Crossbar Addressing Using Core-Shell Nanowires*
Extended Abstract

Eric Rachlin†, John E. Savage
Department of Computer Science
Brown University

Abstract

The nanowire crossbar is a promising nanotechnology for assembling memories and circuits. In both, a small number of lithographically produced mesoscale wires (MWs) must control a large number of nanoscale wires (NWs). Previous strategies for achieving this have been vunerable to misalignment. In this paper, we introduce core-shell NWs, which eliminate misalignment errors. We also give a two-step assembly process that reduces the amount of crossbar control circuitry.

1 Introduction

Nanoscale wires and single-molecule devices offer the tantalizing prospect of shrinking lithographically produced architectures to the nanoscale. Multiple techniques for growing NWs have been discovered [6, 7, 14, 10] as have "programmable molecules" (PMs) exhibiting rectification and hysteresis [1]. PMs store state, changing conductance when exposed to a sufficiently strong electric field.

At present, only highly regular configurations of NWs and PMs can be assembled. A NW crossbar (see Fig. 1), or **nanoarray**, is a grid formed from two sets of parallel NWs separated by a layer of PMs. Pairs of orthogonal NWs can control the molecular devices at their crosspoints [1].

In section 2 we show how nanoarrays can be used as memories. They can also function as general circuits [2]. Either setup requires that NWs be controlled with lithographically-produced technology. In section 3 we introduce **core-shell NWs** [11], which allow for this type of control. Unlike previous approaches, they avoid errors caused by misalignment. They also permit a two-step assembly process, given in Section 4, that reduces the size of support circuitry required by the nanoarray.

*This research was funded in part by NSF Grant CCF-0403674.
†Presenter

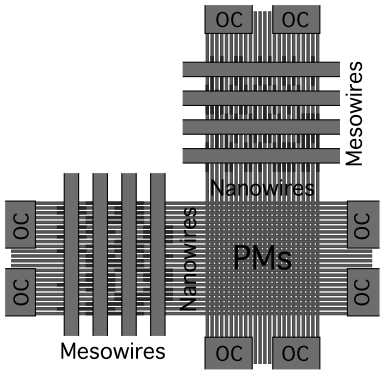

Figure 1: A stochastically assembled nanoarray with lithographically produced control circuitry at the periphery.

2 Nanoarrays and Nanowire Codewords

A stochastically assembled nanoarray (see Fig. 1) is controlled by lithographically produced MWs and ohmic contacts (OCs). The OCs place voltages across groups of consecutive NWs. Activated MWs turn off subsets of the NWs. When a MW carries a voltage, NWs with exposed lightly doped regions underneath the MW become nonconducting.

Two NWs, n_a and n_b, **function independently** if they touch different OCs, or if they can be turned off independently. When a NW functions independently of all other NWs, it can be individually connected to an external source. The current flowing through it can also be measured.

A sufficiently large voltage placed across two orthogonal NWs sets the conductivity of the PMs at their crosspoint. A smaller voltage across only one of the NWs allows the other NW to be used to measure the conductivity of the crosspoint. These two actions act as write and read operations, respectively. Notice that if two NWs do not function independently, they must be used simultaneously, reducing the number of addresses in the array.

MWs and OCs only provide good control of the nanoarray if a large fraction of NWs function independently. NWs

at different OCs always function independently, but OCs are manufactured lithographically, and must touch some minimal number of NWs (about 10). NWs at the same OC only function independently when their exposed lightly doped region permit. Unfortunately the placement of NWs and their regions is the result of a stochastic assembly process.

The set of MWs which, when activated, turns off a NW, is called its **codeword**. One method for assigning codewords to NWs involves random particle deposition [13]. Another uses lithographically defined high-k dielectric regions [5, 4]. A third uses **modulation-doped NWs** grown with patterns of lightly and heavily doped regions along their axis [3]. A stochastic fluidic method can form nanoarrays of these NWs. It also applies to core-shell NWs.

Modulation-doped NWs are grown off-chip using chemical vapor deposition. Many NWs are grown at once and, as they grow, given a precise pattern of lightly and heavily doped regions along their length. NWs grown simultaneously have the same pattern, but many sets of NWs with different patterns can be grown separately and collected in a fluid. Using a fluidic assembly process, known as Langmuir-Blodgett assembly [12], a random subset of NWs is selected, aligned in parallel, and deposited onto a chip. Two layers of parallel NWs, deposited at right angles, form a nanoarray. MWs and OCs are then placed along the peripherally of the array (see Figure 1).

A NW's codeword is determined by its doping pattern and its axial displacement in the array. NW placement is random, so each codeword is equally likely to appear on each NW. When the number of possible codewords is sufficiently large, many different codewords will appear at each OC with high probability. If this condition is met, the doping patterns on the NWs form a stochastically-assembled MW decoder. A detailed exploration of the number of codewords and doping patterns required is given in [4].

Fluidic assembly suffers from two significant problems. First, although it aligns NWs in parallel, they may not be aligned end-to-end. Small axial displacements can prevent doping patterns from aligning with the MWs, and badly aligned NW will not function properly. Second, codeword selection is stochastic, and the codewords present at each OC will vary. Since codewords determine which MWs carry a voltage during a read/write operation, they must be discovered and recorded. This requires programmable address translation circuitry that maps binary memory addresses to pairs of orthogonal NWs. Core-shell NWs can eliminate both difficulties.

3 Core-Shell Nanowires

Core-shell NWs consist of a lightly doped silicon core surrounded by thin layers of insulating material, called **shells**. As with modulation-doped NWs, core-shell NWs are grown off-chip using chemical vapor deposition and then deposited using Langmuir-Blodgett assembly. When manufactured, the core is grown first, then shells are grown radially, one after the other. The ability to grow a single shell on a lightly doped core has been demonstrated [9].

The materials used to form shells, or **shell types**, must be chosen so as to be independently etchable. If consecutive NW shells are never of the same type, they can be etched away one at a time. Given a shell type, x, we use $E(x)$ to denote the etching process that removes only shells of type x. two NWs, n_a and n_b, have materials x and y in their outer shells respectively, $E(x)$ removes the outer shell of NW n_a, but leaves n_b unaffected.

By producing core-shell NWs with many different sequences of shell types, and applying a different sequence of etching processes to the region under each MW, multiple types of decoding circuitry can be produced.

3.1 The Linear Decoder

Assume there are n shell types used to produce core-shell NWs with k shells. Since each shell type must be different from the one in the preceding shell, $n(n-1)^{k-1}$ different shell type sequences are possible. Suppose core-shell NWs are produced with all possible sequences and used to form a nanoarray. The resultant nanoarray can be controlled by a **linear decoder** as follows:

1. Associate a different shell sequence with each MW along each dimension of the nanoarray. Let $s_1^i, s_2^i, ..., s_k^i$ be the sequence associated with MW M_i where s_1^i denotes the material of the inner most shell type, and s_k^i denotes the material of the outermost one.

2. Before producing M_i, expose the region under M_i to the etching process $E(s_k^i), E(s_{k-1}^i), ..., E(s_1^i)$. Here $E(s_k^i), E(s_{k-1}^i), ..., E(s_1^i)$ denotes the application of each of individual etching processes, one after the other. This k-step etching process will expose only the cores of NWs with shell type sequences $s_1^i, s_2^i, ..., s_k^i$.

3. By associating a MW with each shell type sequence, each NW is given a codeword. Each of the $n(n-1)^{k-1}$ types of core-shell NW is turned off by exactly one MW. If NW n_a has a different shell type sequence then all other NWs touching a particular OC, these NWs can be turned off while n_a remains conducting.

The linear decoder we have described requires a different MW for each type of NW. Each NW type results in a different codeword. If the total number of desired codewords is not large, linear decoding is reasonable.

In the immediate future we expect the number of shell types available to be small. We also expect the number of

shells to be small, since adding shells necessarily increases NW pitch. If k is 2 and n is 5, 20 codewords can be produced. If each OC controls 10 NWs, 20 codewords is sufficient to build a nanoarray based memory [4] in which at least half of the NWs are addressable.

3.2 The Linear-Logarithmic Decoder

A standard demultiplexer uses $\log_2 N$ inputs to select one of N outputs. The linear decoder above uses N MWs to select one of N NW types. In fact $2\log_2 N$ MWs suffice. We describe how to efficiently decode with respect to k; we deal with the number of shell types, n, separately. Then we explain how the two approaches can be combined.

Our logarithmic decoder requires that only $n/2$ shell types be used per shell. Previously we allowed $n(n-1)^{k-1}$ NW types and required only that consecutive shells be of different types. Now we divide the n shell types into two equal sized sets (assume n is even) and use the first $m = n/2$ shell types in odd numbered shells, and the remainder in even number shells. This allows for more powerful etching operations, but limits the total number of NW types to $(n/2)^k$. Under these conditions, a nanoarray can be controlled by a **linear-logarithmic decoder**, as shown.

1. Let $s_1^i, s_2^i, ..., s_m^i$ denote the m shell types permitted to appear in a NWs i^{th} shell Let E_i denote the etching process $E(s_1^i), E(s_2^i), ..., E(s_m^i)$ that removes all material in the ith shell (the order is unimportant).

2. Consider each etching processes of the form $E_k, ..., E(s_j^i), ..., E_1$. It removes the $k-i$ outermost shells of every NW, a shell from every NW with shell type s_j^i, and the remaining $i-1$ shells of those NWs that have type s_j^i in the i^{th} shell. Only these NWs have have their cores exposed.

3. Associate each etching process of the form $E_k, ..., E(s_j^i), ..., E_1$ with a different MW along each dimension of the nanoarray. Apply the etching processes to the regions under their associated MWs. Let MW M_i^j be associated with $E_k, ..., E(s_j^i), ..., E_1$. When activated, M_i^j turns off only the the NWs with shell type s_j in their i^{th} shell.

4. Each NW is given a codeword based on each of its shells. To turn off all NWs that do not possess a given codeword, simply activate all MWs that do not affect NWs with that codeword.

The linear-logarithmic decoder controls $N = (n/2)^k$ NW types with $k(n/2)$ MWs. As stated earlier, a large k necessarily increases the pitch of core-shell NWs. The decoder becomes far more practical when n is handled efficiently as well.

3.3 The Fully logarithmic Decoder

Consider the case $k = 1$ in which there are m shell types. A nanoarray constructed from these types is controlled efficiently as follows:

1. Assign each shell type a unique L bit binary number. L need be no larger than $\lceil \log_2 m \rceil$.

2. Let S_b^j be the set of all shell types with b as their j^{th} bit. Let $E(S_d^j)$ be an etching process that removes each shell type in S_d^j, one after the other, in some arbitrary order.

3. Associate each $E(S_d^j)$ with a different MW along each dimension of the nanoarray and apply it to the region under that MW. If MW M_d^i is associated with $E(S_d^j)$ it will turn off only NWs with shell types in S_d^i.

4. A given NW's codeword is the L bit number assigned to its shell type. To turn off all NWs with other codewords, activate all MWs that do not affect the NW.

This decoder controls m shell types with $\lceil \log_2 m \rceil$ MW. It is readily incorporated into the linear-logarithmic decoder, creating a **fully logarithmic decoder**. To do this, the etching processes of the form $E_k, ..., E(s_j^i), ..., E_1$ are replaced with those of the form $E_k, ..., E(S_d^j), ..., E_1$. Here S_d^j is also dependent on the parity of i. The resultant decoder controls $N = (n/2)^k$ NW types with $2\log_2 N = 2k\log_2 n/2$ MWs. Each NWs codeword is the concatenation of binary numbers assigned to its shell types. A detailed description of all three decoders is found in [11].

3.4 Error Correction

The etching processes we have described may behave imperfectly. Shells which should remain may be removed, and shells that should be removed may remain. Either error can alter a NW's codeword.

Consider two NWs, n_a and n_b, with distinct, error-free codewords, and a third NW, n_c, with an erroneous codeword that is a subset of the other two. n_c is conducting when either n_a or n_b is conducting. If all three NWs are attached to the same OC, n_a and n_b cannot be used separately. When errors cause codewords to become subsets of other codeword, the number of useable NWs is reduced.

When NW codewords are represented as binary numbers, errors caused by etching correspond to bit flips. As observed in [8], NW codewords with a hamming distance of $2d+1$ can tolerate up to d errors. Using coding theory, the L bit binary numbers assigned to each shell type can be chosen to have an arbitrarily large hamming distance, h. Using these values to generate a fully logarithmic decoder results in NW codewords with hamming distance kh. Fully logarithmic decoding allows for efficient fault-tolerance.

3.5 Note on Etching

Associated with each MW is an etching process of the form $E_k, ..., E(S_d^j), ..., E_1$. Each etching process involves at most km basic etching operations. For each E_i and $E(S_d^j)$, the order in which etching operations occur is unimportant. This allows all MWs to be etched concurrently using only km operations. To accomplish this, cycle $k/2$ times through all $n = 2m$ basic etching operations, and mask a MW region whenever an etch should not be applied. This **fast etch algorithm** is fully defined in [11].

4 Two-Stage Etching

The unknown codewords present at each OC must be discovered and recorded in programmable address translation circuitry. If the codewords at each OC can be made the same, the size of this circuitry could dramatically reduced. As we show, core-shell NWs make this possible.

When the nanoarray is first etched, the resultant decoder can be used to discover which codewords are present at each OC. If each OC has at least C codewords, the regions under a second set of C MWs can be etched to form a second linear decoder. At each MW, at each OC, select an etching process that exposes one of the C shell type sequences known to be present. Each of the C MWs is then guaranteed to control at least one NW at each OC. When the new decoder is used, each OC contains each codeword. Programmable address translation circuitry is no longer needed.

Two-stage etching can created a deterministic linear decoder if C distinct shell type sequences are present at each OC. After codeword discovery, a custom etching process is used to ensure each of C MWs controls one of C NWs. Remarkably, an arbitrary assignment of MWs to NWs can be achieved if the C sequeneces present meet an additional criteria. Let S be the set of shell sequences present at a particular OC. There must be C sequences in S such that each sequence contains a shell type in some shell that no other sequence in S contains in that shell. If this condition is met, C arbitrary codewords can be deterministically assigned.

Core-shell NWs eliminate misalignment and provide an elegant means of fault tolerance. Two-stage etching, though time-consuming, assigns NWs codewords deterministically. This eliminates the need for programmable address translation circuitry. It also allows nanoarrays to compute functions, since each NW computes the NOR of a set of MWs.

References

[1] 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 configurable molecular-based logic gates. *Science*, 285:391–394, 1999.

[2] A. DeHon. Array-based architecture for FET-based, nanoscale electronics. *IEEE Transactions on Nanotechnology*, 2(1):23–32, Mar. 2003.

[3] A. DeHon, P. Lincoln, and J. E. Savage. Stochastic assembly of sublithographic nanoscale interfaces. *IEEE Transactions on Nanotechnology*, 2(3):165–174, 2003.

[4] B. Gojman, E. Rachlin, and J. E. Savage. Evaluation of design strategies for stochastically assembled nanoarray memories. *Submitted*, 2005.

[5] J. R. Heath, Y. Luo, and R. Beckman. System and method based on field-effect transistors for addressing nanometer-scale devices, US Patent Application 20050006671, Jan. 13, 2005.

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

[7] F. Kim, S. Kwan, J. Akana, and P. Yang. Langmuir-Blodgett nanorod assembly. *Journal of the American Chemical Society*, 123(18):4360–4361, 2001.

[8] P. J. Kuekes, W. Robinett, G. Seroussi, and R. S. Williams. Defect-tolerant Interconnect to Nanoelectronic Circuits. *Nanotechnology*, 16:869–882, 2005.

[9] L. J. Lauhon, M. S. Gudiksen, D. Wang, and C. M. Lieber. Epitaxial core-shell and core-multishell nanowire heterostructures. *Nature*, 420:57–61, 2002.

[10] N. A. Melosh, A. Boukai, F. Diana, B. Gerardot, A. Badolato, P. M. Petroff, and J. R. Heath. Ultrahigh-density nanowire lattices and circuits. *Science*, 300:112–115, Apr. 4, 2003.

[11] J. E. Savage, E. Rachlin, A. DeHon, C. M. Lieber, and Y. Wu. Radial addressing of nanowires, 2005. In prepration.

[12] D. Whang, S. Jin, Y. Wu, and C. M. Lieber. Large-scale hierarchical organization of nanowire arrays for integrated nanosystems. *Nano Letters*, 3(9):1255–1259, 2003.

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

[14] Y. Wu, R. Fan, and P. Yang. Block-by-block growth of single-crystal Si/SiGe superlattice nanowires. *Nano Letters*, 2(2):83–86, 2002.