Deterministic Addressing of Nanoscale Devices Assembled at Sublithographic Pitches

André DeHon, *Member, IEEE*

***Abstract*—Multiple techniques have now been proposed using random addressing to build demultiplexers which interface between the large pitch of lithographically patterned features and the smaller pitch of self-assembled sublithographic nanowires. At the same time, the relatively high defect rates expected for molecular-sized devices and wires dictate that we design architectures with spare components so we can map around defective elements. To accommodate and mask both of these effects, we introduce a programmable addressing scheme which can be used to provide deterministic addresses for decoders built with random nanoscale addressing and potentially defective wires. We describe how this programmable addressing scheme can be implemented with emerging, nanoscale building blocks and show how to build deterministically addressable memory banks. We characterize the area required for this programmable addressing scheme. For 2048 × 2048 memory banks, the area overhead for address correction is less than 33%, delivering net memory densities around 10^{11} b/cm^2.**

***Index Terms*—Defect tolerance, electronic nanotechnology, molecular electronics, stochastic assembly.**

I. Introduction

PREVIOUS work (e.g., [1], [2]) has demonstrated that it is possible to build decoders that allow a small number of microscale wires to uniquely address individual nanowires arranged in a tight array with sublithographic spacing between conductors. Unique nanowire addresses are created stochastically, either by random particle deposition [1] or by random selection of coded nanowires [2]. The result is that we can address individual nanowires and ultimately use the decoder to address individual memory bits in a fully nanoscale memory array. Estimates suggest that nonvolatile nanoscale memories built using these techniques could exceed densities of 10^{11} b/cm^2, providing 3–5 × the projected density of DRAMs at the 22-nm node [3].

However, in these random decoder cases, the set of live addresses is small compared to the total address space which must be used. For example, with the coded nanowire scheme, we can use an address space of 3432 unique codes to assure over 85% likelihood that 33 random wires all have unique codes [2]. Defects in the nanowires will further render some of the present addresses unusable. We typically combat these defects by providing more nanowires in the array than we expect to use. Defective wires and sparing widens further the gap between the present and functional nanowire addresses and the total address space.

In order to address the present and functional nanowires, we must either (re)discover the present addresses when we need to access the device, or we need to store away the set of addresses for known good nanowires so we can address them directly. For applications like programmable logic array (PLA) programming (e.g., [4], [5]), rediscovering the addresses when we want to (re)program the device may be viable. However, for data storage applications, it is unreasonable to search through an address space which is $O(N^2)$ or $O(N^5)$ large in order to find a particular address. Consequently, we will need to store a translation table which maps the good addresses within the total address space. Unfortunately, the size of this translation table is too large to store in a lithographic-scale memory without negating much of the density advantage of the sublithographic memory core.

To resolve this problem, we introduce a programmable address decoder. Our decoder will effectively allow us to build a nanoscale memory to map between a set of externally deterministic addresses and the known good nanowire addresses. The result is that we can program the address decoder for the present and live addresses during a testing phase. During operation, the decoder presents a simple, deterministic and compact set of addresses to the rest of the system. The decoder directly enables us to design nanoscale memory banks which can be addressed using a deterministic set of addresses.

After reviewing our sublithographic building blocks (Section II), we outline the basic scheme in Section III. In Section IV, we describe a simple, programmable decoding scheme built from programmable field-effect devices. We discuss how this deterministic addressing now enables memories with multibit word access in Section V.

II. Background

A. Technology

The key technologies we build upon are nanowires and diode-programmable crosspoints.

1) Nanowires: Semiconducting nanowires can be grown to controlled dimensions on the nanometer scale using seed catalysts (e.g., gold balls) to define their diameter. Nanowires with diameters down to 3 nm have been demonstrated [6], [7]. By controlling the mix of elements in the environment during growth, semiconducting nanowires can be doped to control their electrical properties [8]. Conduction through doped nanowires can be controlled via an electrical field like

Manuscript received December 1, 2003; revised June 30, 2005. This work was supported by the DARPA Moletronics Program under Grant ONR N00014-01-0651.

The author is with the Computer Science Department, California Institute of Technology, Pasadena, CA 91125 USA (e-mail: andre@acm.org; andre@cs.caltech.edu).

Digital Object Identifier 10.1109/TNANO.2005.858587

field-effect transistors (FETs) [9]. The doping profile along the length of a nanowire can be controlled by varying the dopant level in the growth environment over time [10]; as a result, our control over growth rate allows us to control the physical dimensions of these features down to almost atomic precision. The doping profile can also be controlled along the radius of these nanowires, allowing us to sheath nanowires in insulators (e.g., silicon dioxide) [11] to control spacing between conductors [12] and between gated wires and control wires.

Langmuir–Blodgett and flow techniques can be used to align a set of nanowires into a single orientation, close pack them, and then transfer them onto a surface [12], [13]. This step can be repeated with different angles so that we get multiple layers of nanowires. One useful construct is a pair of orthogonal crossed nanowires for building a crossbar array or memory core.

2) Programmable Diode Crosspoints: Over the past few years, many technologies have been demonstrated for molecular-scale memories (e.g., [14]–[18]). These allow us to place a nonvolatile switch at each junction of a nanowire crossbar. So far, all of these crosspoints offer the following features:

1) large change in current flow between "on" and "off" states (large $R_{\rm off}/R_{\rm on}$ ratio);
2) rectification;
3) devices which can be turned "on" or "off" by suitable application of voltage (sometimes in the presence of other environment factors, such as oxygen concentration or high temperature).

See [3] for a longer review of molecular-scale crosspoints.

B. Addressing Nanowires From Lithographic Scale Wires

The preceding technologies allow us to pack nanowires at a tight pitch into crossbars with programmable crosspoints at their junctions. The pitch of the nanowires can be much smaller than our lithographic patterning (e.g., 10–20 nm). We will be using these programmable crosspoints to serve as data memory bits. In order to do this, we need a way to selectively place a defined voltage across a pair of crossed (row and column) nanowires in order to set the state of the crosspoint.

We can give each nanowire an address by varying the doping level along its length. The dimensions of the controllable doped regions can be set to the lithographic pitch so that a set of crossed lithographic wires can be used to address a single nanowire. If we code up all of the nanowires along each dimension of the array with suitably different codes, we can get unique nanowire addressability and effectively implement a demultiplexer between a small number of lithographic wires and a large number of nanowires. We cannot control exactly which nanowire codes appear in a single array or how they are aligned, but if we randomly select nanowires from a sufficiently large code space, we will achieve uniqueness with very high probability. For N nanowires, if we use a code space with $100N^2$ codes, the probability of obtaining a unique nanowire set exceeds 99%; using even-weight codes (Section IV-C), the number of microscale address wires required is only

$$N_{\rm arnd\_codednw} = \lceil 2.2 \log_2(N) \rceil + 11. \quad (1)$$

This scheme is developed in detail in [2].

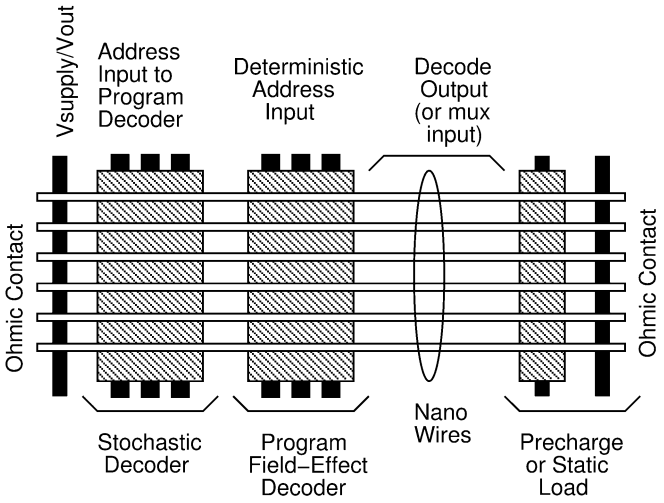

Fig. 1. Address programmable field-effect decoder.

C. Memories

DeHon *et al.* sketch how the coded nanowire address decoder can be combined with programmable diode crosspoints (Section II-A2) to build a functional, nanoscale memory bank [2]. They detail algorithms for discovering the functional addresses and bits in the array. Knowing these addresses, the memory can be written and read.

Chen *et al.* demonstrate a 4×4 memory and a programmable nanoscale decoder based on switchable crosspoint molecules [16] but do not address the general problem of array scaling or bridging efficiently between the microscale and nanoscale wires.

DeHon *et al.* assess the density and performance potential of these nanoscale memories including decoder overhead and defect losses [3]. They derive a number of combinations of lithographic and nanoscale feature sizes and yield rates which will allow us to reach net bit densities in excess of 10^{11} b/cm^2 (net bit areas smaller than 1000 nm^2/b). They abstract the issue of address correction which we address in detail in this paper. This paper will build on the area and yield analysis developed in [3].

III. Programmable Decoder Strategy

Our basic strategy for providing deterministic addressing in the face of random assembly is to make the operational address decoders programmable. This way, we can assign each address we would like to see in the array to a good wire in the array.

However, in order to program up a nanoscale junction, we generally need to address just that junction; that is, we need to place a programming voltage differential across only the microwire and nanowire which make up the junction. Consequently, we will need to start with nanowire addressability in order to bootstrap the programming process.

To achieve this addressability, we build a pair of address decoders on each set of nanowires (see Fig. 1). The first address decoder is built using the previously mentioned stochastic decoder scheme (Section II-B) to achieve unique addressability of nanowires. We can then use this address decoder to configure the programmable address decoder. During operation, we use only the programmed address decoder. Once we have this

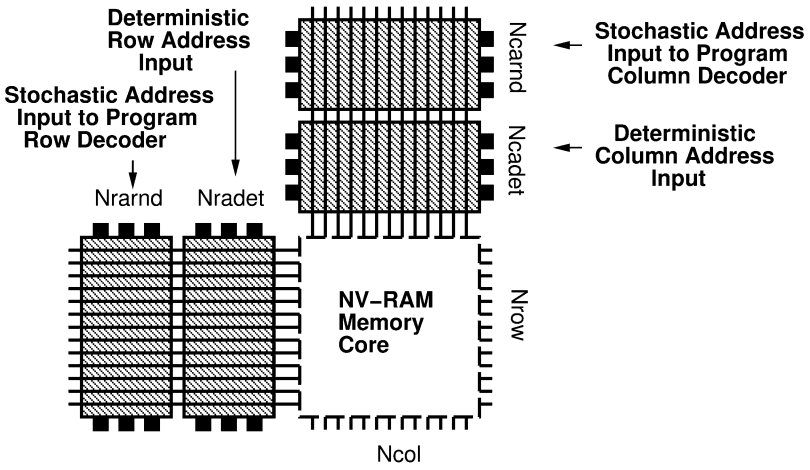

Fig. 2. Address corrected memory bank using programmable field-effect decoders.

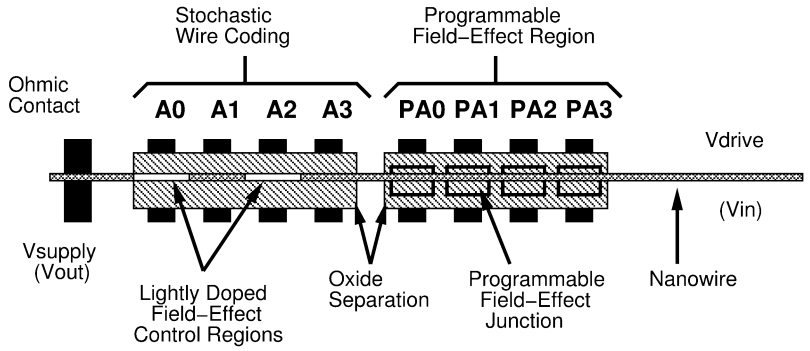

Fig. 3. Programmable address line using programmable field-effect block.

programmable address decoder scheme, we compose a pair of them to build a deterministically addressable memory or memory bank (see Fig. 2).

IV. Programmable Field-Effect Decoder

In this section, we develop the design for a programmable decoder based on programmable field-effect devices. We start by discussing the ideal model of such devices and point to technology developments which may offer nanoscale, programmable field-effect devices. We then develop the detailed design for the decoder and memory and finish up with an area and timing assessment.

A. Device

The ideal device needed for the programmable address decoder is a field-effect junction device which can be programmed to either:

- control conduction through a segment of the nanowire;
- (or) not control conduction through a segment of the nanowire.

We should be able to program the field-effect junction into one of these two states by applying a voltage field across the device junction. With devices like this, we can set each nanowire so that it is controlled by a subset of the input microscale wires. We can use a dual-rail or even-weight code (Section IV-C) to select individual nanowires.

Floating-gate devices would naturally serve this function. De Salvo *et al.* [19] suggest technology which could be used with nanowires to provide nanoscale floating-gate devices. Alternately, Huang *et al.* describe a selective oxide growth scheme which has been used to provide one-time-programmable field-effect junctions [9].

B. Address Decoder

Fig. 3 shows expanded detail on a single nanowire from the programmable field-effect decoder shown in Fig. 1. The conduction path passes from a source voltage (V_{supply} on the left), through the prefabrication doped address code on the wire, through the programmable region, and then to the decoder output (V_{drive} on the right). The lightly doped, pre-fabrication regions are controllable, requiring a high voltage for n-type nanowires (low voltage for p-type nanowires) to allow conduction. The junction squares in the programmable region indicate that each of these junctions can be programmed to be controlling or noncontrolling. Note that it is always possible to drive all microscale address inputs to one of the decoders to a pass-through voltage so that the decoder does not gate conduction. During address discovery, we use only the prefabrication addresses $(A0, A1, A2, \ldots)$ to discover which nanowire addresses are present and functional. We also use these addresses during programming to select a single nanowire and microscale wire junction. Finally, during operation, we drive all of the prefabrication addresses to an enable voltage

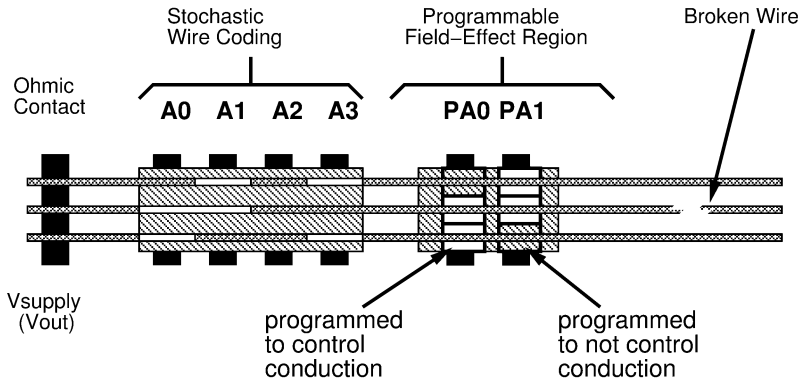

Fig. 4. Programmed programmable decoder using programmable field-effect blocks.

and use the programmable addresses $(PA0, PA1, PA2, \ldots)$ to actually address the array.

Note that there is no directionality to the conduction through the set of controllable junctions. Consequently, this same decoder structure can be used as a multiplexer, allowing conduction between a single one of the nanowires on the right and the common line (V_{out}) on the left of Figs. 1 and 3.

Fig. 4 shows a tiny decoder to illustrate programming and operation of this decoder. We use a 2-hot addressing scheme where each legal code word has exactly two 1's. Of the $\binom{4}{2} = 6$ 4-b, 2-hot codes, we see that only three of them are present in this array (0101, 1100, 1001). We further see that one of them (1100) is associated with a broken nanowire. Consequently, we want to assign our two even-weight addresses (01 and 10) to the two good wires. We program up the first good wire $(A0 : A3 = 0101)$ so it is controlled only by the second programmable address line (PA1); this allows this wire to conduct when the address 01 is presented. We program up the second good wire $(A0 : A3 = 1001)$ so it is controlled only by the first programmable address line, allowing it to conduct when the address 10 is presented. We program up all nonpresent or broken addresses so they are controlled by all lines. In this case, that means the broken wire (1100) is programmed to be controlled by both programmable address lines. Since we never use the 11 code to address any of our good address lines, this keeps the partial line from interfering with operation.

C. Addressing Schemes

Since each nanowire will be strictly n-type or p-type, we can only make junctions controllable or noncontrollable as discussed above. This means the nanowire is actually treating each address bit at the noncontrolled junctions as a "don't-care" bit rather than a bit that must have a particular value to allow conduction. Consequently, we must use coding schemes which are different from our normal binary encodings.

The simplest one to use is a dual-rail code. That is, we bring in both polarities of all address signals. In the example above, we could think about PA0:PA1 as a single address line with $\text{PA1} = \overline{\text{PA0}}$, so that the 01 encoding corresponded to address 0 and the 10 encoding corresponded to address 1. This way we simply place the controllable region under exactly one of the inverted

TABLE I
DEFAULT ORGANIZATION, FEATURE SIZE, AND YIELD PARAMETERS

Param	Value	Description
N_{row}	2048	Number of rows in memory array core
N_{col}	2048	Number of columns in memory array core
W_{litho}	108nm	Minimum lithographic full pitch (108nm is full pitch for 45nm node [20])
W_{nano}	10nm	Nanowire pitch
W_{over}	5nm	Required overlap with a lightly doped nanowire control region [2]
P_c	0.95	Probability of good micro-to-nano contact
P_{gbit}	0.95	Probability a junction can be set, cleared, and will hold its value
P_j	0.9999	Probability a wire has no fault at a junction
P_{mb}	0.98	Target probability for memory bank yield

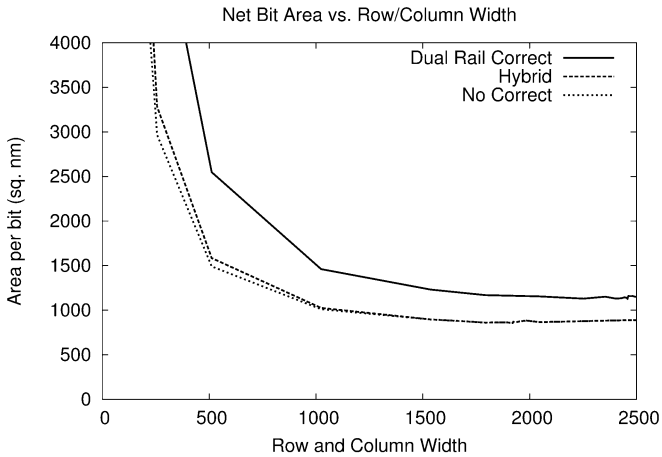

Fig. 5. Net area versus bank row and column width for translation schemes.

or noninverted inputs to assign that address bit to this code. This results in a scheme that requires twice as many address wires (N_a) as a standard binary code

$$N_a = 2\lceil \log_2(N) \rceil . \tag{2}$$

DeHon *et al.* note in [2] that an even-weight address scheme provides denser coding

$$N_a < \lceil 1.1 \log_2(N) \rceil + 3. \tag{3}$$

TABLE II
NET BIT AREA WITH VARIOUS ADDRESSING SCHEMES

Addressing Scheme			Address Bits		2048×2048 Memory	
Deterministic Code	Bootstrap Code	Eqs.	Total Bits	Example N=2048	T_{read} (ns)	A_{netbit} (nm^2)
Dual Rail	Coded Nanowire	Eq. 1 & 2	$4.2\log_2(N)$+15	64	31	1150
Even-Weight	Coded Nanowire	Eq. 1 & 3	$3.3\log_2(N)$+17	54	31	1050
Even-Weight	Hybrid (Sec. IV-F)	Eq. 3	$1.1\log_2(N)$+23	36	30	870
none	Coded Nanowire	Eq. 1	$2.2\log_2(N)$+11	36	30	870

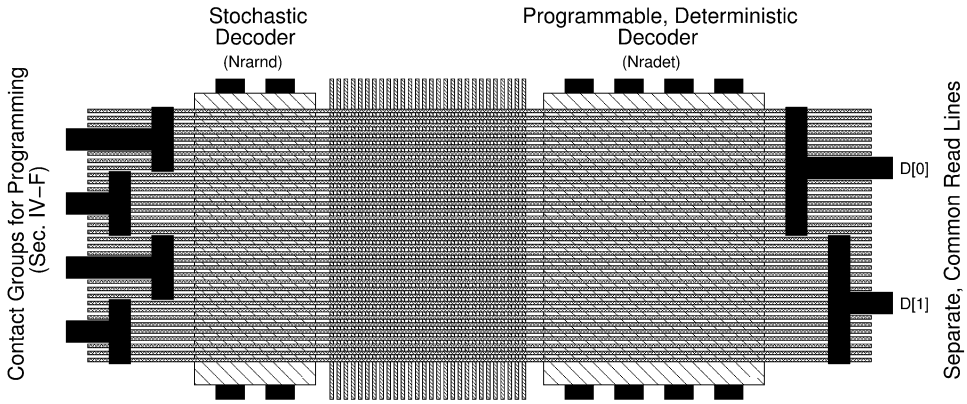

Fig. 6. Row organization for two-output memory bank with hybrid address programming.

D. Address-Corrected Memory

To build a memory, we simply replace the single stochastic row and column decoders in [2] or [3] with the programmable decoder developed here composed of a stochastic decoder and programmable decoder (Fig. 1). Fig. 2 shows the basic topology that results.

With a one-time-programmable field-effect junction, the address programming is permanent. With reprogrammable field-effect junction devices, we can reserve spares in the array and extend the device lifetime by programming around new defects that may arise during operation.

E. Analysis

All the area and timing models from [3] remain the same. We simply need to adjust the number of microscale address lines to account for both the stochastic bootstrap decoder and the programmable operational decoder. That is, we combine the stochastic and deterministic addressing lines to total up the number of address microscale wires actually used

$$N_a = N_{\text{arnd}} + N_{\text{adet}} + 2. \tag{4}$$

The extra "2" accounts for lithographic space added between the microscale wires in case the two regions need to be masked and treated differently.

For illustration we will use the parameter assumptions in Table I and decoders of size $N = 2048$. The 2048×2048 memories are some of the densest memories under these assumptions, as shown in Fig. 5. This size amortizes out the addressing overhead without driving the percentage of broken nanowires too high.

Using the area, timing, and yield models from [3] and the combined addressing schemes, Table II summarizes the net bit area achievable for several addressing schemes. The coded nanowire case with no deterministic code provides a reference point so we can quantify the overhead of the deterministic address translation scheme. The corrected schemes require less than 33% more area per bit than the uncorrected scheme.

F. Hybrid Addressing

DeHon *et al.* [2] described a hybrid addressing scheme that allows us to reduce the number of microscale address lines and reduce the total number of unique nanowire codes needed to achieve suitable uniqueness. The hybrid scheme provides separate contacts at the lithographic scale to groups of wires; that is, we replace the single common row and column lines at the end of the stochastic decoder arrays (see the left-hand side of Fig. 6) with a set of segregated microscale connections. This allows us to only use the stochastic addressing to select among sets of nanowires which cannot be distinguished by the microscale contract group. This means we can likely use $N_{\text{arnd}} = 14$–22 address bits rather than the $N_{\text{arnd}} = 36$ needed when we address all 2048 nanowires monolithically.

The simplest extension is to use the contact group segregation only to reduce the number of stochastic address bits (N_{arnd}), while keeping the programmed deterministic addresses together. This allows us to add minimal CMOS-level control while reducing the number of microscale addressing wires. Since the stochastic addresses are only used during testing and deterministic address programming, the contact groups can be efficiently addressed using a minimal CMOS shift-register rather than requiring a full, microscale decoder.

At 45 nm, we have a tight contact group width of 162 nm ($W_{\text{contact\_group\_len}} \geq 2 \times W_{\text{litho\_pitch}} - W_{\text{metal\_width}}$ using features from [20]). This means we need to use the stochastic addressing to uniquely address each of the 16 nanowires in a

contact group. Using the direct calculation from [21], $N_{\text{arnd}} = 18$ will allow us to uniquely address 16 nanowires 99.7% of the time. We add $N_{\text{adet}} = 16$ to address the 2048 net wires in each row for a total of $N_a = 36$ microscale address wires (4). This reduces the decoder area down to parity with the nonhybrid, uncorrected, coded nanowire case (see Table II).

V. Multibit Words

In memory design, we can increase the bandwidth by reading multiple data bits in parallel. This is typically done either by either of the following methods:

1) truncating the final multiplexer so that we take the row data read and multiplex it down to the desired word width rather than multiplexing it down to a single bit;
2) addressing multiple memory banks with the same address and concatenating the resulting bits or words.

Now that we have a deterministic address decoder, both schemes are viable.

To provide multiple bits from a single memory bank, we simply split the common read line into separate microscale connections to the nanowire array (see the right-hand side of Fig. 6). We then program up the nanowire addresses so that the same address is present in each of the nanowire bundles associated with a distinct microscale output contact.

In the multiple output case, a single selected column line will need to charge the capacitance of multiple row lines. Consequently, read time will increase with the number of parallel outputs from the same array. We refine the precharge read timing model from [3] to include a parallel output width W

$$\begin{aligned} T_{\text{chg\_read}} &= (R_{\text{contact}} + R_{\text{decode}}) \\ &\quad \times (C_{\text{column}} + W \times (C_{\text{row}} + C_{\text{out}})) \\ &\quad + R_{\text{ondiode}} \times (C_{\text{row}} + C_{\text{out}}) \\ &\quad + (R_{\text{contact}} + R_{\text{decode}}) \times C_{\text{out}} \qquad (5) \\ T_{\text{discharge}} &= (R_{\text{contact}} + R_{\text{decode}}) \times C_{\text{column}} \qquad (6) \\ T_{\text{charge\_rows}} &= T_{\text{discharge}} \qquad (7) \\ T_{\text{chg\_read\_cycle}} &= T_{\text{charge\_rows}} + T_{\text{discharge}} + T_{\text{chg\_read}}. \qquad (8) \end{aligned}$$

For the 2048×2048 example array considered here (Table I), we add about 12 ns of delay per output. We saw the single output case had a read cycle of 30 ns; the $W = 2$ case has a delay of 42 ns, and the $W = 4$ case has a delay of 67 ns.

VI. Summary

Nanoscale memories will come with high defect rates and are likely to require stochastic addressing. By using a dual decoder scheme, we can identify the functional nanowires and program deterministic addresses after fabrication to avoid the defective nanowires. Using programmable field-effect devices, the programmable decoder adds less than 33% net area per bit overhead compared to the baseline uncorrected memory case; with the hybrid scheme, we essentially eliminate this overhead. These results demonstrate that we can provide deterministic addressing into nanoscale memories without negating their density benefits.

Acknowledgment

Architecture work at this early stage is only feasible and meaningful in close cooperation with scientists working on device properties and fabrication. Special thanks to C. Lieber for his support for this research and design.

References

[1] S. Williams and P. Kuekes, "Demultiplexer for a Molecular Wire Crossbar Network," U.S. Patent 6 256 767, Jul. 3, 2001.
[2] A. DeHon, P. Lincoln, and J. Savage, "Stochastic assembly of sublithographic nanoscale interfaces," *IEEE Trans. Nanotechnol.*, vol. 2, no. 3, pp. 165–174, Sep. 2003.
[3] A. DeHon, S. C. Goldstein, P. J. Kuekes, and P. Lincoln, "Non-photolithographic nanoscale memory density prospects," *IEEE Trans. Nanotechnol.*, vol. 4, no. 2, pp. 215–228, Mar. 2005.
[4] A. DeHon, "Array-based architecture for FET-based, nanoscale electronics," *IEEE Trans. Nanotechnol.*, vol. 2, no. 2, pp. 23–32, Mar. 2003.
[5] A. DeHon and M. J. Wilson, "Nanowire-based sublithographic programmable logic arrays," in *Proc. Int. Symp. Field-Programmable Gate Arrays*, Feb. 2004, pp. 123–132.
[6] 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.
[7] A. M. Morales and C. M. Lieber, "A laser ablation method for synthesis of crystalline semiconductor nanowires," *Science*, vol. 279, pp. 208–211, 1998.
[8] Y. Cui, X. Duan, J. Hu, and C. M. Lieber, "Doping and electrical transport in silicon nanowires," *J. Phys. Chem. B*, vol. 104, no. 22, pp. 5213–5216, Jun. 2000.
[9] 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, Nov. 2001.
[10] 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.
[11] L. J. Lauhon, M. S. Gudiksen, D. Wang, and C. M. Lieber, "Epitaxial core-shell and core-multi-shell nanowire heterostructures," *Nature*, vol. 420, pp. 57–61, 2002.
[12] D. Whang, S. Jin, and C. M. Lieber, "Nanolithography using hierarchically assembled nanowire masks," *Nanolett.*, vol. 3, no. 7, pp. 951–954, Jul. 2003.
[13] Y. Huang, X. Duan, Q. Wei, and C. M. Lieber, "Directed assembly of one-dimensional nanostructures into functional networks," *Science*, vol. 291, pp. 630–633, Jan. 2001.
[14] C. Collier, G. Mattersteig, E. Wong, Y. Luo, K. Beverly, J. Sampaio, F. Raymo, J. Stoddart, and J. Heath, "A [2]Catenane-Based solid state reconfigurable switch," *Science*, vol. 289, pp. 1172–1175, 2000.
[15] 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.
[16] Y. Chen, G.-Y. Jung, D. A. A. Ohlberg, X. Li, D. R. Stewart, J. O. Jeppesen, K. A. Nielsen, J. F. Stoddart, and R. S. Williams, "Nanoscale molecular-switch crossbar circuits," *Nanotechnol.*, vol. 14, pp. 462–468, 2003.
[17] D. R. Stewart, D. A. A. Ohlberg, P. A. Beck, Y. Chen, R. S. Williams, J. O. Jeppesen, K. A. Nielsen, and J. F. Stoddart, "Molecule-independent electrical switching in pt/organic monolayer/ti devices," *Nanolett.*, vol. 4, no. 1, pp. 133–136, 2004.
[18] 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.
[19] B. D. Salvo, G. Ghibaudo, G. Pananakakis, P. Masson, T. Baron, N. Buffet, A. Fernandes, and B. Guillaumot, "Experimental and theoretical investigation of nano-crystal and nitride-trap memory devices," *IEEE Trans. Electron Devices*, vol. 48, no. 8, pp. 1789–1799, Aug. 2001.
[20] (2003) International Technology Roadmap for Semiconductors. [Online]. Available: http://public.itrs.net/
[21] A. DeHon, "Law of large numbers system design," in *Nano, Quantum and Molecular Computing: Implications to High Level Design and Validation*, S. K. Shukla and R. I. Bahar, Eds. Boston, MA: Kluwer, 2004, ch. 7, pp. 213–241.

André DeHon (S'92–M'96) received the S.B., S.M., 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, Computer Science Department, University of California, Since 1999, he has been an Assistant Professor of computer science with the California Institute of Technology, Pasadena. He is broadly interested in the physical implementation of computations from substrates, including VLSI and molecular electronics, up through architecture, computer-aided design, and programming models. He places special emphasis on spatial programmable architectures (e.g., field-programmable gate arrays) and interconnect design and optimization.