Appearing in *First Workshop on Non-Silicon Computation* (NSC-1), Feb. 3, 2002

Array-Based Architecture for Molecular Electronics

André DeHon
Department of Computer Science, 256-80
California Institute of Technology
Pasadena, CA 91125
andre@acm.org

Abstract

Advances in our basic scientific understanding at the molecular and atomic level place us on the verge engineering designer structures at the molecular scale. This introduces exciting opportunities 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 motivate different computing architectures than we found efficient and appropriate in conventional VLSI. We sketch a basic architecture for molecular electronics based on carbon nanotubes and silicon nanowires which can provide universal logic functionality with all logic and signal restoration operating at the molecular scale. 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.

1 Overview

We show how to organize the carbon nanotube, silicon nanowires, and molecular scale devices which are now being developed into an operational computing system. The molecular-scale wires can be arranged into interconnected, crossed arrays with non-volatile switching devices at their crosspoints; these crossed arrays can function as programmable-logic arrays and programmable interconnect (See Figure 1). Using nanoscale FET devices, we provide both signal restoration and programming support for the non-volatile switches. The result is a programmable logic device which can be configured to compute any logical function and which operates entirely at the nanoscale. Bottom-up synthesis is prone to high defect rates compared to conventional integrated electronics, making defect-tolerance an essential requirement for this architecture.

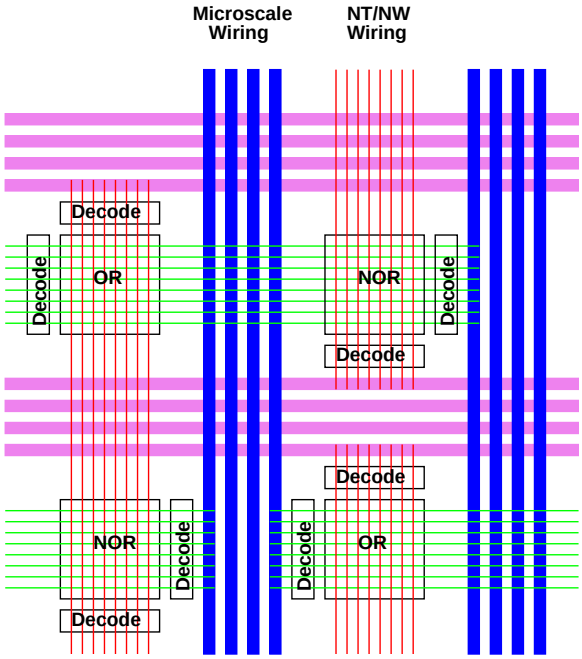


Figure 1: Overall Assembly of Functional Nanoarrays

2 Technology

Wires We can synthesize carbon nanotubes (NT) which are nanometers in diameter and microns long [6]. 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 [9]. Ultimately, these carbon nanotubes can be a single nanometer wide and spaced several nanometers apart. To date, we cannot control the detailed electrical properties (conducting vs. semiconducting) for these nanotubes, but the conduction of even the worst conductors is often adequate for many uses.

At the same time, we are developing technologies to grow silicon nanowires [4] [12] which are also only nanometers in width and can be grown or assembled into sets of long parallel wires [1]. We can control the electrical properties of these silicon nanowires (SiNW) with dopants,

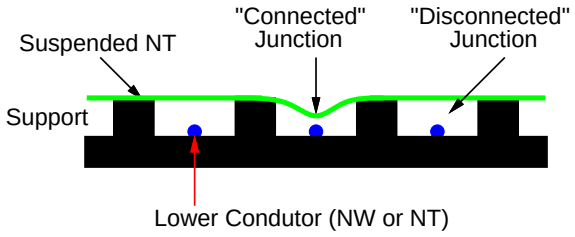


Figure 2: Suspended Nanotube Switched Connection

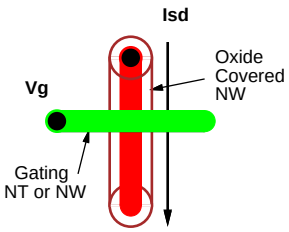


Figure 3: NT/NW FET Device

yielding semiconducting wires [3]. Nanowires can be assembled along with nanotubes when their respective properties complement each another.

Devices Lieber and his students have shown switched devices using suspended nanotubes [14] (See Figure 2). The nanotube-nanotube 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 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 bistable states, effectively setting or resetting the programming of the connection. Silicon nanowires can be substituted for the lower wire, and these junctions can be rectifying such that the connected state exhibits PN-diode rectification behavior.

Doped silicon nanowires exhibit Field-Effect Transistor (FET) behavior [8]. That is, oxide can be grown over the silicon nanowire to prevent direct electrical contact of a crossed conductor (See Figure 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. Carbon nanotubes also demonstrate FET behavior [15] [7].

Heath, Stoddard, and their groups at UCLA and HP have demonstrated molecules which appear to exhibit orders of magnitude different resistance in different states [2]. They sketch how to assemble an aligned, single layer of these molecules between nanoscale conductors such as silicon nanowires or carbon nanotubes.

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 PLA or FPGA, where the area to hold the memory cell and switch are much larger than a primitive wire crossing (*e.g.* $2500\lambda^2$ for a small, pass-gate switch with memory versus $25–50\lambda^2$ for a wire crossing). So, even if we achieve 35nm silicon feature sizes (which might imply 70-90nm wire pitches), the density difference between 20nm spaced nanotubes or silicon nanowires and the 35nm silicon will be greater than the roughly $(80\text{nm}/20\text{nm})^2$ 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 cost in density. This is particularly beneficial in achieving the necessary defect tolerance.

Near Term Based on the current successes and understanding, in the near term (next 5 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.

3 Architectural Strategy

Armed with these building blocks and properties, we consider an architecture based on a collection of interconnected arrays (See Figure 1). The crossed arrays can act as memory cores, Programmable-Logic Array (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] [10] [5]), but a collection of interconnected arrays allows us to both exploit logical

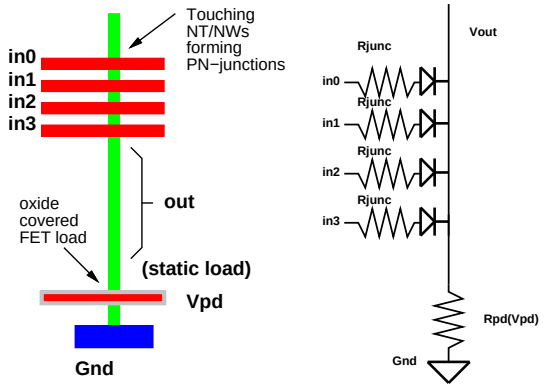


Figure 4: Diode OR Arrangement

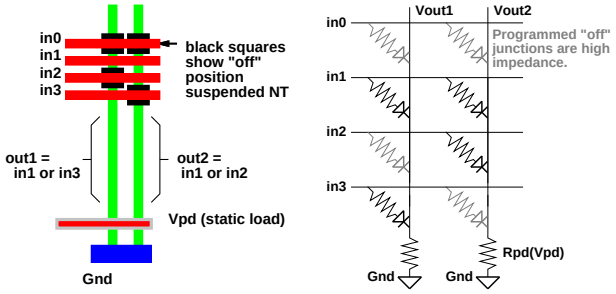


Figure 5: Programmable Diode OR Array

structure and isolate faults.

Key issues in the design include:

1. Achieving gain for signal restoration (Section 4)
2. Interfacing between our conventional, microscale features and the nanoscale circuits (Section 5)
3. Bootstrapping array personalization (Section 5)
4. Configuring functional logic around defective devices (Section 7)

4 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 Figure 4). Using the suspended switching, we can assemble configurable OR planes, with connected wires acting as low-resistance PN-junctions, and distant wires isolated by high resistance (See Figure 5). We can use these passive devices in our switching, 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.

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,

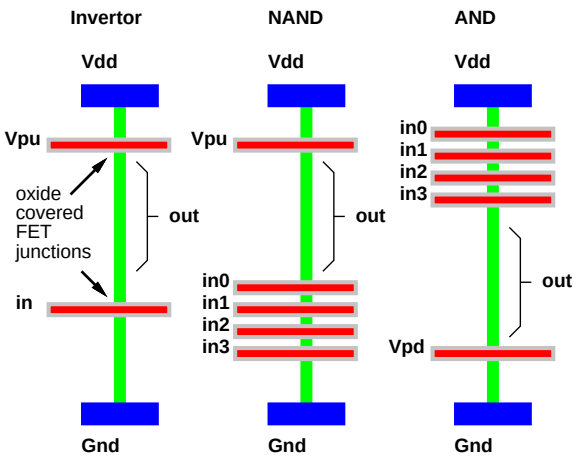


Figure 6: FET Logic Arrangements

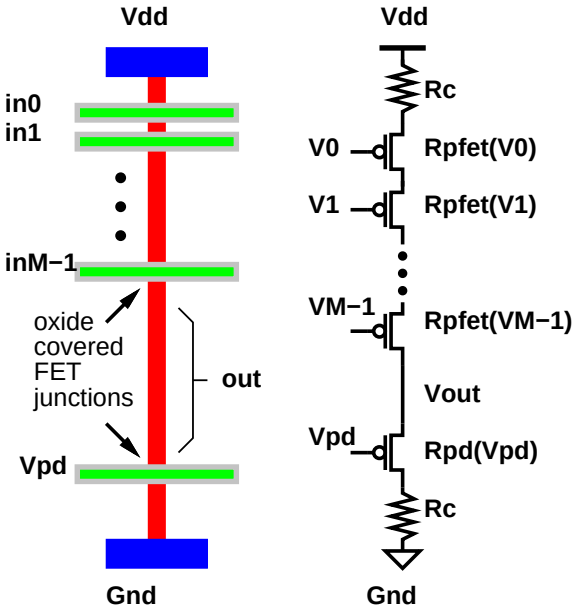
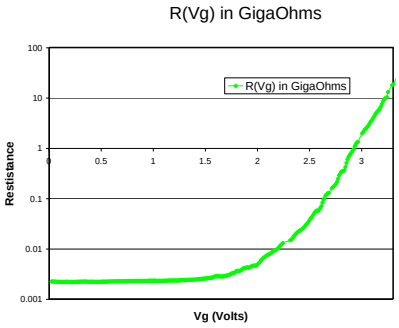


Figure 7: PFET NOR Circuit

AND, NOR, or OR logic (See Figures 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 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.

Figure 7 shows the logical arrangement and corresponding circuit model for a PFET NOR. The depletion-mode



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

Figure 8: PFET Resistance versus V_g

PFETs conduct with low resistance in their default state and increase their resistance as the gate voltage is increased (See Figure 8). We can characterize the output voltage as:

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

M is the number of inputs to the NOR gate (as shown in Figure 7). Current experimental characterization suggests that the contact resistance (R_c) is on the order of 1MΩ; this 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_{pfet} (small V_g) $<<$ $R_c = 1$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_{oh}. As noted, the ON-resistance of the PFETs is low, so as long as we can make $R_{pd} >> R_c$, the pull-up resistance is small compared to the pull-down resistance, and V_{out} becomes close to V_{dd}. Consequently, we want to set V_{pd} such that $R_{pd} = R_{pfet}(V_{pd}) \approx 9R_c$. 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_{ol} voltage. That means: $R_{pfet}(V_{ih}) >> (R_{pd}+R_c) = 10R_c$. The OFF-resistance of the PFETs is in the 100's of GΩs, so this is easily obtainable as well. A sample set of operating voltages derived from the data in Figure 8 is shown in Table 1.

The operating point here is set by the the placement of the high gain region and hence 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 1V supply [11].

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

Table 1: Operating Voltages for PFET NOR assuming R-V characteristics shown in Figure 8

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 ∼10MΩ. The capacitance of a 1μm nanotube will be $C_{wire} \approx 3\times10^{-16}$F (calculation based on data in [3]), and SiNW capacitance is comparable. The RC-delay for pull-down is thus $T_{pd} \approx 1.0\text{M}\Omega \times 3 \times 10^{-16}\text{F} \approx 300$ ps. 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 $R_{pd} = 2R_c + R_{pd}$, or again, roughly 10MΩ. Static power is $P_{nor} = \frac{(V_{dd})^2}{R_{pd}}$. At $V_{dd} = 3.3$V, $P_{nor} \approx 1\mu$W. At 1V, $P_{nor} \approx 0.1\mu$W. The topology for this static-load logic is particularly simple and regular making it compatible with bottom-up fabrication techinques; in future work, we will explore alternatives to reduce or eliminate static power while retaining as much of this simplicity as possible.

5 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.

As noted above (Section 2), 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. So, our first

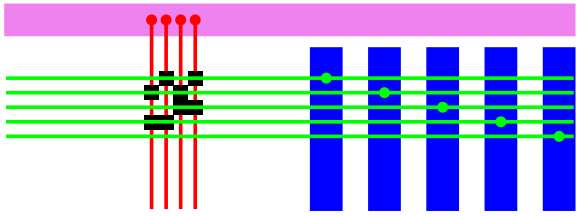


Figure 9: Programmed Decoder

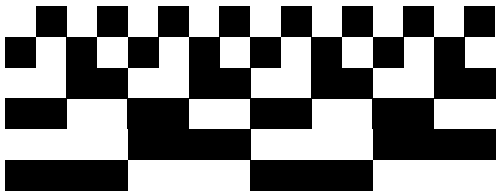


Figure 10: Decoder Imprint Pattern

challenge is to get to the point where we can apply a voltage on each horizontal or vertical NW/NT 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 nanowire array. The decode has N wires which connect to the nanowire array and $2\log(N)+1$ nanowires which connect to an orthogonal set of microscale wires through nanovias (See Figure 9). The $2\log(N)$ address wires come from the fact that we drive both the true and complement of every address bit into the decoder so that we can simply AND together the enables to select each wire (best shown in Figure 9); the extra line is used to the disable the decoder connection.

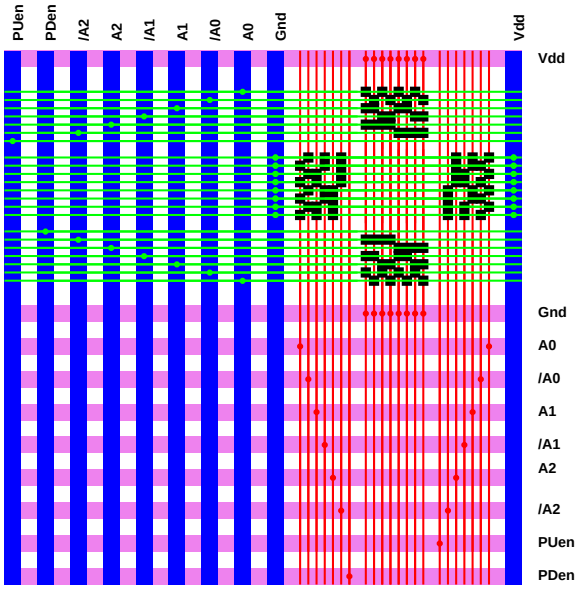
However, we cannot program the decoder at the nano-micro scale interface as we intend to program the core. The address lines which are connected directly to the microscale wires can be driven to a voltage by conventional electronics. However, we have no way to drive the nanoscale wires which drive into 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 Figure 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 nanowires are far enough apart that they do not control each other (See Figure 9). The patterning does not need to be perfect here. What is important is that we have a code which 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. Williams and Kuekes have proposed stochastic self-assembly techniques as an alternate way to build this kind of decoder [16].

These decoders are placed on either side of a nanoscale array in both dimension. Figure 11 shows a simple, but non-operational, arrangement of this bracketing. Using these decoders, it is now possible to drive any single horizontal or vertical tube to a high or low voltage, and leave the other tubes floating. We can drive a tube high by driving all of the PFET nanowire crossings in the decoder low—that would be the pull-up enable and all the address lines necessary to select this tube; driven this way, we have a low-impedance path from this tube to the high voltage supply. Assuming we drive the pull-down enable with a high voltage so that it is in high-impedance mode, and we drive the true and complement address lines with appropriately opposing voltages, 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 the pull-up enable high and the pull-down enable low.

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 enables high, we isolate the array completely from the programming FETs. For PN-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. Figure 12 show 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 Figure 7). By driving all of the PFETs low (*i.e.* driving all the address lines low, including both the true and complement lies), the PFETs will act as wires. If we further drive the pull-up enable low and the pull-down enable with V_{pd}, then this becomes the NOR circuit we identified earlier (Figure 7) with the pull-down enable FET serving as R_{pd}.

We may be able to personalize these FET arrays by using the same suspended tube scheme used for the PN-junctions. We use the FET decoders to move the crossed wires into either a close contact position or separated position (See Figure 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 PN-junction behavior. In the far case, the wires should be sufficiently separated that we



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 logarithmically in array width; so for the larger nanoarray sizes we consider typical, the microscale wiring becomes a thin periphery around a large nanoscale array core.

Figure 11: Array Bracketed with Decoders

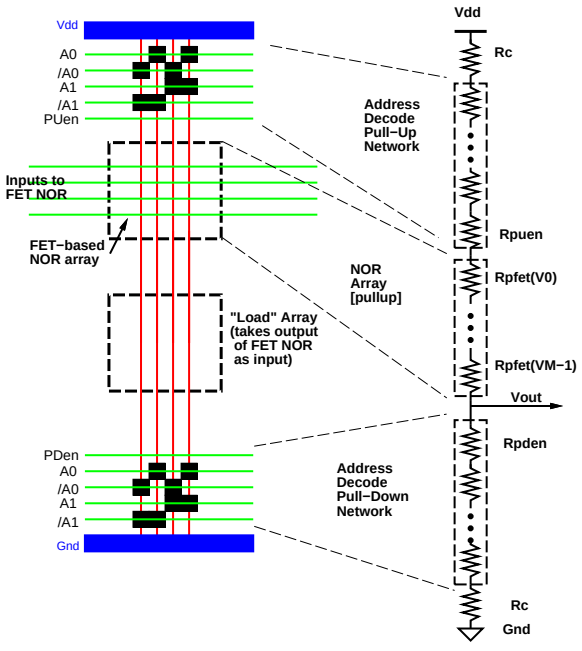


Figure 12: Operating FET NOR Array Bracketed by Decoders

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.

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_{dd} and V_{gnd}) during programming and operation. Further, note that this FET decoder scheme should work with any devices with non-volatile junction state switched using voltages, including, perhaps the UCLA-HP molecular switches [2].

Note that the "output" of each NOR circuit appears on the nanowire 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 Figure 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.

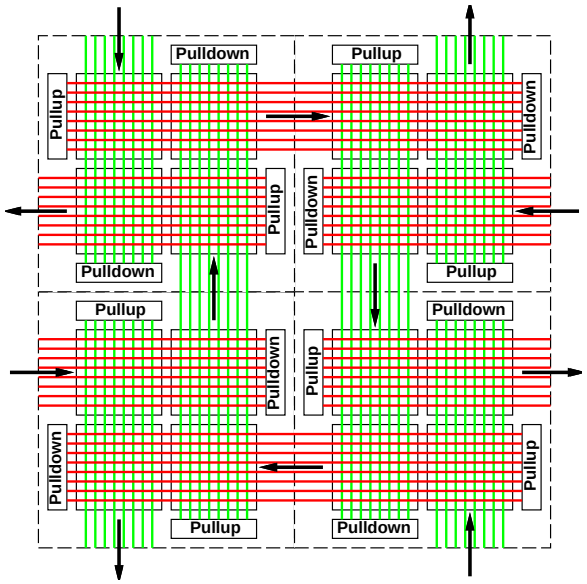
6 Organization

We organize the array cells detailed in the previous section into large arrays. Each array has wires overlapping with adjacent arrays for inter-array communication (See Figure 1 and 12). In simplest form, all arrays 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. Figure 13 shows a NOR-only macro tile which can be abutted horizontally and vertically to allow arbitrary Manhattan routing within the master array. In more complex configuration, we can alternate diode-based arrays with the FET NOR arrays. Notably, if only the diode-arrays are programmable, we can use imprinting to pattern fixed-connectivity NOR stages, allowing the programmable diode-OR and fixed NOR pair to provide both logic and signal restoration, realizing a PAL-like logic structure.

7 Defect Tolerance

When assembled into arrays, some of the nanoscale wires will have poor or non-existent contacts, and individual switches may be non-functional. The 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



In this more realistic topology, we build a logical NOR plane out of a 2×2 arrangement of crossed nano-arrays (Macro-scale wires, as shown in Figure 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.

Figure 13: NOR-only Macrotile for Routing

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 Figure 14). 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.

The simple decoders shown in Figures 9 and 10 have the bad property that a single faulty address line will make it impossible to address half of the lines in the array. Consequently, in practice, we use a sparser encoding for the input decoder in order to guarantee that a faulty address line will render only a small fraction of the array inaccessible.

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.

8 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.

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.

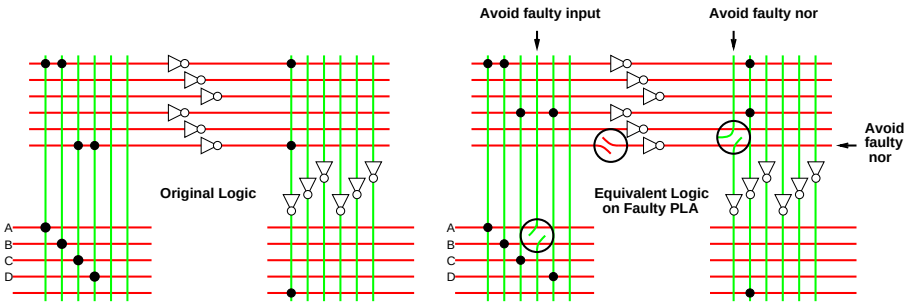
9 Caveats and Open Questions

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 which certainly merit further optimization (*e.g.* energy reduction, 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.

Acknowledgements

This research was funded by the DARPA Moletronics program under grant ONR N00014-01-0651.



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.

Figure 14: Sparing in Crossed-Wire Planes to Avoid Faults

References

[1] Yong Chen, Douglas A. A. Ohlberg, Gilberto Medeiros-Ribeiro, Y. Austin Chang, and R. Stanley Williams. Self-assembled growth of epitaxial erbium disilicide nanowires on silicon (001). *Applied Physics Letters*, 76(26):4004–4006, 2000.

[2] 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*, 285:391–394, 1999.

[3] Yi Cui, Xiangfeng Duan, Jiangtao Hu, and Charles M. Lieber. Doping and electrical transport in silicon nanowires. *Journal of Physical Chemistry B*, 104(22):5213–5216, June 8 2000.

[4] Yi Cui, Lincoln J. Lauhon, Mark S. Gudiksen, Jianfang Wang, and Charles M. Lieber. Diameter-controlled synthesis of single crystal silicon nanowires. *Applied Physics Letters*, 78(15):2214–2216, 2001.

[5] André DeHon. Reconfigurable Architectures for General-Purpose Computing. AI Technical Report 1586, MIT Artificial Intelligence Laboratory, 545 Technology Sq., Cambridge, MA 02139, October 1996.

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

[7] V. Derycke, R. Martel, J. Appenzeller, and Ph. Avouris. Carbon nanotube inter- and intramolecular logic gates. *Nano Letters*, 1(9):453–456, 2001.

[8] Yu Huang, Xiangfeng Duan, Yi Cui, Lincoln Lauhon, Kevin Kim, and Charles M. Lieber. Logic gates and computation from assembled nanowire building blocks. *Science*, 294:1313–1317, 2001.

[9] Yu Huang, Xiangfeng Duan, Qingqiao Wei, and Charles M. Lieber. Directed assembley of one-dimensional nanostructures into functional networks. *Science*, 291:630–633, January 26 2001.

[10] Jack Kouloheris and Abbas El Gamal. Pla-based fpga area versus cell granularity. In *Proceedings of the Custom Integrated Circuits Conference*, pages 4.3.1–4. IEEE, May 1992.

[11] Charles M. Lieber and Xiangfeng Duan. *Personal Communications*, December 2001.

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

[13] Jonathan Rose, Robert Francis, David Lewis, and Paul Chow. Architecture of field-programmable gate arrays: The effect of logic block functionality on area efficiency. *IEEE Journal of Solid-State Circuits*, 25(5):1217–1225, October 1990.

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

[15] Sander J. Trans, Alwin R.M. Verschueren, and Cees Dekker. Room-temperature transistor baed on a single carbon nanotube. *Nature*, 393:49–51, May 7 1998.

[16] Stanley Williams and Philip Kuekes. Demultiplexer for a molecular wire crossbar network. United States Patent Number: 6,256,767, July 3 2001.

Web links for this document: <http://www.cs.caltech.edu/research/ic/abstracts/molecular_nsc2002.html>