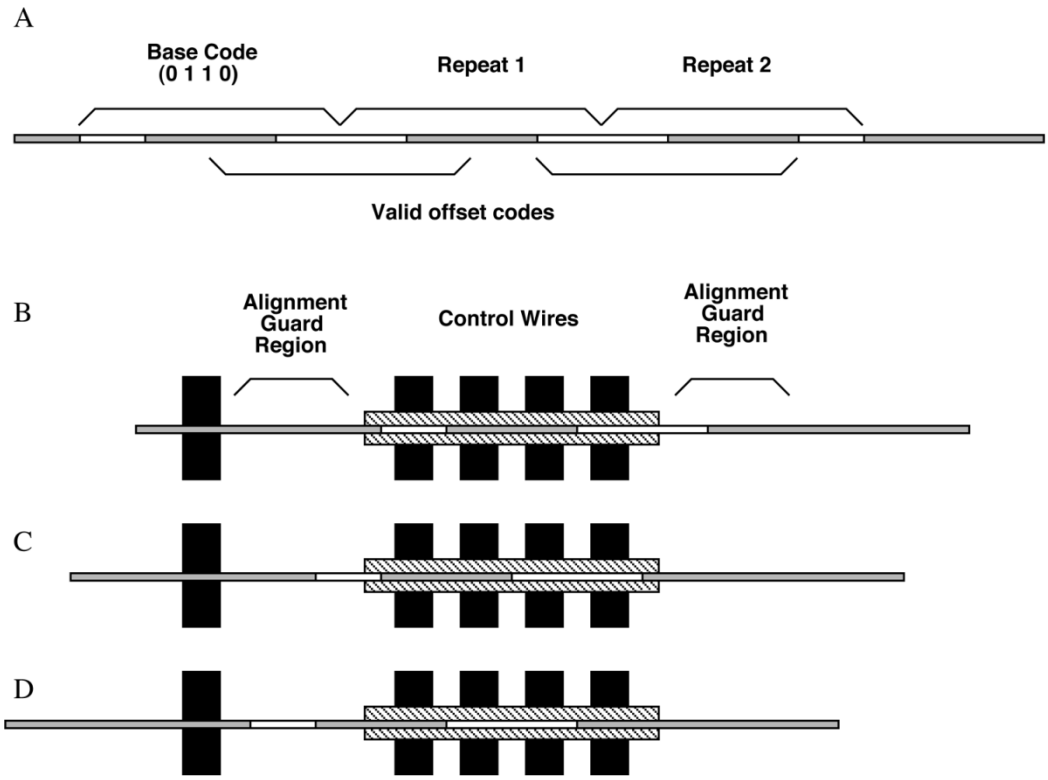

Fig. 8. Repeat code to tolerate misalignment by multiples of bit pitch. (a) NW with three copies of code to tolerate misalignment by ± 1 coding regions. (b)–(d) NW with a partial repeat of two bits to tolerate ± 1 bit position (shown at three different offsets).

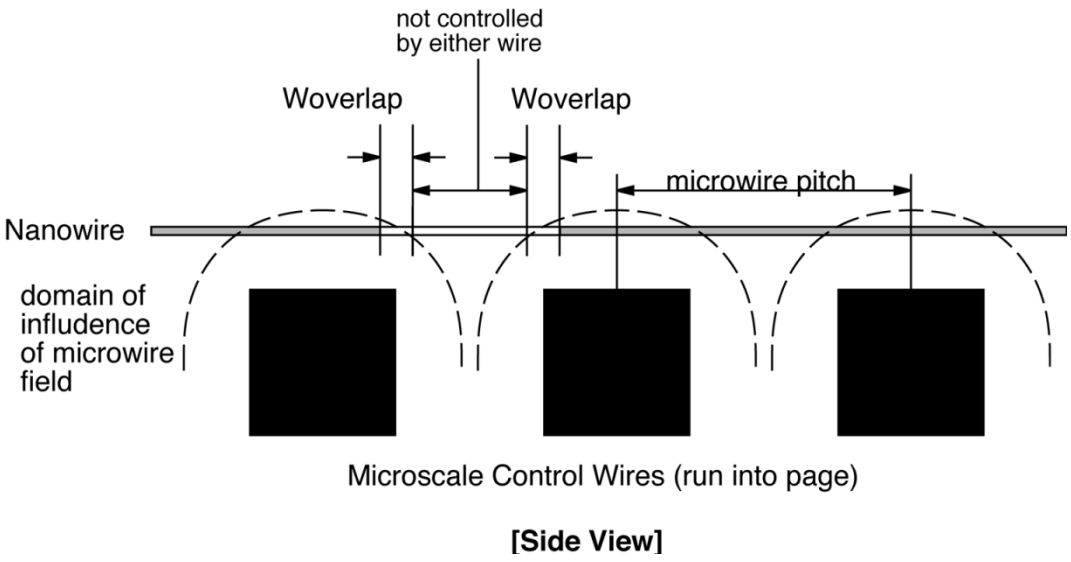

Fig. 9. Design of modulation-doped control region length.

length of the wire, and the fact that the wire is coded along its length will not interfere with operation. For other applications, it may be possible to mask off the address ends at a lithographic scale and bulk dope the nonaddress sections of the nanowires to extinguish the control regions outside of the addressing field. Alternately, if we can guarantee alignment within a few bit pitches, we repeat the code (or a fraction thereof) for a distance equal to the alignment tolerance that we would like to achieve (see Fig. 8). Here, we exploit the fact that the NW conducts across a coded region when there is no field applied; this way, the controllable bit code regions which end up on either side of the control wires will continue to allow signal conduction.

B. Fractions of Bit Pitch

In order to affect a controllable region, we need to have sufficient overlap between the field of one microwire and the doped controllable region (see Fig. 9). We only need to deplete carriers in a small region along the axis of the nanowire in order to stop conduction. Consequently, the necessary overlap W_{overlap} between the microwire field and the NW control region is likely

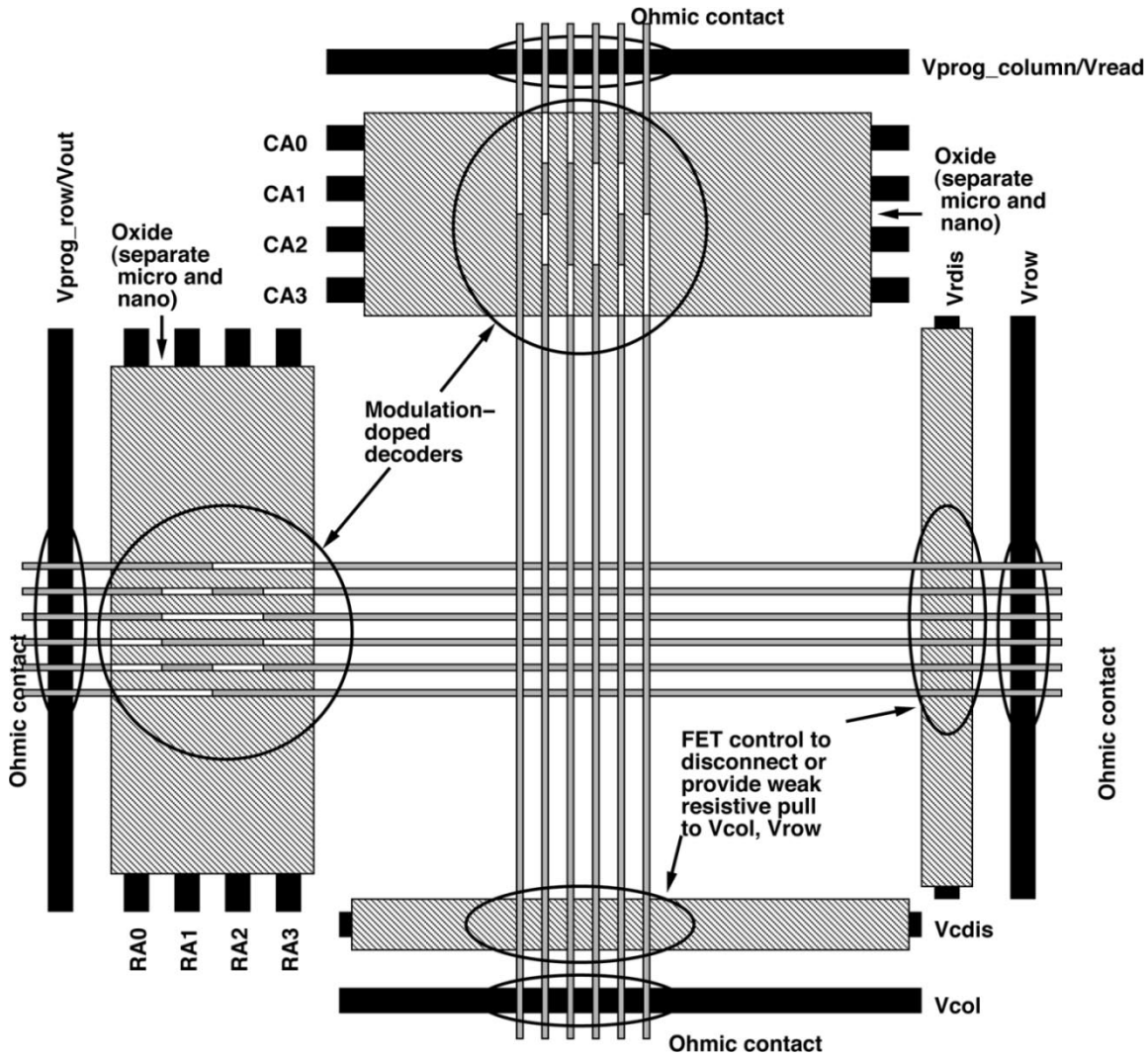

Fig. 10. Nanoscale memory array interfaced using modulation-doped address decoders (shown with only a few nanoscale wires for clarity). A typical array size would have 100–1000 nanoscale wires addressed by only 24–30 microscale wires.

to be small—on the order of a diameter or two of the NW (e.g., 5 nm). Overlaps between 0 and W_{overlap} may only partially turn off conductions, resulting in intermediate current flow levels.

As shown in Fig. 9, we have a noncontrollable region between the fields of adjacent microscale control wires. We consider making the doped controllable region equal to the length of the noncontrollable region of the nanowire plus $2 \times W_{\text{overlap}}$. First, this guarantees there is always at least W_{overlap} under one of the adjacent control fields, making every wire controllable. Second, this means there is a window of alignments of length $2 \times W_{\text{overlap}}$ where the controllable region may be affected by the fields of two microwires. For all other alignments, the controllable region is under only one of the fields. Assuming all sub-bit-pitch misalignments are equally likely, the probability that a region is controlled by only one of the adjacent microscale wire is

$$P_{\text{control}} = \frac{W_{\text{bitpitch}} - 2 \times W_{\text{overlap}}}{W_{\text{bitpitch}}}. \tag{5}$$

W_{bitpitch} is the bit pitch for the microscale wires; e.g., if the bit pitch is 210 nm and we conservatively assume a necessary overlap of 10 nm, then $P_{\text{control}} = 190/210 \geq 0.90$. When the controllable region does overlap multiple fields, it may require both fields to be zero to allow conduction. Since every $n/2$-hot code contains at least one transition between zeros and ones, the overlap case can end up with at least one more control region than any valid code. Consequently, no code in the standard $n/2$-hot code space will enable this small fraction of misaligned wires. Using codes outside of the code space, it may be possible to still address some of these wires.

It is not clear at this point how far the microwire control field will extend beyond the width of each control microwire (Fig. 9). It is possible that the microwires may need to be spaced wider than the minimum microwire pitch to prevent adjacent wire control fields from overlapping. This is one of many areas where further experimental and theoretical work will be necessary to tune designs.

VII. Memory Application

A. Simple Memory Design

A programmable memory can be constructed by placing a decoder of the form described above on both sides of an NW array (see Fig. 10). There are multiple molecular-scale technologies under consideration for placing nonvolatile memory bits at

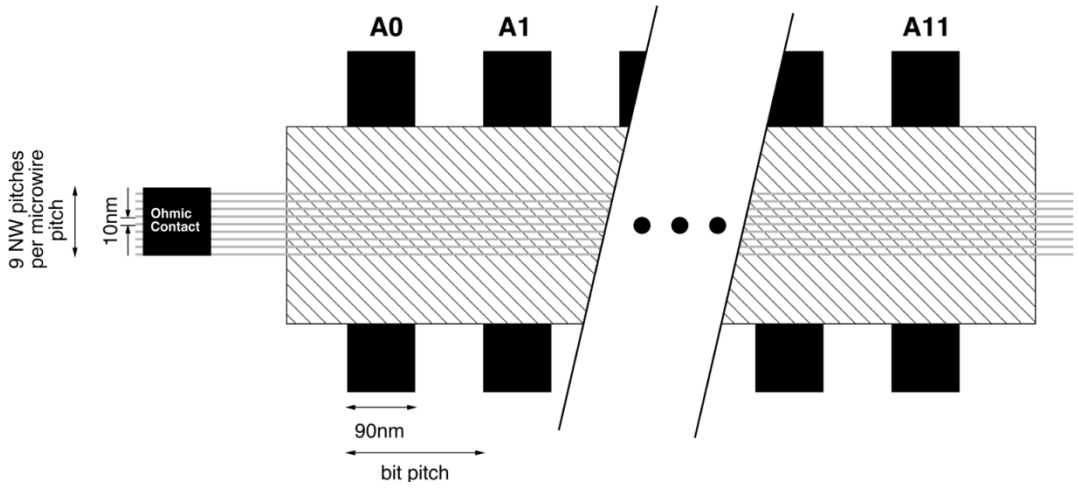

Fig. 11. Small address space decoder.

the crosspoints [6], [7]. These technologies are programmed by placing a large voltage across individual crosspoint junctions and are read by observing the current flowing through a junction, with programmed ON junctions acting as low-resistance paths, while programmed OFF junctions act as high-resistance paths.

Using these addressable NWs, exactly one row and one column wire can be enabled so that we can apply a programming voltage across a single crosspoint. This will require care in the selection of voltage levels such that the crosspoints that are in the same row and column as the intended crosspoint are not also affected. These row and column neighbors will have one side pulled to the programming voltage, while the other is pulled to a nominal voltage, whereas the intended crosspoint is pulled to the programming voltage by both the row and column decoders and, hence, will see a greater voltage differential. We also can generally arrange for the crosspoints to act as diodes to avoid parasitic paths in a partially programmed array.

Data bits are read from the array by again placing the appropriate control bits to enable only a single row and column. A high voltage is placed on the common column line, and the voltage on the common row line is observed. In this manner, only the intended crosspoint sees both a high input on its column line and a low-resistance path to the common row line. If the crosspoint is programmed ON, it will be possible to observe the current flowing out of the selected row line, perhaps raising the row line voltage. If the crosspoint is programmed OFF, there will be less current flow.

B. Hybrid Control Memory

The simple memory described above is easy to understand, but has the drawback that it requires a very large address space and, hence, requires that we first construct a very large collection of differently coded NWs (e.g., 25 million for a 500×500 array). We can use a more modest number of NWs if we observe that we really only need the stochastic addressing to distinguish among the number of wires that we can fit into one microscale wire width. As shown in Fig. 11, we can selectively energize the endpoints of a collection of NWs at the lithographic scale. So, for example, if we have 10-nm pitch nanowires and a 90-nm-wide microwire, we only need to be able to uniquely address nine nanowires at a time. A 6-hot 12-bit code has 942 code words. With 942 code words, we have over a 96% probability that all nine wires in a bundle will have unique codes.

By staggering adjacent microwire contacts, we can maintain the tight NW pitch (see Fig. 12), perhaps losing one wire at the edge of each microwire group. With a contact group length $W_{\text{contact\_group\_len}}$, we need to uniquely address a wire group with N_g wires, as follows:

$$N_g = \frac{W_{\text{contact\_group\_len}}}{W_{\text{nano\_pitch}}}. \tag{6}$$

As shown:

$$W_{\text{contact\_group\_len}} \geq 2 \times W_{\text{litho\_pitch}} - W_{\text{metal\_width}}. \tag{7}$$

For a 90-nm process with $W_{\text{litho\_pitch}} = 210$ nm, we have $W_{\text{contact\_group\_len}} = 330$ nm. With $W_{\text{nano\_pitch}} = 10$ nm, we have $N_g = 33$. A 7-hot 14-bit code has 3432 code words, giving us over an 85% chance of assembling a completely unique set of 33 coded NWs.

It is worthwhile to note that the microscale-to-nanoscale address area does *not* scale up with array size in this hybrid scheme. For each additional group of core wires we add (e.g., $N_g = 33$ nanowires), we will need an additional microscale wire for the contact, but the nanoscale addressing remains constant. To the 14 address bit pitches we need to address these 33 wires, we add two bit pitches for the address contacts and two bit pitches for the load contact on the opposite side of the array. This allows us to calculate the side length of the memory array including the microscale-to-nanoscale address translation $S_{\text{mem\_array}}$:

$$S_{\text{mem\_array}} = 18 \times W_{\text{litho\_pitch}} + N \times W_{\text{nano\_pitch}}. \tag{8}$$

For $W_{\text{nano\_pitch}} = 10$ nm, $N = 500$, and $W_{\text{litho\_pitch}} = 210$ nm (90-nm process), $S_{\text{mem\_array}} = 8780$ nm. Including the decoder support, the memory area for the array is

$$A_{\text{mem\_array}} = S^2_{\text{mem\_array}}. \tag{9}$$

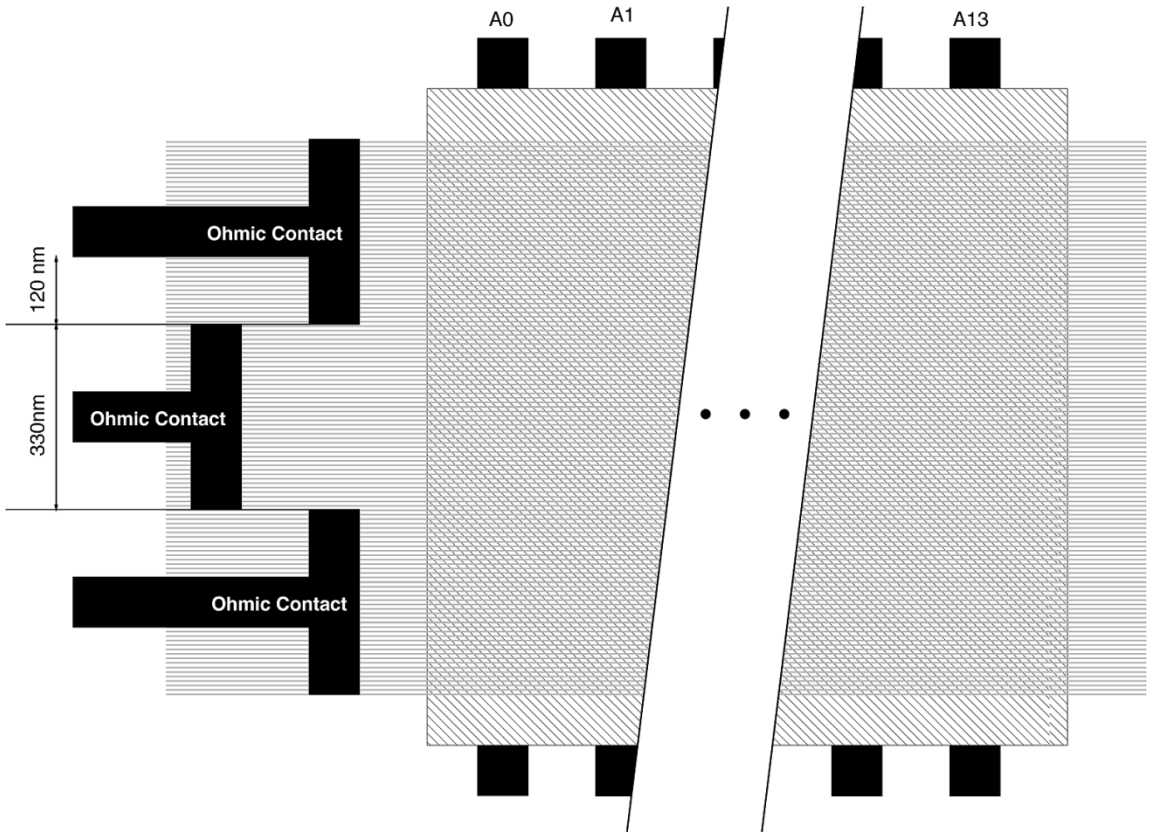

Fig. 12. Small address space decoder with staggered microscale contacts.

TABLE II
MEMORY YIELD FACTORS

Param	Sample Value	Description
P_{unique}	0.85	Probability group of N_g wires contains unique members
		Sample assumes $N_g = 33$, 7-hot, 14-bit code.
		(conservative since we yield a large fraction which are unique)
$P_{control}$	0.90	Probability aligned without overlap (Section VI-B)
$P_{wireend}$	0.97	Assume lose 1 of each group of $N_g = 33$ NWs
$P_{contact}$	0.95	Probability end makes adequate electrical contact [11]
$P_{nobreak}(9\mu\text{m})$	0.91	(assume $P_{break}/W_{nano\_pitch} = 0.0001$)
		N.b. reliable growth of this length in [9]

Consequently, this gives us a raw memory bit area of $A_{\text{mem\_array}}/N^2 = (8780\,\text{nm})^2/500^2 < 310\ \text{nm}^2$. Compared to the 100-nm^2 memory bit area in the NW core, the array with microscale-to-nanoscale decoder is a factor of 3.1 times larger.

We will yield less than N^2 net bits due to a number of factors as summarized in Table II. A wire is good only if it makes contacts at its ends where it connects to microwires and there are no breaks or shorts along its length:

$$P_{\text{goodwire}} = P_{\text{contact}}^2 \times P_{\text{nobreak}}\left(L_{\text{nanowire}}\right). \quad (10)$$

The wire is addressable only if the address group is unique, the wire is properly controllable, and the wire end connects to only one microwire group:

$$P_{\text{addressable}} = P_{\text{unique}} \times P_{\text{control}} \times P_{\text{wireend}}. \quad (11)$$

For a wire to yield, it must both be good and addressable:

$$Y_{\text{wire}} = P_{\text{goodwire}} \times P_{\text{addressable}}. \quad (12)$$

Further, for a bit to yield, both the row and column wire intersecting it must yield:

$$Y_{\text{bit}} = Y_{\text{wire}}^2. \quad (13)$$

For the values in Table II, $Y_{\text{bit}} > 0.38$. Combining with a raw area of 310 nm^2, this gives a yielded bit area of 800 nm^2.

For comparison, note that the DRAM core memory cell area is 49 000 nm^2 in a 90-nm process [15], suggesting that the factor of 8 overhead for the addressing and yield does not negate the NW density benefits and there is additional headroom for lower yield, larger address overlap, and address remapping as necessary. At 45 nm, the DRAM core cell area is 12 000 nm^2; with $W_{\text{litho\_pitch}} = 105$ nm, the raw nanomemory bit area with decode support is < 200 nm^2, with yielded bit area around 500 nm^2.

C. Variations

The basic memory construct which this enables can easily be embellished in a number of ways. We can split the row read line to read or program multiple bits in parallel. We can array many such blocks to build a large defect-tolerant memory. This basic array programming construct also allows us to program the programmable logic subarrays in nanoscale logic devices [8].

VIII. Discovering Code Words

By design, the number of code words is large compared to the number of wires in any row, column, or microscale contact group ($C \gg N$). Consequently, after fabrication we will have to discover which code words actually make up the set of live addresses.

A. Resettable Memory Technology

For the memory arrays based on resettable crosspoints (e.g., [6]), this can be done moderately simply in a manner similar to memory testing. Initially, we can activate the row addresses with all zeros—this will enable all of the row lines. In this manner, we treat each column as a single bit rather than a collection of individual bits. This allows us to attempt to program and read each possible column address. If the column address is programmable, we know that address is present. If it is not programmable, we know it is not present. Once we know which column addresses are present, we can then test each row address using the known column addresses. This process will find both the column and row addresses included and operational in the array and any crosspoint junction faults. For an $N \times N$ array, this will certainly take longer to test than an array with perfect, dense, predictable codes. However, as long as $C = O(N^2)$, as derived above, it will still take only $O(N^2)$ total time.

With the hybrid address scheme, the testing overhead is only $O(N)$. For the 500×500 nanowire array above, we will need to test $(500/33) \times 3432 = 52\,000$ row addresses and an equal number of column addresses to find which addresses are present. Note that there are 250 000 raw bits in the array, so the additional 104 000 tests to find valid addresses will not even double the number of test operations required. Compared to the $250\,000 \times 0.38 > 95\,000$ memory bits we expect to yield from this array, the 354 000 tests is less than four times the number of final yielded bits.

B. One-Time Programmable Crosspoints

Some crosspoints technologies set the junctions permanently during programming (e.g., [7]). For these technologies, we can test for address presence without programming the crosspoints. As shown in Fig. 10, the row (and column) lines are connected to a common line (V_{row}, V_{col}). To test for presence of a row (column) address, we weakly pull down V_{row} (V_{col}) and drive the row (column) address in question. If the address is present, it will be able to pull up V_{row} (V_{col}); if the address is not present, V_{row} (V_{col}) will be pulled down to a low voltage. By observing the voltage on V_{row} (V_{col}), we can detect the presence or absence of the address. This can easily be done at voltage levels below the programming voltages so that no crosspoints are inadvertently programmed during address discovery. Again, as long as $C = O(N^2)$, we only have $O(N^2)$ total row and column addresses to test to establish the set of row and column addresses present. This is reduced to $O(N)$ for the hybrid addressing scheme.

C. Mapping Present Addresses

Since the codes are sparse, it will also be necessary to keep track of the live row and column addresses. There are N live row and column addresses, each of which is $O(\log(N))$ bits long; consequently, we will need $O(N \log(N))$ bits of storage to hold this address translation if we build a monolithically addressed memory or about $14N$ for the hybrid addressing scheme. Since this is asymptotically smaller than the $O(N^2)$ bits in the memory, the memory to hold this translation is smaller than the memory that we are addressing. At the cost of multiple nanoscale reads to resolve an address, we can apply this reduction trick repeatedly to reduce the number of bits needed to an arbitrarily small number which we can then store in a microscale memory.

Using the hybrid addressing scheme, we need 14×500 bits to describe the nanoscale portion of the 500 row wires along with an equal number of bits for the column wires. Thus, we will ultimately need to know 14 000 bits of data in order to retrieve the 95 000+ bits in the memory. The roughly 7 : 1 reduction here is not adequate to reduce the information needed to address this memory to a sufficiently compact amount that it can be efficiently stored in a lithographic-scale memory; consequently, multiple stages of mapping will be necessary. An important area of future work will be the development of multistage nanoscale address mapping architectures where a nanoscale mapper can be programmed, using this same technique, to perform the address translation so as to provide a deterministic external set of memory addresses. Even if we needed two programmed address mappers that were as large as the memory they addressed, we would still see benefits from this scheme.

Note that deterministic externally visible addresses are not necessary for the case of programmable logic array (PLA) programming or for programming address mapper stages. In these cases, we simply (re)discover the codes as part of the programming phase. During operation of the PLA or address mapper, the programming addresses are irrelevant.

IX. Conclusion

The stochastically assembled address decoder allows us to address individual nanoscale features without requiring any lithographic processing at nanoscale dimensions. As an example, this allows us to construct a complete nanoscale memory and obtain independent access to each nanoscale bit without requiring any lithographic processing to achieve the nanoscale features. Nanoscale wire features are controlled by catalysts and reaction time. They are decorated with molecules and assembled using self-assembly techniques. The code properties that we have described allow the assembled devices to tolerate gross misalignment with the microscale wires to which they are interfaced in several scenarios and remain independently addressable. As a result, this provides an important technique for bridging the gap between microscale and nanoscale features and for bootstrapping the programming and customization of nanoscale systems.

Acknowledgment

The authors would like to thank C. Lieber, D. Wang, and Z. Zhong for their support in this work.

REFERENCES

[1] C. Dekker, "Carbon nanotubes as molecular quantum wires," *Phys. Today*, pp. 22–28, May 1999.

[2] Y. Cui, L. J. Lauhon, M. S. Gudiksen, J. Wang, and C. M. Lieber, "Diameter-controlled synthesis of single crystal silicon nanowires," *Appl. Phys. Lett.*, vol. 78, no. 15, pp. 2214–2216, 2001.

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

[4] Y. Huang, X. Duan, Q. Wei, and C. M. Lieber, "Directed assembley of one-dimensional nanostructures into functional networks," *Science*, vol. 291, pp. 630–633, Jan. 2001.

[5] F. Kim, S. Kwan, J. Akana, and P. Yang, "Langmuir–Blodgett nanorod assembly," *J. Amer. Chem. Soc.*, vol. 123, no. 18, pp. 4360–4361, May 2001.

[6] T. Rueckes, K. Kim, E. Joselevich, G. Y. Tseng, C.-L. Cheung, and C. M. Lieber, "Carbon nanotube based nonvolatile random access memory for molecular computing," *Science*, vol. 289, pp. 94–97, 2000.

[7] C. P. Collier, E. W. Wong, M. Belohradsky, F. M. Raymo, J. F. Stoddard, P. J. Kuekes, R. S. Williams, and J. R. Heath, "Electronically configurable molecular-based logic gates," *Science*, vol. 285, pp. 391–394, 1999.

[8] A. DeHon, "Array-based architecture for FET-based nanoscale electronics," *IEEE Trans. Nanotechnol.*, vol. 2, pp. 23–32, Mar. 2003.

[9] M. S. Gudiksen, J. Wang, and C. M. Lieber, "Synthetic control of the diameter and length of semiconductor nanowires," *J. Phys. Chem. B*, vol. 105, pp. 4062–4064, 2001.

[10] S. Williams and P. Kuekes, "Demultiplexer for a molecular wire crossbar network," U.S. Patent 6 256 767, July 3, 2001.

[11] Y. Huang, X. Duan, Y. Cui, L. Lauhon, K. Kim, and C. M. Lieber, "Logic gates and computation from assembled nanowire building blocks," *Science*, vol. 294, pp. 1313–1317, 2001.

[12] M. S. Gudiksen, L. J. Lauhon, J. Wang, D. C. Smith, and C. M. Lieber, "Growth of nanowire superlattice structures for nanoscale photonics and electronics," *Nature*, vol. 415, pp. 617–620, Feb. 2002.

[13] Y. Wu, R. Fan, and P. Yang, "Block-by-block growth of single-crystalline Si/SiGe upperlattice nanowires," *Nano Lett.*, vol. 2, no. 2, pp. 83–86, Feb. 2002.

[14] M. T. Björk, 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 Lett.*, vol. 2, no. 2, pp. 87–89, Feb. 2002.

[15] (2001) International Technology Roadmap for Semiconductors. [Online]. Available: http://public.itrs.net/Files/2001ITRS/

André DeHon (S'92–M'96) received the B.S., M.S., and Ph.D. degrees in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, in 1990, 1993, and 1996, respectively.

From 1996 to 1999, he co-ran the BRASS group in the Department of Computer Science, University of California at Berkeley. Since 1999, he has been an Assistant Professor of computer science with the California Institute of Technology, Pasadena. His research interests focus on spatial programmable architectures and interconnect design and optimization.

Patrick Lincoln received the B.S. degree in computer science from the Massachusetts Institute of Technology, Cambridge, in 1986 and the Ph.D. degree in computer science from Stanford University, Stanford, CA, in 1992.

He joined SRI International, Menlo Park, CA, in 1989, where he is currently the Director of the Computer Science Laboratory. He has previously held positions at MCC and Los Alamos National Laboratory. He has co-chaired DARPA-sponsored Information Science and Technology (ISAT) studies, and serves on the Scientific Advisory Boards of private and public companies including Cable & Wireless. His research focuses on the fields of molecular electronics, formal methods, computer security and privacy, bioinformatics, scalable distributed systems, and programming language design and implementation.

John E. Savage (S'59–M'66–SM'91–F'92–LF'96) received the B.S., M.S., and Ph.D. degrees from the Massachusetts Institute of Technology, Cambridge, in 1961, 1962 and 1965, respectively.

He joined Bell Laboratories, Holmdel, NJ, in January 1965, leaving to join the faculty of Brown University, Providence, RI, in 1967. He is a Founder of Brown's Department of Computer Science and was its second Chair. He has done research on information theory, coding, communication theory, circuit complexity, space-time tradeoffs, I/O complexity, VLSI algorithms and analysis, silicon compilation, parallel algorithms, scientific computing, nanotechnology, and human cognition. He is the author of *The Complexity of Computing* (New York: Wiley,1976) and *Models of Computation* (Reading, MA: Addison-Wesley, 1998) and a coauthor of *The Mystical Machine* (Reading, MA: Addison-Wesley, 1986). He is a co-editor of *Advanced Research in VLSI and Parallel Systems* (Cambridge, MA: MIT Press, 1992). He is a member of the editorial board of the *Journal of Computer and Systems Sciences*.

Dr. Savage is a Guggenheim Fellow and a Fellow of the Association for Computing Machinery and the American Association for the Advancement of Science.