Nanowire-Based Programmable Architectures

ANDRÉ DEHON
California Institute of Technology

Chemists can now construct wires which are just a few atoms in diameter; these wires can be selectively field-effect gated, and wire crossings can act as diodes with programmable resistance. These new capabilities present both opportunities and challenges for constructing nanoscale computing systems. The tiny feature sizes offer a path to economically scale down to atomic dimensions. However, the associated bottom-up synthesis techniques only produce highly regular structures and come with high defect rates and minimal control during assembly. To exploit these technologies, we develop nanowire-based architectures which can bridge between lithographic and atomic-scale feature sizes and tolerate defective and stochastic assembly of regular arrays to deliver high density universal computing devices. Using 10nm pitch nanowires, these nanowire-based programmable architectures offer one to two orders of magnitude greater mapped-logic density than defect-free lithographic FPGAs at 22nm.

Categories and Subject Descriptors: B.6.1 [**Logic Design**]: Design Styles—*Logic arrays*; B.7.1 [**Integrated Circuits**]: Types and Design Styles—*Advanced technologies*

General Terms: Design

Additional Key Words and Phrases: Defect tolerance, Manhattan mesh, nanowires, programmable logic arrays, programmable interconnect, sublithographic architecture, stochastic construction

1. INTRODUCTION

Chemists are now demonstrating bottom-up synthesis techniques which can construct atomic-scale switches, field-effect devices, and wires (Section 2 and Appendix A). While these are key components of a computing system, we must also understand if they can be assembled and organized into useful computing devices. That is, How do we build arbitrary logic from nanowire building blocks and atomic-scale switches?

This research was supported by the Defense Advanced Research Projects Agency under ONR contracts N00014-01-0651 and N00014-04-1-0591. This material is based on work supported by the Department of the Navy, Office of Naval Research. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Office of Naval Research.
Author's Address: Department of Computer Science, 256-80, California Institute of Technology, Pasadena, CA 91125; email: andre@cs.caltech.edu.

—Do we have an adequate set of capabilities to build logic?
—How do we cope with the regularity demanded by bottom-up assembly?
—How do we accommodate the high defect rates and statistical assembly which accompany bottom-up assembly techniques?
—How do we organize and interconnect these atomic-scale building blocks?
—How do we address nanowires from the lithographic scale for testing, configuration, and IO?
—How do we get logic restoration and inversion?
—What net benefit do these building blocks offer us?

The regular synthesis techniques can be used to assemble tight-pitch, parallel nanowires (Section 2.2); this immediately suggests we build programmable crossbar arrays (Section 4.1) as the key building blocks in our architectures. These crossbar arrays can be used as memory cores (Section 5), wired-OR logic arrays (Section 6.1), and programmable interconnect (Section 6.3)—memory, logic, and interconnect—all the key components needed for computation.

We must limit the length of nanowires for yield, performance, and logical efficiency (Section 3.2). Consequently, we organize the nanowires into a collection of modest-sized, interconnected crossbar arrays (Section 6.3). A reliable, lithographic-scale support structure provides power, clocking, control, and bootstrap testing for the nanowire crossbar arrays. Each nanowire is coded so it can be uniquely addressed from the lithographic support wires (Section 4.2). With the ability to address individual nanowires, individual crosspoints can be programmed (Section 9) to personalize the logic function and routing of each array and to avoid defective nanowires and switches (Section 8).

Since we cannot, deterministically, place specific nanowires in precise locations using these bottom-up techniques, we exploit stochastic assembly to get unique addressability (Section 4.2). We further exploit stochastic assembly to provide signal restoration and inversion at the nanoscale (Section 4.3). Remarkably, starting from regular arrays of programmable diode switches and stochastic assembly of non-programmable field-effect controlled nanowires, we can build fully programmable architectures with all logic and restoration occurring at the nanoscale. Section 7 summarizes the complete assembly approach for constructing nanowire-based computing systems.

The resulting architectures (Section 6) provide a high-level view similar to Island-Style FPGAs, and we can adapt conventional logic mapping tools to compile logic to these arrays (Section 10). Owing to the high defect rates likely to be associated with *any* atomic-scale manufacturing technology, all viable architectures at this scale are likely to be post-fabrication configurable (Section 8). That is, while nanowire architectures can be customized for various application domains by tuning their gross architecture (e.g., ratio of logic and memory), there will be no separate notion of custom atomic-scale logic.

Section 11 models the area/delay/energy costs of these structures. Even after accounting for the required, regular structure, high defect rates, stochastic assembly, and the lithographic support structure, we see a net benefit from being able to build with nanowires which are just a few atoms in diameter and

programmable crosspoints that fit in the space of a nanowire junction (Section 12). Mapping conventional FPGA benchmarks from the Toronto20 benchmark set [Betz and Rose 1999b], the designs presented here should achieve one to two orders of magnitude greater density than FPGAs in 22nm CMOS lithography, even if the 22nm lithography delivers defect-free components (Section 12).

The design approach taken here represents a significant shift in design styles compared to conventional lithographic fabrication. In the past, we have relied on virtually perfect and deterministic construction and complete control of features down to a minimum technology feature size. Here, we can exploit very small feature sizes but do not have complete control of device location in all dimensions. Instead, we rely on statistical properties of large ensembles of wires and devices to achieve the desired, aggregate component features. Further, post-fabrication configuration becomes essential to device yield and personalization.

This article describes a complete assembly of a set of complementary technologies and architectural building blocks. The particular ensemble presented is one of several architectural proposals which have a similar flavor (Section 13) based on these kinds of technologies (Appendix A) and building blocks.

2. TECHNOLOGY

2.1 Nanowires

Atomic-scale nanowires can be engineered to have a variety of conduction properties from insulating to semiconducting to metallic. The composition of a nanowire can be varied along its axis and along its radius, offering us powerful heterostructures to provide both controllable devices and interconnect integrated into a single structure.

2.1.1 *Growth.* Semiconducting nanowires (NWs) can be grown to controlled dimensions on the nanometer scale using seed catalysts (e.g., gold balls) to define their diameter. The catalyst constrains the growth of the semiconductor to only a small region so that new atomic layers deposit only in the region of the catalyst (see Figure 1). The catalyst forms the head of the growing NW. The diameter of the grown NWs is closely correlated to the diameter of the seed catalysts [Cui et al. 2001]. NWs with diameters down to 3nm have been demonstrated [Morales and Lieber 1998; Cui et al. 2001]. Seed catalysts with controlled diameter can be produced by self-limiting chemical processes (e.g., Tan et al. [2003]).

2.1.2 *Field-Effect Control.* By controlling the mix of elements in the environment during growth, semiconducting NWs can be doped to control their electrical properties [Cui et al. 2000]. Heavily doped NWs are conducting. Conduction through lightly doped NWs can be controlled via an electrical field like Field-Effect Transistors (FETs) [Huang et al. 2001]. For the depletion-mode P-type devices which have been heavily reported, a low voltage (or no applied voltage) will allow good conduction, whereas a high applied voltage will evacuate carriers from the doped semiconductor preventing conduction along the NW

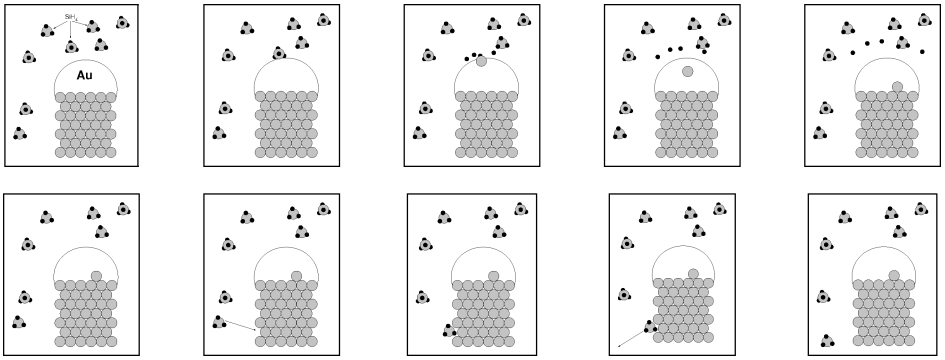


Fig. 1. SiNW growth cartoon. Top row shows how a gold (Au) catalyst allows the SiH_4 molecules to shed their Hydrogen termination so the Silicon can assemble onto the top of the growing NW. Bottom row shows how the Hydrogen termination on the SiH_4 molecule prevents it from assembling onto the edge of the growing NW where the catalyst is not present.

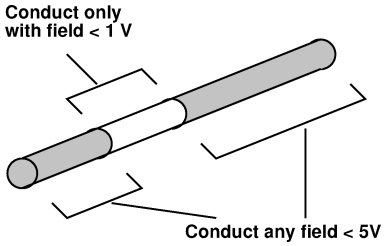


Fig. 2. Axial doping profile places selective gateable regions in a NW.

length. Off resistances (R_{offfet}) can be over 10GΩs and on resistances (R_{onfet}) under 0.1MΩ; off/on resistance ratios are at least 10^4 [Cui et al. 2003]. NW field-effect gating has sufficient gain to build restoring gates [DeHon 2003]. The threshold voltage for the NW can be controlled by material properties (e.g., doping or composition) and geometry factors.

2.1.3 *Axial Profile.* The doping profile or material composition along the length of a NW can be controlled by varying the material concentrations in the growth environment over time [Gudiksen et al. 2002]. This allows us to construct wires which are gateable in some regions but not gateable in others (see Figure 2). Our control over growth rate allows us to control the physical dimensions of these features down to almost atomic precision without lithographic processing [Gudiksen et al. 2002; Wu et al. 2002; Björk et al. 2002].

2.1.4 *Radial Profile.* The doping profile or material composition can also be controlled along the radius of these NWs [Lauhon et al. 2002; Law et al. 2004] (see Figure 3). After completing axial growth, environmental conditions are changed to allow atomic layers to grow over the entire surface of the NW. This allows us to sheath NWs in insulators (e.g., SiO_2) to control spacing between conductors [Whang et al. 2003a] and between gated wires and control wires.

2.1.5 *Silicide.* After a NW has been grown, it can be converted into a metal silicide. For example, by coating select regions of the NW with nickle

Fig. 3. Radial doping profile.

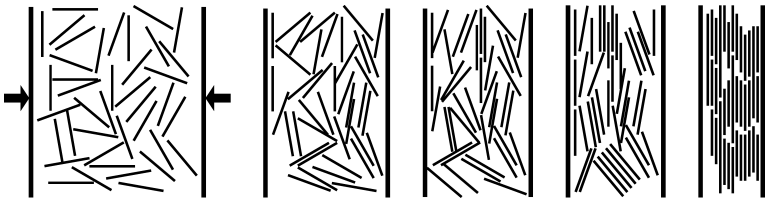


Fig. 4. Langmuir Blodgett alignment of NWs.

and annealing, we can form a Nickle-Silicide (NiSi) NW [Wu et al. 2004]. The NiSi resistivity is $\rho_{NiSi} = 10^{-5}\Omega$-cm which is much lower than the resistivity of highly-doped bulk silicon ($\rho_{Si} = 10^{-3}\Omega$-cm). By treating the ends of NWs which contact to lithographic scale signals in this way, the contact resistance (R_c) can be in the 10KΩ range.

2.2 Assembly

Langmuir-Blodgett (LB) flow techniques can be used to align a set of NWs into a single orientation, close pack them, and transfer them onto a surface [Huang et al. 2001; Whang et al. 2003a] (see Figure 4). The resulting wires are all parallel with nematic alignment. By using wires with an oxide sheath around the conducting core, the wires can be packed tightly. The oxide sheath defines the spacing between conductors and can, optionally, be etched away after assembly. The LB step can be rotated and repeated so that we get multiple layers of NWs [Huang et al. 2001; Whang et al. 2003a] such as crossed NWs for building a crossbar array or memory core (Section 4.1).

2.3 Crosspoints

Many technologies have been demonstrated for nonvolatile, switched crosspoints. Common features include:

(1) resistance which changes significantly between on and off states;
(2) the ability to be made rectifying;
(3) the ability to turn the device on or off by applying a voltage differential across the junction;
(4) the ability to be placed within the area of a crossed NW junction.

Chen et al. [2003] demonstrate a nanoscale Ti/Pt-[2]rotaxane-Ti/Pt sandwich which exhibits hysteresis and nonvolatile state storage showing an order of magnitude resistance difference between on and off states for several write cycles. With 1600nm^2 junctions, the on resistance ($R_{ondiode}$) was roughly 500KΩ,

and the off resistance ($R_{offdiode}$) 9MΩ. After an initial burn-in step, the state of these devices can be switched at ±2V and read at ±0.2V. The basic hysteretic molecular memory effect is not unique to the [2]rotaxane, and the junction resistance is continuously tunable [Stewart et al. 2004]. The exact nature of the physical phenomena involved is the subject of active investigation.

LB techniques can also be used to place the switchable molecules between crossed NWs (e.g., Collier et al. [2000]). The molecules are formed into a single monolayer in an LB trough and then transfered onto a set of parallel NWs [Brown et al. 2000]. An orthogonal set of NWs is then transfered on top creating the conductor-device-conductor sandwich for the crosspoint array. The Chen et al. [2003] 8×8 molecular crossbar was constructed using this approach.

In conventional VLSI, the area of an SRAM-based programmable crosspoint switch is much larger than the area of a wire crossing. A typical, CMOS switch might be $2500\lambda^2$ [DeHon 1996], compared to a $5\lambda \times 5\lambda$ bottom level metal wire crossing, making the crosspoint $100\times$ the area of the wire crossing. Consequently, the nanoscale crosspoints offer an additional device size reduction beyond that implied by the smaller NW feature sizes. This particular device size benefit reduces the overhead for configurability associated with programmable architectures (e.g., FPGAs, PLAs) in this technology compared to conventional CMOS.

Appendix A.2 highlights four more molecular-scale, nonvolatile crosspoint technologies and additional technologies are actively being developed.

2.4 Technology Roundup

We can create wires which are nanometers in diameter and which can be arranged into crossbar arrays with nanometer pitch. Crosspoints which both switch conduction between the crossed wires and store their own state can be placed at every wire crossing without increasing the pitch of the crossbar array. NWs can be controlled in FET-like manner and can be designed with selectively gateable regions. This can all be done without relying on ultrafine lithography to create the nanoscale feature sizes. Consequently, these techniques promise smaller feature sizes and an alternate, perhaps more economical, path to atomic-scale computing structures than top-down lithography.

Each of the capabilities previously described has been demonstrated in a lab setting as detailed in the papers cited. We assume it will be possible to combine these capabilities and scale them into a repeatable manufacturing process.

3. CHALLENGES

In our top-down lithographic model, we define a minimum, lithographically imageable feature size and build devices that are multiples of this imageable feature size (e.g., half pitch). Within the limits of this feature size, we could perfectly specify the size of features and their relative location to each other in three dimensions—both in the two-dimensional plane of each lithographic layer and with adequate registration between layers. This would give us complete flexibility in the design of circuit structures as long as we adhered to the minimum imageable and repeatable feature size rules.

As we approach the atomic-scale, it becomes harder and harder to maintain this model. Precise location of atoms becomes relevant. Discreteness of the underlying atoms begins to show up as a significant fraction of feature size. Variations occur due to statistical doping and dopant placement and interferometric mask patterning. Perfect repeatability may be extremely difficult or infeasible for these feature sizes.

These bottom-up approaches, in contrast, promise us finer feature sizes that are controlled by physical phenomena but do not promise perfect, deterministic alignment in three dimensions. We may be able to get good repeatability of certain kinds of small feature sizes (e.g., NW diameters) and correlation of tiny features within a single NW using axial and radial composition, but we may have little correlation from NW to NW in the plane or between NW planes. This leads us to ask if we can reasonably give up our perfect correlation and complete design freedom in three dimensions in order to exploit smaller feature sizes. The techniques summarized here suggest this is a viable alternative.

3.1 Regular Assembly

The assembly techniques (Sections 2.2 and 2.3) suggest that we can build regular arrays at tight pitch with both NW trace width and trace spacing using controlled NW diameters. While this gives us nanometer pitches and crosspoints that are 10s of nanometers in area, we cannot deterministically differentiate features at this scale, that is, we cannot make one particular crosspoint be different in some way from the other crosspoints in the array.

3.2 Nanowire Lengths

NWs can be grown to hundreds of microns [Wu and Yang 2000] or perhaps millimeters [Zheng et al. 2002] in length. However, at this high length to diameter ratio, they become highly susceptible to bending and ultimately breaking. Assembly puts stresses along the NW axis which can break excessively long NWs. Consequently, we must limit ourselves to modest NW lengths (10s of microns) in order to yield a large fraction of the NWs in a given array. Gudiksen et al. [2001] report reliable growth of SiNWs which are over 9μm long, and Whang et al. [2003a, 2003b] demonstrated collections of arrays of NWs of size $10\mu\text{m}\times10\mu\text{m}$. Even if we could physically build longer NWs, the high resistivity of small diameter NWs would also drive us to keep NWs down to the 10s of micron length range (Section 11.4).

3.3 Defective Wires and Crosspoints

At this scale, we expect wires and crosspoints to be defective in the 1–10% range.

—NWs may break along their axis during assembly as suggested earlier, and the integrity of each NW depends on the $\sim$100 atoms in each radial cross-section.

—NW to microwire junctions depend on a small number of atomic scale bounds which are statistical in nature and subject to variation in NW properties.

—Junctions between crossed NWs will be composed of only 10s of atoms or molecules and individual bond formation is statistical in nature.
—Statistical doping of NWs may lead to high variation among NWs.

For example, Huang et al. [2001] reports that 95% of the wires measured had good contacts, and Chen et al. [2003] reports that 85% of crosspoint junctions measured were usable. Both of these are early experiments and we expect the yield rates to improve. However, based on the physical phenomena involved, we anticipate the defect rates will be closer to the few percent range than the minuscule rates we are familiar with in conventional, lithographic processing.

Consequently, we consider two main defect types.

(1) *Wire Defects*. A wire is either functional or defective. A functional wire has good contacts on both ends, conducts current with a resistance within a designated range, and is not shorted to any other NWs. Broken wires will not conduct current. Poor contacts will increase the resistance of the wire leaving it outside of the designated resistance range. Excessive variation in NW doping from the engineered target can also leave the wire out of the specified resistance range. We can determine if a wire is in the appropriate resistance range during testing (Section 9.1) and can arrange not to use the ones which are defective (Section 8.1).
(2) *Nonprogrammable Crosspoint Defects*. A crosspoint is programmable, nonprogrammable, or shorted into the on state. A programmable junction can be switched between the resistance range associated with the on-state and the resistance range associates with the off-state. A nonprogrammable junction can be turned off, but cannot be programmed into the on-state; a nonprogrammable junction could result from the statistical assembly of too few molecules in the junction or from poor contacts between some of the molecules in the junction and either of the attached conductors. A shorted junction cannot be programmed into the off-state. Based on the physical phenomena involved, we consider nonprogrammable junctions to be much more common than shorted junctions. Further, we expect fabrication can be tuned to guarantee this is the case. Consequently, we will treat shorted junctions like a pair of defective wires and avoid both wires associated with the short.

We do not currently consider bridging of adjacent NWs as a major defect source. Radial shells around the (semi) conducting NW cores prevent the shorting of adjacent NWs. At present, there is insufficient experience to determine if variations in core shell thickness, imperfect planar NW alignment, or other effects may, nonetheless, lead to bridging defects between adjacent NWs. If such bridging were to occur, it could make a pair of NWs indistinguishable, perhaps effectively giving two addresses to the NW pair. These bridged NW pairs could be detected and avoided but their occurrence would necessitate slightly more complicated testing and verification algorithms than the ones detailed in Section 9.

After describing the building blocks and architecture, Section 8 describes how we accommodate our two main defect types within the architecture.

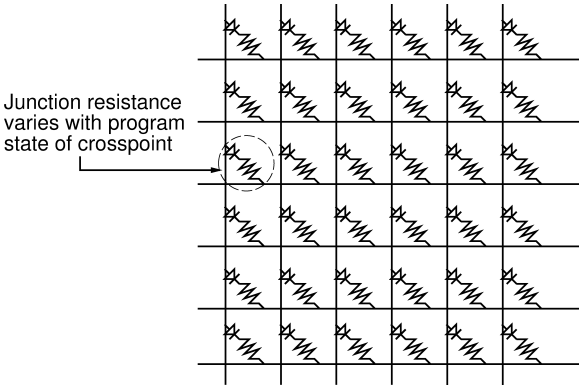


Fig. 5. Logical diode crossbar formed by crossed NWs.

4. BUILDING BLOCKS

Working from the technological capabilities (Section 2) and within the regular assembly requirements (Section 3.1), we can build a few building blocks which enable the construction of a widerange of interesting programmable architectures.

4.1 Crosspoint Arrays

As suggested in Section 2.2 and demonstrated in Chen et al. [2003] and Wu et al. [2005], assembly processes allow us to create tight-pitch arrays of crossed NWs with switchable diodes at the crosspoints (see Figure 5). Assuming for the moment that we can make contact to individual NWs in these tight-pitch arrays (see Section 4.2), these arrays can serve as:

—memory cores,

—programmable, wired-OR planes,

—programmable crossbar interconnect arrays.

4.1.1 *Memory Core.* As noted in Section 2.3, by applying a large voltage across a crosspoint junction, the crosspoint can be switched into a high or low resistance state. Consequently, if we can set the voltage on a single row and a single column line to desired voltages, we can set each of the crosspoints to a particular conduction state. We further noted that we can operate at a lower voltage without resetting the crosspoint. Consequently, we can read back a crosspoint's state by applying a small, test voltage to a column input and observing the current flow, or rate of charging, of a row line to tell if the crosspoint has been set into a high or low resistance state.

4.1.2 *Programmable, Wired-*OR *Plane.* Once we have a way to program the crosspoints into high or low resistance states, we can program OR logic into a crosspoint array. Each row output NW serves as a wired-OR for all of the inputs programmed into the low resistance state. Consider a single row NW, and assume for the moment that we have a way to pull a nondriven NW down to ground. Now if any of the column NWs which cross this row NW are connected

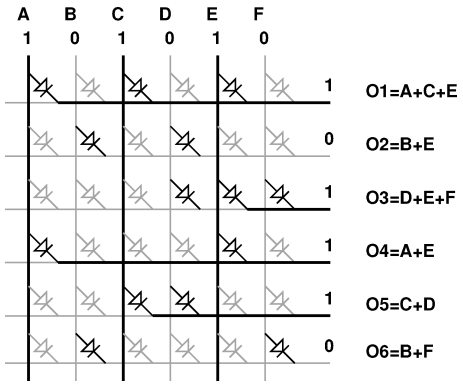


Fig. 6. Wired-OR plane operation. Programmed on crosspoints are shown in black; off crosspoints are shown in grey. Dark lines represent a NW pulled high, while light lines remain low. Output NWs are marked dark, starting at the diode that pulls them high, in order to illustrate current flow; the entire output NW would be pulled high in actual operation.

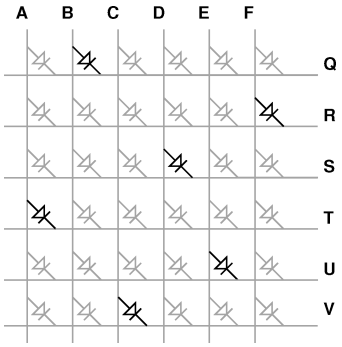


Fig. 7. Example crossbar routing configuration. Programmed on crosspoints are shown in black; off crosspoints are shown in grey. Here we show the crossbar programmed to connect A→T, B→Q, C→V, D→S, E→U, and F→R.

with low resistance crosspoint junctions and are driven to a high voltage level, the current into the column NW will be able to flow into the row NW and charge the row NW up to a higher voltage value (see O1, O3, O4, and O5 in Figure 6). However, if none of the connected column NWs is high, the row NW will remain low (see O2 and O6 in Figure 6). Consequently, the row NW effectively computes the OR of its programmed inputs.

The output NWs do pull their current directly off the inputs and may not be driven as high as the input voltage. So, these outputs will need restoration (Section 4.3).

4.1.3 *Programmable Crossbar Interconnect Arrays.* A special use of the Wired-OR programmable array is for interconnect. That is, if we restrict ourselves to connecting a **single** row wire to each column wire, the crosspoint array can serve as a crossbar switch. This allows us to route any input (column) to any output (row) (e.g., see Figure 7). This structure is useful for

post-fabrication programmable routing to define a logic function and to avoid defective resources (Section 3.3).

4.2 Decoders

A key challenge is bridging the length scale between the lithographic-scale wires we can create using conventional top-down lithography and the small diameter NWs we can grow and assemble into tight-pitch arrays. As noted in Section 4.1.1, we must be able to establish a voltage differential across a single row and column NW to write a bit in the tight-pitch NW array. Further, we must be able to drive and sense individual NWs to read back the memory bit. By building a decoder between the coarse-pitch lithographic wires and the tight-pitch NWs, we can bridge this length scale and address a single NW at this tight pitch [Williams and Kuekes 2001; DeHon et al. 2003; Gojman et al. 2004; DeHon 2004].

4.2.1 *NW Coding.* One way to build such a decoder is to place an address on each NW using the axial doping or material composition profile described previously (Section 2.1.3). To interface with lithographic-scale wires, we mark off address bit regions at the lithographic pitch. Each such region is then either doped heavily so that it is oblivious to the field applied by a crossed lithographic-scale wire or is doped lightly so that it can be controlled by a crossed lithographic-scale wire. In this way, the NW will only conduct if all of the lithographic-scale wires crossing its lightly doped, controllable regions have a suitable voltage to allow conduction. If any of the lithographic-scale wires crossing controllable regions provide a suitable voltage to turn off conduction, then the NW will not be able to conduct.

Note that we can only make each bit position either controllable or noncontrollable with respect to the lithographic-scale control wire. We cannot make different bit positions sensitive to different polarities of the input. Consequently, we must encode our addresses differently from the dense, binary address we normally use for memories. One simple way to generate addresses is to use a dual-rail binary code. That is, for each logical input bit, we provide the value and its complement. This results in two bit positions on the NW for each logical input address bit, one for the true sense and one for the false sense. To code a NW with an address, we simply code either bit position to be sensitive to exactly one sense of each of the bit positions (see Figure 8). This results in a decoder which requires $2\log_2(N)$ address bits to address N NWs.

We can achieve denser addressing using $N_a/2$-hot codes. That is, rather than forcing one bit of each pair of address lines to be off and one to be on, we simply require that half of the address bits, N_a, be set to a voltage which allows conduction and half to be set to a voltage that prevents conduction. This scheme requires only $1.1\log_2(N) + 3$ address bits [DeHon et al. 2003].

4.2.2 *Decoder Assembly.* If each NW in our array has a unique address in our selected coding scheme, we can uniquely address each individual NW in the array. However, our NW assembly techniques (Section 2.2) do not allow us to place particular NWs in particular locations. We can only arrange to create a tight-pitch parallel ensemble of a collection of NWs.

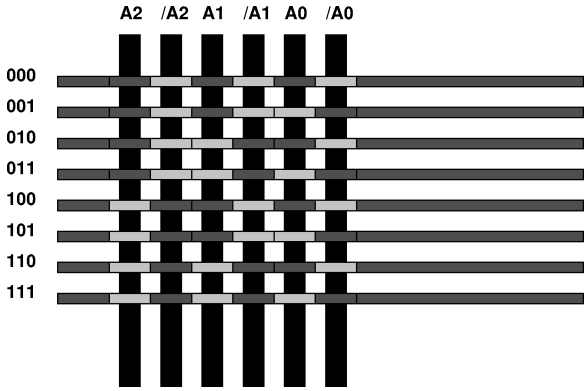


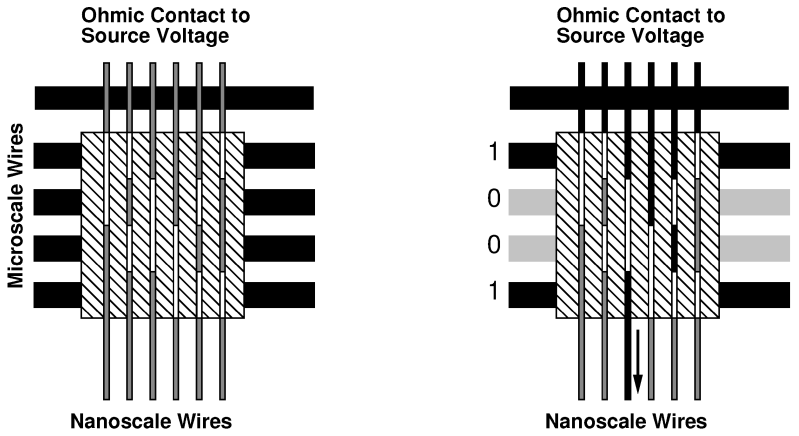
Fig. 8. Dual-rail address coding.

Instead, we notice that if we make the code space for the NWs large compared to the size of the array we need to address, we can statistically guarantee with arbitrarily high probability that every NW in an array has a unique address. That is, we start with a very large number of NW codes. We mix up the NWs before assembly so that we are randomly selecting which NW codes go into each of the crosspoint arrays we are assembling. As long as the array formed is sufficiently small compared to the code space for the NWs, we can provide strong guarantees that each array contains NWs with unique codes [DeHon et al. 2003]. It turns out that we do not need a large number of address bits in order to guarantee this uniqueness. For example, the $N_a/2$-hot codes need a total of only $\lceil 2.2 \log_2(N) \rceil + 11$ bits to achieve over a 99% probability that all NWs in an array will have unique addresses. If a few duplicates are tolerable, then the codes can be much tighter [Gojman et al. 2004; DeHon 2004].

Wires can be coded to tolerate misalignment during assembly. Hybrid addressing schemes which segment the collection of NWs in an array into lithographic-scale contact groups can be used to reduce the size of the NW codespace needed. See DeHon et al. [2003] for further details.

4.2.3 *Decoder and Multiplexer Operation.* Now that we know we can assemble uniquely coded NWs into an array, we can see how the decoder operates. Assume all the NWs are either precharged or weakly pulled to a nominal voltage. We then apply the desired NW address to the lithographic-scale address lines. We also apply the desired drive voltage to a common line attached to all the NWs. If the selected address is present in the array, it will allow conduction from the common line into the array charging up the selected NW. All other NWs will differ in at least one bit position and will thus be disabled by the address lines. Consequently, only the selected NW is charged strongly to the voltage driven on the common line, and all other NWs are held at the nominal voltage (see Figure 9).

Note that there is no directionality to the decoder. Consequently, this same unit can serve equally well as a multiplexer. That is, when we apply an address to the lithographic-scale wires, it allows conduction through the addressing region for only one of the NWs. Consequently, we can sense the voltage on the



(a) Decoder Configuration: white NW regions are coded and controllable, while grey regions are not controllable and act as wires.

(b) Dark represents lines driven high; light grey shows lines low or undriven. Only the coding on the third line matches the applied address (1001) and allows conduction. All other cases have a high address voltage crossing a lightly doped region which prevents conduction.

Fig. 9. Coded NW decoder.

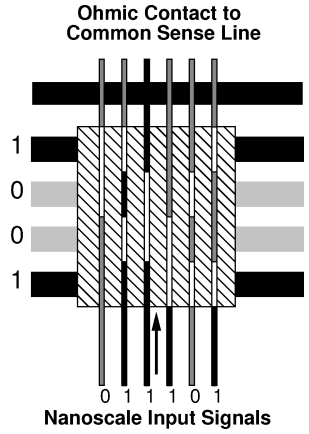


Fig. 10. Coded NW multiplexer operation.

common line rather than drive it. Now the one line allowed to conduct through the array can potentially pull the common line high or low. All other lines have a high resistance path across the lithographic-scale address wires and will not be able to strongly effect the common line. This allows us to sense a single NW at a time (see Figure 10) as we need to read out crosspoint state as described in Section 4.1.1.

4.3 Restoration and Inversion

As noted in Section 4.1.2, the programmable, wired-OR logic is passive and nonrestoring, drawing current from the input. Further, OR logic is not universal.

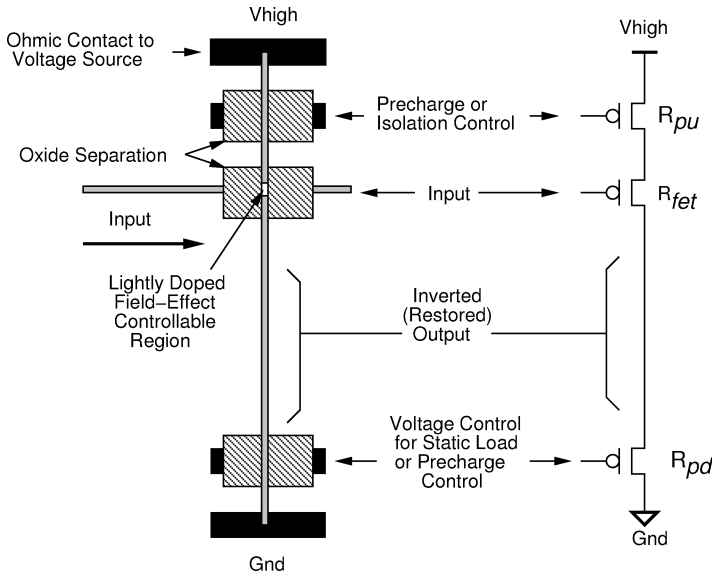


Fig. 11. NW inverter.

To build a good composable logic family, we will need to be able to isolate inputs from output loads, restore signal strength and current drive, and invert signals.

Fortunately, NWs can be field-effect controlled. This gives us the potential to build FET-like gates for restoration. However, to realize them, we must find ways to create the appropriate gate topology within our regular assembly constraints (Sections 3.1).

4.3.1 *NW Inverter and Buffer.* If we separate two NWs by an insulator, perhaps using an oxide core shell (Section 2.1.4), we can potentially use the field from one NW to control the other NW. Figure 11 shows an inverter built using this basic idea. The horizontal NW serves as the input and the vertical NW as the output. This gives a voltage transfer equation:

$$V_{out} = V_{high}\left(\frac{R_{pd}}{R_{pd} + R_{fet}(\text{Input}) + R_{pu}}\right). \tag{1}$$

For the sake of illustration, the vertical NW has a lightly doped P-type depletion mode region at the input crossing forming a FET controlled by the input voltage (R_{fet}(Input)). Consequently, a low voltage on the input NW will allow conduction through the vertical NW ($R_{fet} = R_{onfet}$ is small), and a high input will deplete the carriers from the vertical NW and prevent conduction ($R_{fet} = R_{offfet}$ is large). As a result, a low input allows the NW to conduct and pull the output region of the vertical NW up to a high voltage. A high input prevents conduction and the output region remains low. A second crossed region on the NW is used for the pull down (R_{pd}); this region can be used as a gate for predischarge so the inverter is pulled low before the input is applied, then left high to disconnect the pulldown voltage during evaluation. Alternately, it can be used as a static load for PMOS-like ratioed logic. By swapping the location of the high- and low-power

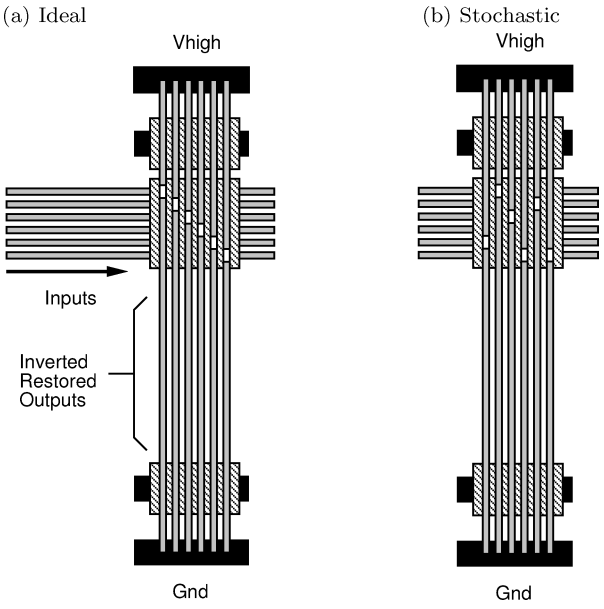


Fig. 12. Restoration array.

supplies, this same arrangement can be used to buffer rather than invert the input.

Note that the gate only loads the input capacitively. Consequently, we get current isolation at this inverter or buffer. Further, NW field-effect gating has sufficient nonlinearity so that this gate provides gain to restore logic signal levels [DeHon 2003].

4.3.2 *Ideal Restoration Array.* In many scenarios, we need to restore a set of tight-pitch NWs such as the outputs of a programmable, wired-OR array (Section 4.1.2). To do this, we would like to build a restoration array as shown in Figure 12(a). This array is a set of crossed NWs which we can assemble using our NW assembly techniques (Section 2.2). If each of the NWs was sensitive to all of the crossed inputs, we would end up with all of the outputs actually computing the NOR of the same set of inputs. To avoid computing a redundant set of NORs and instead simply invert each of the inputs independently, we code these NWs using an axial doping or material composition profile (Section 2.1.3). This way, each NW is field-effect sensitive to only a single NW and hence provides the NW inversion described for a single one of the crossed NWs and is oblivious to the inputs of the other NWs.

The only problem here is that we do not have a way to align and place axially-doped NWs so that they provide exactly this pattern. Assembly treats all NWs as identical.

4.3.3 *Restoration Array Construction.* The region for active FETs is a nanoscale feature but it does not require small pitch or tight alignment. As such, there may be ways to mask and provide material differentiation along a diagonal as required to build this decoder.

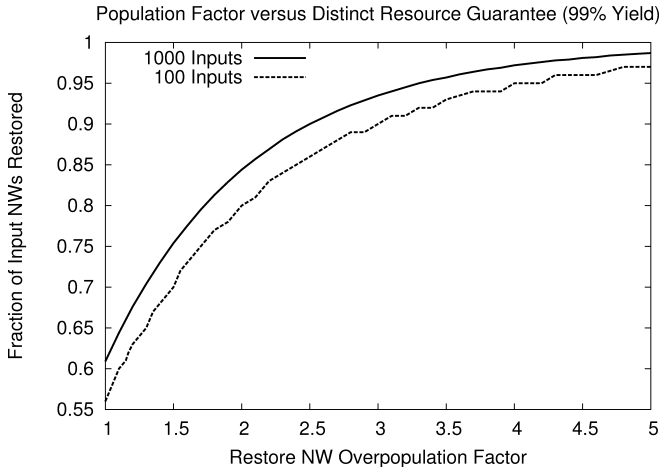


Fig. 13. Fraction of input NWs restored as a function of restoration overpopulation.

Nonetheless, it is also possible to stochastically construct this restoration array in a manner similar to the construction of the address decoder. That is, we provide assembly with a set of NWs with their restoration regions in various locations. The restoration array will be built by randomly selecting a set of restoration NWs for each array (see Figure 12(b)).

Two things differ compared to the address decoder case.

(1) The code space will be the same size as the desired restoration population.
(2) Duplication is allowed.

Our question then is how large a fraction of the inputs will we succeed in restoring for a given number of randomly selected restoration NWs? This is an instance of the Coupon Collectors Problem [Maunsell 1937]. If we populate the restoration array with the same number of NWs as inputs, the array will typically contain restoration wires for 50–60% of the NW inputs. One way to look at this is that we have to populate the array with 1.7–2$\times$ as many wires as we hope to yield due to these stochastic assembly effects. If we increase the number of restoration wires relative to the number of input NWs, we can restore a higher fraction of the inputs as shown in Figure 13. See Appendix B and DeHon [2004] for further details on these yield calculations.

5. MEMORY ARRAY

Combining the crosspoint memory cores (Section 4.1.1) with a pair of decoders (Section 4.2), we can build a tight-pitch, NW-based memory array [DeHon et al. 2005]. Figure 14 shows how these elements come together in a small memory array. The entire array is formed using crossed, tight-pitch NWs. Programmable diode crosspoints are assembled in the NW-NW crossings. Lithographic-scale address wires form row and column addresses. Write operations into the memory array can be performed by driving the appropriate write voltages onto a single row and column line. Read operations occur by driving a reference voltage

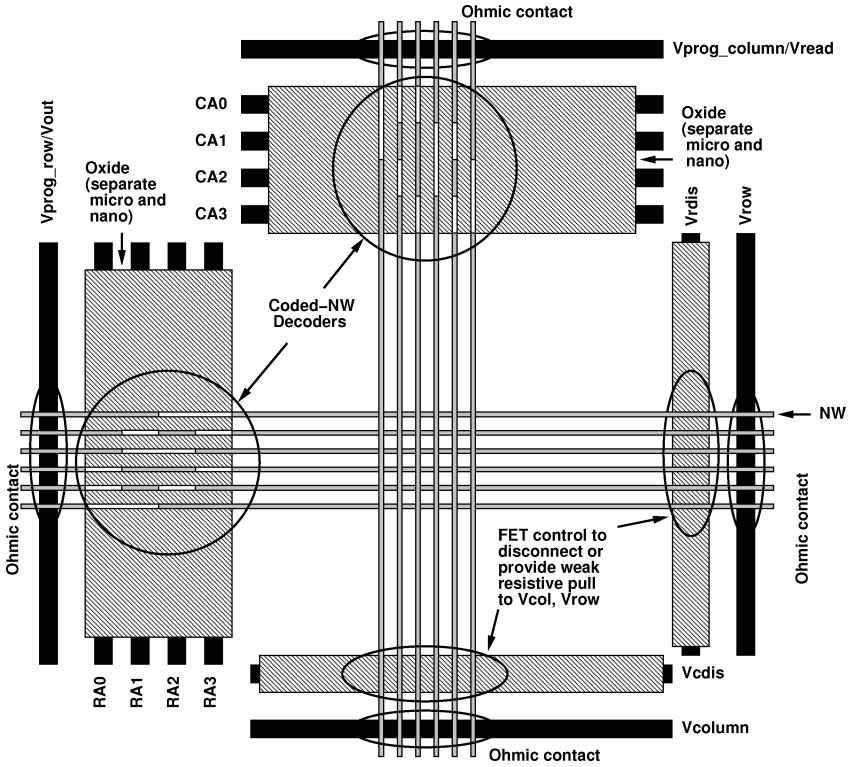


Fig. 14. Memory array built from coded NW decoder and crosspoint memory core.

onto the common column line, setting the row and column addresses, and sensing the voltage on the common row read line.

Limitations on reliable NW length (Section 3.2) and the capacitance and resistance of long NWs prevent us from building arbitrarily large memory arrays. Instead we break up large NW memories into banks similar to the banking used in conventional DRAMs (see Figure 15). Reliable, lithographic-scale wires provide address and control inputs and data inputs and outputs to each of the NW-based memory banks. We expect to yield only a fraction of the NWs in the array due to wire defects (Section 3.3). Error-correcting codes (ECC) can be used to tolerate nonprogrammable crosspoint defects (Section 3.3). After accounting for defects, ECC overhead, and lithographic control overhead, net densities on the order of 10^{11} bits/cm^2 appear achievable, using NW pitches around 10nm [DeHon et al. 2005].

6. LOGIC ARCHITECTURE

Combining the building blocks introduced in Section 4, we can construct complete, programmable logic architectures with all logic, interconnect, and restoration occurring in the atomic-scale NWs. Diode crosspoints organized into Wired-OR logic arrays (Section 4.1.2) provide programmable logic, field-effect restoration arrays (Section 4.3) provide gain and signal inversion, and

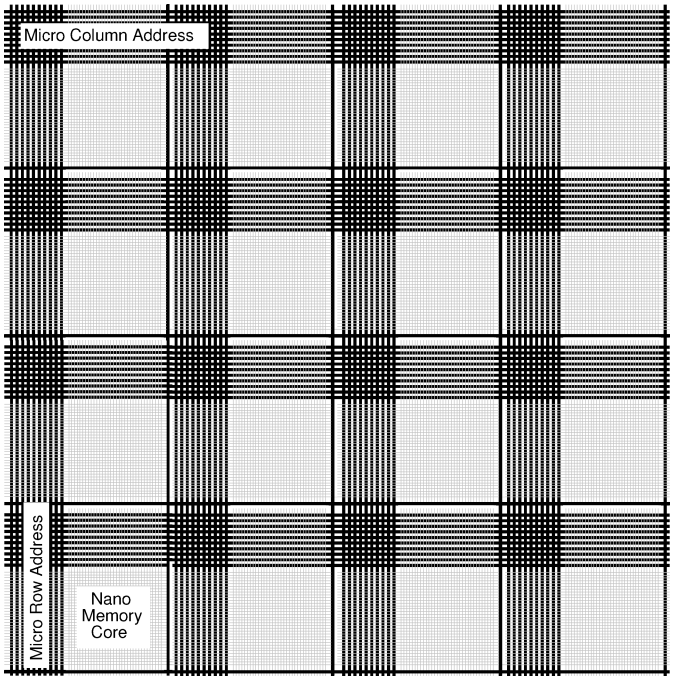


Fig. 15. Tile of NW-based memory banks to construct large-scale memory.

the NWs themselves provide interconnect among arrays. Lithographic scale wires provide a reliable support infrastructure which allows device testing and programming (Section 9), addressing individual NWs using the decoders introduced in Section 4.2. Lithographic-scale wires also provide power and control logic evaluation.

6.1 Logic

Figure 16 shows a simple Programmable Logic Array (PLA) built using the building blocks from Section 4 and first introduced in DeHon and Wilson [2004]. The design includes two interconnected logic planes. Each plane is composed of a programmable wired-OR array (Section 4.1.2), followed by a restoration array (Section 4.3). Note here that we actually use two restoration arrays, one providing the inverted sense of the OR-term logic and one providing the noninverted buffered sense. This arrangement is similar to conventional PLAs where we provide the true and complement sense of each input into each PLA plane. Since wired-OR logic NWs can be inverted in our nanoPLA, each plane effectively serves as a programmable NOR plane. The combination of the two coupled NOR-NOR planes can be viewed as an AND-OR PLA with suitable application of DeMorgan's Laws and signal complementation.

6.1.1 *Construction.* The entire construction is simply a set of crossed NWs as allowed by our regular assembly constraints (Section 3.1). Lithographic-scale etches are used to differentiate regions (e.g., programmable-diode regions

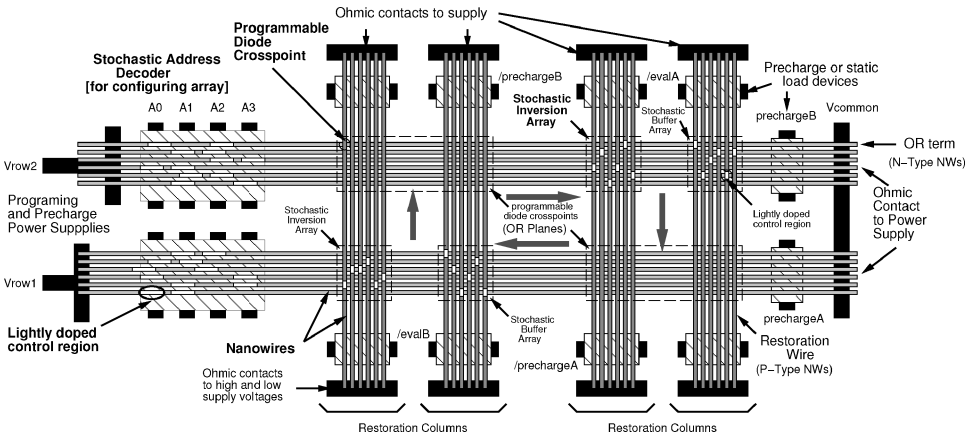


Fig. 16. Simple nanoPLA block.

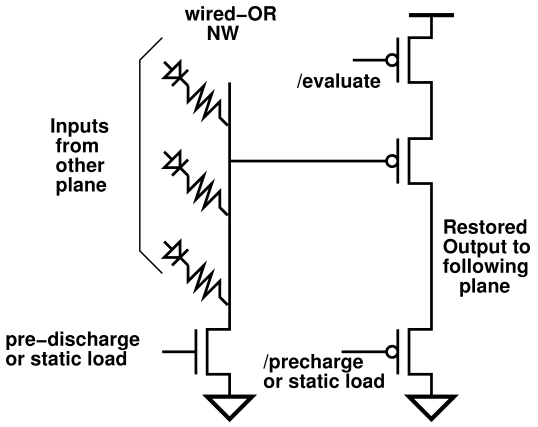


Fig. 17. Rough circuit equivalent for each nanoPLA plane.

for the wired OR). The topology allows the same NWs that perform logic or restoration to carry their outputs across as inputs to the array that follows it.

6.1.2 *Logic Circuit.* The logic gates in each PLA plane are composed of a diode-programmable wired-OR NW, followed by a field-effect buffer or inverter NW (see Figure 17). The field-effect stage provides isolation as there is no current flow between the diode stage and the field-effect stage output. That is, the entire OR stage is capacitively loaded rather than resistively loaded. The OR stage simply needs to charge up its output which provides the field for the field-effect-based restoration stage. When the field is high enough (low enough for P-type NWs) to enable conduction in the field-effect stage, the NW will allow the source voltage to drive its output.

6.1.3 *Programming.* On the left of Figure 16, we form a decoder as introduced in Section 4.2 using the vertical microscale wires A0–A3. These lithographic-scale wires allow us to select individual NWs for programming.

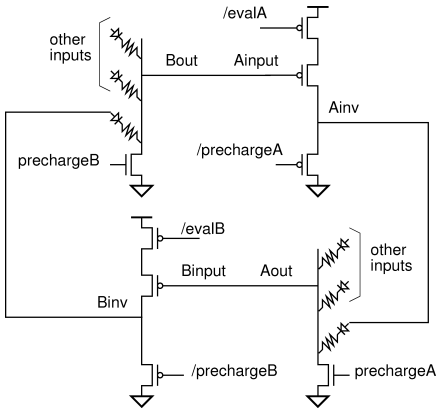


Fig. 18. Precharge clocked INV-OR-INV-OR (NOR-NOR, AND-OR) cycle.

Each usable vertical restoration NW is driven by a horizontal NW. Consequently, we only need decoders to address the horizontal NWs (see Section 9).

6.2 Registers and Sequential Logic

With slight modification in how we drive the control signals on the identified logic stages, we can turn this into a clocked logic scheme. An immediate benefit is the ability to create a finite-state machine out of a single pair of PLA planes. A second benefit is the ability to use precharge logic evaluation for inverting restoration stages.

6.2.1 *Basic Clocking.* The basic nanoPLA cycle shown in Figure 16 is simply two restoring logic stages back-to-back (see Figure 18). For our clocking scheme, we evaluate the two stages at altering times.

First, note that if we turn off all three of our control transistors in our restoring stages (restoring precharge and evaluate and diode precharge, e.g., evalA and prechargeA in Figure 18), there is no current path from the input to the diode output stage. We effectively isolate the input from the output. Since the output stage is capacitively loaded, the output will hold its value. As with any dynamic scheme, eventually leakage on the output will be an issue which will set a lower bound on the clock frequency.

With a stage isolated and holding its output, we can evaluate the following stage. It computes its value from its input, the output of the previous stage, and produces its result by suitably charging its output line. Once this is done, we can isolate this stage and evaluate its succeeding stage which, in this simple case, is also its predecessor. This is the same strategy as two-phase clocking in conventional VLSI (e.g., Mead and Conway [1980] and Weste and Harris [2005]).

In this manner, we never have an open current path all the way around the PLA (see Figures 18 and 19). In the two phases of operation, we effectively have a single register on any PLA outputs which feed back to PLA inputs.

6.2.2 *Precharge Evaluation.* For the inverting stage, we drive the pull-down gate hard during precharge and turn it off during evaluation. In this

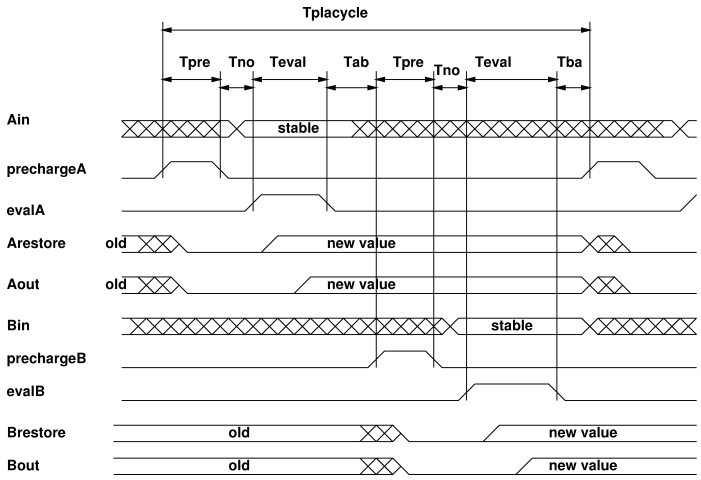


Fig. 19. Clocking/precharge timing diagram.

manner, we precharge the line (Ainv) low and pull it up only if the input (Ainput) is low. This works conveniently in this case because the output will also be precharged low. If the input is high, then we do not want to pullup the output and simply leave it low. If the input is low, we enable the current path to pullup the output. The net benefit is that inverter pulldown and pullup are both controlled by strongly driven gates and can be fast, whereas in a static logic scheme, the pulldown transistor must be weak, making pulldown slow compared to pullup. Typically, the weak pulldown transistor would be set to have an order of magnitude higher resistance than the pullup transistor so this can be a significant reduction in worst-case gate evaluation latency (see Section 11.3).

Unfortunately, we can neither precharge to high nor turn off the weak pullup resistor in the buffer case so we do not get comparable benefits there. Perhaps new devices or circuit organizations will eventually allow us to build precharge buffer stages.

6.3 Interconnect

We know from VLSI that large PLAs do not always allow us to exploit the structure which exists in logic. For example, an n-input XOR requires an exponential number of product terms to construct in the two-level logic of a single PLA. Further, our limitation on NW length (Section 3.2) bounds the size of the PLAs we can reasonably build. Consequently, to scale up to large capacity logic devices, we must interconnect modest size nanoPLA blocks; we extend these nanoPLA blocks to include input and output to other nanoPLA blocks and assemble them into a large array (see Figure 20) as first introduced in DeHon [2005].

6.3.1 *Basic Idea.* The key idea for interconnecting nanoPLA blocks is to overlap the restored output NWs from each such block with the wired-OR input region of adjacent nanoPLA blocks (see Figure 20). In turn, this means each nanoPLA block receives inputs from a number of different nanoPLA blocks.

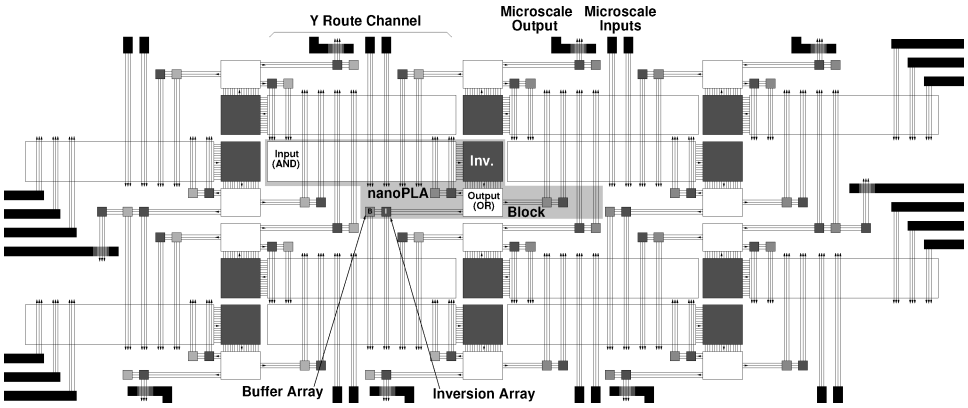


Fig. 20. nanoPLA block tiling with edge IO to lithographic scale.

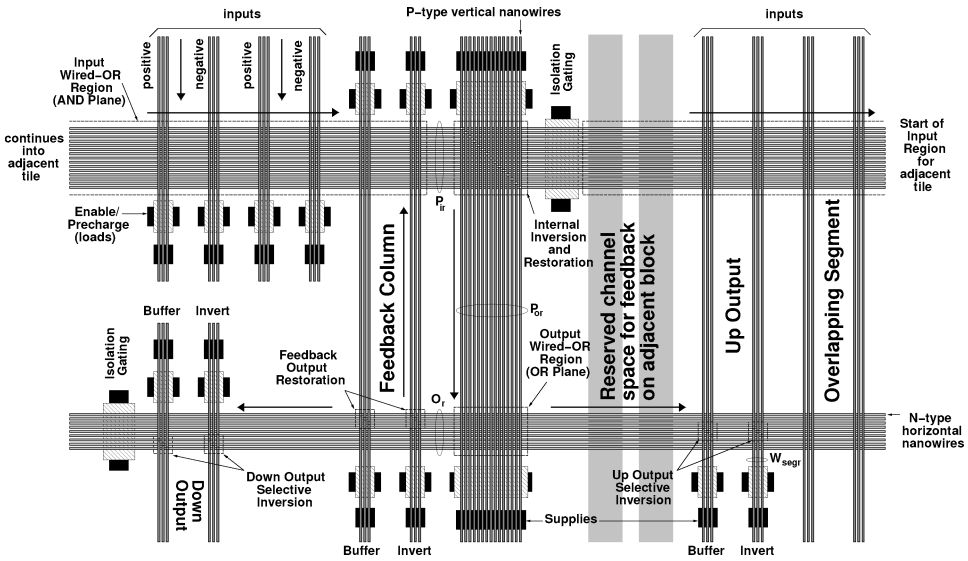


Fig. 21. nanoPLA block tile.

With multiple input sources and outputs routed in multiple directions, this allows the nanoPLA block to also serve as a switching block. By arranging the overlap appropriately, we can support Manhattan routing, allowing the array of nanoPLA blocks to be configured to route signals between any of the blocks in the array.

6.3.2 *nanoPLA Block*

6.3.2.1 *Input wired-*OR *region.* One or more regions of programmable crosspoints serves as the input to the nanoPLA block. Figures 20 and 21 show a nanoPLA block design with a single such input region. The inputs to this region are restored output NWs from a number of different nanoPLA blocks. The programmable crosspoints allow us to select the inputs which participate in

each logical product term (PTERM) building a wired-OR array (Section 4.1.2) as in the base nanoPLA (Section 6.1).

6.3.2.2 *Internal inversion and restoration array.* The NW outputs from the input block are restored by a restoration array (Section 4.3). We arrange the restoration logic at this stage to be inverting so that we provide the logical NOR of the selected input signals into the second plane of the nanoPLA.

6.3.2.3 *Output* OR *plane.* The restored outputs from the internal inversion plane become inputs to a second programmable crosspoint region. Physically, this region is the same as the input plane. Each NW in this plane computes the wired OR of one or more of the restored PTERMs computed by the input plane.

6.3.2.4 *Selective output inversion.* The outputs of the output OR plane are then restored in the same way as the internal restoration plane. On this output, however, we use the selective inversion scheme introduced in Section 6.1. This gives us both polarities of each output so we can provide them to the succeeding input planes. This selective inversion plays the same role as a local inverter on the inputs of conventional, VLSI PLA; here we place it with the output to avoid introducing an additional logic plane into the design.

As with the nanoPLA block, these two planes provide NOR-NOR logic. With suitable application of DeMorgan's laws, these can be viewed as a conventional AND-OR PLA.

6.3.2.5 *Feedback.* As shown in Figures 20 and 21, one set of outputs from each nanoPLA block feeds back to its own input region. This completes a PLA cycle similar to the nanoPLA design (Section 6.1). These feedback paths serve the role of intracluster routing similar to internal feedback in conventional Island-style [Betz et al. 1999] FPGAs. The nanoPLA block implements registers by routing signals around the feedback path (Section 6.2.1). We can route signals around this feedback path multiple times to form long register delay chains for data retiming.

6.3.3 *Interconnect*

6.3.3.1 *Block outputs.* In addition to self feedback, output groups are placed on either side of the nanoPLA block and can be arranged so they cross input blocks of nanoPLA blocks above or below the source nanoPLA block (see Figure 20). Like segmented FPGAs [Brown et al. 1996; Betz and Rose 1999a] output groups can run across multiple nanoPLA block inputs (i.e., Connection Boxes) in a given direction. The nanoPLA block shown in Figure 21 has a single output group on each side, one routing up and the other routing down. We will see that the design shown is sufficient to construct a minimally complete topology.

Since the output NWs are directly the outputs of gated fields: (1) an output wire can be driven from only one source, and (2) it can only drive in one direction. Consequently, unlike segmented FPGA wire runs, we must have directional wires which are dedicated to a single producer. If we coded multiple control regions into the NW runs, conduction would be the AND of the producers crossing

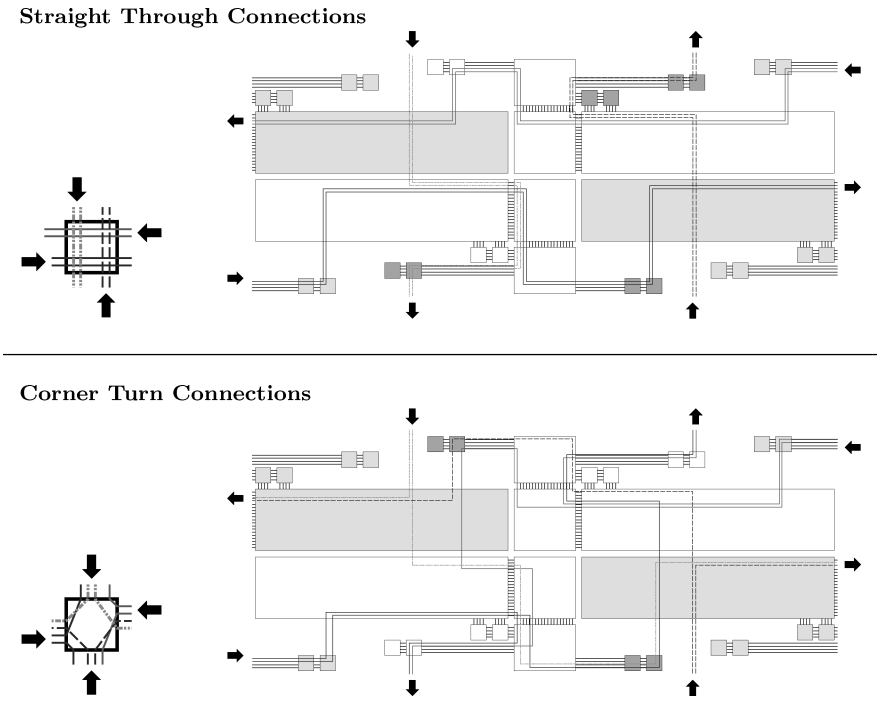


Fig. 22. Routing view of nanoPLA logic block.

the coded regions. Single direction drive arises from the fact that one side of the gate must be the source logic signal being gated so the logical output is only available on the opposite side of the controllable region. Interestingly, recent work suggests that conventional, VLSI-based FPGA designs would also benefit from directional wires [Lemieux et al. 2004].

6.3.3.2 *Y route channels.* With each nanoPLA block producing output groups which run one or more nanoPLA block heights above or below the array, we end up with vertical (Y) routing channels between the logic cores of the nanoPLA blocks (see Figure 20). The segmented, NW output groups allow a signal to pass a number of nanoPLA blocks. For longer routes, the signal may be switched and rebuffered through a nanoPLA block (see Figure 22). Because of the output directionality, we end up with separate sets of wires for routing up and routing down in each channel.

6.3.3.3 *X routing.* While Y route channels are immediately obvious in Figure 20, the X route channels are less apparent. All X routing occurs through the nanoPLA block. As shown in Figure 21, we place one output group on the opposite side of the nanoPLA block from the input. In this way, one can route in the X direction by going through a logic block and configuring the signal to drive a NW in the output group on the opposite side of the input. If all X routing blocks had their inputs on the left, then we would only be able to route from left

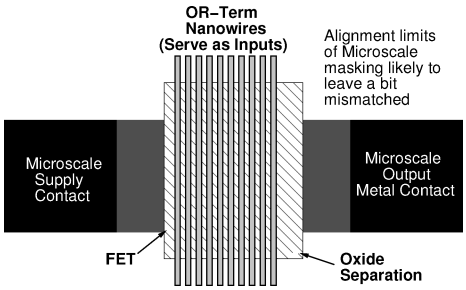


Fig. 23. Nanoscale to lithographic-scale FET output structure.

to right. To allow both left-to-right and right-to-left routing, we alternate the orientation of the inputs in alternate rows of the nanoPLA array (see Figures 20 and 22). In this manner, even rows provide left-to-right routing while odd rows allow right-to-left routing.

6.3.3.4 *Relation to Island-style Manhattan design.* Logically viewed, this interconnected nanoPLA block is very similar to conventional, Island-style FPGA designs, especially when the Island-style designs use directional routing [Lemieux et al. 2004]. As shown in Figure 22, we have X and Y routing channels, with switching to provide X-X, Y-Y, and X-Y routing.

6.4 CMOS IO

These nanoPLAs will be built on top of a lithographic substrate. The lithographic circuitry and wiring provides a reliable structure from which to probe the NWs to map their defects and to configure the logic (Section 9).

For input and output to the lithographic scale during operation, we can provide IO blocks to connect the nanoscale logic to lithographic-scale wires much as we connect lithographic-scale wires to bond pads on FPGAs. The simplest arrangement resembles the traditional, edge IO form of a symmetric FPGA with inputs and outputs attached to NWs at the edges of the routing channels (see Figure 20).

NW inputs can easily be driven directly by lithographic-scale wires. Since the lithographic-scale wires are wider pitch, a single lithographic wire will connect to a number of NWs. With the lithographic wire connected to the NWs, the NW crosspoints in the nanoPLA block inputs can be programmed in the same way they are for NW inputs.

It is possible to connect outputs in a similar manner. Such a direct arrangement could be particularly slow as the small NWs must drive the capacitance of a large, lithographic-scale wire. Alternately, the NWs can be used as gates on a lithographic-scale Field-Effect Transistor (FET) (see Figure 23). In this manner, the NWs are only loaded capacitively by the lithographic-scale output and only for a short distance. We can tune the NW thresholds and the lithographic FET thresholds into comparable voltage regions so the NWs can drive the lithographic FET at adequate voltages for switching. As shown, multiple NWs will cross the lithographic-scale gate. The OR-terms driving these outputs are all

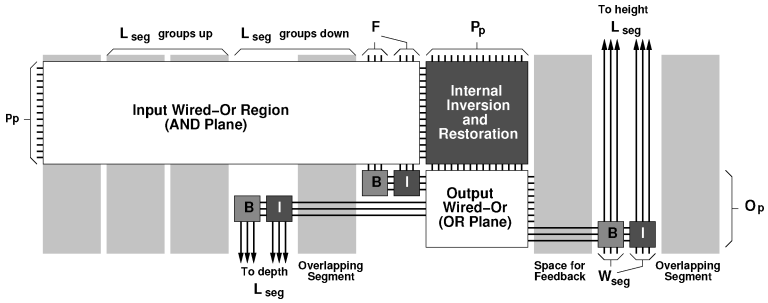


Fig. 24. nanoPLA block parameters.

programmed identically, allowing the multiple-gate configuration to provide strong switching for the lithographic-scale FET.

6.5 Parameters

Figure 24 shows the key parameters in the design of the nanoPLA block.

—W_{seg} is the number of NWs in each output group.

—L_{seg} is the number of nanoPLA block heights up or down which each output crosses; equivalently, the number of parallel wire groups across each Y route channel in each direction. We show $L_{seg} = 2$ in Figure 20, and maintain $L_{seg} = 2$ throughout this article.

—F is the number of NWs in the feedback group. For simplicity, we take $F = W_{seg}$ throughout this article.

—P is the number of logical PTERMS in the input (AND) plane of the nanoPLA logic block.

—O_p is the number of totals outputs in the OR plane. Since each output is driven by a separate wired-OR NW, $O_p = 2 \times W_{seg} + F$ for the nanoPLA block we focus on in this article with two routing output groups and a feedback output group.

—P_p is the number of total PTERMS in the input (AND) plane. Since these are also used for route-through connections, this is larger than the number of logical PTERMS in each logic block.

$$P_p \leq P + 2 \times W_{seg} + F \tag{2}$$

That is, in addition to the P logical PTERMS, we may need one physical wire for each signal that routes through the array for buffering; there will be at most O_p of these.

Additionally, we could parameterize the number and distribution of inputs (e.g., one side (as shown in Figure 24), from both sides, subsets of PTERMS from each side), the output topology (e.g., route both up and down on each side of the array), and segment length distributions. However, we focus on this simple topology with $L_{seg} = 2$ in this article. Consequently, the main physical parameters determining nanoPLA array size are W_{seg} and P_p.

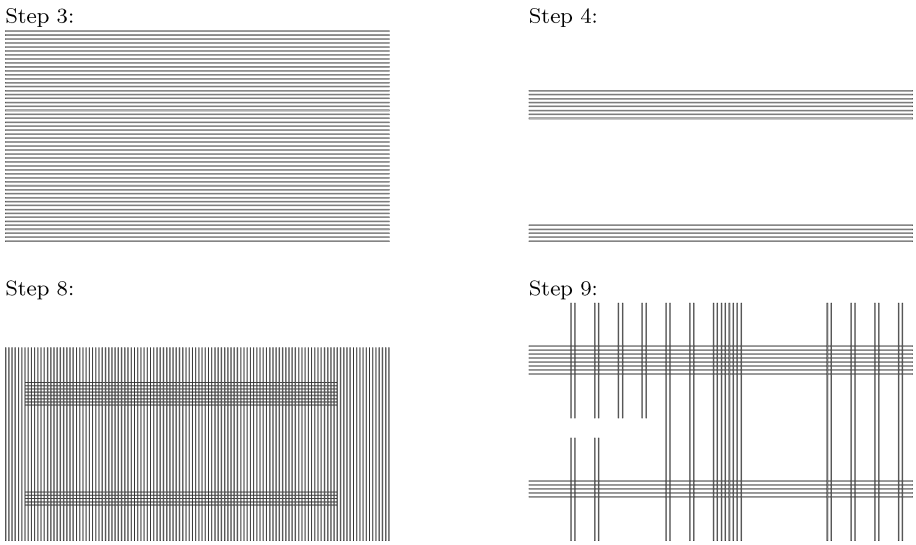


Fig. 25. Illustration of the key steps in the sublithographic fabrication strategy for the nanoPLA block tile.

7. ASSEMBLY AND MANUFACTURING

As noted in Sections 1 and 3, a key challenge is devising an architecture that can be realized with the limited bottom-up assembly techniques introduced in Section 2. Consequently, it is worthwhile to note that we can fabricate the nanoPLA structure with the following basic strategy.

(1) Prepare individual NWs – grow NWs [Cui et al. 2001; Morales and Lieber 1998] with axial differentiation [Gudiksen et al. 2002] and radial differentiation [Lauhon et al. 2002]. Use radial differentiation to place an oxide shell around the (semi) conducting NW core.
(2) Prepare a lithographic substrate with a flat surface.
(3) Use LB techniques to align NWs in a single direction, tight pack them, and transfer them to a surface [Whang et al. 2003b]. The oxide shell defines the spacing between NW conductors (see Figure 25).
(4) Lithographically etch breaks in the NWs to distinguish conduction regions (see Figure 25).
(5) Use material-specific lithographic etches to remove the oxide coating and expose the (semi) conducting core of the NWs where appropriate (e.g., contacts, some crosspoints) [Whang et al. 2003a].
(6) Lithographically mask and deposit metal coatings and anneal to convert desired portions of NWs into metal silicide [Wu et al. 2004].
(7) Use LB techniques to construct and transfer a uniform layer of molecules over the NW conductors, if appropriate [Brown et al. 2000].
(8) Repeat the LB transfer of an orthogonal layer of NWs to provide crossed NWs (see Figure 25).

(9) Repeat lithographically defined etching to segment the orthogonal NW layer (see Figure 25).
(10) Repeat material-specific lithographic etches to remove oxide coating and expose the (semi) conducting core of the NWs were appropriate.
(11) Repeat metal silicide conversion.
(12) Add additional lithographic layers for contacts.

As a result, we can have tight-pitch NWs in both directions. We can deterministically define the extents of these regions by lithographic etches. We cannot deterministically cut NWs, define their lengths, or place contacts on NWs below the lithographic resolution. We can differentiate the NWs at NW pitch by defining features in the NWs using timed growth when the NWs are initially prepared as described in Sections 2, 4.2.1, and 4.3.1. As a result, we are driven to regular architectures (Section 3.1) that use a large number of parallel NWs; the length of the NWs is of lithographic scale, as is the width of the ensemble of parallel NWs.

8. DEFECT TOLERANCE

As introduced in Section 3.3, we are likely to see a small percentage of wires which are defective and crosspoints which are nonprogrammable. Further, stochastic assembly (Sections 4.2.2 and 4.3.3) and misalignment will also result in a percentage of NWs which are unusable. Fortunately, NWs are interchangeable and the crosspoints are small (Section 2.3). Consequently, we can provision spare NWs in an array (e.g., overpopulate compared to the desired P_p and W_{seg}), test NWs for usability (Section 9.1), and configure the array using only the nondefective NWs. Further, a NW need not have a perfect set of junctions to be usable (Section 8.4).

8.1 NW Sparing

Tolerating wire defects is a simple matter of provisioning adequate spares, separating the good wires from the bad, and configuring the nanoPLA blocks accordingly. For a given PLA design, we want each block to have a minimum number of usable wires (P_p and W_{seg}). Since we will have wire losses, we design the physical array to include a larger number of physical wires to assure the yield of enough usable wires to meet our logical requirements.

Using the restoration scheme described in Section 4.3, wires work in pairs. A horizontal OR-term wire provides the programmable computation or programmable interconnect, and a vertical restoration wire provides signal restoration and perhaps inversion. A defect in either wire will result in an unusable pair. Consequently, each logical OR-term or output will yield only when both wires yield. Let P_{wire} be the probability that a wire is not defective. The probability of yielding each OR-term is:

$$P_{or} = (P_{input\_wire} \times P_{restore\_wire}). \tag{3}$$

We can perform an M-choose-N calculation to determine the number of wires we must physically populate (N) to achieve a given number of functional wires

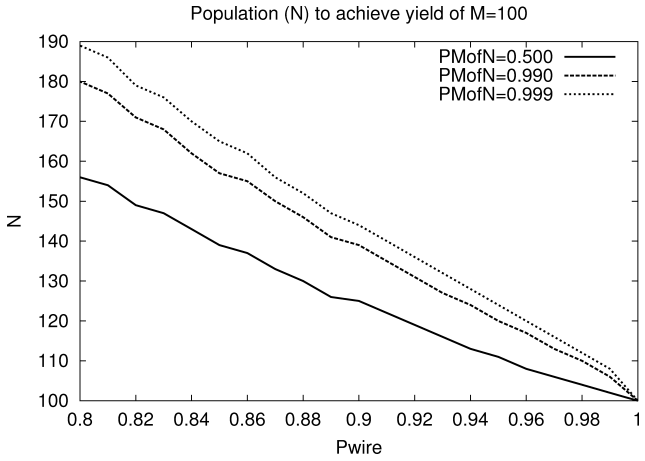


Fig. 26. Physical population (N) of wires to achieve 100 restored OR-terms (M).

(M) in the array. The probability that we will yield exactly i restored OR-terms is:

$$P_{yield}(N,i) = \left(\binom{N}{i} (P_{or})^i (1 - P_{or})^{N-i} \right). \tag{4}$$

That is, there are $\binom{N}{i}$ ways to select i functional OR-terms from N total wires, and the yield probability of each case is: $(P_{or})^i \left(1 - P_{or}\right)^{N-i}$. We yield an ensemble with M items whenever M or more items yield so our system yield is actually the cumulative distribution function:

$$P_{MofN} = \sum_{M \le i \le N} \left(\binom{N}{i} (P_{or})^i \left(1 - P_{or}\right)^{N-i} \right). \tag{5}$$

Given the desired probability for yielding at least M functional OR-terms, P_{MofN}, Equation (5) gives us a way of finding the number of physical wires, N, we must populate to achieve this. For our interconnected nanoPLA blocks, the product terms (P_p) and interconnect wires (W_{seg}) will be the M's in Equation (5), and we will calculate a corresponding pair of raw numbers N to determine the number of physical wires we must place in the fabricated nanoPLA block. We will use P_r to refer to the number of raw product term NWs we need to assemble and W_{segr} to refer to the number of raw interconnect NWs. Figure 26 plots the N required to achieve 50%, 99%, and 99.9% yield rates (P_{MofN}) as a function of $P_{wire} = P_{input\_wire} = P_{restore\_wire}$, when building $M = 100$ wire arrays.

8.2 NW Defect Modeling

A NW could fail to be usable for several reasons.

—The NW may make poor electrical contact to microwires on either end (let P_c be the probability the NW makes a good connection on one end).

—The NW may be broken along its length (let P_j be the probability that there is no break in a NW in a segment of length L_{unit}).

—The NW may be poorly aligned with address region (wired-OR NWs) or restoration region (restoration NW) (let P_{ctrl} be the probability that a the NW is aligned adequately for use).

Consequently, we have a base NW yield which looks like:

$$P_{wire} = (P_c)^2 \times (P_j)^{(L_{wire}/L_{unit})} \times P_{ctrl}. \quad (6)$$

We typically take $P_c = 0.95$ after Huang et al. [2001] and $P_j = 0.9999$ with $L_{unit} = 10$nm after Gudiksen et al. [2001] (see DeHon [2003, 2005]). P_{ctrl} can be calculated from the geometry of the doped regions [DeHon et al. 2003]. P_{wire} is typically around 0.8.

8.3 Net NW Yield Calculation

A detailed calculation for NW population includes both wire defect effects and stochastic population effects. Starting with a raw population number for the NWs in each piece of the array, we:

—calculate the number of nondefective wired-OR wires within our confidence bound (Eqs. (6) and (5));

—calculate the number of those which can be uniquely addressed (Eq. (31));

—calculate the number of net nondefective restored wire pairs within our confidence bound (Eqs. (3), (6), and (5));

—calculate the number of uniquely restored OR terms (Eq. (31)).

These calculations tell us how to get P_r and W_{segr} to achieve a target P_p and W_{seg}.

8.4 Tolerating Nonprogrammable Crosspoints

As we will see (Table III), we typically build PLA crosspoint arrays with around 100 net junctions. If we demanded that all 100 crosspoint junctions on a NW were programmable in order for the NW to yield, we would require an unreasonably high yield rate per crosspoint. That is, assuming a crosspoint is programmable with probability P_{pgm} and a NW has N_{wire} input NWs and hence crosspoint junctions, then the probability that all junctions on a NW are programmable is

$$P_{pgmwire} = (P_{pgm})^{N_{junc}}. \quad (7)$$

To have $P_{pgmwire} \geq 0.5$, we would need $P_{pgm} > 0.993$. However, as we noted in Section 3.3, we expect to see nonprogrammable crosspoint defect rates in the 1–10% range ($0.9 \leq P_{pgm} \leq 0.99$).

As introduced in Naeimi and DeHon [2004], we notice that we can still use a NW with nonprogrammable crosspoints to implement a particular OR-term as long as it has programmable crosspoints where the OR-term needs onprogrammed junctions. Further, since the array has a large number of otherwise interchangeable NWs (e.g., 100), we can search through the array for NWs that can implement each particular OR-term.

For example, if a logic array (AND or OR plane) of a nanoPLA has defective junctions as marked in Figure 27, the OR-term $f = A + B + C + E$ can be

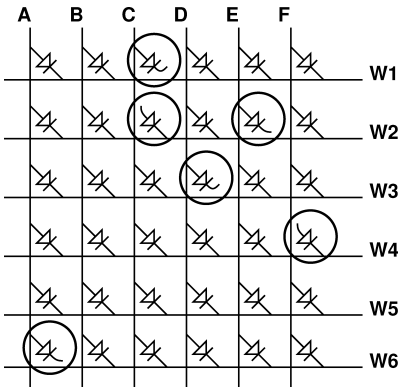


Fig. 27. OR array with defective junctions.

assigned to NW $W3$, despite the fact that it has a defective (nonprogrammable) junction at $(W3, D)$, that is, the OR-term f is compatible with the defect pattern of NW $W3$.

Since the number of programmed junctions needed for a given OR-term is usually small (e.g., 8–20) compared to the number of inputs in an array (e.g., 100), the probability that a NW can support a given OR-term is much larger than the probability that the NW has no junction defects. Assuming C is the fanin to the OR-term and assuming random junction defects, the probability the NW can support the OR-term is

$$P_{support}(C) = (P_{pgm})^C. \tag{8}$$

For example, in a 100 NW array, if $P_{pgm} = 0.95$, $P_{support}(13) \approx 0.51$ while $P_{pgmwire} \approx 0.006$. Further, since we can try multiple NWs in an array to find a compatible match, we only fail to map a NW if there are no compatible NWs in the array.

$$P_{match}(C, N_{wire}) = \left(1 - (1 - P_{support}(C))^{N_{wire}}\right). \tag{9}$$

So, the probability that we fail to find a match for our $C = 13$ OR term in a 100 NW array is $1 - P_{match}(13, 100) \leq 10^{-31}$. Alternately, this means we have a 99% chance of finding a match after checking only 8 NWs ($P_{match}(13, 8) > 0.99$).

Naeimi and DeHon [2004] develop the analysis and mapping strategy in greater detail for tolerating nonprogrammable crosspoints. DeHon and Naeimi [2005] expand the mapping strategy to the interconnected nanoPLAs described in Section 6.3 and show that nonprogrammable defect rates up to 5% can be accommodated with no additional overhead.

9. BOOTSTRAP TESTING

9.1 Discovery

Since addressing and restoration is stochastic, we will need to discover the live addresses and their restoration polarity. Further, since we will have defective NWs, we must identify which NWs are usable and which are not. We

use the restoration columns (see Figures 16 and 21) to help us identify useful addresses. The gate side supply (e.g., top set of lithographic wire contacts in Figures 16 and 21) can be driven to a high value, and we look for voltage on the opposite supply line (e.g., bottom set of lithographic wire contacts in Figure 16; these contacts are marked V_{high} and Gnd but will be controlled independently as described here during discovery). There will be current flow into the bottom supply only if the control associated with the P-type restoration wire can be driven to a sufficiently low voltage. We start by driving all the row lines high using the row precharge path. We then apply a test address and drive the supply (V_{row}) low. If a NW with the test address is present, only that line will now be strongly pulled low. If the associated row line can control one or more wires in the restoration plane, the selected wires will now see a low voltage on their field-effect control regions and enable conduction from the top supply to the bottom supply. By sensing the voltage change on the bottom supply, we can deduce the presence of a restored address. Broken NWs will not be able to effect the bottom supply. NWs with excessively high resistance due to doping variations or poor contacts will not be able to pull the bottom supply contact up quickly enough. We sense the buffering and inverting column supplies separately so we will know whether the line is buffering, inverting, or binate.

We need no more than $O((P_p)^2)$ unique addresses to achieve virtually unique row addressing [DeHon et al. 2003], so the search will require at most $O((P_p)^2)$ such probes. A typical address width for the nanoPLA blocks is $N_a = 14$ which provides 3,432 distinct 7-hot codes, and a typical number of OR terms might be 90 (see Table III). We might thus need to probe 3,432 addresses to find 90 live row wires.

Once we know all the present addresses in an array and the restoration status associated with each address, we can assign logic to logical addresses within each plane based on the required restoration for the output. With logic assigned to live addresses in each row, we can now use the address of the producing and consuming row wires to select and program a single junction in a diode-programmable OR plane.

9.2 Programming

To program any diode crosspoint in the OR planes (e.g., Figure 16), we drive one address into the top address decoder and the second into the bottom. The stochastic restoration performs the corner turn for us so that we effectively place the desired programming voltage differential across a single crosspoint. We set the voltages and control gating on the restoration columns to define which programmable diode array is actually programmed during a programming operation (e.g., in Figure 16, the contacts marked V_{high} and Gnd are the control voltages; V_{pu} and V_{pd} are the signals used for control gating).

9.3 Scaling

Note that each nanoPLA array is addressed separately from its set of microscale wires ($A0$, $A1$, ... and V_{row}, V_{bot}, and V_{top} in Figure 30, illustrated in

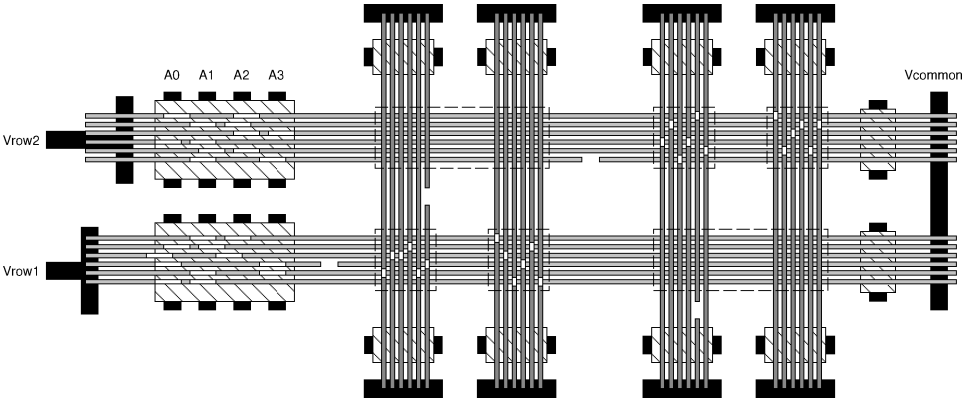


Fig. 28. Illustration nanoPLA with broken NWs.

Section 9.4.2). Consequently, the programming task is localized to each nanoPLA plane, and the work required to program a collection of planes (e.g., Figure 20) only scales linearly with the number of planes.

9.4 Example

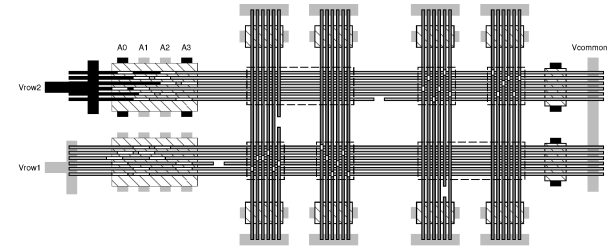
To illustrate the discovery and programming process, this section walks through the steps involved in discovering and programing the illustration PLA in Figure 16 to perform a 2-input XOR. We will assume four of the wires are broken as shown in Figure 28 to better illustrate defect handling.

9.4.1 *Discovering Present Addresses.* First we must discover which addresses are present in each of the two planes. For this example, we have 4 address lines for addressing the OR-term NWs. We use a 2-hot code, meaning there are $\binom{4}{2} = 6$ possible addresses for the OR terms in each plane (1100, 1010, 0110, 1001, 0101, 0011). So, for each plane, we need to test each of the 6 addresses.

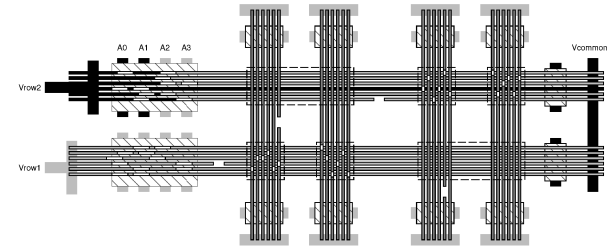
To test for an address' presence, we

(1) drive the right end common supply (V_{common} in Figure 28) to ground, then release it;
(2) drive the address lines ($A0$, $A1$, ...) to the test address;
(3) drive the common row line (i.e. V_{row1} or V_{row2} in Figure 28) to high;
(4) observe the voltage on the common line (V_{common}).

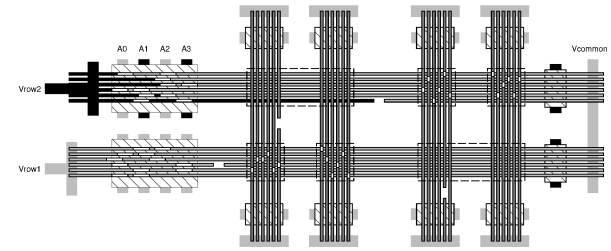
The common line will be raised to high *only if* the test address is present, allowing a complete path between V_{row} and V_{common}. Figure 29(a) shows an attempt to read the address 1001 on the top plane. Since the 1001 wire is not present, this results in no current path from V_{row2} to V_{common}, and V_{common} remains low. Figure 29(b) shows an attempt to read the address 1100 on the top plane. Since the 1100 wire is present and unbroken, this does succeed in raising the voltage on V_{common}. In Figure 29(c), we see that an attempt to read address 0101 does not raise V_{common} since the wire has a break in it. After testing all six addresses, we know that the present and functional addresses in the top plane



(a) 1001 Address (nonpresent)



(b) 1100 Address (present)



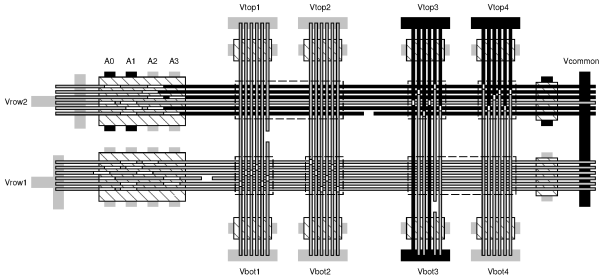
(c) 0101 Address (defective NW)

Fig. 29. Testing the functional address presence. Light grey lines are at ground; black lines driven to high voltage.

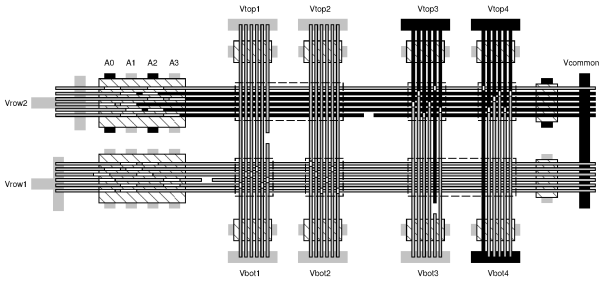
are 1100, 1010, 0110, and 0011. Similar testing for the bottom plane tells us the present and function addresses there are 1100, 1010, 0110, and 0101.

9.4.2 *Discovering Polarities.* Knowing which addresses are present, we can determine which polarities they provide. We drive each good address to a low voltage, while keeping the other wires high and observe if the output is restored. For each address we

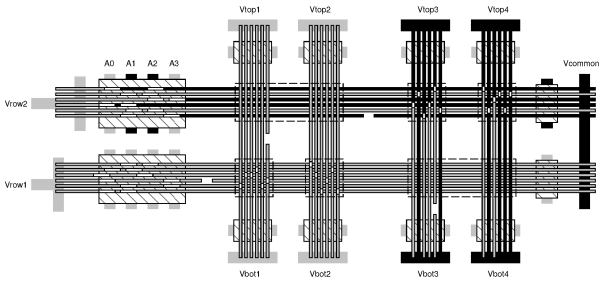
(1) set the gateside supplies on the restoration column (V_{top} in Figure 30) to a low voltage;
(2) drive the opposite supplies (V_{bot} in Figure 30) to a low voltage and release;
(3) use V_{common} and V_{row} to precharge all lines to a high voltage. Drive the addresses ($A0$, $A1$, ...) to all ones during this precharge. Note that we drive high from both sides. This way even wires with a single break are charged to a high voltage;



(a) 1100 NW (inverting)



(b) 1010 NW (noninverting)



(c) 0110 NW (binate)

Fig. 30. Discovering restoration polarities.

(4) release V_{common}, V_{row} and return the addresses to zeros;
(5) drive the intended address on the address lines;
(6) drive V_{row} to a low voltage;
(7) after the row line has had time to discharge, drive the gateside supplies on the appropriate restoration columns (V_{top} in Figure 30) to a high voltage;
(8) observe the voltage on the opposite supply (e.g. V_{bot}) once the restoration line has had a chance to charge.

The restoration wires are P-type NWs. A high voltage across their lightly-doped control region will deplete carries and prevent conduction, while a low voltage will allow conduction. In Steps 3–6, we guarantee that only the addressed row

is low; all other rows are driven to a high value. As a result, we will only see conduction between V_{top} and V_{bot} in a column if the addressed NW controls some NW in that column.

Figure 30(a) shows how we test the 1100 wire. As described earlier, we drive only the 1100 wire to a low voltage. The restoration columns for this wire are bracketed by V_{top3}/V_{bot3} and V_{top4}/V_{bot4}, so we drive V_{top3} and V_{top4} to high voltages and watch the voltage on V_{bot3} and V_{bot4}. Notice that the 1100 NW intersects with two control regions in restoration column 3 and no control regions in restoration column 4. Consequently, V_{bot3} is pulled high while V_{bot4} remains low. By convention (see Figure 16), column 3 is our inverting column. This tells us the 1100 OR term can only be used in its inverting sense.

In Figure 30(b), we show testing of the 1010 wire. The 1010 wire controls restoration wires in both columns 3 and 4. However, the restoration wire in column 3 is broken. Consequently, only the restoration in column 4 is usable. V_{bot4} is pulled high, but V_{bot3} remains low because of the broken column 3 wire. We conclude that we can use the 1010 OR term only in its noninverting sense.

Figure 30(c) shows the result of testing the 0110 wire. There are two 0110 wires in the top plane. Consequently, when we address 0110, we affect both of them. Since the two wires have the same address, we will always select them in tandem. Both OR terms are charged low. It further turns out that there are multiple wires in both column 3 and column 4 controlled by either of the two 0110 OR terms. Both V_{bot3} and V_{bot4} are driven high, telling us that we can use both polarities of the 0110 OR-term (i.e., it is binate). Testing the 0011 OR-term in a similar manner, we learn that it can only be used in the noninverted sense.

We can perform similar tests for the lower OR plane. In this case, the outputs of this OR plane are restored by columns 1 and 2. We drive the high test values into V_{bot1} and V_{bot2} and observe the voltages at V_{top1} and V_{top2}; the role of top and bottom supplies are reversed compared to the top OR plane to match the fact that the position of the restoration array and the succeeding OR array are reversed. After performing the test, we learn the that 1100 and 1010 are binate, 0110 is noninverting, and 0101 is inverting.

9.4.3 *Programming Diode Crosspoints.* Knowing which polarities are available from each of the present addresses, we can program up the intended function.

Figure 31 shows our assignment of known good OR terms to the XOR calculation. We bring in the inputs A and B on the bottom OR terms 1100 and 1010. We need both polarities of A and B, and we know both of these terms are binate. We compute $\overline{A} + B$ on the top OR term 1100 knowing it is inverting, and we compute $A + \overline{B}$ on the top OR term 0110 which is binate so can provide an inverted output. Finally, we use the bottom OR term 0110 to OR together $\overline{\overline{A} + B}$ and $\overline{A + \overline{B}}$ to produce the XOR of A and B.

To program up each crosspoint, we must apply suitable voltages to both the wires in the junction. For example, to make the restored B an input to the $\overline{A} + B$ in the top plane, we set the low address to 1010 to select B's OR term and the high address to 1100 to select the $\overline{A} + B$ OR wire (see Figure 32(a)). Similar to polarity testing above, we precharge the bottom plane wires to high

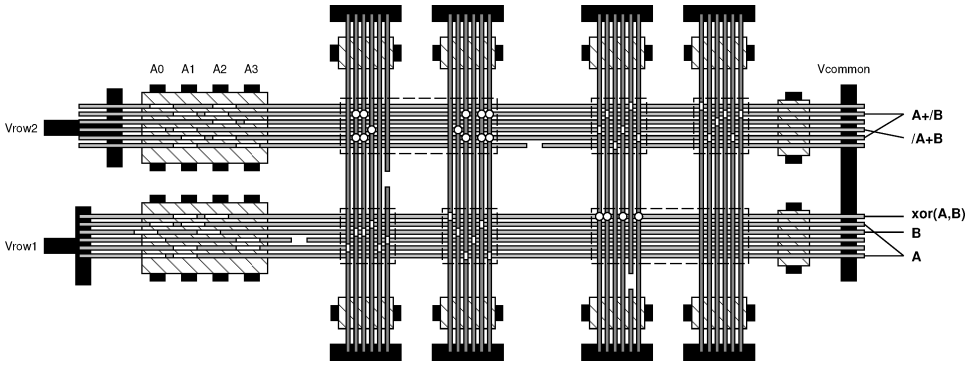
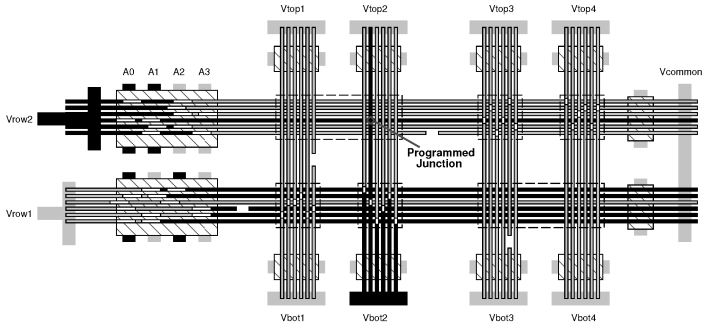
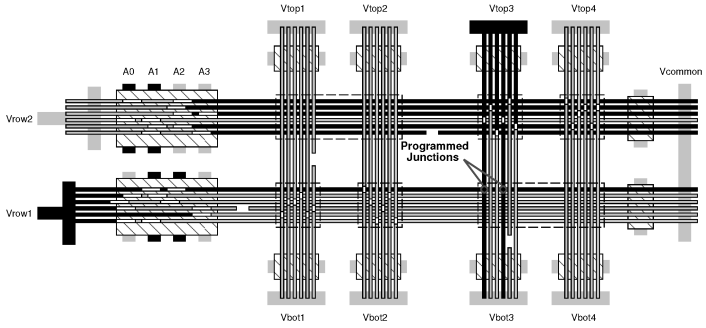


Fig. 31. Assign OR-terms for Computing XOR2.



(a) Program B (bottom 1010) to $\overline{A} + B$ (top 1100) junction



(b) Program $\overline{A} + B$ (top 1100) to XOR (bottom 0110) junction

Fig. 32. Programming diode junctions. Black lines are at suitable programming voltages or are driven high for control; light lines are at nonprogramming voltages.

and then drive V_{row1} to low so that only the 1010 address is low and enables conduction to the OR plane. We drive V_{row2} directly to the low voltage needed for junction programming. We drive V_{bot2} to the high voltage needed for junction programming and leave V_{bot1} at a nominal voltage so that we only program on the noninverting B input. We keep V_{top3}, V_{top4}, V_{bot3}, V_{bot4} at nominal voltages

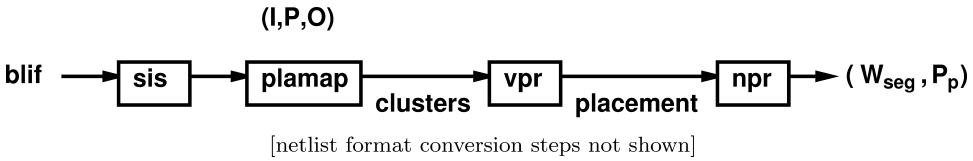


Fig. 33. Design automation flow for nanoPLA mapping.

so that we do not program any junctions in the bottom-right OR plane while we are programming our intended junction in the top-left OR plane.

When we want to program a junction in the bottom-right OR plane, we suitably drive programming voltages on V_{top3} or V_{top4} and keep V_{bot1} and V_{bot2} at nominal voltages. For example, in Figure 32(b), we show programing of the $\overline{\overline{A} + B}$ to XOR junction. Here V_{top3} is placed at the high programming voltage since we want an inverting connection and V_{top4} is held at a nominal voltage along with V_{bot1} and V_{bot2}.

10. CAD MAPPING

To map from standard logic netlists (e.g., BLIF [Sentovich et al. 1992]) to the nanoPLA arrays, we used a combination of conventional and custom tools as shown in Figure 33. SIS [Sentovich et al. 1992] performs standard technology-independent optimizations and decomposes the logic into small fanin nodes for covering. PLAMAP [Chen et al. 2003] covers the logic into (I,P,O) PLA clusters, where

—I is the number of inputs to PLA block;

—P is the number of PTERMS in PLA block;

—O is the number of outputs to PLA block.

These clusters can then be placed with VPR [Betz 1999; Betz and Rose 1997]. While VPR can also route designs, the routing architecture for the nanoPLA array is sufficiently different to merit separate treatment. Consequently, we developed our own nanoPLA router (npr) for routing. Npr reports the physical design parameters W_{seg} and P_p.

The cluster mapping variables to PLAMAP (I,P,O) only account for the logical mapping. I and O will impact W_{seg}; routing along with P will impact P_p.

The nanoPLA router is a global directional wire router using Pathfinder-like history [McMurchie and Ebling 1995]. Since the nanoPLA inputs are effectively a fully populated crossbar, there are no detail routing limitations; inputs can be switched in from just about any channel on which they arrive. Similarly, outputs can be placed on any wire channel by programming the output channel's wired OR appropriately in the OR plane of the PLA block. The route search proceeds through each nanoPLA logic block it encounters, accounting for the extra PTERMS required for such route-through logic so that P_p is measured and minimized.

11. COST MODELS

In this section, we gather the equations for calculating the total area, delay, and energy for nanoPLA blocks.

Table I. Key Technology Parameter Assumptions

Symbol	Description	Value	Reference
ϵ_{SiO_2}	Dielectric constant of SiO_2	3.4×10^{-11}F/m	
ρ_{Si}	Resistivity of heavily doped Si	$10^{-3}\Omega$-cm	
ρ_{NiSi}	Resistivity of NiSi	$10^{-5}\Omega$-cm	[Wu et al. 2004]
R_c	Micro-to-nanoscale contact resistance	10KΩ	[Wu et al. 2004]
$R_{ondiode}$	diode on resistance	100KΩ	[Rueckes et al. 2000]
$R_{offdiode}$	diode off resistance	$>$1GΩ	[Rueckes et al. 2000]
R_{onfet}	FET on resistance	$<$100KΩ	[Cui et al. 2003]
R_{offfet}	FET off resistance	$>$10GΩ	[Cui et al. 2003]

11.1 Technology Parameters

We use the following feature size parameters.

—W_{litho} is the lithographic interconnect pitch. E.g., for the 45nm node, $W_{litho} =$ 105nm [ITRS 2001].

—W_{dnano} is the NW pitch for NWs which are inputs to diodes (i.e., Y route channel segments and restored PTERM outputs).

—W_{fnano} is the NW pitch for NWs which are inputs to field-effect gated NWs; this may be larger than W_{dnano} in order to prevent inputs from activating adjacent gates and to avoid short-channel FET limitations.

Further, Table I summarizes resistance and capacitance parameters assumed for the analysis which follows.

11.2 Area

From Figures 21 and 24, we can see the basic area composition of each tile.

$$\begin{aligned} TW &= (3 + 4(L_{seg} + 1)) \times W_{litho} \\ &\quad +(P_{or} + 4(L_{seg} + 1)W_{segr}) \times W_{dnano} \qquad (10) \\ TH &= 12 \times W_{litho} + (O_r + P_{ir}) \times W_{fnano} \qquad (11) \\ AW &= (N_a + 2) \times W_{litho} \qquad (12) \\ Area &= (AW + TW) \times TH \qquad (13) \end{aligned}$$

P_{or}, P_{ir}, O_r, and W_{segr} (shown in Figure 21) are the raw number of wires we need to populate in the array in order to yield P_p restored inputs, O_p restored outputs, and W_{seg} routing channels (Section 8.3). The two 4's in TW (Tile Width) arise from the fact that we have $L_{seg} + 1$ wire groups on each side of the array ($2\times$) and each of those is composed of a buffer/inverter selective inversion pair ($2\times$). We charge a lithographic spacing for each of these groups since they must be etched for isolation (Section 7) and controlled independently by lithographic-scale wires. The twelve lithographic pitches in TH (Tile Height) account for the 3 lithographic pitches needed on each side of a group of wires for the restoration supply and enable gating. Since we end and begin segmented wire runs between the input and output horizontal wire runs, we pay for these three lithographic pitches four times in the height of a single nanoPLA block: once at the bottom

of the block, twice between the blocks for segments begin/ends, and once at the top of the block (see Figure 21).

N_a is the number of microscale address wires needed to address individual, horizontal nanoscale wires [DeHon et al. 2003]; for the nanoPLA blocks in this work, N_a is typically 14–20. Two extra wire pitches in the Address Width (AW) are the two power supply contacts at either end of a shared address run.

11.3 Timing

A single cycle in the interconnected nanoPLA scheme (Section 6.3, Figure 20) evaluates both the input (AND) plane and the output (OR) plane.

$$T_{cycle} = T_{input\_plane} + T_{output\_plane} \tag{14}$$

The timing of these planes is slightly different. The input plane will be slower because the Y route segment is longer than the internal restored PTERM segment.

The time for a single plane evaluation (see Figure 19) is

$$T_{plane} = T_{precharge} + T_{no} + T_{eval} + T_{ab}. \tag{15}$$

Precharge just needs to discharge NW capacitances through a contact (resistance R_c) and an on field-effect NW junction (resistance R_{onfet}).

$$\begin{aligned} T_{precharge} &= (R_c + R_{onfet})C_{wire} \\ &\quad + 0.5R_{wire} \times C_{wire} \end{aligned} \tag{16}$$

We calculate $T_{precharge}$ for each of the four wires (PTERM input, PTERM restore, final OR/output, and Y-route) case.

$$T_{in\_precharge} = \max(T_{yroute\_precharge}, T_{pin\_precharge}) \tag{17}$$

$$T_{out\_precharge} = \max(T_{prestore\_precharge}, T_{out\_precharge}) \tag{18}$$

Evaluation in the precharge scheme charges a number of rows through on diode junctions (resistance $R_{ondiode}$).

$$\begin{aligned} T_{in\_eval} &= (R_c + 2R_{onfet}) \times (C_{yroute} + f_{yr} \times C_{pin}) \\ &\quad + 0.5R_{yroute} \times (C_{yroute} + f_{yr} \times C_{pin}) \\ &\quad + R_{ondiode} \times (C_{pin}) + 0.5R_{pin} \times (C_{pin}) \end{aligned} \tag{19}$$

$$\begin{aligned} T_{out\_eval} &= (R_c + 2R_{onfet}) \times (C_{prestore} + f_p \times C_{out}) \\ &\quad + 0.5R_{prestore} \times (C_{prestore} + f_p \times C_{out}) \\ &\quad + R_{ondiode} \times (C_{out}) + 0.5R_{out} \times (C_{out}) \end{aligned} \tag{20}$$

f_{yr} is the maximum output fanout in the Y route channel, and f_p is the maximum PTERM fanout following PTERM restoration. We assume $T_{no} = T_{ab} = T_{precharge}$.

11.4 NW Resistance and Capacitance

Wire resistance will be a function of length and materials.

$$R_{wire} = \frac{\rho L_{wire}}{\pi \left(\frac{d}{2}\right)^2} \tag{21}$$

For doped silicon NWs, $\rho = \rho_{Si} = 10^{-3}\Omega$-cm. After NiSi conversion (Section 2.1.5), $\rho = \rho_{NiSi} = 10^{-5}\Omega$-cm [Wu et al. 2004]. As suggested in Section 7, at a lithographic scale, we convert regions which do not need to be semiconducting into NiSi NWs. The restoration NWs only need the small input region to be gateable silicon. The wired-OR NWs may need doped silicon on their input portion.

If the wires are simply doped silicon, the 10μm NWs used in these arrays will have R_{wire} (e.g., R_{yroute}, R_{pin}, $R_{prestore}$, R_{out}) on the order of megaohms. However, with NiSi conversion, the long Y route channel and restored PTERM (vertical) NWs can have their resistance reduced to tens of kiloohms. The diode (horizontal) NWs have long diode regions and will have hundreds of kiloohms of resistance.

NW capacitance comes both from junctions and adjacent NWs.

$$C_{wire} = C_{nwoverlap} + 2 \times C_{nw\text{-}nw} + C_{mwoverlap} \tag{22}$$

$$C_{nwoverlap} = N_{crossednws} \times C_{nanoj} \tag{23}$$

$$C_{mwoverlap} = N_{crossedmws} \times C_{microj} \tag{24}$$

$$C_{nw\text{-}nw} = L_{wire} \times C_{unit\text{-}parallel\text{-}nw} \tag{25}$$

We charge two $C_{nw\text{-}nw}$ capacitances to include both the left and right neighbor of a NW run.

For NW-NW capacitance,

$$C_{unit\text{-}parallel\text{-}nw} = \epsilon \left(\frac{\pi}{\ln\left(1 + \frac{2t_{sh}}{d}\left(1 + \sqrt{1 + \frac{d}{t_{sh}}}\right)\right)} \right). \tag{26}$$

Here t_{sh} is the width of the shell around each conductor used for space such that the total spacing between conductor is $2t_{sh}$. We take $\epsilon = \epsilon_{SiO_2}$.

For junction capacitance, conservatively, treating the NW junctions as over a conductor plane of length d

$$C_{junc} = \epsilon \left(\frac{2\pi \times d}{\ln\left(1 + \frac{2t_{junc}}{d}\left(1 + \sqrt{1 + \frac{d}{t_{junc}}}\right)\right)} \right). \tag{27}$$

For C_{nanoj}, d is the NW diameter which is roughly half the NW pitch (W_{dnano}, W_{fnano}). For C_{microj}, $d = W_{litho}/2$. $t_{junc} = t_{sh}$ for FET junctions, and we conservatively assume $t_{junc} = 1$nm for diode junctions.

Total capacitance of a 10μm NWs is typically around a femtofarad.

11.5 Energy and Power

The nanoPLAs will dissipate active energy, charging and discharging the functional and configured NWs.

$$E_{NW} = \frac{1}{2} C_{wire} V^2 \tag{28}$$

The previous section discussed the calculation of C_{wire}. To tolerate variations in NW doping, it is likely the operating voltage will need to be 0.5 to 1V.

We can discount the raw E_{NW} by the fraction of NWs typically used in a routing channel or a logic array, F. This tends to be 70-80% with the current tools (Section 10) and designs (Section 12.1). Using the selective inversion scheme, we will typically be driving both polarities of most signals guaranteeing close to a 50% activity factor, A.

Assuming an operating frequency of f, power for a nanoPLA tile is

$$P_{array} = \sum_{\text{all NWs}} (A \times F \times E_{NW} \times f). \tag{29}$$

Power density is then

$$P_{density} = \frac{P_{array}}{Area}. \tag{30}$$

Here *Area* is the area for the tile as calculated in Equation (13).

12. DESIGN SPACE EXPLORATION AND ANALYSIS

Technology developments suggest we can build and assemble 10nm pitch NWs with crosspoints at every NW-NW crossing. To use these, we must pay for lithographic addressing overhead, use regular architectures, and tolerate defects. To understand the net benefits, we analyze the characteristics of composite designs in this section. Section 12.1 maps conventional FPGA benchmarks to NW logic with $W_{litho} = 105$nm, $W_{fnano} = W_{dnano} = 10$nm. Section 12.2 then looks at the sensitivity of these characteristics to technology parameters and assumptions.

12.1 Mapped Logic Density

To assess the density benefits of these sublithographic PLAs, we mapped 19 designs from the Toronto 20 benchmark suite [Betz and Rose 1999b] to various PLAs using the flow described in Section 10.

—Original source was the 4-LUT covered BLIFs for the Toronto 20 benchmark set.

—These were re-optimized using `script.algebraic` in `sis`.

—They were then decomposed with `tech_decomp` (available in the RASP suite version of `sis` [Cong et al. 2004]).

—PLAMAP [Chen et al. 2003] mappings were performed without depth reduction. The parameter set for our exploration was I = {12,14,16,18,20}, O={2,4,6,8}, and P = {24,28,32,36,40,44,48}.

—We set the VPR architecture IO Ratio to 16.

—After routing with `npr`, we extracted P_p and W_{seg} and used these in the area models shown in Sections 8 and 11.

—We report nanoPLA areas in Table III; area listed includes the entire rectangle in which the design is placed and routed.

Table II. 22nm CMOS FPGA Area

Design	4-LUTs	Area ($\times 10^{11}$nm^2)
alu4	1522	1.52
apex2	1878	1.88
apex4	1262	1.26
bigkey	1707	1.71
clma	8382	8.38
des	1591	1.59
diffeq	1497	1.50
dsip	1370	1.37
elliptic	3604	3.60
ex1010	4598	4.60
ex5p	1064	1.06
frisc	3556	3.56
misex3	1397	1.40
pdc	4575	4.58
s298	1931	1.93
s38417	6406	6.41
seq	1750	1.75
spla	3690	3.69
tseng	1047	1.05

Table III depicts the minimum area mappings for $W_{litho} = 105$nm, $W_{fnano} = W_{dnano} = 10$nm, and ideal restoration. Table III also compares this minimum area to lithographic 4-LUT FPGAs at the 22nm node [ITRS 2001]. For the 4-LUT areas in 22nm, we took the known LUT counts and simply multiplied these by 10^8nm^2 (see Table II). Here we make use of the fact that 4-LUT blocks run about 1Mλ^2 [DeHon 1996], with $\lambda \approx 11$nm for the 22nm roadmap node. We round 121×10^6nm^2 to 10^8nm^2 for the estimation used here. Electrical length in Table III is the sum of the Y route channel length ($2 \cdot TH$) and the PTERM input length (TW).

Tables II and III show that routed nanoPLA designs are one to two orders of magnitude smaller than 22nm lithographic FPGAs even after accounting for lithographic addressing overhead, defects, and statistical addressing. The fact that many designs achieve their minimum area point at the extremes of the explored parameter suggests the need to expand the parameter search further to see the full potential density benefits.

12.2 Technology Impact

In the previous section, we looked at a single set of technology parameters. Area will vary with technology and fabrication assumptions. To calibrate ourselves on the impact of these different technology assumptions, we examine in this section variations in three technology features as a function of our physical parameter W_{seg}.

(1) Lithographic pitch (W_{litho}). We evaluate [ITRS 2001]:
   —current, 90nm lithography ($W_{litho} = 210$nm)
   —45nm lithography ($W_{litho} = 105$nm)
   —22nm lithography ($W_{litho} = 50$nm)

Table III. Area Minimizing Design Points (Ideal Restoration, $W_{litho} = 105$nm, $W_{fnano} = 10$nm, $W_{dnano} = 10$nm)

	Parameters					nanoPLA			Electrical
	Map			Physical		array	Area	Area	Length
Design	I	P	O	P_p	W_{seg}	org.	$\times 10^8$nm^2	Ratio	(μm)
alu4	18	44	2	60	8	5×5	4.5	339	11.2
apex2	20	24	8	54	15	14×14	47.2	39	12.5
apex4	12	48	2	62	7	6×6	6.0	208	10.8
bigkey	16	24	8	44	13	11×11	24.7	69	11.3
clma	20	48	8	104	28	23×23	278.6	30	19.3
des	18	28	8	78	25	12×12	59.6	26	16.8
diffeq	16	44	8	86	21	11×11	46.5	32	16.0
dsip	20	24	6	58	18	9×9	23.0	59	13.7
elliptic	18	24	8	78	27	17×17	130.1	27	17.7
ex1010	20	48	4	66	9	9×9	16.0	287	11.9
ex5p	12	32	8	67	18	3×3	2.7	389	14.2
frisc	18	24	8	92	34	18×18	198.5	17	21.2
misex3	18	48	4	64	8	7×7	9.1	153	11.5
pdc	16	48	8	74	13	7×7	12.6	363	12.8
s298	18	48	8	79	15	8×8	18.3	105	13.8
s38417	14	32	8	76	22	23×23	199.3	32	15.9
seq	20	36	8	72	18	9×9	25.2	69	14.4
spla	20	44	8	68	12	5×5	5.8	632	12.1
tseng	16	28	8	78	25	11×11	50.1	20	16.8

(2) W_{dnano}, W_{fnano}. We examine the impact of diode NW pitch being half the FET NW pitch.

(3) Deterministic vs. stochastic construction. We compare three different scenarios for NW feature construction.

12.2.1 *Stochastic vs. Deterministic Construction.* As discussed in Sections 4.2 and 4.3.3, we note that addressing and the restoration array can be manufactured using stochastic population. We also noted that it may be possible to construct the restoration array without using stochastic population (Section 4.3.2). We compute and compare the area required under various assumptions:

(1) stochastic restore, stochastic address,

(2) ideal restore, stochastic address,

(3) ideal restore, ideal address.

Figure 34 compares the area of the array as a function of W_{seg} for the technology point $W_{litho} = 105$nm, $W_{fnano} = W_{dnano} = 10$nm. For larger arrays, we see the stochastic construction of the restoration unit costs us roughly a factor of three in density. This comes from the need to overpopulate both the input (P_{ir}, O_r) and restoration (P_{or},W_{segr}) NWs to yield the desired P_p and W_{seg} unique wires; the larger number of raw wires also makes all the wires longer, reducing NW yield.

We also see that the ideal addressing has only a modest impact on area; in fact, as the final curve in Figure 34 shows, the impact of ideal addressing is

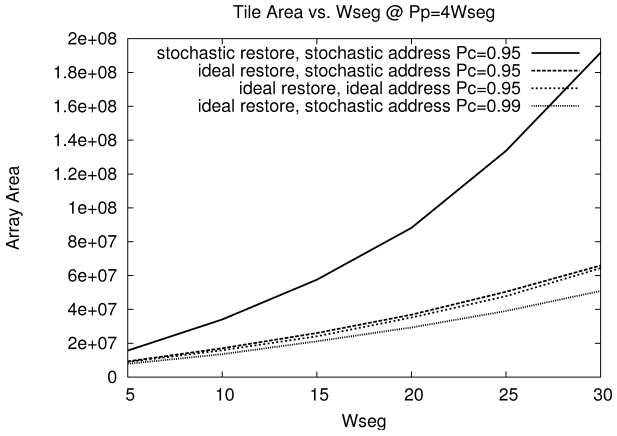


Fig. 34. Impact of stochastic vs. deterministic construction at $W_{litho} = 105$nm, $W_{fnano} = W_{dnano} = 10$nm.

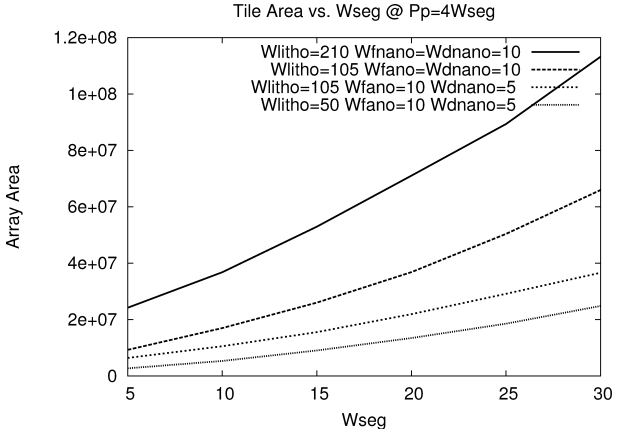
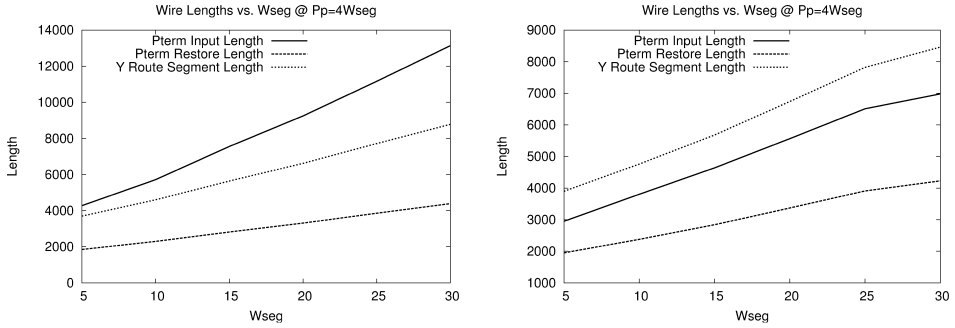


Fig. 35. Impact of technology features for ideal restore, stochastic address construction.

less than the impact of improving the yield in making nanoscale-to-microscale contacts (P_c). For addressing, it only takes a few additional address lines to get mostly unique addressing. The overhead area added by stochastic addressing is only these extra address lines and a small percentage of overpopulation in P_{ir} and O_r in order to accommodate the few redundant addresses which do appear. See DeHon [2004] for a more detailed treatment of uniqueness versus address space size.

12.2.2 *Feature Sizes.* Figure 35 shows the impact of lithographic support technology and reduced diode pitch. Overall, we see almost a factor of 9 in area difference between the 90nm lithography with 10nm NWs and 22nm lithography with 10nm FET NWs and 5nm diode NWs for small arrays where the lithographic overhead dominates. For large arrays, this gap narrows just below a factor of 5. For 45nm lithography, we show both 10nm and 5nm diode NW



$W_{dnano} = 10\text{nm}$ $W_{dnano} = 5\text{nm}$

Fig. 36. NW lengths as a function of W_{seg} for ideal restore, stochastic address case with $W_{litho} = 105\text{nm}$, $W_{fnano} = 10\text{nm}$.

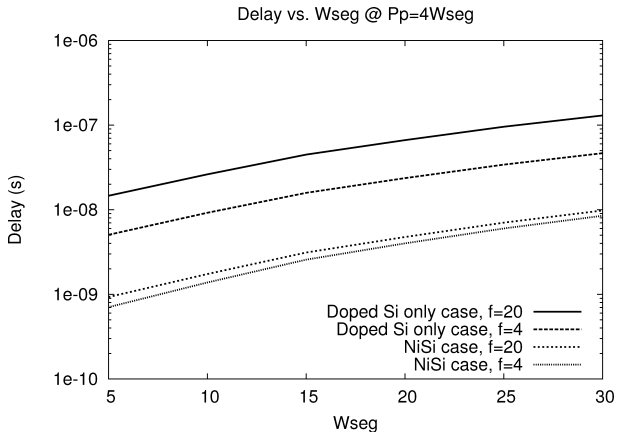


Fig. 37. Delay as a function of W_{seg} for ideal restore, stochastic address case with $W_{litho} = 105\text{nm}$, $W_{fnano} = 10\text{nm}$, $W_{dnano} = 10\text{nm}$.

pitch and see the smaller diode pitch reduces area just over 30% at the low end and over 45% at the high end.

12.2.3 *NW Lengths.* Figure 36 shows the lengths of the key NW features as a function of W_{seg} for the equal FET and diode pitch case and the reduced diode pitch case. For small arrays, NW lengths are 2–4μm due to the lithographic spacing required to supply and separate array features. For larger arrays, with $L_{seg} = 2$, the Y Route segment is around 10μm long; we currently expect to reliably yield assembled NWs around 10–20μm long.

12.2.4 *Delay.* Assuming on-diode resistance ($R_{ondiode}$) of 100KΩ, delay is a strong function of NW resistance and capacitance which, in turn, is a function of NW length. Figure 37 shows the total cycle delay under various fanout ($f_{yr} = f_p = f$) and technology assumptions. The graphs show that the NiSi conversion reduces delay by an order of magnitude. With NiSi and low fanout, cycle delays under a nanosecond may be feasible. Reducing W_{dnano} may further reduce cycle time.

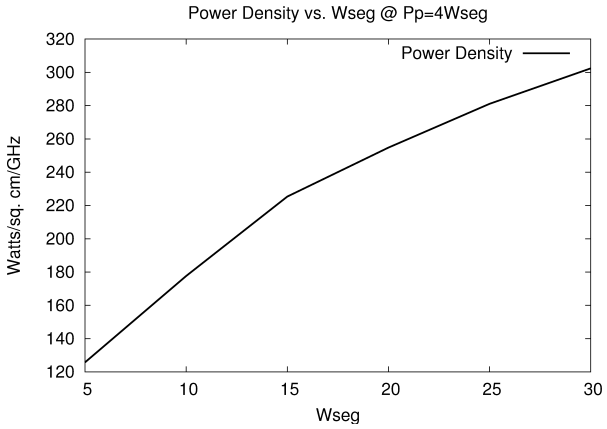


Fig. 38. Power density as a function of W_{seg} for ideal restore, stochastic address case with $W_{litho} =$ 105nm, $W_{fnano} =$ 10nm, $W_{dnano} =$ 10nm.

12.3 Power Density

Figure 38 shows the power density associated with interconnected nanoPLAs suggesting the designs may dissipate a few hundred Watts per cm^2. In typical designs, compute arrays would be interleaved with memory banks (Section 5) which have much lower power densities. Nonetheless, this suggests power management is as much an issue in these designs as it is in traditional, lithographic designs.

13. VARIATIONS

Several groups have been studying variants of these nanowire-based architectures (see Table IV). Heath et al. [1998] articulated the first vision for constructing defect-tolerant architectures based on molecular switching and bottom-up construction. Luo et al. [2002] elaborated the molecular details and diode-logic structure. Williams and Kuekes [2001] introduced a random particle decoder scheme for addressing individual NWs from lithographic-scale wires. These early designs assumed diode logic was restored and inverted using lithographic-scale CMOS buffers and inverters.

Goldstein and Budiu [2001] described an interconnected set of these chemically-assembled diode-based devices. Goldstein and Rosewater [2002] uses only two-terminal non-restoring devices in the array, but add latches based on resonant-tunneling diodes (RTDs) for clocking and restoration.

Snider et al. [2004] suggests nanoFET-based logic and also tolerates nonprogrammable crosspoint defects by matching logic to the programmability of the device.

Strukov and Likharev [2005] also explore crosspoint-programmable nanowire-based programmable logic. They use lithographic-scale buffers with an angled topology and nanovias so that each long NW can be directly attached to a CMOS-scale buffer.

These designs all share many high-level goals and strategies as described in this article. They suggest a variety of solutions to the individual technical

Table IV. Comparison of NW-Based Logic Designs

Component	HP/UCLA	CMU nanoFabric	SUNY CMOL	This Article
Crosspoint Technology	diode	diode	single-electron transistor	diode
NW	imprint lithography	nanoPore templates	interferometric lithography	catalyst NWs
Litho↔NW	random particles		offset angles	coded NWs
Restoration	CMOS	RTD latch	CMOS	NW FET
References	[Heath et al. 1998; Luo et al. 2002; Williams and Kuekes 2001]	[Goldstein and Budiu 2001; Goldstein and Rosewater 2002]	[Strukov and Likharev 2005]	

components including the crosspoint technologies (Appendix A.2), NW formation (Appendix A.1), lithographic-scale interfacing, and restoration (see Table IV). The wealth of technologies and construction alternatives identified by these and other researchers increases our confidence that there are options to bypass any challenges which may arise realizing any single technique or feature in these designs.

14. RESEARCH ISSUES

While the key building blocks have been demonstrated as previously cited, considerable research and development remains in device synthesis, assembly, integration, and process development. We do not have a complete fundamental understanding of the device physics at these scales. Detailed and broader characterization of devices, junctions, interconnects, and assemblies are necessary to refine our models, better predict system properties, and drive architectural designs and optimization.

Mapping results in Section 12.1 are area- and defect-tolerance driven. For high performance designs, additional techniques, design transformations, and optimizations will be needed, including interconnect pipelining (e.g., Tsu et al. [1999]) and fanout management (e.g., Hoover et al. [1984]).

Section 8 noted that we can tolerate high defect rates when the defects occur before operation. New defects are likely to arise during operation, and additional techniques and mechanisms will be necessary to detect the occurrence of new defects, guard the computation against corruption when they occur, and rapidly reconfigure around the new defects.

Further, we expect these small feature devices will see transient faults during operation. The exact fault rates are unknown at this point, but are certainly expected to be larger than we have traditionally seen in lithographic silicon. This suggests the need for new lightweight techniques and architectures for fault identification and correction.

15. CONCLUSIONS

Bottom-up synthesis techniques can produce single nanometer-scale feature sizes (Section 2). Using decorated NWs (e.g., varying composition at nanometer scales both axially and radially), we get our key nanoscale features built into our NWs (Section 2.1). The NWs can be assembled at tight, nanoscale pitch

into dense arrays, contacted to a reliable, lithographic-scale infrastructure, and individually addressed from the lithographic scale (Section 4.2). The aggregate set of synthesis and assembly techniques appears adequate to build arbitrary logic at the nanoscale even if the only programmable elements are nonrestoring diodes.

Bottom-up self-assembly demands that we build highly regular structures (Section 3.1). These structures can be differentiated stochastically for addressing (Section 4.2) and restoration (Section 4.3). NW field-effect gating provides signal restoration and inversion while keeping signals at the dense, nanoscale pitch. Post-fabrication configuration allows us to define any, deterministic computation on top of the regular array, despite random differentiation and high rates of randomly placed defects (Section 6). When these NWs are assembled into modest-sized interconnected PLA arrays, we estimate one-to-two orders of magnitude higher net density than defect-free lithographic-scale FPGAs built in 22nm CMOS (Section 12). This gives us a path to exploit nanometer-pitch devices, interconnect, and systems without pushing lithography to provide these smallest feature sizes.

APPENDIXES

A. ADDITIONAL TECHNOLOGY OPTIONS

A.1 Wires

There have been numerous techniques proposed for manufacturing nanoscale wires. In this section, we briefly note some of the alternatives to the NW growth and assembly described in Section 2.1:

—NWs can be grown on a lattice mismatched substrate so they naturally grow in a single dimension yielding straight, parallel NWs which are a few nanometers across and spaced several nanometers apart [Chen et al. 2000].

—A vertically grown superlattice structure can be used to create a pattern for etching; timed vertical growth of different materials defines features down to a few nanometers without lithography. The resulting structure can be cut orthogonally, differentially etched to expose one of the materials, and used to transfer an etch mask pattern [Melosh et al. 2003]. Parallel wires with pitches as small as 14nm have been demonstrated [Austin et al. 2004].

—Porous media, such as Alumina can serve as a template for NW growth [Kovtyukhova et al. 2003]. This technique may allow the incorporation of a broader set of materials in the construction of compositionally differentiated NWs.

—Carbon Nanotubes (CNT) can also be grown from seed catalysts with diameters down to roughly 1nm in diameter and microns long. They can be semiconducting, allowing field-effect control, or metallic, perhaps offering superior electrical properties to silicon NWs or even Copper. Currently, we do not know how to selectively synthesize CNTs with particular properties (e.g., metallic, semiconducting), however, techniques are actively being developed to selectively sort CNTs [Krupke et al. 2003]. We do not currently know how

to differentiate CNTs along their length as we do with NWs, and aligning CNTs into straight, parallel arrays is not as well developed as NW alignment.

—Interferometric lithography with frequency multiplication may be able to produce regular features like the parallel array of NWs needed for crossbars [Brueck 2002].

A.2 Crosspoints

In recent years, numerous technologies have been proposed and demonstrated which can serve as programmable crosspoints that fit within a NW-NW crossing. These include.

—Mononitro oligo (phenylene ethynylene) molecules, bridging discontinuous gold islands and gold nanorods, demonstrate hysteretic switching with off/on resistance ratios up to 10^4 at room temperature [Tour et al. 2003]. Metal nanofilament and molecular electronic effects have been hypothesized as mechanisms for the voltage controllable resistances with more recent evidence pointing toward filamentary metal effects.

—Ag-TCNQ NWs can be grown and exhibit switching off/on ratios of 10^4 [Fan et al. 2005]. These NWs incorporate the programmable crosspoint material into the NW, eliminating the need for a separate assembly step to place a material between the crossed NWs forming a crosspoint array.

—Floating Gate devices based on Silicon Nanocrystals may provide switching at these scales, demonstrating off/on ratios of several orders of magnitude [Brault et al. 2005].

—Suspended CNTs can realize a bistable junction with an energy barrier between the two states [Rueckes et al. 2000]. The top tube is held in the far state above the lower conductor by mechanical forces and the distance between conductors is large enough to make tunneling current small (GΩs of resistance). When the tubes come into contact, they are held together via molecular forces and there is little resistance (100KΩ) between the tubes. By charging the tubes to the same or opposite polarities with an applied voltage, electrical charge attraction/repulsion allows the tubes to cross the energy gap between the two stable states, effectively setting or resetting the programming of the connection. Junctions can be directional such that the connected state exhibits PN-diode rectification behavior to support independent addressability of the memory crosspoints.

B. RESTORATION UNIQUENESS CALCULATIONS

For cases like the restoration array where we allow duplication, we can derive a recurrence relationship for calculating the probabilities for all cases of C (number of possible restoration wires), N (total number of restoration NWs populated), and u (number of unique NWs in the array):

$$P_{different}(C, N, u) = \left(\frac{C - (u-1)}{C}\right) \times P_{different}(C, N-1, u-1) + \left(\frac{u}{C}\right) \times P_{different}(C, N-1, u) \tag{31}$$

The recurrence in Equation (31) says we can extend the probability distribution from $N-1$ to N in one of two ways. That is, either

(1) we have $u-1$ things in the $N-1$ case, and we select a new item not in the $u-1$ things already selected,
(2) or we already have u different things in the $N-1$ case, and we extend that by selecting among the u different things we already have.

The base cases are

—$P_{different}(C, 1, 1) = 1$ (if we pick one thing, we get one different thing);
—$P_{different}(C, 1, u \neq 1) = 0$ (if we pick one, we will get exactly one different thing);
—$P_{different}(C, 0, 0) = 1$ (if we pick nothing, we get nothing);
—$P_{different}(C, 0, u > 0) = 0$ (if we pick nothing, we get nothing);
—$P_{different}(C, N, u < 0) = 0$ (we cannot have less than nothing).

This recurrence counts each code once even if it appears multiple times which is what we want if we allow duplications.

REFERENCES

AUSTIN, M. D., GE, H., WU, W., LI, M., YU, Z., WASSERMAN, D., LYON, S. A., AND CHOU, S. Y. 2004. Fabrication of 5 nm linewidth and 14 nm pitch features by nanoimprint lithography. *Applied Physics Letters 84*, 26 (June), 5299–5301.

BETZ, V. 1999. VPR and T-VPack: Versatile packing, placement and routing for FPGAs. Version 4.3. Available at http://www.eecg.toronto.edu/vaughn/vpr/vpr.html.

BETZ, V. AND ROSE, J. 1997. VPR: A new packing, placement, and routing tool for FPGA research. In *Proceedings of the International Conference on Field-Programmable Logic and Applications*, W. Luk, P. Y. K. Cheung, and M. Glesner, Eds. Lecture Notes in Computer Science, vol. 1304. Springer, 213–222.

BETZ, V. AND ROSE, J. 1999a. FPGA routing architecture: Segmentation and buffering to optimize speed and density. In *Proceedings of the International Symposium on Field-Programmable Gate Arrays*. 59–68.

BETZ, V. AND ROSE, J. 1999b. FPGA place-and-route challenge. Available at http://www.eecg.toronto.edu/~vaughn/challenge/challenge.html.

BETZ, V., ROSE, J., AND MARQUARDT, A. 1999. *Architecture and CAD for Deep-Submicron FPGAs*. Kluwer Academic Publishers, Norwell, MA.

BJÖRK, M. T., OHLSSON, B. J., SASS, T., PERSSON, A. I., THELANDER, C., MAGNUSSON, M. H., DEPPER, K., WALLENBERG, L. R., AND SAMUELSON, L. 2002. One-dimensional steeplechase for electrons realized. *Nano Letters 2*, 2 (Feb.), 87–89.

BRAULT, J., SAITOH, M., AND HIRAMOTO, T. 2005. Channel width and length dependence si nanocrystal memories with ultra-nanoscale channel. *IEEE Trans. Nanotech. 4*, 3, 349–354.

BROWN, C. L., JONAS, U., PREECE, J. A., RINGSDORF, H., SEITZ, M., AND STODDART, J. F. 2000. Introduction of [2]catenanes into langmuir films and langmuir-blodgett multilayers. A possible strategy for molecular information storage materials. *Langmuir 16*, 4, 1924–1930.

BROWN, S., KHELLAH, M., AND VRANESIC, Z. 1996. Minimizing fpga interconnect delays. *IEEE Des. Test Comput. 13*, 4, 16–23.

BRUECK, S. R. J. 2002. *International Trends in Applied Optics*. SPIE Press, Bellingham, WA, Chapter 5, 85–109.

CHEN, D., CONG, J., ERCEGOVAC, M., AND HUANG, Z. 2003. Performance-driven mapping for cpld architectures. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 22*, 10 (Oct.), 1424–1431.

CHEN, Y., JUNG, G.-Y., OHLBERG, D. A. A., LI, X., STEWART, D. R., JEPPESEN, J. O., NIELSEN, K. A., STODDART, J. F., AND WILLIAMS, R. S. 2003. Nanoscale molecular-switch crossbar circuits. *Nanotech. 14*, 462–468.

CHEN, Y., OHLBERG, D. A. A., LI, X., STEWART, D. R., WILLIAMS, R. S., JEPPESEN, J. O., NIELSEN, K. A., STODDART, J. F., OLYNICK, D. L., AND ANDERSON, E. 2003. Nanoscale molecular-switch devices fabricated by imprint lithography. *Applied Physics Letters 82*, 10, 1610–1612.

CHEN, Y., OHLBERG, D. A. A., MEDEIROS-RIBEIRO, G., CHANG, Y. A., AND WILLIAMS, R. S. 2000. Self-assembled growth of epitaxial erbium disilicide nanowires on silicon. *Applied Physics Letters 76*, 26, 4004–4006.

COLLIER, C., MATTERSTEIG, G., WONG, E., LUO, Y., BEVERLY, K., SAMPAIO, J., RAYMO, F., STODDART, J., AND HEATH, J. 2000. A [2]catenane-based solid state reconfigurable switch. *Science 289*, 1172–1175.

CONG, J., CHEN, D., DING, E., HUANG, Z., HWANG, Y.-Y., PECK, J., WU, C., AND XU, S. 2004. RASP_SYN release B 2.1: FPGA/CPLD technology mapping and synthesis package. Available at http://ballade.cs.ucla.edu/software_release/rasp/htdocs/.

CUI, Y., DUAN, X., HU, J., AND LIEBER, C. M. 2000. Doping and electrical transport in silicon nanowires. *J. Phys. Chem. B 104*, 22 (June), 5213–5216.

CUI, Y., LAUHON, L. J., GUDIKSEN, M. S., WANG, J., AND LIEBER, C. M. 2001. Diameter-controlled synthesis of single crystal silicon nanowires. *Applied Physics Letters 78*, 15, 2214–2216.

CUI, Y., ZHONG, Z., WANG, D., WANG, W. U., AND LIEBER, C. M. 2003. High performance silicon nanowire field effect transistors. *Nanoletters 3*, 2, 149–152.

DEHON, A. 1996. Reconfigurable architectures for general-purpose computing. AI Tech. rep. 1586 (oct.)., MIT Artificial Intelligence Laboratory, Cambridge, MA.

DEHON A. 2003. Array-based architecture for FET-based, nanoscale electronics. *IEEE Trans. Nanotech. 2*, 1 (March) 23–32.

DEHON, A. 2004. 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. Kluwer Academic Publishers, Boston, MA. Chapter 7, 213–241.

DEHON, A. 2005. Design of programmable interconnect for sublithographic programmable logic arrays. In *Proceedings of the International Symposium on Field-Programmable Gate Arrays*. 127–137.

DEHON, A., GOLDSTEIN, S. C., KUEKES, P. J., AND LINCOLN, P. 2005. Non-photolithographic nanoscale memory density prospects. *IEEE Trans. Nanotech. 4*, 2, 215–228.

DEHON, A., LINCOLN, P., AND SAVAGE, J. 2003. Stochastic assembly of sublithographic nanoscale interfaces. *IEEE Trans. Nanotech. 2*, 3, 165–174.

DEHON, A. AND NAEIMI, H. 2005. Seven strategies for tolerating highly defective fabrication. *IEEE Design Test Comput. 22*, 4 (July–August), 306–315.

DEHON, A. AND WILSON, M. J. 2004. Nanowire-based sublithographic programmable logic arrays. In *Proceedings of the International Symposium on Field-Programmable Gate Arrays*. 123–132.

FAN, Z., MO, X., LOU, C., YAO, Y., WANG, D., CHEN, G., AND LU, J. G. 2005. Structures and electrical properties for ag-tetracyanoquinodimetheane organometallic nanowires. *IEEE Trans. Nanotech. 4*, 2 (March), 238–241.

GOJMAN, B., RACHLIN, E., AND SAVAGE, J. E. 2004. Decoding of stochastically assembled nanoarrays. In *Proceedings of the International Symposium on VLSI*.

GOLDSTEIN, S. C. AND BUDIU, M. 2001. NanoFabrics: Spatial computing using molecular electronics. In *Proceedings of the International Symposium on Computer Architecture*. 178–189.

GOLDSTEIN, S. C. AND ROSEWATER, D. 2002. Digital logic using molecular electronics. In *IEEE ISSCC Digest of Tech. Papers*. 204–205.

GUDIKSEN, M. S., LAUHON, L. J., WANG, J., SMITH, D. C., AND LIEBER, C. M. 2002. Growth of nanowire superlattice structures for nanoscale photonics and electronics. *Nature 415*, 617–620.

GUDIKSEN, M. S., WANG, J., AND LIEBER, C. M. 2001. Synthetic control of the diameter and length of semiconductor nanowires. *J. Phys. Chem. B 105*, 4062–4064.

HEATH, J. R., KUEKES, P. J., SNIDER, G. S., AND WILLIAMS, R. S. 1998. A defect-tolerant computer architecture: Opportunities for nanotechnology. *Science 280*, 5370 (June), 1716–1721.

HOOVER, H. J., KLAWE, M. M., AND PIPPENGER, N. J. 1984. Bounding fan-out in logical networks. *J. ACM 31*, 1 (Jan.), 13–18.

HUANG, Y., DUAN, X., CUI, Y., LAUHON, L., KIM, K., AND LIEBER, C. M. 2001. Logic gates and computation from assembled nanowire building blocks. *Science 294*, 1313–1317.

HUANG, Y., DUAN, X., WEI, Q., AND LIEBER, C. M. 2001. Directed assembly of one-dimensional nanostructures into functional networks. *Science 291*, 630–633.

ITRS. 2001. International technology roadmap for semiconductors. http://public.itrs.net/Files/2001ITRS/.

KOVTYUKHOVA, N. I., MALLOUK, T. E., AND MAYER, T. E. 2003. Templated surface sol-gel synthesis of sio_2 nanotubes and sio_2-insulated metal nanowires. *Advanced Materials 15*, 780–785.

KRUPKE, R., HENNRICH, F., V. LÖHNEYSEN, H., AND KAPPES, M. M. 2003. Separation of metallic from semiconducting single-walled carbon nanotubes. *Science 301*, 344–347.

LAUHON, L. J., GUDIKSEN, M. S., WANG, D., AND LIEBER, C. M. 2002. Epitaxial core-shell and core-multi-shell nanowire heterostructures. *Nature 420*, 57–61.

LAW, M., GOLDBERGER, J., AND YANG, P. 2004. Semiconductor nanowires and nanotubes. *Ann. Rev. Material Sci. 34*, 83–122.

LEMIEUX, G., LEE, E., TOM, M., AND YU, A. 2004. Directional and single-driver wires in fpga interconnect. In *Proceedings of the International Conference on Field-Programmable Technology*. 41–48.

LUO, Y., COLLIER, P., JEPPESEN, J. O., NIELSEN, K. A., DELONNO, E., HO, G., PERKINS, J., TSENG, H.-R., YAMAMOTO, T., STODDART, J. F., AND HEATH, J. R. 2002. Two-dimensional molecular electronics circuits. *ChemPhysChem 3*, 6, 519–525.

MAUNSELL, F. G. 1937. A problem in cartophily. *The Math. Gazette 22*, 328–331.

MCMURCHIE, L. AND EBLING, C. 1995. PathFinder: A negotiation-based performance-driven router for FPGAs. In *Proceedings of the ACM International Symposium on Field-Programmable Gate Arrays*. 111–117.

MEAD, C. AND CONWAY, L. 1980. *Introduction to VLSI Systems*. Addison-Wesley.

MELOSH, N. A., BOUKAI, A., DIANA, F., GERARDOT, B., BADOLATO, A., PETROFF, P. M., AND HEATH, J. R. 2003. Ultrahigh-density nanowire lattices and circuits. *Science 300*, 112–115.

MORALES, A. M. AND LIEBER, C. M. 1998. A laser ablation method for synthesis of crystalline semiconductor nanowires. *Science 279*, 208–211.

NAEIMI, H. AND DEHON, A. 2004. A greedy algorithm for tolerating defective crosspoints in NanoPLA design. In *Proceedings of the IEEE International Conference on Field-Programmable Technology*. 49–56.

RUECKES, T., KIM, K., JOSELEVICH, E., TSENG, G. Y., CHEUNG, C.-L., AND LIEBER, C. M. 2000. Carbon nanotube based nonvolatile random access memory for molecular computing. *Science 289*, 94–97.

SENTOVICH, E. M., SINGH, K. J., LAVAGNO, L., MOON, C., MURGAI, R., SALDANHA, A., SAVOJ, H., STEPHAN, P. R., BRAYTON, R. K., AND SANGIOVANNI-VINCENTELLI, A. 1992. Sis: A system for sequential circuit synthesis. UCB/ERL M92/41 (May) University of California, Berkeley, CA.

SNIDER, G., KUEKES, P., AND WILLIAMS, R. S. 2004. Cmos-like logic in defective, nanoscale crossbars. *Nanotech. 15*, 881–891.

STEWART, D. R., OHLBERG, D. A. A., BECK, P. A., CHEN, Y., WILLIAMS, R. S., JEPPESEN, J. O., NIELSEN, K. A., AND STODDART, J. F. 2004. Molecule-independent electrical switching in pt/organic monolayer/ti devices. *Nanoletters 4*, 1, 133–136.

STRUKOV, D. B. AND LIKHAREV, K. K. 2005. Cmol fpga: a reconfigurable architecture for hybrid digital circuits with two-terminal nanodevices. *Nanotech. 16*, 6 (June), 888–900.

TAN, Y., DAI, X., LI, Y., AND ZHU, D. 2003. Preparation of gold, platinum, palladium and silver nanoparticles by the reduction of their salts with a weak reductant–potassium bitartrate. *J. Material Chem. 13*, 1069–1075.

TOUR, J. M., CHENG, L., NACKASHI, D. P., YAO, Y., FLATT, A. K., ANGELO, S. K. S., MALLOUK, T. E., AND FRANZON, P. D. 2003. Nanocell electronic memories. *J. Amer. Chem. Soc. 125*, 43, 13279–13283.

TSU, W., MACY, K., JOSHI, A., HUANG, R., WALKER, N., TUNG, T., ROWHANI, O., GEORGE, V., WAWRZYNEK, J., AND DEHON, A. 1999. HSRA: High-speed, hierarchical synchronous reconfigurable array. In *Proceedings of the International Symposium on Field-Programmable Gate Arrays*. 125–134.

WESTE, N. H. E. AND HARRIS, D. 2005. *CMOS VLSI Design: A Circuits and Systems Perspective*, 3rd ED. Addison-Wesley.

WHANG, D., JIN, S., AND LIEBER, C. M. 2003a. Nanolithography using hierarchically assembled nanowire masks. *Nanoletters 3*, 7 (July), 951–954.

WHANG, D., JIN, S., WU, Y., AND LIEBER, C. M. 2003b. Large-scale hierarchical organization of nanowire arrays for integrated nanosystems. *Nanoletters 3*, 9 (Sept.), 1255–1259.

WILLIAMS, S. AND KUEKES, P. 2001. Demultiplexer for a molecular wire crossbar network. United States Patent Number 6,256,767.

WU, W., JUNG, G.-Y., OLYNICK, D., STRAZNICKY, J., LI, Z., LI, X., OHLBERG, D., CHEN, Y., S.-Y.WANG, LIDDLE, J., TONG, W., AND WILLIAMS, R. S. 2005. One-kilobit cross-bar molecular memory circuits at 30-nm half-pitch fabricated by nanoimprint lithography. *Applied Physics A 80*, 1173–1178.

WU, Y., FAN, R., AND YANG, P. 2002. Block-by-block growth of single-crystalline si/sige superlattice nanowires. *Nano Letters 2*, 2 (Feb.), 83–86.

WU, Y., XIANG, J., YANG, C., LU, W., AND LIEBER, C. M. 2004. Single-crystal metallic nanowires and metal/semiconductor nanowire heterostructures. *Nature 430*, 61–64.

WU, Y. AND YANG, P. 2000. Germanium Nanowire Growth via Simple Vapor Transport. *Chem. Materials 12*, 605–607.

ZHENG, B., WU, Y., YANG, P., AND LIU, J. 2002. Synthesis of ultra-long and highly-oriented silicon oxide nanowires from alloy liquid. *Advanced Materials 14*, 122.

Received June 2005; accepted July 2005