Array-Based Architecture for FET-Based, Nanoscale Electronics

André DeHon, *Member, IEEE*

***Abstract*—Advances in our basic scientific understanding at the molecular and atomic level place us on the verge of engineering designer structures with key features at the single nanometer scale. This offers us the opportunity to design computing systems at what may be the ultimate limits on device size. At this scale, we are faced with new challenges and a new cost structure which motivates different computing architectures than we found efficient and appropriate in conventional very large scale integration (VLSI). We sketch a basic architecture for nanoscale electronics based on carbon nanotubes, silicon nanowires, and nano-scale FETs. This architecture can provide universal logic functionality with all logic and signal restoration operating at the nanoscale. The key properties of this architecture are its minimalism, defect tolerance, and compatibility with emerging bottom-up nanoscale fabrication techniques. The architecture further supports micro-to-nanoscale interfacing for communication with conventional integrated circuits and bootstrap loading.**

***Index Terms*—Bootstrapping, electronic nanotechnology, molecular electronics, nanoscale FET logic, programmable architecture.**

I. Introduction

WE SHOW how to organize the carbon nanotubes (CNTs), silicon nanowires (SiNWs), and molecular-scale devices that are now being developed into an operational computing system. The molecular-scale wires can be arranged into interconnected, crossed arrays with nonvolatile switching devices at their crosspoints; these crossed arrays can function as programmable-logic arrays and programmable interconnect (see Fig. 1). Using nanoscale FET devices, we provide both signal restoration and programming support for the nonvolatile switches. The result is a programmable logic device that can be configured to compute any logical function and that operates entirely at the nanoscale. Defect-tolerance is an essential component of this architecture allowing it to cope with the high defect rates associated with bottom-up synthesis.

A. Technology

1) Wires: Today, chemists can synthesize CNTs which are nanometers in diameter and microns long [1]. We can control the growth and alignment of these nanotubes such that they can be assembled into parallel rows of conductors and layered into arrays [2]. Ultimately, these CNTs can be a single nanometer wide and spaced several nanometers apart. To date, we cannot control the detailed electrical properties (conducting versus semiconducting) for these nanotubes, but the conduction of even the worst conductors is often adequate for many uses.

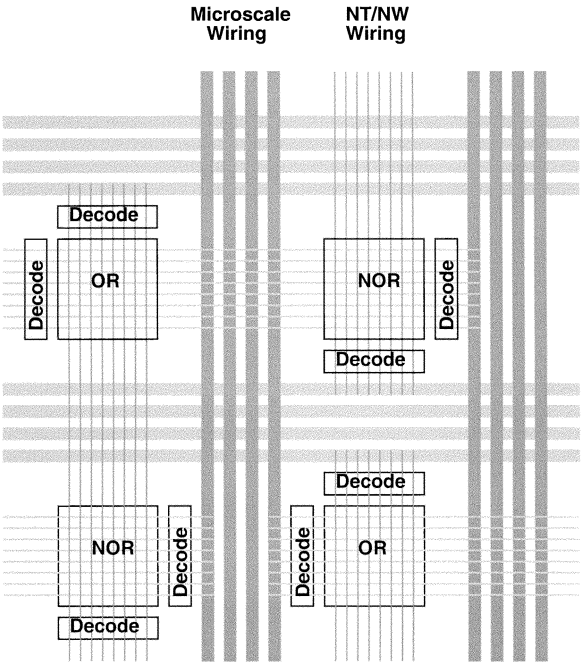


Fig. 1. Overall assembly of functional nanoarrays.

At the same time, we are developing technologies to grow silicon and germanium NWs [3], [4], which are also only nanometers in width (e.g., wires as small as 3 nm in diameter have been reported) and can be grown or assembled into sets of long parallel wires [5]. We can control the electrical properties of these SiNWs with dopants, yielding semiconducting wires [6]. NWs can be assembled along with nanotubes when their respective properties complement each another.

2) Devices: Lieber and his students have shown switched devices using suspended nanotubes [7] (see Fig. 2). The NT–NT junction is bistable with an energy barrier between the two states. In one state, the tubes are "far" apart and mechanical forces keep the top wire from descending to the lower wire. At this distance the tunneling current between the crossed conductors is small, resulting, effectively, in a very high resistance between the conductors (GΩs). In the second state, the tubes come into contact and are held together via molecular forces. In this state, there is little resistance (100 kΩ) between the tubes. By applying a voltage to the tubes, one can charge them to the same or opposite polarities and use electrical charge attraction/repulsion to cross the energy gap between the two

Received July 27, 2002; revised October 22, 2002. This work was supported by the Defense Advanced Research Project Agency (DARPA) Moletronics Program under Grant ONR N00014-01-0651.

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

Digital Object Identifier 10.1109/TNANO.2003.808508

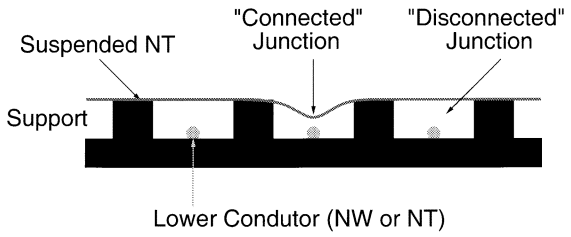


Fig. 2. Suspended NT switched connection.

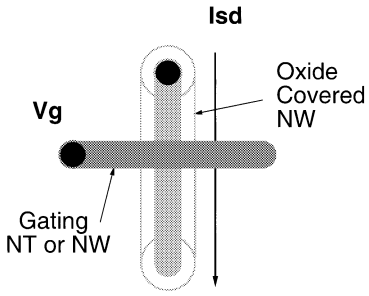


Fig. 3. NT–NW FET device.

bistable states, effectively setting or resetting the programming of the connection. SiNWs can be substituted for the lower wire, and these junctions can be rectifying such that the connected state exhibits p-n-diode rectification behavior.

Doped SiNWs exhibit FET behavior [8]. That is, oxide can be grown over the SiNW to prevent direct electrical contact of a crossed conductor (see Fig. 3). The electrical field of one wire can then be used to "gate" the other wire—locally evacuating a region of the doped SiNW of carriers to prevent conduction. FET resistance varies from ohms (likely, but not currently measured) to gigaohms. CNTs also demonstrate FET behavior [9]–[11].

Further the Heath and Stoddard groups at University of California, Los Angeles (UCLA) and Hewlett-Packard (HP) have demonstrated molecules which appear to exhibit orders of magnitude different resistance in different states [12]. The molecules can be irreversibly disconnected by applying a voltage across the junction. They sketch how to assemble an aligned, single layer of these molecules between nanoscale conductors such as SiNWs or CNTs. The result can be used as a one-time programmable memory array.

An interesting consequence of all these devices is the ability to store state and implement switching at a wire crossing. That is, the switch device itself holds its state. Contrast this with a programmable switchpoint in an SRAM-based programmable-logic array (PLA) or field-programmable gate array (FPGA), where the area to hold the memory cell and switch are much larger than a primitive wire crossing (e.g., 2500 λ^2 for a small pass-gate switch with memory versus 25–50 λ^2 for a wire crossing). So, even if we achieve 35-nm silicon feature sizes (which might imply 70–90-nm wire pitches), the density difference between 20-nm spaced nanotubes or SiNWs and the 35-nm silicon will be greater than the roughly (80 nm/20 nm)^2 wire feature size difference. This difference in relative costs also has an impact on architecture. Whereas, full crossbars in silicon are switch dominated, motivating us to depopulate them for compactness, crossbars in this technology can be fully populated with no density penalty. This is particularly beneficial in achieving the necessary defect tolerance.

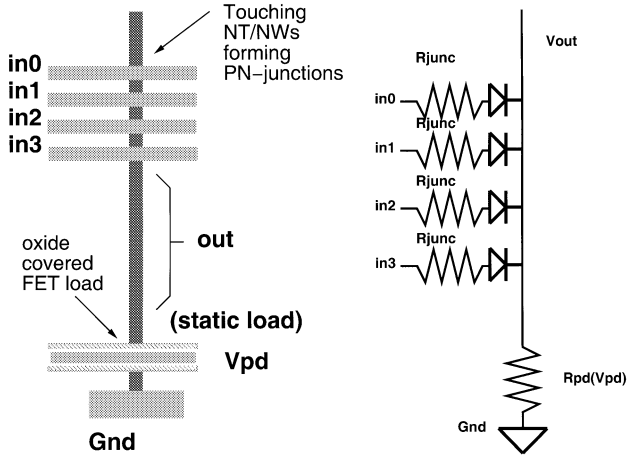


Fig. 4. Diode OR arrangement.

3) Near Term: Based on the current successes and understanding, in the near term (next five years), it appears plausible we will be able to assemble modest size arrays of crossed conductors with one or more of the aforementioned device effects at the junctions of wires. Regular arrays of uniform length wires and identical junctions at the nanoscale look feasible. Defects in this regular structure will exist, as we rely on synthesis procedures and statistical assembly which offers only probabilistic yield of wires and connections. Varying the lengths of wire runs or device properties can be done only at the microscale, where we have traditional lithographic techniques to specify differentiated growth and assembly conditions.

B. Architectural Strategy

Armed with these building blocks and properties, we consider an architecture based on a collection of interconnected arrays (see Fig. 1). The crossed arrays can act as memory cores, PLA planes and crossbars—memory, compute, and interconnect—all the key elements we need to implement computations. Further, each of these structures is amenable to sparing and remapping to avoid inevitable faults in the base array. A single, monolithic memory, PLA, or crossbar would not be useful or efficient (e.g., [13]–[15]), but a collection of interconnected arrays allows us to both exploit logical structure and isolate faults.

Key issues in the design include the following:

1) achieving gain for signal restoration (Section II);
2) interfacing between our conventional, microscale features and the nanoscale circuits (Section III);
3) bootstrapping array personalization (Section III);
4) configuring functional logic around defective devices (Section IV-B).

C. Related Work

The strategy detailed here follows the high-level vision articulated by Heath [16]. We provide a complete sketch showing how these technologies can be organized into a functional architecture.

Goldstein introduces nanoFabrics [17], an architecture based on molecular-scale electronic building blocks. Goldstein care-

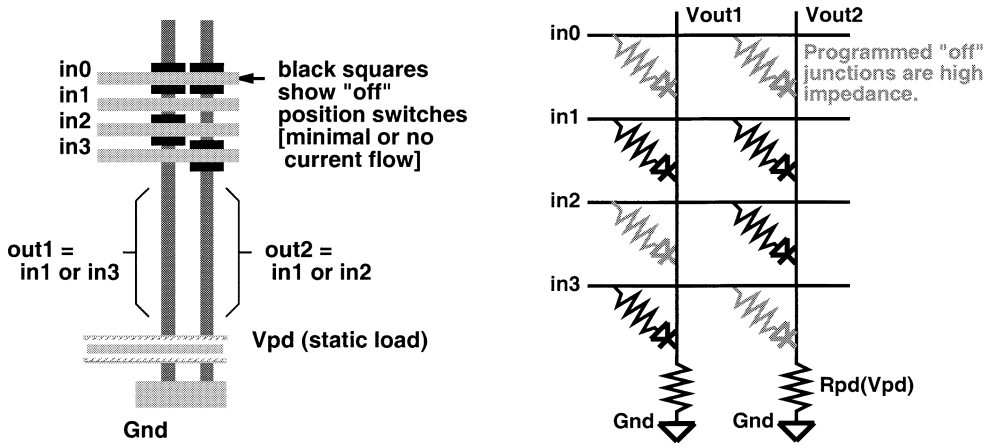


Fig. 5. Programmable diode OR array.

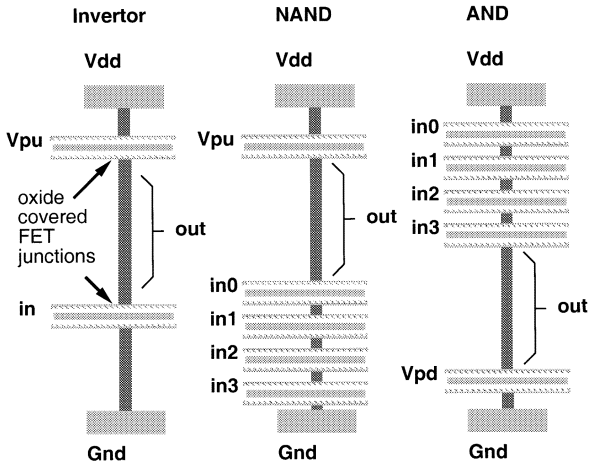


Fig. 6. FET logic arrangements.

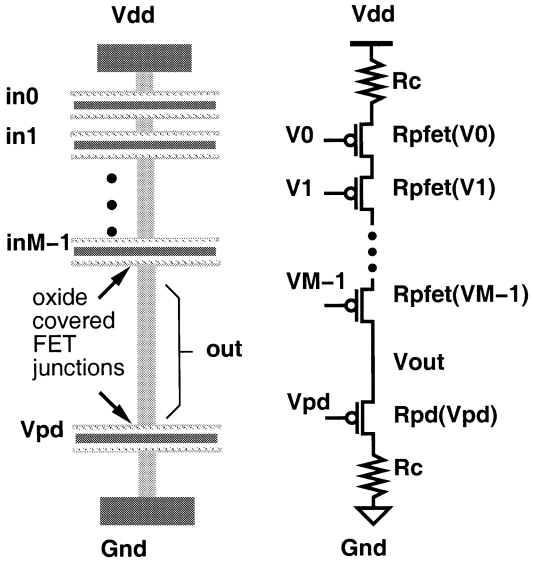


Fig. 7. PFET NOR circuit.

fully restricts the nanoFabric to use only two-terminal devices. In contrast, we show array designs which are enabled by the SiNW and CNT FETs, which are now emerging. We show how FET circuits allow direct signal restoration and detail how they enable nanoscale addressing. The resulting designs may be simpler to assemble and repair.

II. Electrical Operation

At present the switch molecules and suspended tube diode junctions appear to act entirely as passive devices. The tube diode connections allow us to build wired-OR logic (see Fig. 4). Using the suspended switching, we can assemble configurable OR planes, with connected wires acting as low-resistance p-n-junctions and distant wires isolated by high resistance (see Fig. 5). We can use these passive devices in our switching to implement programmable logic arrays, but since they do not provide gain, we cannot build closed systems entirely out of these devices. We must bracket them with restoring logic either at the microscale or at the nanoscale in order to build robust digital logic.

The FET SiNW junctions appear to be our current best technology for signal restoration at the nanoscale. Using these devices, we can build NMOS-like inverters, NAND, AND, NOR, or OR logic (see Figs. 6 and 7). We can build these into fixed logic arrays for restoration between programmable, suspended tube or switched molecule arrays, or we can build these as programmable logic array stages themselves.

For brevity we will focus on the electrical operation of the restoring FET NOR stage using a p-type SiNW and a PMOS-like logic discipline. Logically, using only NOR arrays is sufficient to achieve universal logic. The inverter and OR stages are straightforward variations on this arrangement.

Fig. 7 shows the logical arrangement and corresponding circuit model for a PFET NOR. The depletion-mode PFETs conduct with low resistance in their default state and increase their resistance as the gate voltage is increased (see Fig. 8). We can characterize the output voltage as

$$V_{\rm out} = V_{\rm dd}\left(\frac{R_{\rm pd}+R_c}{R_c+\sum_{i=0}^{M-1}\left(R_{\rm PFET}(V_i)\right)+R_{\rm pd}+R_c}\right).$$

M is the number of inputs to the NOR gate (as shown in Fig. 7). Current experimental characterization suggests that the contact resistance (R_c) is on the order of 1 MΩ [8], [18]; this resistance

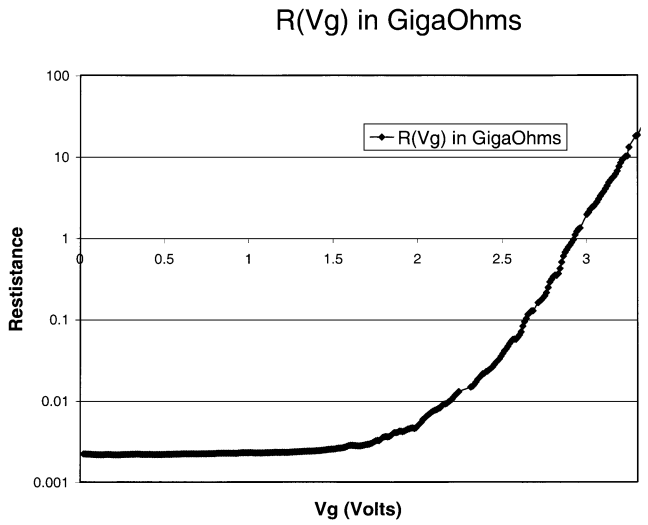


Fig. 8. PFET resistance versus gate voltage (V_g) from [8]: at the low voltage end, the 2.2 MΩ measured is due to the contact resistance of the measurement setup not the FET ON resistance.

may decrease as our mastery of this technology improves. For low voltages, the resistance of the PFETs is so small as to not be measurable compared to the contact resistance ($R_{\rm PFET}$ (small V_g) $\ll R_c = 1$ MΩ).

Qualitatively, when all the inputs are low, the output should go to a high value—close to the rail and above our designated $V_{\rm oh}$. As noted, the ON-resistance of the PFETs is low, so as long as we can make $R_{\rm pd} \gg R_c$, the pull-up resistance is small compared to the pull-down resistance, and $V_{\rm out}$ becomes close to $V_{\rm dd}$. Consequently, we want to set $V_{\rm pd}$ such that $R_{\rm pd} = R_{\rm PFET}(V_{\rm pd}) \approx 9R_c$. In order for the logic function to work, it must also be possible for a single input with a logical high input voltage to make the resistance of the pull-up large compared to the pull-down resistance so the output goes below our designated $V_{\rm ol}$ voltage. That means: $R_{\rm PFET}(V_{\rm ih}) \gg (R_{\rm pd} + R_c) = 10R_c$. The OFF-resistance of the PFETs is in the 100s of gigaohms, so this is easily obtainable as well. A sample set of operating voltages derived from the data in Fig. 8 is shown in Table I.

The operating point here is set by the placement of the high gain region and, hence, the effective threshold voltage. With care controlling the doping and geometry of the NWs, it is possible to lower the threshold voltage. Recent experiments have placed the entire high-gain region below half a volt, suggesting it may be possible to operate with a 1-V supply [18].

The slowest operating time for this gate will be charging the output node through the large pull-down resistance. The pull-down path resistance will be ∼10 MΩ. The capacitance of a 1-μm NT will be $C_{\rm wire} \approx 3{\times}10^{-16}$ F (calculation based on data in [6]), and SiNW capacitance is comparable. The *RC*-delay for pull-down is thus $T_{\rm pd} \approx 10$ MΩ $\times 3 \times 10^{-16}$ F ≈ 3 ns. Note that this speed is largely set by the contact resistance and can be reduced as better control of the manufacturing process allows us to reduce the contact resistance.

Worst-case static power comes from the voltage divider when the path resistance is minimum; that is, when all the inputs are low. The resistance here is $R = 2R_c + R_{\rm pd}$, or roughly

TABLE I
OPERATING VOLTAGES FOR PFET NOR ASSUMING R–V CHARACTERISTICS SHOWN IN FIG. 8

V_{dd}	3.3V
V_{oh}	3.0V
V_{ih}	2.8V
V_{il}	0.5V
V_{ol}	0.15V
V_{pd}	2.4V

10 MΩ. Static power is $P_{\rm NOR} = (V_{\rm dd})^2/R$. At $V_{\rm dd} = 3.3$ V, $P_{\rm NOR} \approx 1$ μW. At 1 V, $P_{\rm NOR} \approx 0.1$ μW. The topology for this static-load logic is particularly simple and regular making it compatible with bottom-up fabrication techniques. In future work, we will explore alternatives to reduce or eliminate static power while retaining as much of this simplicity as possible; if noise can be contained sufficiently, precharge logic structures might be a reasonable alternative. Precharge would further allow us to avoid the ratioed pull-down, making the critical delay term proportional to the contact resistance (R_c) instead of ten times the contact resistance as shown above.

III. BOOTSTRAPPING

Bootstrapping presents several challenges. The fabricated device will have no personalization and contain numerous defects. We must:

1) connect between the microscale lithographic world and the nanoworld;
2) do so in a manner which allows us to retain the nanoscale pitch;
3) be able to program the nanoscale connections before we can use them;
4) arrange for the programming facilities not to interfere with normal operation of the device.

A. Nanoscale Addressing

As noted above (Section I-A2), if we can apply a voltage to a horizontal and vertical NW or NT, we can change the state of the device at their intersection. Our first challenge is to get to the point where we can selectively apply a voltage to a single horizontal and vertical NW/NT pair when packed at nanoscale density. If we simply drove each nanoscale wire directly from a lithographic microscale wire, we would achieve wire densities no greater than that of the lithographic wire. To exploit the increased density, we use FET decoders to allow a small number of microscale wires to connect to a larger number of nanoscale wires.

We place a small, nanoscale decoder block on the edge of a NW array. The decoder has N wires which connect to the core NW array and a smaller number of address wires, N_a, which connect to an orthogonal set of microscale wires through nanovias (see Fig. 9). N_a could be as small as $O(\log(N))$ wires; however, if we use such a dense encoding a single fault in the

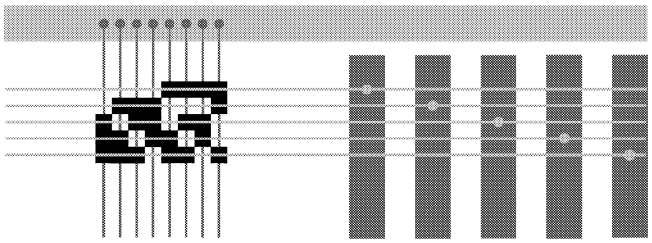


Fig. 9. Programmed decoder.

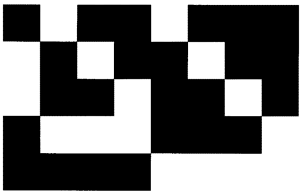


Fig. 10. Decoder imprint pattern.

address wires could render large portions of our array inaccessible (e.g., a single address line fault in the densest codes will render half of the array inaccessible). Instead, we are considering a two-hot coding scheme where every core wires is enabled by AND-ing together a pair of address wires. This makes $N_a = O(\sqrt{N})$ and guarantees that we only lose $O(\sqrt{N})$ wires on any address fault. Further note that we reserve one code which will not select any of the core wires for the case where all the array wires should be disconnected from the associated supply.

We cannot program the decoder at the nano-micro scale interface as we intend to program the core. The address lines which are connected directly to the microscale wires can be driven to a voltage by conventional electronics. However, we have no way to drive the nanoscale wires which drive into the array. To address this, we customize the decoder pattern during fabrication. For example, we may imprint the pattern of blocks between the orthogonal layers of nanoscale wires in order to personalize the decoders (see Fig. 10). Where the pattern leaves openings, the two layers are allowed to contact producing a strongly coupled FET arrangement. Where the blocks prevent the crossed wires from contacting, the crossed NWs are far enough apart that they do not control each other (see Fig. 9). The patterning does not need to be perfect here. What is important is that we have a code that allows us to address most of the nanoscale wires independently; it does not matter which code addresses which nanoscale wire, and we can tolerate not being able to address a small fraction of the nanoscale wires. This may allow us to use emerging techniques for nano-imprinting which avoid direct, lithographic limitations (e.g., [19]). The decode is the only feature of this design that may require direct patterning of nanoscale features. We are exploring ways to avoid even this requirement. For example, Williams and Kuekes [20] have proposed stochastic self-assembly techniques as an alternate scheme for constructing this kind of decoder without being limited by to photolithographic dimension.

These decoders are placed on either side of a nanoscale array in both dimension. Fig. 11 shows a simple, but nonoperational, arrangement of this bracketing. Using these decoders, it is now

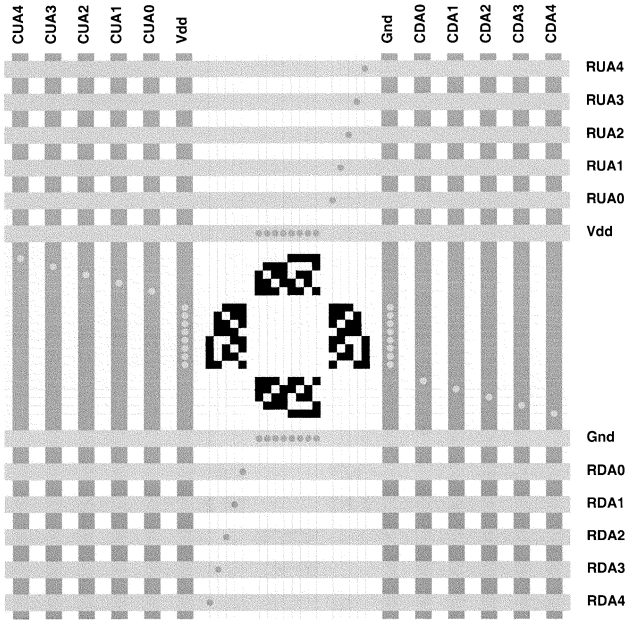


Fig. 11. Array bracketed with decoders: Shown here is an 8×8 nanoscale wire array bracketed by the decoders used to program the array and connections to microscale wires. As shown, the array is small compared to the microscale wires. Note, however, that the number of microscale wires scales as the square root of the array width; for the larger nanoarray sizes we consider typical, the microscale wiring becomes a thin periphery around a large nanoscale array core.

possible to drive any single horizontal or vertical tube to a high or low voltage and leave the other tubes floating, as we need to do for programming. We can drive a tube high by driving the exposed PFET NW crossings in the decoder low—that would be all the address lines necessary to select this tube; driven this way, we have a low-impedance path from the core portion of the selected tube to the high-voltage supply. Assuming we drive the pull-down network with a code which places all the pull-down paths in a high-impedance state, this means that *only* this line is driven and all the other lines are left to float to high impedance. We can drive a tube low in a similar manner by driving appropriate address into the pull-down network and a disable address into the pull-up network.

B. Operation

During normal operation, we do not want the decoders to drive the nanoscale wires. Rather, the nanoscale wires will be performing logic of their own. By driving both the pull-up and pull-down decoders with high addresses, we isolate the array completely from the programming FETs. For p-n-diode connected arrays such as the suspended NT devices, we will need to isolate the programming from the array in this manner.

For the FET logical arrays described earlier, the programming FETs perform a dual function; during operation these FETs can serve as the static pull-down (or pull-up) load. Fig. 12 shows a typical setup and the equivalent logical circuit for a single PFET NOR. The decoding FETs are placed in series between the contact resistance and the output or input FETs (compare Fig. 7). By driving all of the pull-up PFETs low (i.e., driving all the address lines low), the PFETs will act as wires. If we further drive the pull-down decoder with a suitable $V_{\rm pd}$, then

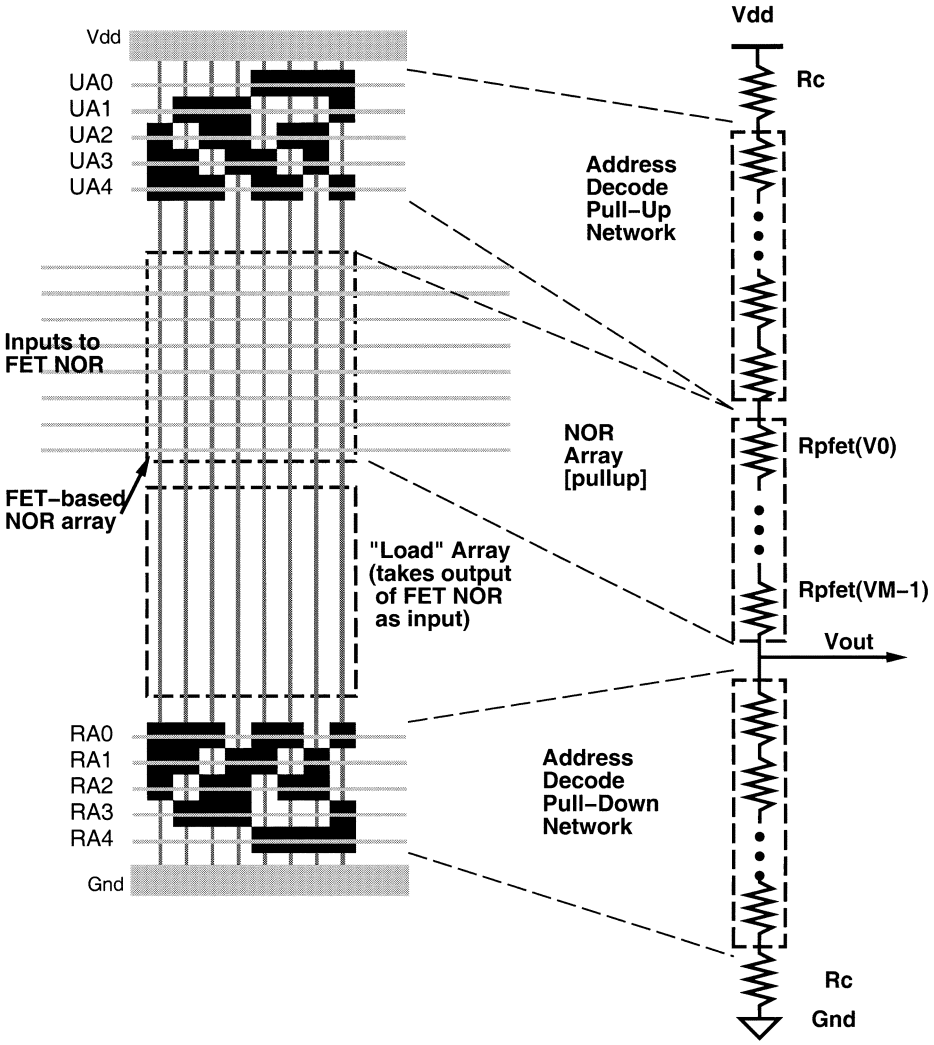


Fig. 12. Operating FET NOR array bracketed by decoders.

this becomes the NOR circuit we identified earlier (Fig. 7) with the pull-down FET network serving as $R_{\rm pd}$.

We may be able to personalize these FET arrays by using the same suspended tube scheme used for the p-n-junctions. We use the FET decoders to move the crossed wires into either a close contact position or separated position (see Fig. 2). In this case, however, one or both of the wires has an oxide coating so that the close coupled case exhibits FET rather than p-n-junction behavior. In the far case, the wires should be sufficiently separated that we get small field effects between the crossed wire. In this manner, we can "program" the behavior of the FET array similar to the way we would program the behavior of the NOR plane in a conventional PLA.

Alternately, we can alternate diode-based nanoarrays with the FET NOR nanoarrays. Notably, if only the diode-arrays are programmable, we can use imprinting to pattern fixed-connectivity NOR stages. Together, the programmable diode OR and fixed NOR pair provide both logic programmability and signal restoration, realizing a PAL-like logic structure [21], [22].

In either case, the programming voltages to switch the state of a wire junction should be higher than the operating voltages for the FET or diode logic. This is necessary to prevent the devices from being inadvertently reprogrammed during normal operation. To achieve this, we will place different voltages on the decoder's supply voltages (nominally $V_{\rm dd}$ and $V_{\rm gnd}$) during programming and operation. Further, note that this FET decoder scheme should work with any devices with nonvolatile junction state switched using voltages, including, perhaps the UCLA–HP molecular switches [12].

Note that the "output" of each NOR circuit appears on the NW between the input array of crossed wires and the pull-down enable. To use these as subsequent inputs to another stage of logic we simply arrange to place the other array orthogonal to this array such that its input aligns with this array's output (see Fig. 12). A similar situation occurs for any of the kinds of array logic (e.g., OR, NAND, AND); the output will be some portion of the wire, and we arrange for that portion of the wire to cross an orthogonal array as the intended inputs. This allows us to use a simple manufacturable topology of crossed NTs or NWs while achieving efficient interconnection of functions.

IV. Organization

We organize the nanoarray cells detailed in the previous section into large arrays. Each nanoarray has wires overlapping

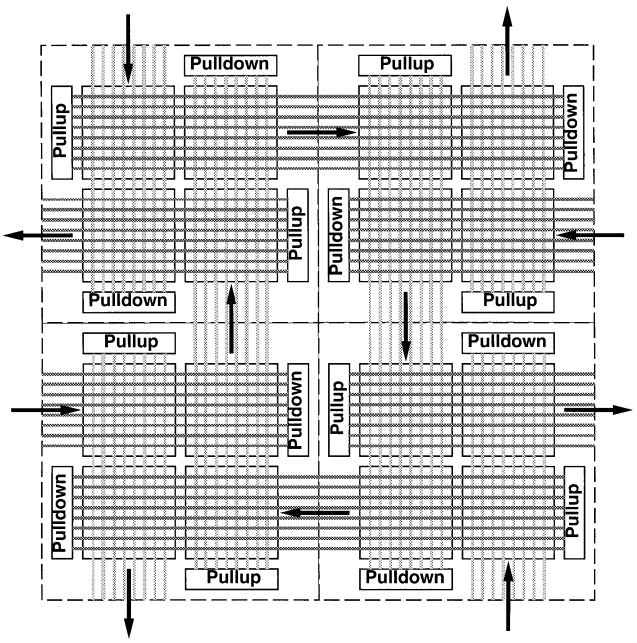


Fig. 13. NOR-only macrotile for routing. In this more realistic topology, we build a logical NOR plane out of a 2×2 arrangement of crossed nano-arrays (microscale wires, as shown in Fig. 1, exist but are omitted here to simplify the diagram). This arrangement allows inputs to enter from either side of the NOR-plane and outputs to depart in either orthogonal direction. Assembled into the macrotile shown, array entry and exit freedom allows us to route signals in both dimensions, providing arbitrary Manhattan routing. This macrotile is abutted in both dimensions to build larger devices.

with adjacent arrays for interarray communication (see Figs. 1 and 12). In simplest form, all nanoarrays can be FET-based NOR arrays. Careful arrangement of overlap topologies and array inversions (e.g., OR and NOR) will allow routing and signal polarity control. Fig. 13 shows a NOR-only macrotile, which can be abutted horizontally and vertically to allow arbitrary Manhattan routing within the master array. In more complex configurations, we can alternate diode and FET-based nanoarrays as described in the previous section.

A. Raw Crosspoint Density

Within the core of a nanoarray, we get one crosspoint every molecular-scale wire pitch ($W_{\text{molecular}}$) such that each crosspoint takes up $W^2_{\text{molecular}}$ area. The effective density is lower than this due to the CMOS and address support needed for each subarray. Reviewing Fig. 1, we see that each subarray core is bracketed by a decoder and a set of microscale address lines. The total width of an N-tube wide nanoarray tile is

$$S_{\text{array}} = W_{\text{cmos}} \times (N_a) + W_{\text{molecular}} \times (N + N_a). \quad (1)$$

W_{cmos} is the CMOS wire pitch. A minimum 2-hot addressing scheme requires

$$N_a = \left\lceil \sqrt{2N} \right\rceil + 1. \quad (2)$$

From this, we can calculate the effective area of each crosspoint bit

$$A_{\text{bit}} = \frac{S^2_{\text{array}}}{N^2}. \quad (3)$$

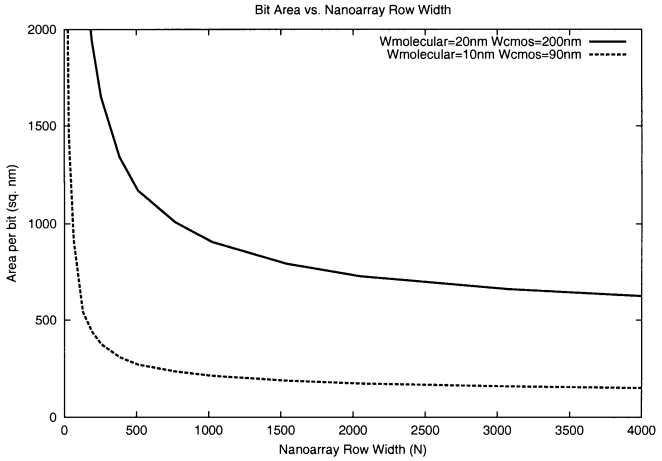


Fig. 14. Raw effective crosspoint density.

Fig. 14 shows the raw crosspoint density for $W_{\text{molecular}} = 20$ nm and $W_{\text{cmos}} = 200$ nm, a design point which might be achievable in a few years, and $W_{\text{molecular}} = 10$ nm and $W_{\text{cmos}} = 90$ nm, a design point which might be achievable in 2010 [23]. Densities here should be compared to the raw area per bit in the core of 400 nm^2 for a 20-nm molecular scale pitch and 100 nm^2 for a 10-nm pitch. For these sizes we achieve 50% of the core cell density (800 nm^2/cell, 200 nm^2/cell) with nanoarray widths around $N = 1500$ and 1000, respectively.

B. Defect Tolerance

When assembled into arrays, some of the nanoscale wires will have poor or nonexistent contacts and individual switches may be nonfunctional. This architecture is designed to tolerate these defects by both local wire sparing and array sparing.

There is no logical significance to which wire we use to collect the output of a logical OR or logical NOR function. As long as we fabricate more wires in the array than we actually need, we can simply avoid the faulty wires and switches and perform our logical operations on the functional wires (see Fig. 15). We pick the base array size and the level of sparing included in the array based on the specific defect rate we expect at any point in time in much the same way one designs spare rows and columns in conventional DRAM memories.

Sparing is done hierarchically as well. There will be many different instances of the base crossed-wire array in any system. We designate some of these arrays as spares. If the number of faulty wires in some arrays or decoders exceeds the designed level of sparing, we can then discard those entire arrays, using only the repairable arrays which remain in the design. Multiple, independent paths through different arrays in the design allow us to route completely around any such faulty arrays.

C. Net Density With Faults

We consider two main causes of defects in the NT/NW structures:

- contact connection fails—with probability P_c the contact at one end of the NT or NW is sufficiently poor as to be unusable;

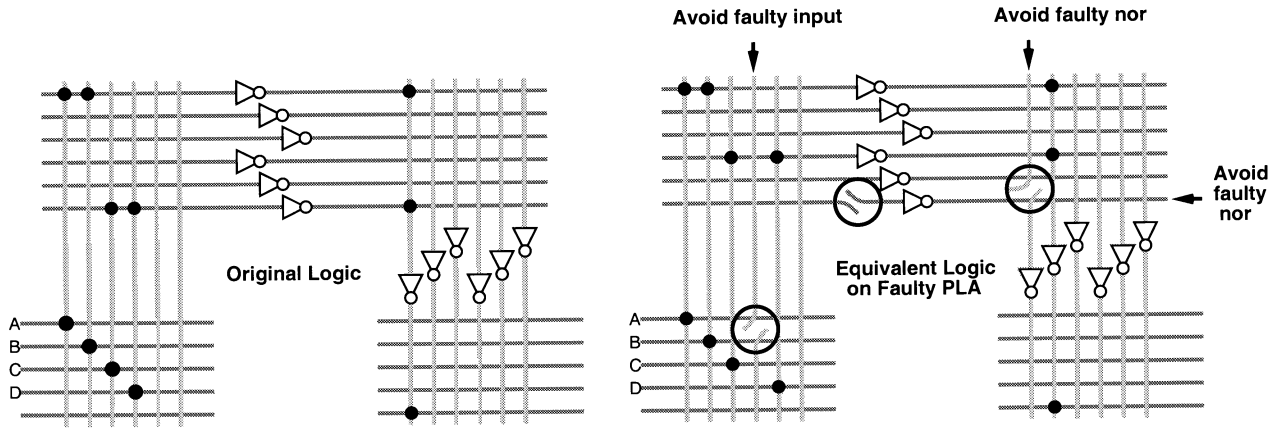


Fig. 15. Sparing in crossed-wire planes to avoid faults: All lines in a PLA or crossbar are equivalent. With spare lines, we can use this property to avoid faulty lines. In the cartoon PLA above, dots show programmed (enabled) connections. The right figure shows how we use this equivalence along with device configuration to avoid defective wires.

- length or junction failures—with probability P_j there is a break or short in the NT or NW at the junction.

For an N long tube to yield, it must contain no failures

$$P_{\text{tube}} = (1 - P_c)^2 (1 - P_j)^N. \tag{4}$$

Current experiences suggests that contact faults are likely to occur in the single-digit percentages and breaks and shorts are quite unlikely. For example, [8] reports over 95% yield of junctions with controllable electronic characteristics ($P_c < 0.05$); [24] reports reliable growth of SiNWs, which are over 9 μm long (i.e., no breaks over a distance equivalent to 900 10-nm junction lengths). These reported data represent yield levels obtainable in research labs and we expect mature manufacturing to achieve higher levels of yield. Nonetheless, no one has experience building large arrays to date and we expect to refine our yield models as the technology develops.

We must further account for faults in the address decoders. If we use a 2-hot code where each line is driven by asserting two of the address lines, then the number of lines addressed by N_a address lines is

$$N_{\text{la}}(N_a) = \frac{N_a \times (N_a - 1)}{2}. \tag{5}$$

We can now approach the yield of the array in the following two parts:

1) look at the yield of the address decoder(s);
2) based on the yielded address decoder, look at the yield of the addressed tubes.

The expected number of addressable wires is then:

$$E(N_{\text{addressable\_rows}}) = \sum_{m=0}^{m=N_a} N_{\text{la}}(N_{\text{a-m}}) \times C(N_{\text{a, m}}) \times P_{\text{tube}}^{N_a - m} (1 - P_{\text{tube}})^m \tag{6}$$

where $C(N, M)$ is the number of combinations of N things taken M at a time. By symmetry, we will expect a similar number of addressable rows and columns. The net row yield is then

$$E(Y_{\text{net\_row}}) = \left(\frac{E(N_{\text{addressable\_rows}})}{N_{\text{rows}}} \right) \times E(Y_{\text{row}}). \tag{7}$$

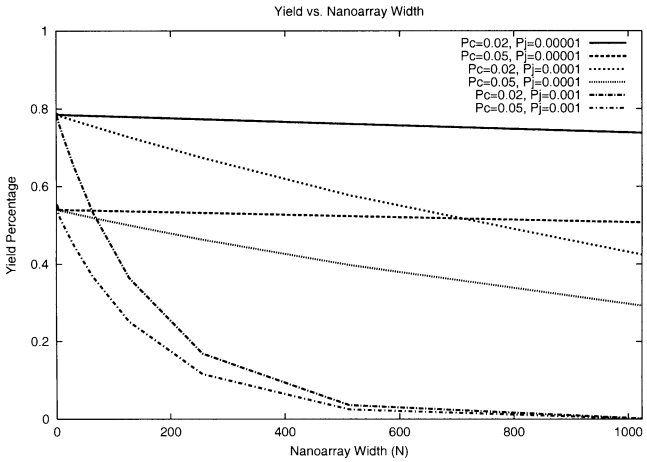


Fig. 16. Crosspoint yield rates based on subarray size.

By symmetry, we expect a similar column yield. Together, this gives us a net yield

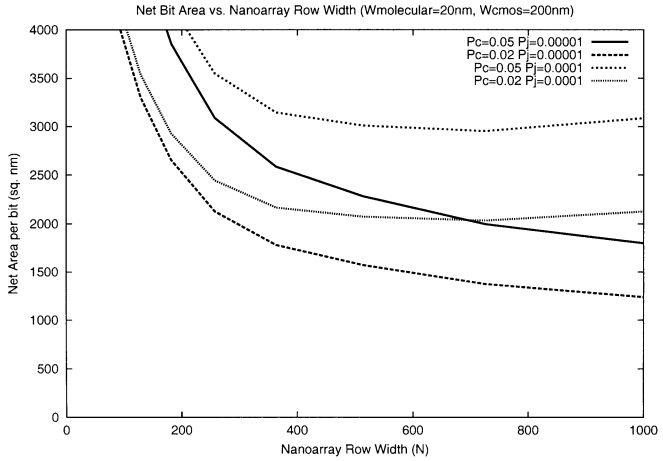
$$E(Y_{\text{net\_bits}}) = E(Y_{\text{net\_row}}) \times E(Y_{\text{net\_column}}). \tag{8}$$

From this, we can compute the expected yield rate for bits in the core and show sample trends in Fig. 16. Combining yield with our area calculation, we can compute the net area per bit after considering both yield rates and support overheads (see Fig. 17). This data suggests modest arrays with 500–1000 tubes per side will offer the highest net density.

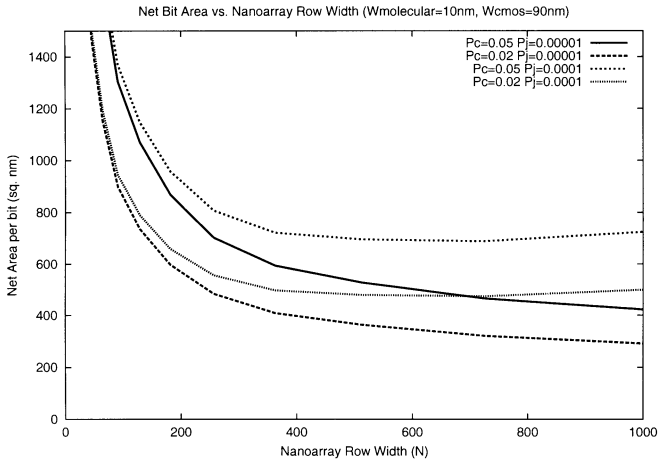
The net power density in a full NOR–NOR architecture is roughly:

$$P_{\text{density}} = \frac{P_{\text{NOR}}}{N \times A_{\text{netbit}}}. \tag{9}$$

That is, the extent of each NOR is the length of its output wire, so it burns P_{NOR} in an area equal to one bit pitch times the length of the NOR wire. Each NOR wire is roughly $2N$ nanoscale pitches long since it spans two arrays. There are two wire layers in each NOR array. The two factors of two cancel each other giving us Equation (9). Using $P_{\text{NOR}} \approx 0.1\,\mu$W from Section II, and 500×500 arrays ($N = 500$), we get 40 W/cm^2 when $A_{\text{netbit}} =$ 500 nm^2 ($W_{\text{molecular}} = 10$ nm) and 10 W/cm^2 when $A_{\text{netbit}} =$



(a)



(b)

Fig. 17. Net bit area.

2000 nm^2 ($W_{\text{molecular}} = 20$ nm) (see Fig. 17). As noted earlier, more complicated circuit architectures may allow us to further reduce static power requirements.

V. Summary

We have shown a complete architectural style built entirely out of large arrays of crossed NWs and/or NTs. The key feature of this organization is that it provides a sufficient set of capabilities for performing logic, restoration, routing, and bootstrap programming using only large, crossed wire arrays. Strategic breaks in conductors exist between arrays at regular intervals and are essential for achieving complete and efficient logic operation. The breaks are large compared to the nanoscale features and can be generated lithographically—either by patterning blocks to NT/NW growth or by cutting grown structures.

Nanoscale FET devices allow us to define a restoring logic discipline, making it possible to compute through an arbitrary number of logic stages. Collections of NOR gates are universal, so this substrate is sufficient to perform any computation. Gross topology, doping, and device selection will allow us to include or mix-and-match other kinds of logical arrays to improve architectural efficiency.

VI. Caveats and Open Questions

The architecture sketched here is an existence proof, demonstrating a complete, plausible scheme for achieving molecular-scale logic from these building blocks. There are numerous components of the architecture that certainly merit further optimization (e.g., energy reduction, decoder fabrication, array customization, self programming, yield enhancements). We are attacking many of these issues as part of our ongoing work.

At this point, even the detailed behavior of the basic wires and devices are highly experimental. Assembly procedures and reliability are active areas of current research. Many of the components here may not be feasible or operational as currently envisioned. Nonetheless, there are many technological alternatives available for each of the key components, and it seems likely that we can find at least one viable path through the emerging set of technologies. Simultaneous development of architecture with technology allows us to see what the emerging technology can and cannot do and push back on the technology development to engineer the essential features, which will make the technology viable for implementing computations.

Acknowledgment

The author would like to thank C. Lieber, X. Duan, D. Wang, and Z. Zhong for their support in this work. Architecture work at this early stage is only feasible and meaningful in close cooperation with scientists working on device properties and fabrication.

References

[1] C. Dekker, "Carbon nanotubes as molecular quantum wires," *Phys. Today*, pp. 22–28, May 1999.
[2] Y. Huang, X. Duan, Q. Wei, and C. M. Lieber, "Directed assemble of one-dimensional nanostructures into functional networks," *Science*, vol. 291, pp. 630–633, Jan. 2001.
[3] 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.
[4] A. M. Morales and C. M. Lieber, "A laser ablation method for synthesis of crystalline semiconductor nanowires," *Science*, vol. 279, pp. 208–211, 1998.
[5] Y. Chen, D. A. A. Ohlberg, G. Medeiros-Ribeiro, Y. A. Chang, and R. S. Williams, "Self-assembled growth of epitaxial erbium disilicide nanowires on silicon (001)," *Appl. Phys. Lett.*, vol. 76, no. 26, pp. 4004–4006, 2000.
[6] 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, June 8, 2000.
[7] 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.
[8] 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.
[9] S. J. Trans, A. R. M. Verschueren, and C. Dekker, "Room-temperature transistor based on a single carbon nanotube," *Nature*, vol. 393, pp. 49–51, May 7, 1998.
[10] V. Derycke, R. Martel, J. Appenzeller, and Ph. Avouris, "Carbon nanotube inter- and intramolecular logic gates," *Nano Lett.*, vol. 1, no. 9, pp. 453–456, 2001.
[11] S. J. Wind, J. Appenzeller, R. Martel, V. Deycke, and Ph. Avouris, "Vertical scaling a of carbon nanotube field-effect transistors using top gate electrodes," *Appl. Phys. Lett.*, vol. 80, no. 20, pp. 3817–3819, 2002.

[12] 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.
[13] J. Rose, R. Francis, D. Lewis, and P. Chow, "Architecture of field-programmable gate arrays: The effect of logic block functionality on area efficiency," *IEEE J. Solid-State Circuits*, vol. 25, no. 5, pp. 1217–1225, Oct. 1990.
[14] J. Kouloheris and A. El Gamal, "Pla-based FPGA area versus cell granularity," in *Proc. IEEE Custom Integrated Circuits Conf.*, Boston, MA, May 1992, pp. 4.3.1–4.3.4.
[15] A. DeHon, "Reconfigurable architectures for general-purpose computing," MIT Artif. Intell. Lab., Cambridge, MA, AI Tech. Rep. 1586, Oct. 1996.
[16] J. R. Heath, P. J. Kuekes, G. S. Snider, and R. S. Williams, "A defect-tolerant computer architecture: Opportunities for nanotechnology," *Science*, vol. 280, pp. 1716–1721, June 12, 1998.
[17] S. C. Goldstein and M. Budiu, "Nanofabrics: Spatial computing using molecular electronics," in *Proc. 28th Annu. Int. Symp. Computer Architecture*, Gotenbörg, Sweden, June 2001, pp. 178–189.
[18] C. M. Lieber and X. Duan, "Nanofet threshold voltages," unpublished, Dec. 2001.
[19] S. Y. Chou, P. R. Krauss, W. Zhang, L. Guo, and L. Zhuang, "Sub-10 nm imprint lithography and applications," *J. Vacuum Sci. Technol. B*, vol. 15, no. 6, pp. 2897–2904, Nov./Dec. 1997.
[20] S. Williams and P. Kuekes, "Demultiplexer for a molecular wire crossbar network," U.S. Patent 6 256 767, July 3, 2001.
[21] V. J. Coli, "Introduction to programmable array logic," *BYTE*, pp. 207–219, Jan. 1987.
[22] Monolithic Memories, Inc., *PAL Handbook*, 3rd ed. Santa Clara, CA: Monolithic Memories, Incorporated, 1983.
[23] (2001) International technology roadmap for semiconductors. [Online]. Available: http://public.itrs.net/Files/2001ITRS/
[24] 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.

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 (MIT), Cambridge, MA, in 1990, 1993, and 1996 respectively.

From 1996 to 1999, he co-ran the BRASS Group in the Computer Science Department, University of California at Berkeley. Since 1999, he has been an Assistant Professor of Computer Science at the California Institute of Technology, Pasadena. He is broadly interested in how to physically implement 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., FPGAs) and interconnect design and optimization.