INSTITUTE OF PHYSICS PUBLISHING NANOTECHNOLOGY
Nanotechnology **18** (2007) 035204 (11pp) doi:10.1088/0957-4484/18/3/035204

Nano/CMOS architectures using a field-programmable nanowire interconnect

Gregory S Snider and R Stanley Williams

Hewlett Packard Laboratories, 1501 Page Mill Road, Palo Alto, CA 94304-1126, USA

Received 11 July 2006, in final form 29 September 2006
Published 3 January 2007
Online at stacks.iop.org/Nano/18/035204

Abstract
A field-programmable nanowire interconnect (FPNI) enables a family of hybrid nano/CMOS circuit architectures that generalizes the CMOL (CMOS/molecular hybrid) approach proposed by Strukov and Likharev, allowing for simpler fabrication, more conservative process parameters, and greater flexibility in the choice of nanoscale devices. The FPNI improves on a field-programmable gate array (FPGA) architecture by lifting the configuration bit and associated components out of the semiconductor plane and replacing them in the interconnect with nonvolatile switches, which decreases both the area and power consumption of the circuit. This is an example of a more comprehensive strategy for improving the efficiency of existing semiconductor technology: placing a level of intelligence and configurability in the interconnect can have a profound effect on integrated circuit performance, and can be used to significantly extend Moore's law without having to shrink the transistors. Compilation of standard benchmark circuits onto FPNI chip models shows reduced area (8× to 25×), reduced power, slightly lower clock speeds, and high defect tolerance—an FPNI chip with 20% defective junctions and 20% broken nanowires has an effective yield of 75% with no significant slowdown along the critical path, compared to a defect-free chip. Simulations show that the density and power improvements continue as both CMOS and nano fabrication parameters scale down, although the maximum clock rate decreases due to the high resistance of very small (<10 nm) metallic nanowires.

1. Overview

Micro/nano hybrid architectures have been proposed which integrate nanowire crossbars with a CMOS chip (figure 1, right) [6, 25, 26, 18, 20]. Each crossbar 'junction' (formed by one nanowire crossing over another separated by a small distance (figure 1, left)) is generally hypothesized to be an electrically configurable or reconfigurable device, the simplest being an antifuse. Metallic 'pins' on the surface of the chip connect down into CMOS gates and provide contact points for electrically attaching nanowires in the crossbar. The architects of these hybrids must decide: (1) how to split functionality between the CMOS and nano layers; (2) how to interconnect the two layers by appropriate placements of pins and nanowires; and (3) how to deal with the high defect rates and device variability found in the nanowire crossbars.

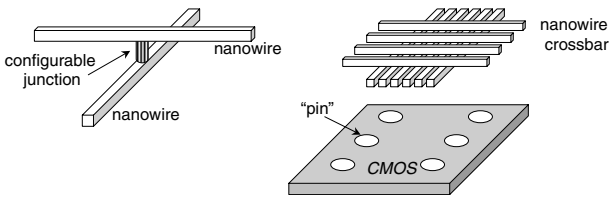

Figure 1. Left: crossing nanowires separated by a molecular layer form 'junctions' that may be electrically configured as electronic devices. Right: nanowire crossbars connected to a CMOS chip via metallic 'pins' on the CMOS surface.

One of the earliest ideas proposed that demultiplexers, implemented in the nanowire crossbars, would allow a small number of pins to control a large number of

nanowires [12, 6, 24]. Although progress has been made with this approach [13, 19, 9, 14], demultiplexers are difficult to build without nonlinear devices. Demultiplexers also present architectural challenges since they are often called upon to do double-duty: to configure selected junctions in the crossbars as well as shuttle data between the CMOS and nano layers.

1.1. CMOL

A more recent approach, CMOL (CMOS/molecular hybrid) [20, 22], proposes moving the most difficult functions necessary for logic—inversion and gain—along with the demultiplexing into the CMOS layer, using the nanowires and selected configured junctions only for wired-OR logic and signal routing. By distributing pins uniformly across the surface of the CMOS chip and structuring the crossbars such that each nanowire connects to exactly one pin, CMOL achieves simplicity in junction configuration and maximal signal bandwidth between the CMOS and nano layers. Preliminary studies have shown that CMOL field-programmable gate arrays (FPGAs) might provide circuits about two orders of magnitude denser than conventional CMOS FPGAs and with similar performance [22].

The cleverness and chief virtues of CMOL are its simplicity, density, and clean separation of configuration and data communication, but it does present some operational and fabrication issues. Because it uses non-complementary, wired-OR logic, keeping static power dissipation within bounds (200 W cm^{-2}) requires careful optimization of the closed-junction resistance, pass-transistor resistance, and supply voltage. An initial estimate places the supply voltage, V_{dd}, at about 0.3 V [22], far lower than any projected by the International Technology Roadmap for Semiconductors (ITRS) through the year 2020 [10]. The configurable junctions are assumed to be extremely nonlinear antifuses in order to implement the wired-OR function, but device variability or insufficient nonlinearity might limit the fan-in to less than desired (a fan-in of 7 was assumed in [22]), decreasing the circuit density. The wired-OR logic also restricts the architecture to nonlinear nanodevices, excluding the use of more linear devices that might be easier to fabricate.

CMOL pins present a fabrication challenge, as shown in the left half of figure 2—the pins are actually 'nanopins,' just a few nanometres in diameter. Half of the nanopins (the blue pins in figure 2) must be taller (by about 8 nm) than the other half (red pins), and they must be surrounded by an insulating layer (shown as grey cladding in the figure). This might be accomplished by projecting the taller pins above the surface of the CMOS layer, which would be flush with the tops of the shorter pins, but the projecting nanopins would severely complicate subsequent circuit processing steps.

Another challenge is registration error or uncertainty in the locations of the nanopins. The protruding blue nanopins of figure 2 are designed to extend through and break the lower level nanowires at regular intervals in order to make contact with the upper level nanowires, but sufficient variation in nanopin location could lead to missing or extra breaks in the nanowires. Although in principle this need not significantly reduce the CMOL connectivity, it does present problems to a compiler attempting to map a circuit onto a CMOL chip—not only must the compiler be aware of defective junctions and broken or shorted wires, it must also have a complete characterization of the actual nanowire lengths and connectivity.

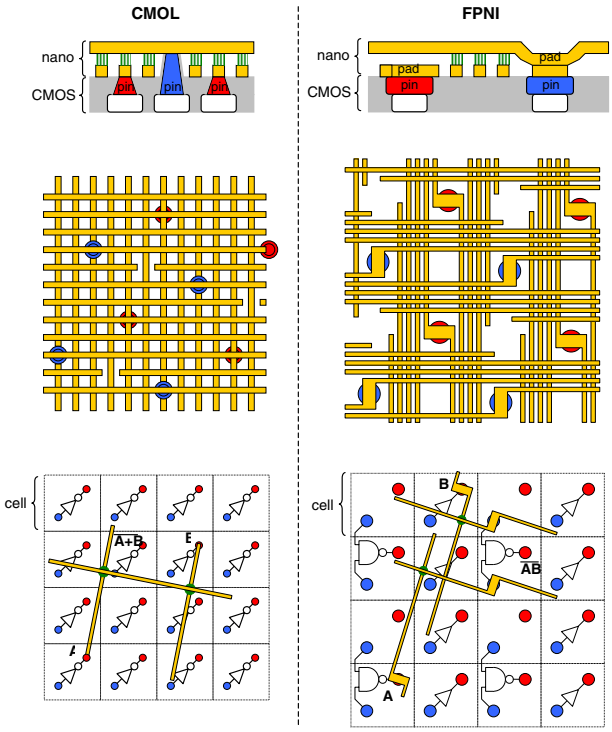

Figure 2. Schematic diagrams of hybrid circuits. The CMOL (left column) places a nanowire crossbar on top of a sea of CMOS inverters. The crossbar is slightly rotated so that each nanowire is electrically connected to one pin extending up from the CMOS layer. Electrically configured, nonlinear antifuses (green, bottom panel) allow wired-OR logic to be implemented, with CMOS supplying gain and inversion. The FPNI (right column) places a sparser crossbar on top of CMOS gates and buffers. Nanowires are also rotated so that each one connects to only one pin, but configured junctions (green, bottom panel) are used only for programmable interconnect, with all logic done in CMOS.

The inverter-based structure of CMOL presents a challenge to routing algorithms, which almost always assume routing networks of non-inverting buffers and switches. This structure requires either the development of new routing algorithms that keep track of signal polarity of nets during routing, or a pairing up of inverters into buffers, reducing the density but allowing conventional algorithms to be used.

One final challenge is the nanowire size (4.5 nm) and pitch (9 nm) chosen for CMOL. These values are far beyond any demonstrated lithographic capabilities, and essentially represent an extrapolation of the ITRS out to the year 2030 [10]. Thus, CMOL as proposed and described is a decade or more away from being realized.

1.2. FPNI

In this paper we present a general hybrid architectural approach named FPNI (field-programmable nanowire interconnect) that trades off some of the speed, density and defect-tolerance of CMOL in exchange for easier fabrication, lower power

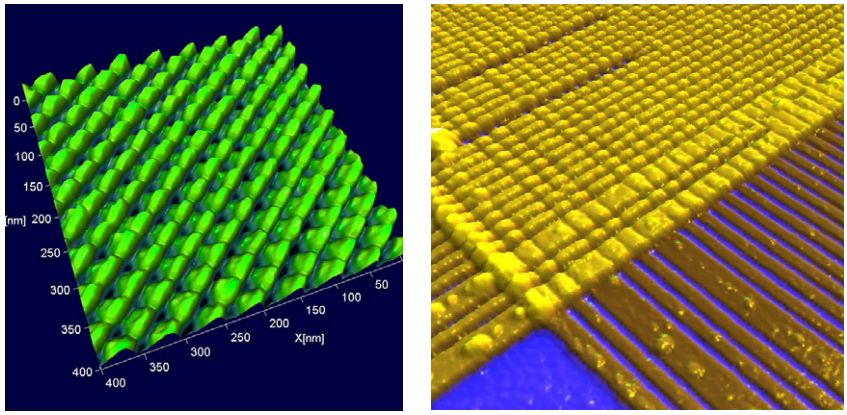

Figure 3. Atomic force microscope topographs of a nearly defect-free region (left) and highly defective region (right) in a 34 nm pitch nanowire crossbar [11].

dissipation, and greater freedom in the selection of nanodevices in the crossbar junctions. The key differences are as follows.

(1) In FPNI architectures, logic is done only in CMOS, routing only in the nanowires (figure 3 shows some nanowire crossbars we have fabricated in our laboratory). This significantly reduces static power dissipation, and allows us to use linear (or approximately linear) antifuses for the nanowire junctions. In addition, the FPNI routing network is buffer-based, not inverter-based, which simplifies the routing.
(2) Alignment of the FPNI nanowire crossbar with the CMOS pins is required, but the alignment accuracy is at the same scale as the CMOS.
(3) The FPNI uses conventional CMOS processes and voltages and provides a planar silicon surface for nanowires; the CMOL requires reduced supply voltage ($V_{dd} = 0.3$ V), reduced signalling voltage swing (40 mV) and 'nanopins' of two different heights on a nonplanar silicon surface [22].

Figure 2 compares the geometry of nanowires, pins and underlying CMOS for the two approaches. The CMOL (left column of figure 2) assumes a sea of inverters regularly connected to pins on the surface of the silicon. The nanowire crossbar on top is rotated slightly to allow each nanowire to contact exactly one pin; the approximately horizontal nanowires connect only to inverter inputs and the vertical only to inverter outputs. Selected junctions (shown in green in the bottom panel) are configured as nonlinear resistors to implement wired-OR logic (in conjunction with a pull-down transistor in the CMOS), with the CMOS inverters providing gain and inversion.

The FPNI (right column of figure 2), on the other hand, assumes a sea of logic gates, buffers and other components in the CMOS layer, and uses the nanowires only for the interconnect. The nanowires include large 'pads' to cover the pins (much larger than the CMOL nanopins), and there is a similar crossbar rotation so that each nanowire connects to only one pin. Selected junctions (green, bottom panel) are configured as resistors to interconnect the underlying logic. As the pin size and alignment error approach zero, the CMOL nanowire layout emerges as a special case of FPNI.

Table 1. FPNI architectural parameters.

Parameter	Description
P_{nano}	The nanowire pitch. This is the centre-to-centre spacing of adjacent, parallel nanowires.
W_{nano}	Mean nanowire width.
W_{pin}	Mean pin diameter.
W_{pinvar}	Plus/minus variation in pin size due to fabrication limitations.
W_{cell}	The height of a CMOS cell. All cells are square, laid out in a regular grid.
W_{align}	Maximum alignment error, in any direction, of nanowires relative to CMOS.
W_{sep}	The minimum planar separation between a CMOS pin and a passing nanowire that must be electrically isolated from it.
R_{closed}	Closed junction resistance.

2. Architecture

2.1. Nanowires

The nanowire and cell geometries are derived from the first seven architectural parameters listed in table 1. We route the nanowires diagonally (with a slight rotation to maximize their length) in order to maximize routability. The two nanowire 'arms' emanating from the central 'pad' are of equal length to minimize worst-case RC (resistance–capacitance) delays. The core nanowire fabric is derived from the parameters of table 1 using a little geometry and trigonometry.

2.2. CMOS

The CMOS layer is divided into an array of square cells, with each cell connected to one input pin (for reading a signal driven from a nanowire) and one output pin (for driving a signal from a gate to a nanowire). A buffer is implemented in a single cell, while logic gates and flipflops require multiple cells.

The logic gate used in this architecture is an n-input NAND/AND gate, implemented in n cells. It computes the logical AND of its inputs and drives the true (AND) and complemented (NAND) forms of the result. The motivation for such a nonstandard gate derives from the equal numbers

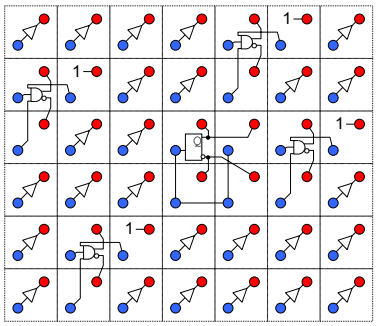

Figure 4. A hypercell consisting of four three-input NAND/AND gates, one flipflop and 26 buffers.

of input and output pins in the cell array. A simple, n-input NAND gate would waste one or more output pins for $n \geqslant 2$; the NAND/AND makes more effective use of the pins and also eliminates the need for inverters. For large n, the true and complemented outputs can be replicated to improve routability and defect tolerance. For all $n > 2$, at least one output pin per gate is used to drive a logical '1' signal, which is connected to unused gate inputs to prevent them from floating.

A flipflop is implemented in four cells. The four input pins are all connected to the D input of the flipflop, allowing the compiler to connect to any of the input pins to reach the D input. Two of the four output pins are driven by the Q output of the flipflop, the other two are driven by the −Q output.

Primary inputs and outputs are implemented with a pair of cells that together handle one input and one output signal. An input signal is brought in from circuitry external to the cell array and driven out in true and complemented form on the two output pins. An output signal is driven through a nanowire to one of the two input pins and from there delivered to the outside world. These I/O cell pairs occupy the outer cell edges of the cell array.

Logic gates, buffers and flipflops are collected together into a rectangular region known as a 'hypercell,' a structure analogous to a configurable logic block (CLB) in a conventional FPGA. An FPNI chip consists of a rectangular array of identical hypercells, surrounded by a periphery of I/O cell pairs. An example is shown in figure 4.

2.3. Configuration

The junction configuration uses the same scheme proposed for use in CMOL [20]. A junction is electrically configured by driving appropriate voltages onto the two nanowires that define it. Configuration lines in the CMOS chip (figure 5), running through each of the cells, provide this capability. During the configuration of a junction, the buffers, gates and flipflops in the cells are disabled; decoders along the edge of the cell array each drive a single configuration line with a programming voltage while grounding the remaining configuration lines. Driving appropriate voltages through the decoders causes two transistors, typically in different cells, to drive two different voltages onto a selected 'output' nanowire and a selected 'input' nanowire. If the two nanowires share a junction, the voltage drop across it can be used to configure the junction's state. For example, a positive voltage drop across an antifuse junction might drive it into a low-impedance state, while a negative voltage drop might return it to a high-impedance state.

Once a circuit has been configured, the configuration lines are driven to turn off the configuration transistors in each cell, and the buffers, gates and flipflops are then enabled to allow programmed circuit operation.

2.4. Fabrication

Since the nanoscale electronics will almost by definition be too small to fabricate by any existing generation of photolithography, the likely approach for fabricating the nanoelectronics will be imprint lithography with alignment capabilities at the scale of (or better than if available, but not necessarily) the alignment required for the CMOS circuitry, in order to achieve registry between the pads and the nanowire connections. It will be possible to begin with a planarized surface on which the pads for both levels of the nanoelectronics are at the same height. A representative process flow is as follows (figure 6).

(1) The first layer of connectors and wires are defined by nanoimprint [11], with the pads of the bottom nanowires aligned over one set of pins on the substrate.

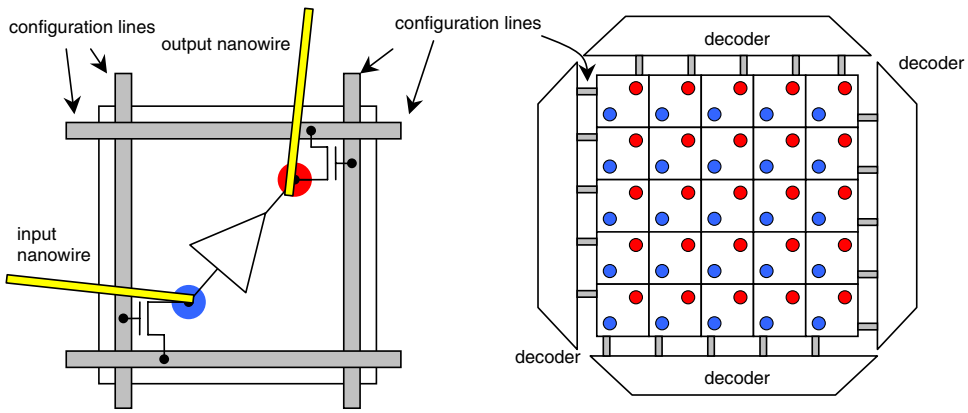

Figure 5. Junction configuration. Each decoder along the edge of the cell array drives a configuration line through each cell. When a pair of configuration lines connected to a transistor (such as the transistor in the upper right of the cell above) is driven with appropriate voltages, the nanowire connected to that transistor will be driven with a voltage. Driving two different nanowires that share a junction with different voltages can be used to change that junction's conductance state, e.g. to close or open a switch connecting the wires.

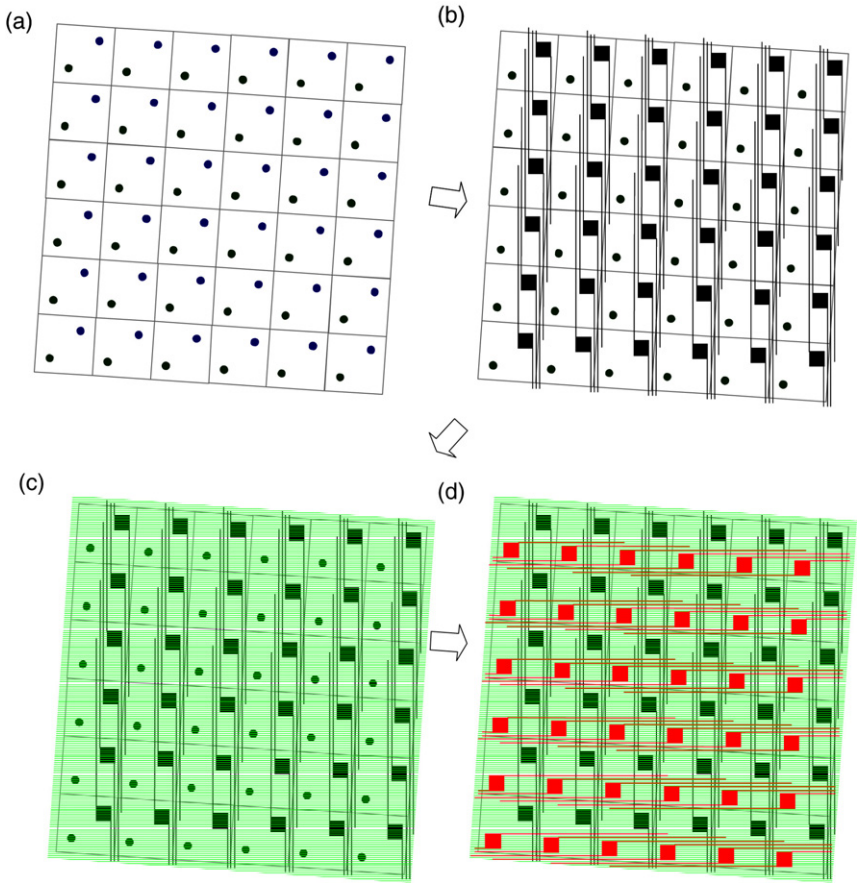

Figure 6. The CMOS cells (a) are rotated slightly relative to the imprinted nanowires. The first layer of vertical nanowires (b) is imprinted so the nano pads (black squares) make contact with the blue CMOS pins. A switching layer is overlaid (c) and the second layer of horizontal nanowires (d) are imprinted so that their nano pads (red squares) make contact with the green nano pins.

(2) The entire surface of the chip is coated with whatever switching layer or layers are necessary, followed by a thin blanket of a protective material such as Ti [2].
(3) Using standard lithography, a mask layer is deposited with openings over the second set of substrate pins, the materials covering these pins are etched away to expose the pins, and the mask layer is removed.
(4) The second layer of nanowires can be defined and deposited in a manner similar to the first, with the pads aligned over the exposed pins.
(5) An etching process is used to remove all the switching and protective materials that are not directly under the upper nanowires, which electrically isolates parallel sets of nanowires, e.g. the formation of the switching junctions is a self-aligned process.

This process flow will require that the upper nanowires 'climb over' the lower nanowires that they cross, as shown in figure 3. While this may lead to breaking of the top nanowires as the wire width shrinks, it has not been a problem for crossbars with 65 nm or greater half-pitch [2, 11], and we have developed a strategy to mitigate this issue for wires narrower than 65 nm. Afterwards, a layer of dielectric material can be deposited over the nanowire layer, which is then planarized to prepare the system for subsequent processing steps. Alternately, a planarization process can be inserted between steps 1 and 2 above, if this is considered to be important and appropriate.

2.5. Electrical model

Calculating the performance and dynamic power of an application compiled onto an FPNI chip requires an electrical model of the nanowires, junctions and CMOS components. For nanowires we need to know the capacitance and resistance per unit length, the closed-junction resistance, and must take into account the geometry of the wires and their interconnections through closed junctions. For the CMOS we need to know the intrinsic gate delay.

Nanowire capacitance per unit length is difficult to estimate because of the non-regular nanowire crossbar structure in the FPNI along with the fact that the top layer nanowires created by nanoimprint are not parallelepipeds, but undulate as they cross over the spaces between nanowires in the layer beneath (figure 3). We begin with Strukov's model for a regular, parallelepiped nanowire crossbar [20], where nanowires within a layer are separated by a distance equal to their width. Given a 3 nm thick switching layer separating the two nanowire layers, a nanowire width of 15 nm, and an insulator between and around all nanowires with a dielectric constant of 3.9 (that of SiO_2), that model predicts a capacitance per length of approximately 2.8 pF cm^{-1}. FPNI crossbars are somewhat sparse, though (see figures 2 and 8), reducing the capacitance between layers. In addition, switching layers that we anticipate using to separate nanowire layers typically

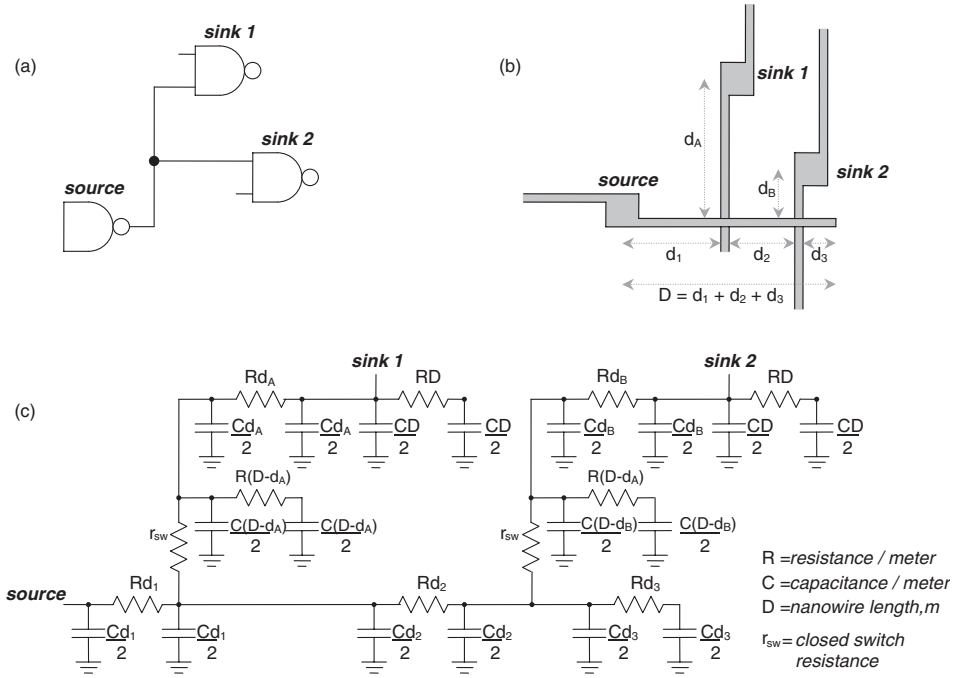

Figure 7. Electrical model of nanowires and junctions. A signal with a fanout of 2 (a) is implemented by electrically closing junction switches between the nanowire driven by the source and the two nanowires connected to the sinks (b). The electrical model used for estimating signal delay (c) uses the physical coordinates of the nanowires and their closed junctions to derive nanowire resistance and capacitance. Delay is estimated using the Elmore delay model; in this example, *delay* (source → sink1)
$\approx Rd_1(4D + d_1/2 + d_2 + d_3)C + 2r_{sw}DC + Rd_A(D + d_A/2)C$.

have a dielectric constant of about 2.5, reducing the interlayer capacitance still further. A passivation layer covering the top nanowire layer need not be SiO_2, allowing us to choose a material that protects the nanowire layers while having a somewhat lower dielectric constant—perhaps 3.5 or so. Nanowire pads add additional capacitance, but since their area is quite small compared to that of the nanowires, we neglect their contribution. From these considerations we have estimated the nanowire capacitance at 2.0 pF cm^{-1}.

Nanowire resistance per unit length depends upon the effective resistivity of the nanowire material. We assume here nanowires of copper (the metal specified in the ITRS roadmap), which allows us to estimate the resistivity for wires down to 15 nm by interpolating the ITRS projections. For example, Cu wires with a line width of 15 nm are projected to have an effective resistivity of approximately 8 $\mu\Omega$ cm, so a square Cu nanowire, 15 nm on a side, would have a resistance of about 355 $\Omega\ \mu m^{-1}$.

Nanowire resistivity, ρ, is difficult to model for very small (<10 nm) wires. Strukov [22] used a common approximation

$$\rho/\rho_0 = 1 + 0.75(1 - p)(\lambda/d) \qquad (1)$$

with p (the fraction of electrons scattered specularly at the surface) assumed to be 0.67, λ (the mean free path) equal to 40 nm, ρ_0 (the bulk resistivity) equal to 2 $\mu\Omega$ cm, and d set to the nanowire width. However, this model is known to underestimate the effective resistivity for small wires [21] and assumes negligible increased resistivity due to scattering at grain boundaries (which is possible for very large grain sizes). For our study we adopted a more conservative model, using Matthiessen's rule to combine the above surface scattering model with the Mayadas–Shatzkes grain boundary scattering model [21], assuming an average grain size equal to the nanowire width (which might require annealing to achieve). We then fitted the resulting model to the ITRS resistivity model, finding a reasonable fit for $p = 0.6$ and a grain boundary reflectivity coefficient of 0.43. Extrapolation yielded an estimated resistivity of about 24 $\mu\Omega$ cm for 4.5 nm Cu nanowires; the uncertainty of this estimate, though, is quite high.

Closed-junction resistance depends on the materials used to build the nanowire crossbar, but we have experimentally observed as a rule of thumb that it is difficult to configure a closed junction to a resistance less than the sum of the resistances of the nanowires from their junction to their respective drivers. Our baseline experiments assume a nanowire width of 15 nm and a maximum nanowire length of about 7.11 μm, yielding a maximum nanowire resistance from pad to tip of about 2.5 kΩ. Based on this and experimental work [15] we have chosen a value of 24 kΩ as a reasonably conservative estimate of obtainable closed-junction resistance for 30 nm pitch nanowires; smaller values have been observed repeatedly in experimental investigations of the closed-switch state for switchable junctions [15]. For higher-resistance, 4.5 nm nanowires, the closed-junction resistance cannot be much less than 120 kΩ.

CMOS gate delay was estimated to be 10 ps by noting the projected n-FET switching time of 0.39 ps for the year 2010 from the ITRS roadmap [10]. Our analysis is not sensitive to this value, though, since circuit timing is strongly dominated by the RC delays of the nanowires.

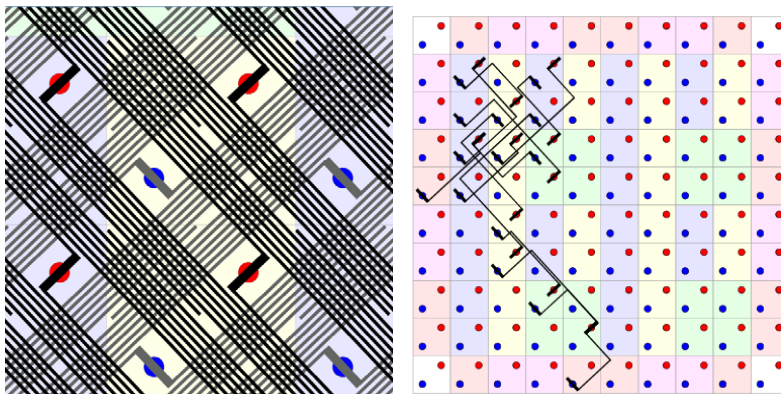

Figure 8. Left: FPNI fabric for FPNI 30 nm parameters. Right: a two-bit counter compiled onto a small FPNI chip. For clarity, only active electrical nanowire connections are shown. The blue cells are two-input NAND/AND gates (two cells per gate), the yellow cells are buffers, and the green cells are flipflops (four cells per flipflop).

Table 2. Experimental parameters for two FPNI architectures.

Parameter	Description	FPNI 9 nm	FPNI 30 nm
P_{nano}	Nanowire pitch	9 nm	30 nm
W_{nano}	Nanowire width	5 nm	15 nm
W_{pin}	Pin diameter	45 nm	90 nm
W_{pinvar}	Pin size variation	20 nm	20 nm
W_{cell}	Cell height	450 nm	840 nm
W_{align}	Alignment error	30 nm	40 nm
W_{sep}	Pin/wire separation	5 nm	15 nm
R_{closed}	Closed junction resistance	120 kΩ	24 kΩ
ρ	Nanowire resistivity	24 $\mu\Omega$ cm	8 $\mu\Omega$ cm
	Nanowire length	5087 nm	7115 nm
	Nanowire resistance	58 kΩ	2.53 kΩ

Evaluating the speed of a circuit mapped onto an FPNI fabric requires a detailed calculation of delay through every component and interconnect, and then extracting the critical timing path by searching for the longest delay through a chain of wires from any flipflop output (or primary input) to any flipflop input (or primary output). Ideally one would use a program like SPICE [23] to do this analysis, but this is practical only for small circuits. Instead we use the simpler Elmore delay model [7, 4, 16] to estimate the delay for each path through nanowires and junctions (see figure 7).

Dynamic power analysis tallies the number of nanowires allocated by our compiler to implement a circuit onto an FPNI target, and computes the dynamic power required to charge and discharge the capacitance of those wires using the formula [5]

$$\text{Dynamic power} = \tfrac{1}{2} A N C V_{\text{dd}}^2 f \qquad (2)$$

where A is the average 'activity' of a signal, N is the number of allocated nanowires, C is the capacitance of a single nanowire, V_{dd} is the supply voltage used by the CMOS, and f is the maximum clock frequency determined by timing analysis. We have chosen an activity of 0.1, following Davis [5], and V_{dd} of 1.0 V from the ITRS roadmap for the year 2010 [10]. Note that we have not computed the static power dissipation in the CMOS gates and flipflops nor the dynamic power needed to drive the CMOS clock tree.

Table 3. Area and delay as function of logic gate inputs for 17 benchmark circuits.

Gate inputs	Total area (μm^2)	Average total critical path delay (ns)
2	657 540	179.3
3	598 730	164.4
4	680 698	169.2
5	647 450	169.9

3. Experiments

Since there is no simple way of analytically comparing the FPNI architecture with CMOL and conventional CMOS FPGAs, we use the time-honoured tradition of modelling and simulation using standard benchmarks. We chose 17 benchmark circuits[1] from the 'FPGA place-and-route challenge' suite [8].

We created two different FPNI architectural models for this study: a conservative model, FPNI 30 nm, that we believe is technologically viable by 2010 for both nanowires and CMOS; and an aggressive model, FPNI 9 nm, that uses the same CMOS and nanowire technology assumptions used by CMOL [22] (most likely aimed at the year 2020) so that we may compare FPNI with CMOL. The CMOS parameters were based on values from the ITRS roadmap for the year 2010 [10] and on discussions with CMOS fabrication engineers. The parameters for both architectures are shown in table 2.

Our initial set of experiments was designed to establish (1) the optimal number of NAND/AND inputs; and (2) the optimal hypercell composition (proportion of gates, buffers and flipflops). To do this we created an ensemble of FPNI chip models meeting the FPNI 30 nm architectural parameters of table 2, but that varied in the number of NAND/AND gate inputs and hypercell composition. We then compiled each of the circuits onto each chip of the ensemble, sizing the FPNI chip in each case to be the smallest hypercell array capable of containing that circuit's logic (although not necessarily its routing). From the results of those experiments, we then fixed

[1] We eliminated the three I/O-limited benchmark circuits from the suite since it was difficult to make meaningful area comparisons with them.

Table 4. Performance comparison: CMOS versus CMOL versus FPNI (data in the CMOS and CMOL columns are from [22]).

	Area (μm^2)				Critical path delay (ns)			Dynamic power (mW)	
Circuit	CMOS	CMOL 9 nm	FPNI 30 nm	FPNI 9 nm	CMOS	FPNI 30 nm	FPNI 9 nm	FPNI 30 nm	FPNI 9 nm
alu4	137 700	1 004	17 513	5 026	5.1	6.53	28.7	0.48	0.061
apex2	166 050	914	18 983	5 448	6	7.10	32.5	0.47	0.059
apex4	414 619	672	13 457	3 862	5.5	5.98	27.1	0.44	0.054
clma	623 194	9 308	78 020	22 391	13.1	19.70	85.5	0.78	0.103
diffeq	100 238	1 194	18 983	5 448	6	6.86	30.6	0.33	0.044
elliptic	213 638	4 581	43 493	12 482	8.6	12.48	56.1	0.50	0.066
ex1010	391 331	3 486	41 252	11 839	9	10.03	44.4	0.84	0.106
ex5p	100 238	829	11 050	3 171	5.1	5.42	23.8	0.37	0.047
frisc	230 850	4 199	43 493	12 482	11.3	14.02	61.8	0.52	0.068
misex3	124 538	1 004	14 750	4 233	5.3	5.52	25.7	0.50	0.061
pdc	369 056	4 979	48 153	13 819	9.6	12.74	58.0	0.90	0.110
s298	166 050	829	20 513	5 887	10.7	12.74	58.5	0.25	0.032
s38417	462 713	9 308	84 220	24 170	7.3	12.94	63.1	0.93	0.114
s38584.1	438 413	9 872	66 329	19 036	4.8	7.80	39.4	1.29	0.153
seq	151 369	1 296	17 513	5 448	5.4	6.55	28.9	0.51	0.066
spla	326 025	2 994	43 493	12 482	7.3	10.92	48.6	0.84	0.108
tseng	78 469	1 194	17 513	5 026	6.3	7.10	29.0	0.25	0.037
Total	4494 491	57 663	598 728	172 250	126.4	164.45	741.6	10.18	1.29
Total relative	1.0	0.013	0.133	0.038	1.0	1.30	5.87	1.0	0.13

our optimal gate size and hypercell composition and continued compiling the benchmarks onto an ensemble to explore

- the performance (power, clock speed, area); and
- the defect tolerance (stuck-open junctions and broken nanowires).

The left side of figure 8 shows a close-up of the nanowires and pins derived from the FPNI 30 nm parameters. The right side shows the result of compiling a small circuit onto a very small FPNI fabric—for clarity, only the active electronic connections are shown. The compiler is described in the appendix.

Table 5. Tolerance of 'stuck-open' junctions.

Stuck-open defect rate	Yield	Average critical path (ns)
0.0	1.00	9.67
0.1	1.00	9.70
0.2	1.00	9.74
0.3	1.00	9.77
0.4	1.00	9.86
0.5	1.00	9.94
0.6	0.95	9.89
0.7	0.94	10.01
0.8	0.89	10.18
0.9	0.43	12.21

3.1. Logic gate and hypercell

Our first set of experiments explored varying the number of inputs to NAND/AND gates from two to five and varying the relative proportions of gates, buffers and flipflops within a hypercell. The goal was to determine the parameters that would minimize the area while maximizing the clock speed. Since we wish to compare our results with conventional FPGAs, which often use a four-input look-up table (LUT) and flipflop as their basic logic element, we needed to determine the computational power of n-input NAND/AND gates relative to four-input LUTs. Technology mapping experiments showed that this relationship is strongly circuit dependent, but suggested an approximation. Our hypercells were thus constructed with a single flipflop combined with either five two-input gates, four three-input gates, three four-input gates or three five-input gates. This left only the number of buffers in a hypercell to be determined.

For each value of n from 2 to 5, we constructed a set of hypercells with the number of n-input gates and flipflops just described, but with varying numbers of buffers that kept the hypercell compactly rectangular. Every benchmark circuit was compiled onto FPNI chips composed of the smallest possible grid for a given hypercell. For each n, we selected the smallest hypercell (the one with the fewest buffers) that all 17 benchmarks could be compiled onto. To compare the results, we summed up the total area and total critical path for all circuits onto chips made of that hypercell. Because our placer and router are nondeterministic, we did this 25 times and averaged the results, shown in table 3.

We were somewhat surprised that neither the area nor the critical path delay was strongly sensitive to the type of logic gate used. But the three-input NAND/AND proved to be the best on both counts, and the smallest three-input hypercell that worked for all circuits was 6×7 cells (shown in figure 4). That hypercell, containing four three-input gates, one flipflop and 26 buffers, was used for the remainder of our experiments.

3.2. Performance (power, speed, area)

To determine the FPNI performance, we compiled each circuit 25 times onto each of the two architectures, FPNI 30 nm and FPNI 9 nm (table 2) and averaged the results for each circuit. The dynamic power calculations assumed $V_{dd} = 1.0$ V, activity $= 0.1$ and circuits clocked at the maximum rate (equal to 1/critical path delay). The results (table 4) show that FPNI

Table 6. Tolerance of 'stuck-open' junctions and broken nanowires.

Stuck-open defect rate	Broken nanowire defect rate	Yield	Average critical path (ns)
0.2	0.0	1.00	9.72
0.2	0.1	0.82	9.00
0.2	0.2	0.75	8.86
0.2	0.3	0.62	8.76
0.2	0.4	0.48	8.81
0.2	0.5	0.35	8.42
0.2	0.6	0.24	8.38
0.2	0.7	0.12	8.54
0.2	0.8	0.04	9.60
0.2	0.9	0.02	8.86

30 nm requires only about one eighth of the area of a CMOS FPGA (with the same 45 nm node semiconductor technology) while running about 22% slower. CMOL 9 nm is far smaller than FPNI 30 nm, but because of our more conservative resistance model, it is not possible to directly compare the CMOL and FPNI performance.

The FPNI 9 nm architecture is only about 4% the size of the CMOS FPGA (though still three times larger than CMOL), but is much slower. The rapidly increasing resistance of shrinking nanowires overwhelms the reduction in total circuit capacitance from the corresponding shortening of nanowires, reducing the clock rate. Note that the reduced power dissipation of the 9 nm architecture is due both to reduced capacitance (due to shorter nanowires) and reduced clock rate. If we normalize the clock rate, the 9 nm architecture dissipates only 57% as much dynamic power per cycle as the 30 nm architecture.

3.3. Defect tolerance

Nanowire antifuse crossbars are typically fabricated with all junctions initially in the 'open' or high-impedance state. The most common defect expected in such crossbars is the 'stuck-open' switch—a high-impedance junction that cannot be configured to a low-impedance state. To study how those defects impact the yield and critical path timing, we compiled each of the 17 benchmarks onto 'FPNI 30 nm' chips, varying the 'stuck-open' junction probability from 0, 0.1, 0.2, . . . , 0.9. To collect sufficient statistics, the compilation was done 100 times for each (circuit, defect rate) pair, for a total of 17 000 compilations. The results are summarized in table 5. Defect rates of 50% have almost no impact on the yield (99.7%) or critical path timing (an average increase of less than 3%). Even defect rates of 80% have respectable yield (88.5%) and degradation in critical path delay[2] (5%).

We expect broken nanowires to be fairly common. To study their impact, we repeated the previous experiments on the 17 benchmarks with the 'stuck-open' defect rate fixed at 0.2, and varied the broken nanowire defect rate (0.0, 0.1, 0.2, . . . , 0.9). Each (circuit, broken nanowire rate) pair was compiled 100 times, for a total of 17 000 compiles, and the results averaged. For this experiment we defined a nanowire to be a single nanowire 'arm' connected to the 'pad' over the pin. Nanowire arms were selected at random with the given rate to be defective, and were broken at a random position along their length with uniform distribution. The experimental results (table 6) show a yield of about 75% when 20% of the nanowires are broken and 20% of the junctions are stuck open. The critical path delay appears to decrease as the nanowire breakage increases—one might expect this because of reduced nanowire capacitance, but we believe in this case it is merely uncovering instabilities in our routing algorithm. Note that the yield shown is pessimistic because our placer is not defect aware; a single gate placed such that one of its outputs has two broken (and extremely short) nanowires extending from its pad would cause the entire compilation to fail.

[2] 'Critical path delay' is the delay of the slowest path limiting the performance of a synchronous circuit; it is equal to the smallest clock period at which the circuit will function correctly.

4. Discussion

FPNI architectures offer a path for continued shrinking of field programmable logic arrays. Simulation shows that the approach is extremely tolerant of the high defect rates likely to be found in nanoscale structures, and that clock rates need not be sacrificed. The eight-fold density increase for FPNI 30 nm compared to a CMOS-only FPGA for the ITRS 45 nm node is equivalent to leaping ahead on the ITRS roadmap by three generations, or nine years to the ITRS 16 nm node (i.e. 2019). According to our simulation results, FPNI can simultaneously improve three performance issues with respect to CMOS-only FPGAs: circuit density, power, and defect tolerance, without requiring improvements in the transistors themselves. In addition, we used fairly conservative estimates for wire and switch resistances—our estimates for the critical path delays would improve significantly if we assumed larger grain sizes in the wires and best-case experimental measurements for ON-state switches.

Variations in nanowire and junction electrical properties will present challenges in modelling ultimate device performance—it is likely that the power and clock rate will be need to be determined empirically for each chip. Device ageing will also need to be addressed: we do not yet know, for example, for how long a configured junction will maintain its state [3]. Perhaps an FPNI chip's configuration would need to be periodically 'refreshed' to continue correct operation.

Compilation presents economic challenges since a manufacturer cannot afford to expend hours of computation on each chip in order to find all defects and then place and route around them. We believe that the extremely high switch redundancy in FPNI, shown in the defect-tolerance experiments, can be exploited to make compilation viable in a reasonable amount of time (in, say, less than 1 min per chip). One approach might involve doing a global place and route on a generic model of a target FPNI chip, being careful to spread the placement out more sparsely than would be needed for a defect-free chip, and then perturbing that mapping as needed during the incremental configuration of the circuit. Algorithms that combine defect characterization with compilation might be most appropriate.

Scaling down both nano and CMOS fabrication dimensions causes dynamic power dissipation to scale down as well, primarily due to reduced wire capacitance from shorter wires. Unfortunately, static power dissipation in CMOS increases as feature sizes decrease, and nanowire resistance increases

rapidly as cross-sectional area shrinks, causing RC delays to reduce performance. Circuit designers at the nanoscale will be forced to make trade-offs between clock rate, area, power, and fabrication cost.

Acknowledgments

We thank Dmitri Strukov for supplying the benchmark circuits, Zhiyong Li and G-Y Jung for the AFM images of nanowire crossbars, Ted Kamins for discussions about CMOS, and Will Tong, Duncan Stewart, and Alex Bratkovski for discussions about nanowire resistivity.

Appendix. FPNI compiler

The FPNI compiler maps logic circuits onto FPNI chips. It takes two files as inputs—a circuit file and an FPNI chip description file—and produces a mapping of that circuit onto the chip. The flow through the compiler is linear.

(1) *Model building* reads a file containing a set of architectural parameters describing an FPNI chip (table 1) and builds models of the chip (e.g., logic gates, nanowire geometry, nanowire electrical properties, chip area) used by the remaining compiler passes.
(2) *Technology mapping* converts a circuit's implementation from n-input NAND gates, inverters and flipflops to n-input NAND/AND gates and flipflops (and no inverters).
(3) *Clustering* groups NAND/AND gates and flipflops together into clusters, with each cluster conforming to the resources available in the target FPNI chip's logic hypercell.
(4) *Placement* places the clusters onto hypercells with the objective of minimizing the total number of nanowires needed to implement the circuit.
(5) *Routing* allocates switches (junctions) to interconnect the circuit's gates, flipflops, primary inputs and primary outputs.
(6) *Timing analysis* (which is also done during routing) analyses delay through a successfully placed and routed circuit to determine the maximum clock speed for the circuit.
(7) *Power analysis* estimates the dynamic power required to drive the allocated nanowires at the maximum clock speed.

Technology mapping begins by replacing each NAND gate with a NAND/AND gate with the same number of inputs, and replacing each primary input with a bipolar primary input that has both true and complemented outputs. Inverters are removed by replacing the output signal driven by the inverter with the inverted form of the inverter's input signal from the appropriate NAND/AND gate or bipolar primary input. Flipflops and primary outputs are left untouched.

Clustering implements Singh's greedy algorithm [17] with a Rent exponent of 0.667. One logic cluster corresponds to a single logic hypercell, so we add additional code to ensure that the number of gates and flipflops packed into a cluster does not exceed the number available in the hypercell. One I/O cluster is allocated for each primary input and each primary output.

Placement uses the simulated annealing algorithm described by Betz [1]. I/O clusters may only be placed on the periphery of the chip in I/O hypercells, and logic clusters may only be placed in the interior logic hypercells. Only one logic cluster may be placed in a logic hypercell, but multiple I/O clusters may be placed in an I/O hypercell if there are sufficient resources to support it (generally the case).

The router implements the timing-driven, directed-search maze algorithm described by Betz [1]. The algorithm requires multiple iterations, with each iteration consisting of routing all nets, recording wire congestion resulting from the attempt, and ripping up the routings. The iterations continue until all nets are successfully routed without overusing any routing resource, up to a maximum of 50 iterations, at which point we declare a circuit to be unrouteable. Wire delays are estimated during routing using a simple, linear-delay model; a more sophisticated delay model (such as the Elmore delay of figure 7) would probably lead to faster critical paths, but at the cost of significant additional computational overhead during routing.

Timing analysis is done at the end of each routing iteration. The Elmore delay is evaluated for every allocated wire and switch, and the critical path is extracted by searching for the longest delay through a chain of wires from any flipflop output (or primary input) to any flipflop input (or primary output). This information is used by the router in the subsequent iteration, helping it to preferentially allocate shorter paths to signals along the critical path.

Power analysis tallies the number of nanowires allocated by the router and computes the dynamic power required to charge and discharge the capacitance of those wires using equation (2).

References

[1] Betz V, Rose J and Marquardt A 1999 *Architecture and CAD for Deep-Submicron FPGAs* (New York: Kluwer Academic)
[2] 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 *Nanotechnology* **14** 462–8
[3] Chen Y, Ohlberg D A A, Li X and Stewart D R 2003 Nanoscale molecular-switch devices fabricated by imprint lithography *Appl. Phys. Lett.* **82** 1610–2
[4] Cong J, He L, Koh C and Madden P 1996 Performance optimization of VLSI interconnect layout, integration *VLSI J.* **21** 1–94
[5] Davis J A, Vivek K De and Meindl J D 1998 A Stochastic wire-length distribution for gigascale integration (GSI)—Part II: applications to clock frequency, power dissipation, and chip size estimation *IEEE Trans. Electron Devices* **45** 590–7
[6] DeHon A 2002 Array-based architecture for molecular electronics *Proc. 1st Workshop on Non-Silicon Computation (Feb. 2002)*
[7] Elmore W 1948 The transient response of damped linear networks with particular regard to wideband amplifiers *J. Appl. Phys.* **19** (January) 55–63
[8] *FPGA Place-and-Route Challenge* Available online at http://www.eecg.toronto.edu/~vaughn/challenge/challenge.html
[9] Gopalakrishnan K *et al* 2005 The micro to nano addressing block (MNAB) *Electron Devices Mtg, 2005. IEDM Technical Digest* (Piscataway, NJ: IEEE International) pp 471–4
[10] International Technology Roadmap for Semiconductors (ITRS) 2005 Available online at http://public.itrs.net/
[11] Jung G Y *et al* 2006 Circuit fabrication at 17 nm half-pitch by nanoimprint lithography *Nano Lett.* **6** 351–4

[12] Kuekes P J and Williams R S 2001 Demultiplexer for a molecular wire crossbar network (MWCN DEMUX) *US Patent Specification* 6,256,767
[13] Kuekes P J, Warren R, Gadiel S and Stanley W R 2005 Defect-tolerant interconnect to nanoelectronic circuits: internally redundant demultiplexers based on error-correcting codes *Nanotechnology* **16** 869–82
[14] Kuekes P, Robinett W, Roth R, Seroussi G, Snider G and Williams R S 2006 Resistor-logic demultiplexers for nanoelectronics based on constant-weight codes *Nanotechnology* **17** 1052–61
[15] Lau C N, Stewart D R, Bockrath M and Williams R S 2005 Scanned probe imaging of nanoscale conducting channels in Pt/alkanoic acid monolayer/Ti devices *Appl. Phys.* A **80** 1373–8
[16] Okamoto T and Cong J 1996 Buffered Steiner tree construction with wire sizing for interconnect layout optimization *Proc. 1996 IEEE/ACM Int. Conf. Comput. Aided Design ICCAD* pp 44–9
[17] Singh A, Parthasarathy G and Marek-Sadowska M 2002 Efficient circuit clustering for area and power reduction in FPGAs *ACM Trans. Des. Autom. Electron. Syst.* **7** 643–63
[18] Snider G, Kuekes P and Williams R S 2004 CMOS-like logic in defective, nanoscale crossbars *Nanotechnology* **15** 881–91
[19] Snider G and Robinett W 2005 Crossbar demultiplexers for nanoelectronics based on n-hot codes *IEEE Trans. Nanotechnol.* **4** 249–54
[20] Strukov D B and Likharev K K 2005 CMOL FPGA: a reconfigurable architecture for hybrid digital circuits with two-terminal nanodevices *Nanotechnology* **16** 888–900
[21] Steinhögl W, Schindler G, Steinlesberger G and Engelhardt M 2002 Size effects in the electrical resistivity of polycrystalline nanowires *Phys. Rev.* B **66** 075414
[22] Strukov D B and Likharev K K 2006 A reconfigurable architecture for hybrid CMOS/nanodevice circuits *FPGA '06 (Monterey, CA, USA, Feb. 2006)*
[23] Vladimirescu A 1994 *The SPICE Book* (New York: Wiley) ISBN 0-471-60926-9
[24] Zhong Z, Wang D, Cui Y, Bockrath M W and Lieber C M 2003 Nanowire crossbar arrays as address decoders for integrated nanosystems *Science* **302** 1377–9
[25] Ziegler M M and Stan M R 2003 CMOS/nano co-design for crossbar-based molecular electronic systems *IEEE Trans. Nanotechnol.* **2** 217–30
[26] Ziegler M M and Stan M R 2003 The CMOS/nano interface from a circuits perspective *ISCAS '03: Proc. 2003 Int. Symp. on Circuits and Systems (May 2003)* vol 4, pp IV-904–7