Nanotechnology **16** (2005) 888–900 doi:10.1088/0957-4484/16/6/045

# CMOL FPGA: a reconfigurable architecture for hybrid digital circuits with two-terminal nanodevices

**Dmitri B Strukov and Konstantin K Likharev**

Stony Brook University, Stony Brook, NY 11794-3800, USA

Received 4 February 2005, in final form 15 March 2005
Published 19 April 2005
Online at stacks.iop.org/Nano/16/888

**Abstract**
This paper describes a digital logic architecture for 'CMOL' hybrid circuits which combine a semiconductor–transistor (CMOS) stack and two levels of parallel nanowires, with molecular-scale nanodevices formed between the nanowires at every crosspoint. This cell-based, field-programmable gate array (FPGA)-like architecture is based on a uniform, reconfigurable CMOL fabric, with four-transistor CMOS cells and two-terminal nanodevices ('latching switches'). The switches play two roles: they provide diode-like $I$–$V$ curves for logic circuit operation, and allow circuit mapping on CMOL fabric and its reconfiguration around defective nanodevices. Monte Carlo simulations of two simple circuits (a 32-bit integer adder and a 64-bit full crossbar switch) have shown that the reconfiguration allows one to increase the circuit yield above 99% at the fraction of bad nanodevices above 20%. Estimates have shown that at the same time the circuits may have extremely high density (approximately 500 times higher than that of the usual CMOS FPGAs with the same design rules), while operating at higher speed at acceptable power consumption.

(Some figures in this article are in colour only in the electronic version)

## 1. Introduction

The simple, uniform structure of semiconductor field-programmable gate arrays (FPGAs) makes them very cost-effective and allows FPGA chips to compete with custom and application-specific integrated circuits for many important applications [1–3]. The most important feature of FPGA circuits, the possibility to reconfigure them after fabrication, makes this approach even more valuable as the leading semiconductor transistor technology, CMOS, approaches the end of scaling [4]. Indeed, as individual transistors are scaled down, their fabrication yield decreases, and the possibility to reconfigure an integrated circuit around bad devices becomes a very attractive option. Most probably, this feature will become absolutely necessary beyond the scaling limit of the purely CMOS technology, when further progress will require its augmentation with novel nanoscale (e.g., molecular [5–8]) devices, because the yield of fabrication and/or self-assembly of these components will hardly ever approach 100%.

Indeed, the application of such classical techniques of providing fault tolerance as von Neumann multiplexing [9, 10] or R-MD redundancy [10] to defects becomes rather inefficient for high defect rates. For example, the recently improved von Neumann multiplexing approach requires a 10-fold redundancy for a bad device fraction $q$ as low as $\sim 10^{-5}$ and a 100-fold redundancy for $q \approx 3 \times 10^{-3}$ [11]. In contrast, reconfigurable computer architectures, which allow one to locate bad components first and then to implement an optimum reconfiguration of the system, may provide high defect tolerance even in these conditions. For example, the Teramac computer [5] can be reconfigured to run a number of real-world tasks even when up to 3% of its resources are defective.

Several reconfigurable architectures for digital nanoelectronic circuits have been proposed (see the recent review [8] and subsequent publications [12, 13]); most of them are based on FPGA-like structures. In FPGAs based on lookup tables (LUTs), all possible values of an $m$-bit Boolean function of $n$ binary operands are kept in $m$ memory arrays, of size $2^n \times 1$ each. (For $m = 1$, and some representative applications, the best resource utilization is achieved with $n$ close to 4 [14], while the Teramac computer [5, 15] uses LUT blocks with $n = 6$ and $m = 2$). The main problem with the idea [5] of application of this approach to hybrid CMOS/nanodevice circuits is that the memory arrays of the LUTs based on realistic nanodevices cannot provide address decoding and output signal sensing (recovery). This means that those functions should be implemented in the CMOS subsystem. The corresponding overhead may be estimated, for example, using our recent results for hybrid memories [16]. In particular, they show that for a memory with $2^6 \times 2$ bits, performing the function of a Teramac's LUT block, and for realistic parameters of CMOS transistors and nanodevices, the area overhead would be above four orders of magnitude, so it would lose the density (and hence performance) competition even to a purely CMOS circuit performing the same function. Increasing the memory array size up to the optimum value calculated in [16] is not a viable option either, because the LUT performance scales (approximately) only as a log of its capacity [17].

The alternative, programmable-logic-array (PLA) FPGAs are based on the fact that an arbitrary Boolean function can be rewritten in the canonical form, i.e., in the two-level logical representation. As a result, it may be implemented as a connection of two crossbar arrays, for example one performing the AND, and the other the OR function [8].

The first problem with the application of this approach to the CMOS/nanodevice hybrids is the same as in the case of LUTs: the optimum size of the PLA crossbars is finite and typically small [18], so that the CMOS overhead is extremely large. Moreover, any PLA logic built with diode-like nanodevices faces an additional problem of high power consumption. In contrast with LUT arrays, where it is possible to have current only through one nanodevice at a time, in PLA arrays the fraction of open devices is of the order of one half [8].

The power consumption may be reduced by using a dynamic logic style, but this approach requires more complex nanodevices. For example, reference [12] describes an interesting dynamic-mode PLA-like structure using several types of molecular-scale devices, most importantly including field-effect transistors (FETs) formed at crosspoints of two nanowires. In such a transistor, one (semiconductor) nanowire would serve as a drain/channel/source structure, while the perpendicular nanowire would play the role of the gate. However, because of an exponential dependence of the threshold voltage on the transistor dimensions, semiconductor FETs with a channel a few nanometres long are irreproducible [19, 20]. (Similar problems are faced by the architecture described in [13], since it is entirely based on crossed-nanowire FET transistors.)

The goal of this paper is to present an alternative reconfigurable architecture for hybrid CMOS/nanodevice circuits, whose structure is similar to the so-called cell-based FPGAs [1, 28]. The architecture has been developed for the recently suggested 'CMOL' variety of the hybrid circuits [20, 21]. As in several earlier proposals [5, 8], nanodevices in CMOL circuits are formed (or self-assembled) at each crosspoint of a 'crossbar' array, consisting of two levels of nanowires (figure 1). However, in order to overcome the CMOS/nanodevice interface problems pertinent to earlier proposals, in CMOL circuits the interface is provided by pins that are distributed all over the circuit area, on the top of the CMOS stack. (The technology necessary for fabrication of tips

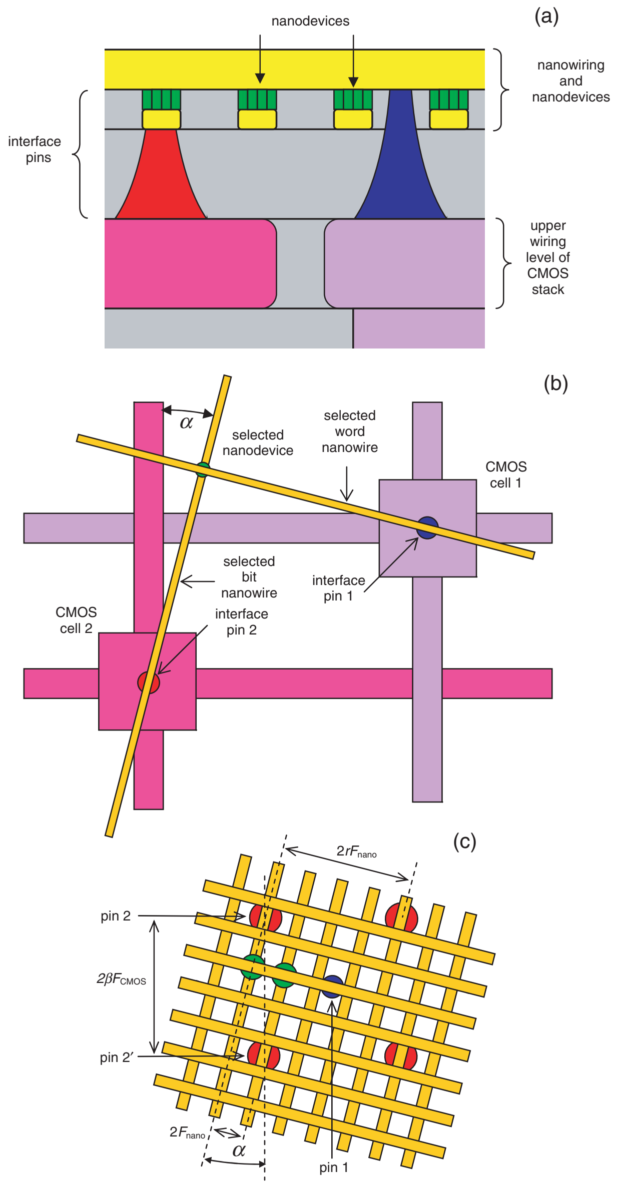

**Figure 1.** Low-level structure of the generic CMOL circuit: (a) schematic side view; (b) the idea of addressing a particular nanodevice, and (c) zoom-in on several adjacent pins to show that any nanodevice may be addressed via the appropriate pin pair (e.g., pins 1 and 2 for the left of the two shown devices, and pins 1 and 2′ for the right device). In panel (b), only the activated CMOS lines and nanowires are shown, while panel (c) shows only two devices. (In reality, similar nanodevices are formed at all nanowire crosspoints.) Also disguised in panel (c) are CMOS cells and wiring. The incline angle $\alpha \ll 1$ and dimensionless parameter $\beta$ satisfy two conditions, $\sin\alpha = F_{nano}/\beta F_{CMOS}$ and $\cos\alpha = rF_{nano}/\beta F_{CMOS}$, where $r$ is an integer.

with nanometre-scale points has been already developed in the context of field-emission arrays [23].) As figure 1(c) shows, pins of each type (reaching to the lower and upper nanowire level) are arranged into a square array with side $2\beta F_{CMOS}$, where $F_{CMOS}$ is the half-pitch of the CMOS subsystem, while $\beta$ is a dimensionless factor larger than 1, that depends on the CMOS cell complexity. The nanowire crossbar is turned by angle $\alpha = \arcsin(F_{nano}/\beta F_{CMOS})$ relative to the CMOS pin array, where $F_{nano}$ is the nanowiring half-pitch. By activating

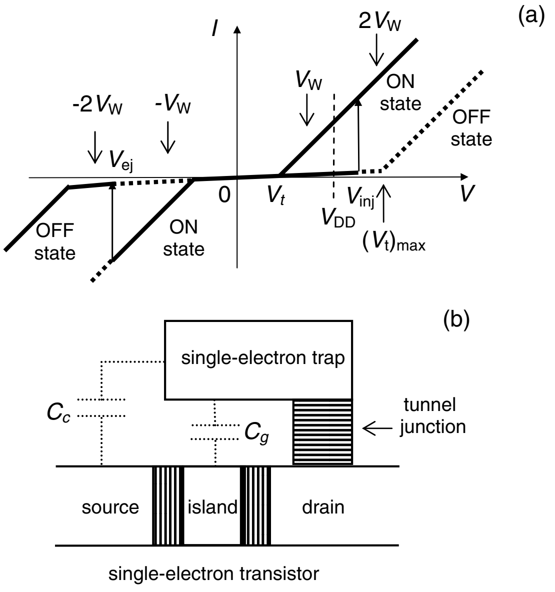

**Figure 2.** Two-terminal latching switch: (a) the $I$–$V$ curve that has been assumed in our analysis (the results are virtually unaffected by the exact shape of the curve) and (b) the single-electron implementation. In the OFF stage of the switch, the single-electron transistor has a high Coulomb blockade threshold $(V_t)_{max} > V_{DD}$. If the source–drain voltage $V$ exceeds a certain value $V_{inj} \sim (V_t)_{max}$, an additional electron is injected into the single-electron trap, and its electric field suppresses the Coulomb blockade threshold to a lower value $V_t < V_{DD}$, enabling current to flow. (The ON state of the latch.) The device may be turned OFF by applying voltage below $V_{ej}$ and thus ejecting the additional electron from the trap island.

two pairs of perpendicular CMOS lines, two pins (and two nanowires they contact) may be connected to CMOS data lines (figure 1(b)). As figure 1(c) illustrates, this approach allows a unique access to any nanodevice, even if $F_{nano} \ll F_{CMOS}$; see [21] for a detailed discussion of this point. If the nanodevices have a sharp current threshold, like the usual diodes, such access allows one to test each of them. Moreover, if the device may be switched between two internal states (figure 2(a)) as, for example, the single-electron latching switches (figure 2(b), [20–22]), each device may be turned into the desirable (ON or OFF) state by applying voltages $\pm V_W$ to the selected nanowires, so that the voltage $V = \pm 2V_W$ applied to the selected nanodevice exceeds the corresponding switching threshold, while half-selected devices (with $V = \pm V_W$) are not disturbed.

We see at least two key advantages of CMOL circuits over other crossbar-type hybrids:

(i) Due to the uniformity of the nanowiring/nanodevice levels of CMOL, they do not need to be precisely aligned with each other and the underlying CMOS stack [21]. This fact allows the use of advanced patterning techniques [24, 25], which lack precise alignment, for nanowire formation.

(ii) CMOL circuits may work with two-terminal nanodevices (e.g., single-electron latching switches) whose fabrication and/or self-assembly is substantially less challenging than that for their three-terminal counterparts. As will be shown below, the relatively low functionality of two-terminal nanodevices may be compensated by (relatively sparse) transistors of the CMOS subsystem.

Recently, we have shown that the CMOL approach allows one to reach high defect tolerance, together with high performance, in digital terabit-scale memories [16] and mixed-signal neuromorphic networks [26]. (For a recent review of these results, see [21].) In this paper, we will show that by using a cell-based FPGA architecture, a similar combination of high performance and defect tolerance may also be reached in Boolean-logic CMOL circuits. So far, we have analysed only two, relatively simple circuits: a 32-bit Kogge–Stone adder and a 64-bit fully connected crossbar. However, these first results are so encouraging that we have decided to publish them right away.

## 2. Architecture

For FPGA applications, it is more convenient (though not absolutely necessary) to turn the nanowire crossbar by almost 45° relative the square array of CMOS cells and interface pins. More exactly, the requirements for the angle $\alpha$ and the dimensionless factor $\beta$ that determines the CMOS cell area $A = (2\beta F_{CMOS})^2$ now take the form:

$$\cos\alpha = \frac{rF_{nano}}{\beta F_{CMOS}}, \qquad \sin\alpha = \frac{(r-1)F_{nano}}{\beta F_{CMOS}}, \qquad (1)$$

where $r$ is a positive integer number[1]. The nanowires are fabricated with small breaks repeated with period $L = 2\beta^2 F_{CMOS}^2/F_{nano}$. With this arrangement, each nanowire segment is connected to one interface pin[2]. As a result, each input or output of a CMOS cell can be connected through a pin–nanowire–nanodevice–nanowire–pin link to each of $M = 2r(r-1) - 1$ other cells located within a square-shaped 'connectivity domain' around the initial cell; see figure 3(a). (For infinitesimal gaps, $M$ would equal $2r(r-1)$, but for a more feasible gap width of the order of $2F_{nano}$, the connectivity domain is by one cell smaller. This is also convenient for analysis, since the resulting connectivity domain is symmetric.)

Each CMOS cell (figure 3(b)) consists of an inverter and two pass transistors that serve two pins (one of each type) serving as the cell input and output, respectively. During the configuration stage, all inverters are disabled by an appropriate choice of global voltages $V_{dd}$ and $V_{gnd}$ (figure 3(b)), and testing and setting of all nanodevices is carried out absolutely similarly to the procedure described in the introduction (figure 1(b); see also [16] and [21]).

When the configuration stage has been completed, the pass transistors are used as pull-down resistors, while the nanodevices set into ON (low-resistive) state are used as pull-up resistors. Together with CMOS inverters, these components

[1] Though our analysis is valid for arbitrary $r$, the best use of CMOL capabilities is achieved at $F_{nano} \ll \beta F_{CMOS}$, when angle $\alpha \approx \pi/4 - 1/r$ is very close to 45°, the integer $r \approx \beta F_{CMOS}/\sqrt{2}F_{nano}$ is large, and the spectrum of possible values of $\beta$, $\beta = (2r^2 - 2r + 1)^{1/2} \times (F_{nano}/F_{CMOS})$, is so dense that choosing one of them that is convenient for the CMOS cell design is not a problem.

[2] The best performance is achieved if the pin contacts the wire fragment in its middle, and our analysis has been carried out with this assumption. This may be assured if the nanowire breaks are provided by features of the same lithographic mask that defines interface pin positions. It is also straightforward to show that at $r \gg 1$, a modest misalignment of the pin and breaks (by $\sim F_{CMOS}$) reduces the circuit performance only by a small factor of the order of $1/\beta \ll 1$.

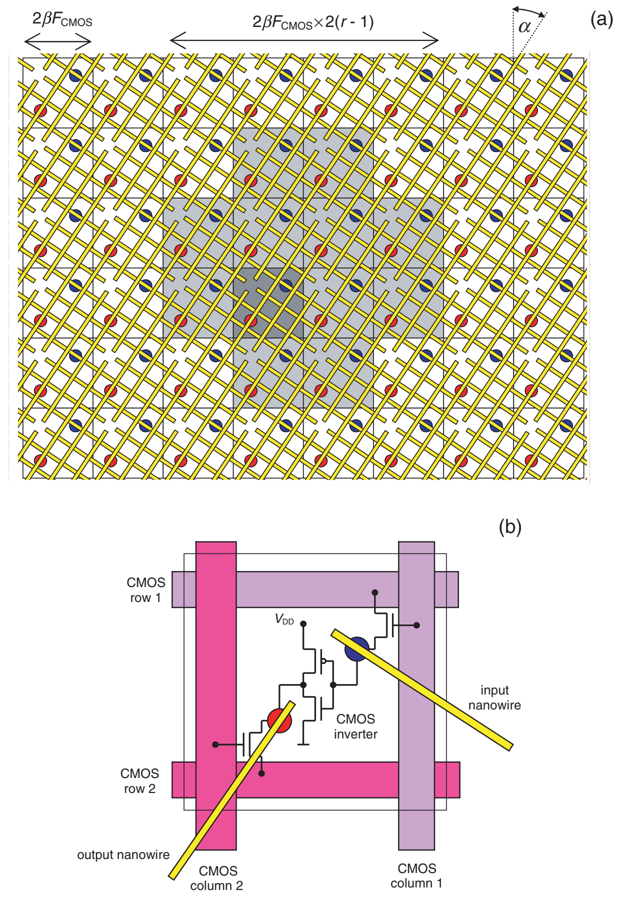

**Figure 3.** CMOL FPGA: (a) the topology and (b) logic cell schematics. In panel (a), $M = 2r(r - 1) - 1$ CMOS cells painted light-grey (in the shown case, $r = 3$, $M = 11$) form the 'connectivity domain' for the input pin of the cell painted dark-grey. (The output connectivity domain has as many cells.) Note that there are $r$ nanowires of one orientation and $(r - 1)$ of the perpendicular orientation per CMOS cell side.

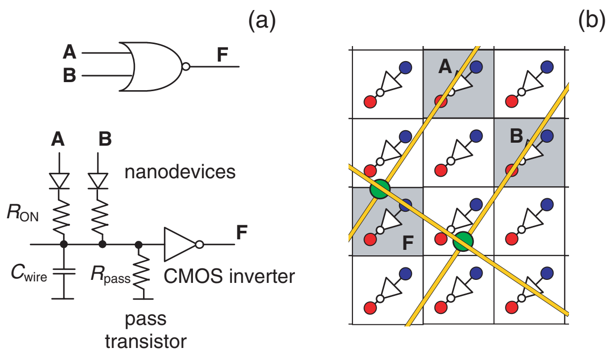

**Figure 4.** CMOL wired-NOR gate: (a) schematics and (b) one of (many possible) configurations.

```
INPUT:
A) design_list (list of gates mapped onto a perfect CMOL fabric, with entries holding
   positions of the initial cell and its input and output gates
B) defect_pattern (locations of defective bits; in our work simulated randomly)
C) Width and height of CMOL array, and parameter r

START:
1: Take the next cell (cur_gate) from design_list
2: If cur_gate is not the end of the list {
3:   If some of the connections from cur_gate to its input and output gates are defective
     for defect_pattern {
4:     Create a list (cur_candidate_list) through the following sequence of steps:
         (i) Based on location of inputs and outputs of the cur_gate, create the list of
       cells ("repair region") where cur_gate could be moved, if no other gates were
       involved
         (ii) In this list eliminate all the cells occupied by other gates which cannot be
       swapped with cur_cell
         (iii) Sort the cur_candidate_list by the connection length penalty $F$ - see Eq. (2)
5:     Take the next cell (candidate_cell) from cur_candidate_list
6:     If candidate_cell is not the end of the list {
7:       If all the connections from candidate_cell to cur_gate's input and output gates
         are defect-free for defect_pattern {
8:         Move cur_gate into candidate_cell, exchanging it with the gate (if any) that
           occupied the cell
9:       } Else { Go to step 7 }
10:    } Else { Exit without success }
11: } Else { Go to step 1 }
12: } Else { Exit with success }
```

**Figure 5.** Pseudo-code of the algorithm used for CMOL FPGA reconfiguration around bad nanodevices. For detailed explanations, see the text.

may be used to form the basic 'wired-NOR' gates (figure 4). For example, if only the two nanodevices shown in figure 4(b) are in the ON state, while all other latches connected to the input nanowire of cell F are in the OFF (high resistance) state, then cell F calculates the NOR function of signals A and B. Clearly, gates with high fan-in and fan-out (broadcast) may be readily formed as well by turning ON the corresponding latching switches. Having these primitives is sufficient to implement any Boolean function, as well as to perform routing, providing that the hardware resources are sufficient. Moreover, our circuits are inherently defect-tolerant, since they have $M = 2r(r-1)-1 \gg 1$ nanodevices per CMOS cell, and only a few of them (on the average, the same as the gate fan-out) are required for circuit operation.

## 3. Reconfiguration

Generally, there may be many different algorithms to reconfigure the CMOL FPGA structure around known defects, including quasi-optimal, exhaustive-search options which are impracticable, because the resources required for their implementation are exponential in circuit size. Here we will describe a very simple linear-time algorithm, whose execution timescales just as $NM$ (where $N$ is the number of gates in the circuit), which nevertheless gives very good results. In this approach, the CMOL FPGA configuration is carried out in two stages: first, mapping the desired circuit on the apparently perfect (defect-free) CMOL fabric, and second, its reconfiguration around defective components.

For our initial analysis of a few simple circuits we have performed the first step manually, though the mapping of more complex circuits will certainly require the development of dedicated CAD tools, quite similar to those already developed for conventional FPGAs; see, for example, [27].

For the second stage we have developed an automatic procedure, so far assuming only one defect type: the absence of nanodevices (latching switches) at certain nanowire crosspoints. (Circuit-wise, such a defect is equivalent to the 'stuck-on-open' fault.) This model has been used in most other works on nanoelectronic circuits (see, for example, [13, 16, 39]) because it is adequate for molecular electronics where such defects result from the failure of molecular self-assembly. In our modelling, the defects have been assumed to be randomly distributed among the crosspoints, with probability $q < 1$.

Our algorithm (formally presented in figure 5) is based on sequential attempts to move each gate from a cell with bad

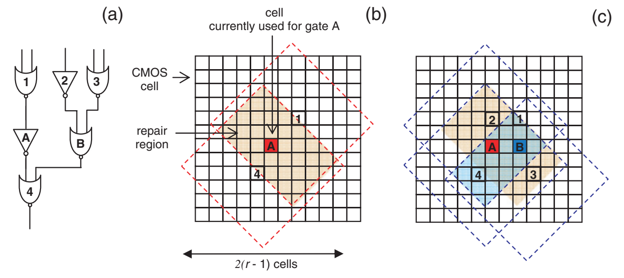

**Figure 6.** Example of a circuit fragment reconfiguration. (a) Circuit whose gate A is to be relocated, because at least one of its connections (with its either input gate 1 or output gate 4) is faulty. (b) The 'repair region' of gate A (painted pink) is the intersect of the connectivity domains (shown by dashed lines) of its input and output gate cells. (c) If a cell of the 'repair region' of A already houses another gate B, the repair domain of the latter cell (painted light blue) is also calculated. Since in this case A is within the repair domain of B, these gates may be swapped, connection quality permitting. For clarity, in this figure $r = 6$; optimal values of $r$ are typically larger (see below).

input or/and output connections to a new cell, while keeping its input and output gates in fixed positions. (Note that according to the CMOL FPGA topology shown in figure 3(a), in each position the cell uses a different set of nanodevices.) At such a move, the gate may be swapped with another one, provided that all connections of the swapped gates can be realized with the CMOL fabric and are not defective.

In order to implement this idea, we first calculate the 'repair region' of the gate, where it could be moved if there were no other cells around; this region is just the overlap of the connectivity domains of all its input and output cells. For example, for the circuit shown in figure 6(a), gate A can be moved to any cell of the repair region painted pink in figure 6(b), which is the intersection of the connectivity domains of its output and input gates 1 and 4. If some cell of the repair region is already occupied by another gate, for example gate B (figure 6(c)), then a similar region is calculated for that gate as well. (For example, in figure 6(c) the repair region for gate B is the intersection of the connectivity domains of gates 2, 3, and 4.) If the original gate lies in that new repair region, then these two gates can be swapped, keeping the circuit functional (provided that all the connections are good).

If there are several cells in the initial gate's repair domain (i.e., several positions this gate may be moved to), higher priority is assigned to positions providing smaller interconnect length. More exactly, for each position we calculate the penalty function

$$F = \sum_i [(\Delta x_i)^2 + (\Delta y_i)^2]^f, \qquad (2)$$

where $x$ and $y$ are the horizontal and vertical coordinates of each cell, and $f$ is an empirically selected exponent. (We have got the best results for $f = 2$.) The summation in equation (2) is over all potential interconnects; if the move requires a cell swap, interconnections of both cells are counted. For example, in figure 6(c) five connections (from gate A to 1 and 4, and from gate B to 2, 3, and 4) give contributions to this sum. (Typically, though not always, this rule gives higher priority to a gate moving into an initially empty cell.)

After the list of all possible moving options has been compiled, they are checked, in the order of increasing penalty $F$, for defective interconnects. The first met option with all good connections is implemented. The case when there are no possible moving options with good connections is considered a reconfiguration failure.

An approximate analysis of this reconfiguration algorithm shows that most reconfiguration failures come from the longest initial connections, corresponding to the very periphery of the cell connectivity domains. This is why we have found that from the point of view of defect tolerance it is beneficial to carry out the initial design for artificially confined connectivity domains. They are similar in shape to that shown in figure 3(a), but have only $M' < M$ cells. In the discussion below, we will mostly quote the linear size scale ('radius') $r'$ of the confined connectivity domain, defined by relation $M' = 2r'(r' - 1) - 1$ (similar to that relating $M$ and $r$).

## 4. First case study: Kogge–Stone adder

### *4.1. Initial mapping*

As the first example, we will consider the CMOL FPGA implementation of an integer, parallel-prefix adder which is one of the key digital logic circuits in digital design; see, for example, [28]. Among such adders, the Kogge–Stone adder [37] has the most regular structure (figures 7(a), (b)) and therefore we could carry out its manual mapping on the CMOL FPGA fabric. First, the 32-bit adder circuitry has been converted into a netlist of fan-in-two NOR gates (figure 7(c)) and then mapped onto a rectangular CMOL block (figure 8), with interleaved inputs A[31:0] and B[31:0] on the top side and outputs S[31:0] and $C_{\text{out}}$ on the bottom side. (For simplicity, $C_{\text{in}}$ is assumed to be always '0'). The mapping procedure was first performed for one bit slice and then repeated for the rest of the circuit, for several values of the connectivity domain radius $r' \leqslant r$. For example, figure 8(a) shows the map for $r' = 10$. (As a reminder, in CMOL hardware each of these straight lines actually consists of two mutually perpendicular nanowires, connected with a nanoscale latching switch; see figures 3(a) and 4(b)). For this case the connectivity domain's diagonal has $2(r' - 1) = 18$ cells; however, in the last (fourth) logic stage the signal vector G (generate) has to span over 32 cells in the horizontal direction. To implement such connections, two additional inverters have been added in the design in each bit slice. Also, assuming that the inputs to the adder are provided by CMOS lines, the broadcast of the input signal vectors A and B (figure 7(b)) has been avoided by adding another logic level (figure 7(c)).

For this particular value of $r'$, the final 'logic depth' (the number of logic levels in the critical path) is 21, the number to be compared to 13 levels for the conventional implementation, and 7 levels for the implementation with [4:1] LUTs. Figure 9 shows the depth as a function of $r'$. Smaller values of $r'$ result in larger depth and hence a larger total number of CMOS cells, up to the point $r' = r_{\text{min}}$, at which the layout becomes impossible. However, a reduction of $r'$ is beneficial for defect tolerance; see the next section.

### *4.2. Reconfiguration results*

The defect tolerance of the circuit immediately after the initial mapping is very poor, with the circuit yield going down

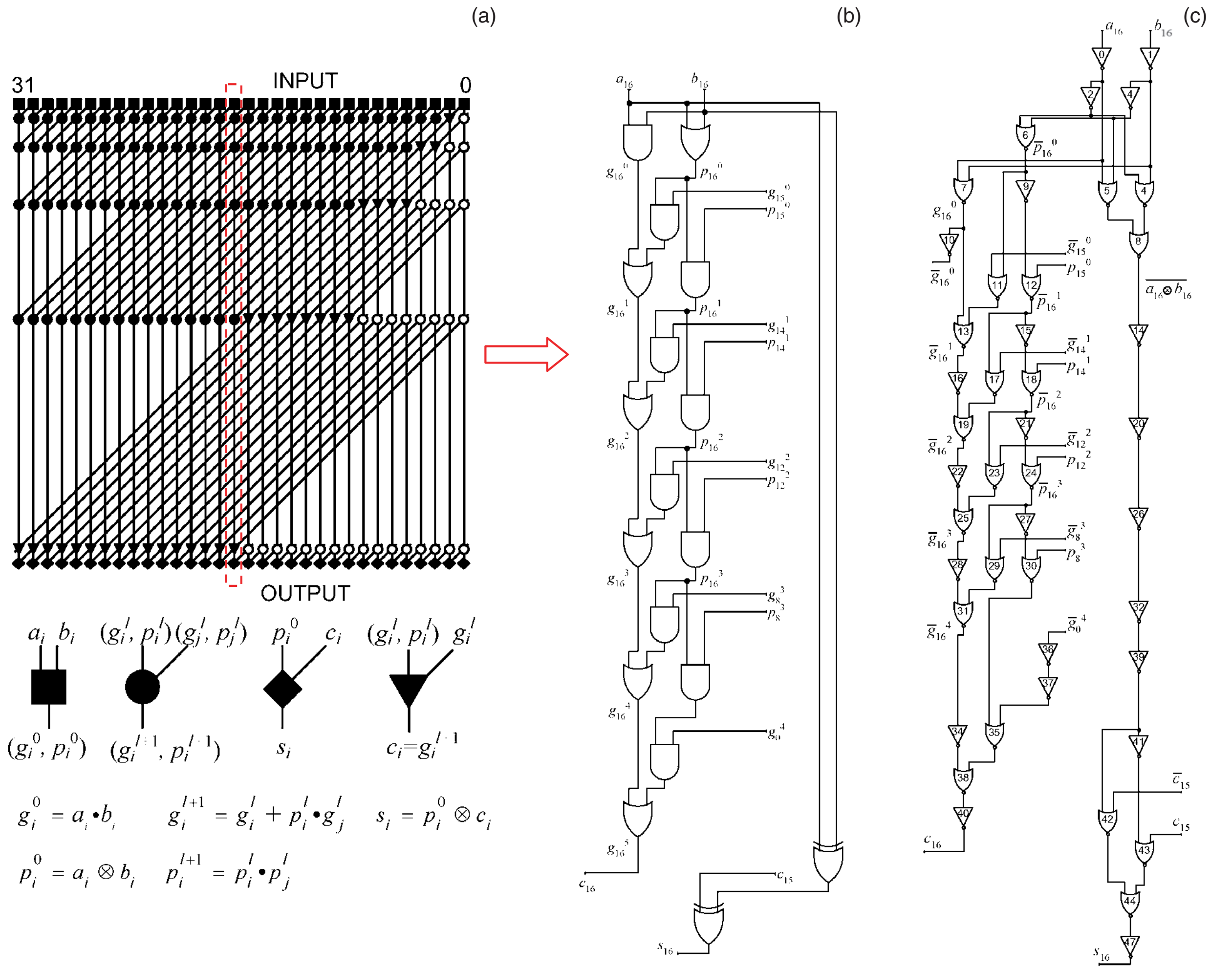
(a)
(b)
(c)
31
INPUT
0
OUTPUT
$a_i$ $b_i$
$(g_i^l, p_i^l)(g_j^l, p_j^l)$
$p_i^0$ $c_i$
$(g_i^l, p_i^l)$ $g_i^l$
$(g_i^0, p_i^0)$
$(g_i^{l+1}, p_i^{l+1})$
$s_i$
$c_i=g_i^{l+1}$
$g_i^0 = a_i \bullet b_i$
$g_i^{l+1} = g_i^l + p_i^l \bullet g_j^l$
$s_i = p_i^0 \otimes c_i$
$p_i^0 = a_i \otimes b_i$
$p_i^{l+1} = p_i^l \bullet p_j^l$
$a_{16}$
$b_{16}$
$s_{16}$
$c_{16}$
$c_{15}$
$\overline{a_{16} \otimes b_{16}}$

**Figure 7.** (a) The 32-bit Kogge–Stone adder and ((b), (c)) its single (16th) bit slice implemented with: (b) AND, OR, and XOR gates, and (c) NOR gates only.

rapidly at the bad device fraction $q$ as low as $\sim 5 \times 10^{-5}$, since the damage of any of $\sim$2300 actively used nanodevices leads to the circuit failure. The reconfiguration increases the acceptable $q$ dramatically. The increase has been calculated using numerical Monte Carlo simulation of the reconfiguration on our group's supercomputer cluster *Njal* (http://njal.physics.sunysb.edu/). For each initially mapped circuit, the program has been run 10 000 times with a randomly chosen set of defects, formed with the same probability $q$. For some (randomly chosen) successful reconfiguration runs the final layout has been functionally simulated to verify the correctness of the design. This was achieved by first saving the layout in the blif format and then converting it into the structural VHDL code with the help of the SIS package [29]. The verification has been fully successful.

Figure 8(b) shows the final connection map of the same adder as in figure 8(a) ($r' = 10$), after a typical successful reconfiguration with $r = 12$ and $q = 0.5$, while figure 10 shows the layout of a small fragment of this circuit, with defective nanodevices marked black. We were very much impressed how resilient the circuit was, retaining full functionality after reconfiguration around as many as 50% of bad devices. Actually, the defect tolerance could be even higher if we allowed the input and output cells of the adder to be moved (as can be done at a joint reconfiguration of several functional units).

Figures 11(a) and (c) show the fault tolerance of the adder as a function of $r$ and $r'$. If we choose not to confine the initial mapping additionally (i.e., take $r' = r$), the circuit becomes more defect tolerant as $r$ is increased. (With a fixed CMOS technology, $F_{\text{CMOS}}$, this requires scaling down the nanowire and nanodevice half-pitch $F_{\text{nano}}$.) If $r$, i.e., fabrication technology, is fixed, the defect tolerance may still be improved remarkably by taking just a slightly lower $r'$. The practical limit for this reduction is imposed by the explosive growth of the logic depth at $r' \to r_{\min}$ (figure 9), as the corresponding performance degradation; see section 6 below.

The results show, for example, that at realistic parameters ($r = 12$, $r' = 10$) the circuit may have a fabrication yield above[3] 99% at the fraction of bad nanodevices as high as $\sim$22% (figure 11(c)). Surprisingly enough, this is much better than our results for CMOL memories [16]. Probably, this

[3] Our estimates have shown that for hierarchically organized VLSI chips, this circuit reliability is sufficient for high total chip yield, with very minor additional circuit-level redundancy. For example, a CMOL analogue of the Teramac computer [5, 15] with circuits of this quality would have a total yield in excess of 99%, at circuit redundancy between 2 and 3.

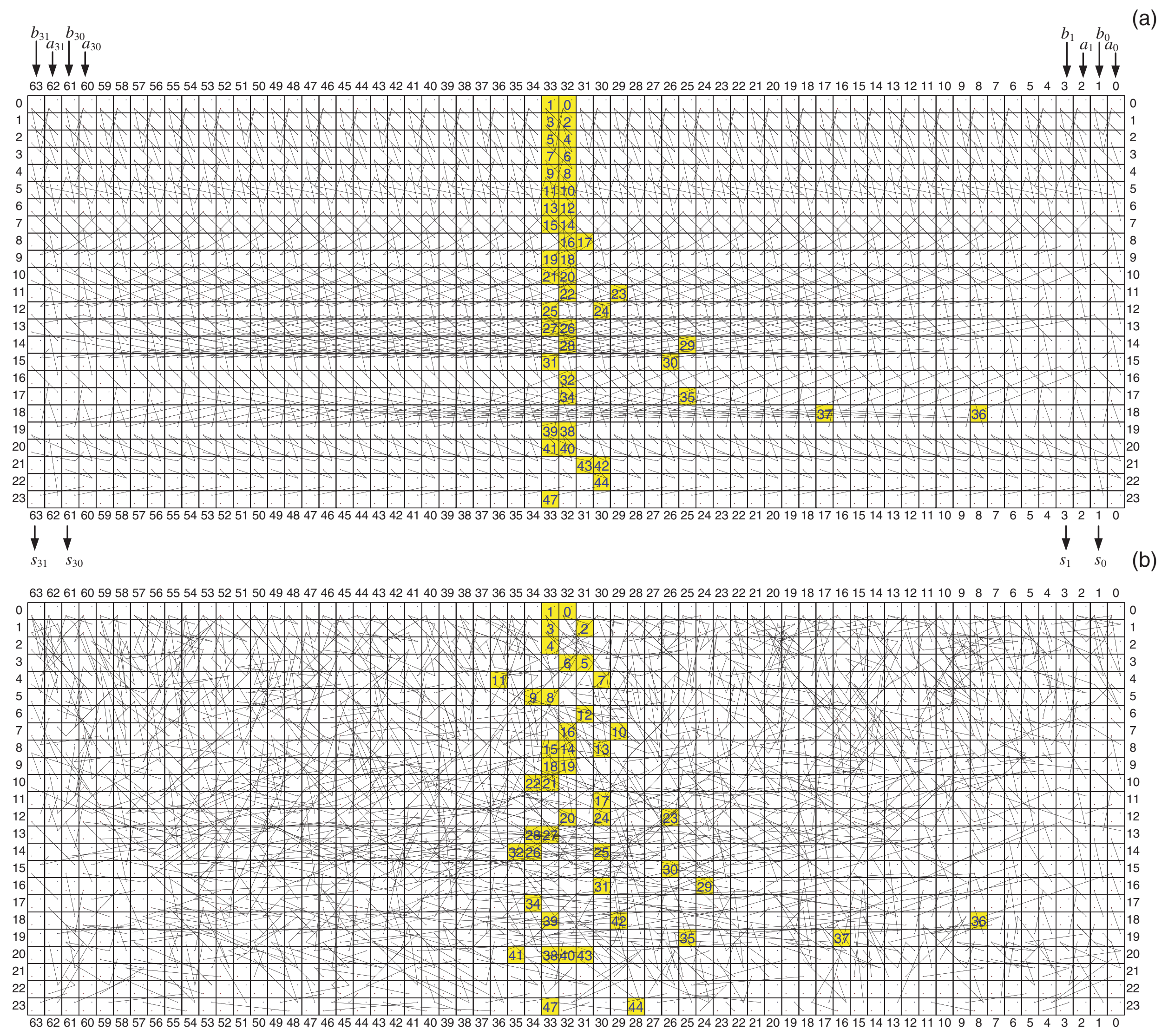

**Figure 8.** Mapping of the 32-bit Kogge–Stone adder on CMOL FPGA fabric with $r' = 10$: (a) the corresponding initial map of cell connections, and (b) the connection map after the successful reconfiguration of the circuit around as many as 50% of bad nanodevices (for $r = 12$). Gates of the 16th bit slice (see the dashed line in figure 7(a)) are painted yellow and numbered in accordance with figure 7(c).

means that the memory architecture we have analysed may be considerably improved.

## 5. Second case study: full crossbar

### 5.1. Initial mapping

Routing resources are a very important part of conventional FPGAs, as well as more exotic reconfigurable systems such as the Teramac computer [5]. This is why as our second case we have chosen the fully connected crossbar (figure 12(a)). For this circuit even the initial mapping on a rectangular CMOL array (with gates working as simple inverters) may be readily automated, for example using the simple 'greedy' algorithm[4]. In this procedure, the I/O pairs to be connected are mapped onto the array one-by-one. Each pair is first assigned a perfect-world Manhattan route, using the vertical rows of the input and output cells and some horizontal row (figure 12(b)). The algorithm checks that the vertical fragments of various routes do not overlap, while uniformly distributing their horizontal fragments among the array rows.

Then, to create an actual path for each I/O pair, the algorithm tries to allocate cells which are closest to the perfect route and are within each other's connectivity radius. (Of course, the cells used in mapping of the previously routed pairs cannot be used again.) Just as in the previous case, an artificial reduction of the connectivity to radius $r' \leqslant r$ (at the initial mapping only) improves the final defect tolerance.

In order to use this (or any other) routing algorithm practically, one needs to select the vertical size $m$ of the CMOL array first (figure 12(a)). In general, we are interested in the smallest value of $m$, because this leads to the smallest area and logic depth of the crossbar. Such a value can be calculated considering the worst possible combinations of the I/O pairs, which result in the largest aggregate data flow ($n$ routes) across the middle cross-section S of the rectangular

[4] Since the number $n$ of I/O pairs is typically much larger than the connectivity radius $r$, inputs and outputs cannot be connected directly.

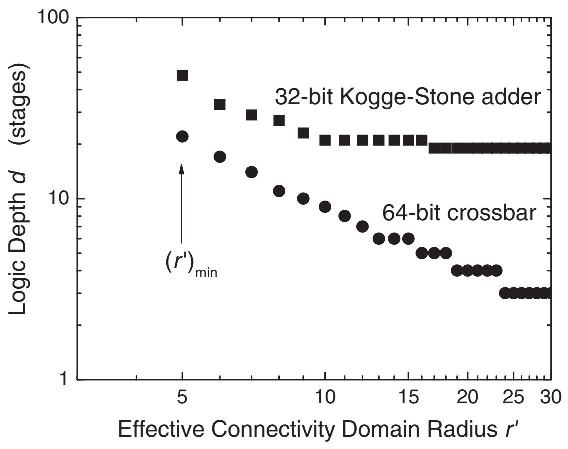

**Figure 9.** Logic depth (critical path length) of the two studied circuits as a function of the effective connectivity radius $r'$.

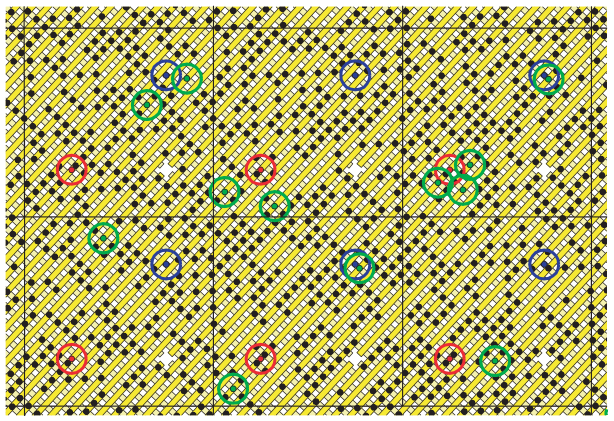

**Figure 10.** A small fragment of the adder after the same reconfiguration as in figure 8(b). Bad nanodevices (50% of the total number) are shown in black, good used devices in green, and unused devices are not shown, for clarity. Coloured circles are only a help for the eye, showing the location of interface pins (red and blue points) and nanodevices used. Thin vertical and horizontal lines show CMOS cell borders.

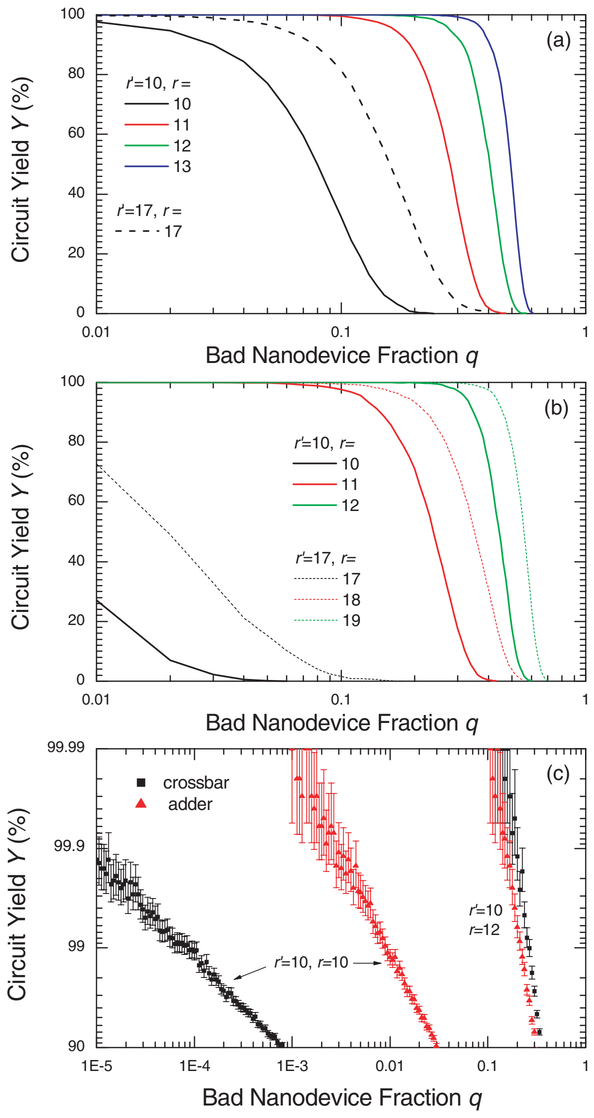

**Figure 11.** The final (post-reconfiguration) defect tolerance of ((a), (c)) the 32-bit Kogge–Stone adder and ((b), (c)) the 64-bit full crossbar for several values of $r$ and $r'$. Panel (c) shows the defect tolerance of the circuits on the log scale, which makes the results visible for the most interesting (high) values of yield. This panel is for the same values of $r$ and $r'$ which have been used for figure 8 and the performance estimates in section 6.

array (figure 12(c)). Since CMOL fabric has $(r-1)$ nanowires passing over each CMOS cell in the least favourable direction (figure 3(a)), and only $(r'-1)$ of them are used at the constrained-radius mapping, there are only $m(r'-1)$ nanowires overall to serve the critical cross-section S. This is why the crossbar height should satisfy the condition $m(r'-1) \geqslant n$. Moreover, in our simple 'greedy' algorithm, nanowires of the same critical cross-section may be used to provide vertical transport of (in the worst case) $n$ routes, so that a more strict condition should be satisfied: $m(r'-1) \geqslant n+m$, i.e., $m \geqslant n/(r'-2)$. Finally, including two input and output rows, the minimal crossbar height is $m_{\min} = \lceil n/(r'-2)\rceil + 2$.

From here, the maximum logic depth $d$ (the number of cell-to-cell hops) of the crossbar may be calculated as $\lceil (n+m)/(r'-2)\rceil$, because the longest route has the length of $(n+m)$ cells and each cell-to-cell hop allows one to move along this route by $(r'-2)$ cells in the worst (left-to-right) case. The resulting dependence of the logic depth of a 64-bit crossbar on $r'$ is shown in figure 9; it is substantially smaller than the depth of the 32-bit adder which has the same total input vector width $(32+32=64)$.

### 5.2. Reconfiguration results

Figures 11(b) and (c) show the yield results for the 64-bit crossbar after its reconfiguration using the same algorithm (figure 5), for several values of $r$ and $r'$. The most important difference from the adder (figures 11(a), (c)) is that without the artificial connectivity domain confinement (i.e., at $r'=r$) the crossbar is substantially less defect-tolerant than the adder. (This is a result of a larger fraction of long interconnects.) However, as soon as the difference $(r-r')$ is increased by the confinement, the defect tolerance of the crossbar improves very quickly, and becomes even better than that of the adder. For example, for the realistic case $r=12$ and $r'=10$, the 99% yield is actually achieved at ~25% of bad nanodevices, slightly higher than ~22% for the adder (figure 11(c)). Such rapid improvement is explained by the fact that the lower fan-in

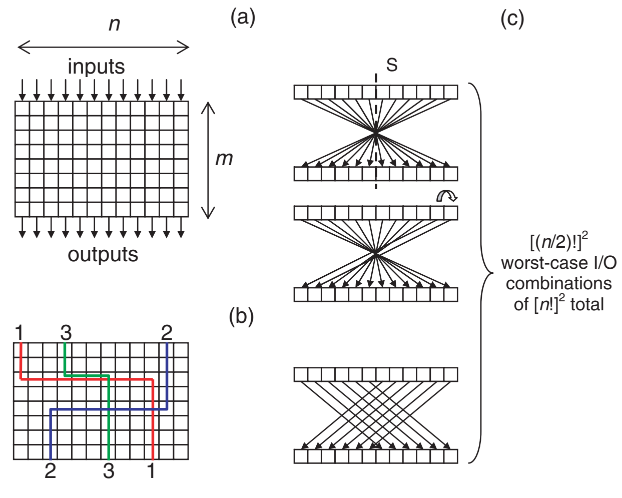

**Figure 12.** Full crossbar: (a) general configuration of the CMOL fabric, (b) perfect Manhattan routes used by the 'greedy' algorithm, and (c) the family of worst-case I/O pairs.

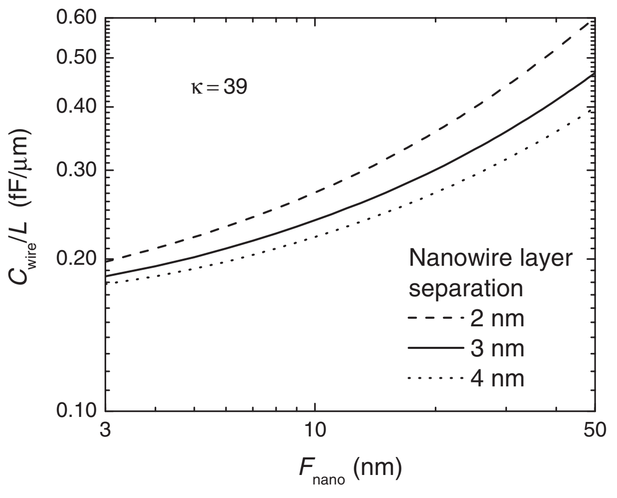

**Figure 13.** Specific capacitance of a nanowire with $F_{\text{nano}} \times F_{\text{nano}}$ cross-section, in a crossbar with several values of interlayer spacing (for dielectric constant $\kappa = 3.9$).

of crossbar gates (inverters) ensures larger repair domain size and hence more room for successful reconfiguration.

## 6. Performance

In this section, we will describe approximate estimates of density, speed, and power of CMOL FPGA circuits, using the following considerations and assumptions.

### *6.1. Nanodevices*

We have assumed that each latching switch is implemented as a parallel connection of several ($D$) single-electron devices of the type shown in figure 2(b)[5]. The most important parameters of the devices are the maximum Coulomb blockade threshold voltage $(V_t)_{\max}$ of the single electron transistor (which gives the scale of the power supply voltage $V_{DD}$, see figure 2(a)), and the ratio of its dynamic resistances in the ON and OFF states. Both these quantities depend on the single-electron addition energy

$$E_a = e(V_t)_{\max}, \tag{3}$$

which is generally contributed by both the single-electron island charging energy $E_c$ and quantum confinement energy $E_k$. For the sub-1 nm island size necessary for reliable room-temperature operation of the switches (see, for example, figure 13 of [20]), $E_k \gtrsim E_c$, so that we can use the formulae valid for the strong confinement limit [30, 31]

$$R_{\text{OFF}}/R_{\text{ON}} \approx \min[\cosh^2(E_a/2k_BT),\, R_{\text{ON}}/R_Q], \tag{4}$$

where $R_Q \equiv \hbar/e^2 \approx 4.1\ \text{k}\Omega$ is the quantum unit of resistance. The first term in the square brackets of equation (4) describes the effect of classical thermal fluctuations [30], while the second one gives a crude estimate for the second-order quantum effect, elastic co-tunnelling [31]. For our parameters (see below), equation (4) shows that the minimally acceptable $V_{DD}$ is of the order of 0.2–0.3 V. This is compatible with estimates of optimal $V_{DD}$ for most promising CMOS devices, double-gate SOI MOSFETs [19, 20], especially taking into account that CMOL circuits do not require the transistors to reach deep current saturation.

### *6.2. Nanowires*

The specific capacitance $C_{\text{wire}}/L$ of nanowires has been calculated using the well-known FASTCAP code [32] for the crossbar structure (figure 1(a)) in which both the width and the thickness of the nanowire, as well as the horizontal distance between the wires, were assumed to be all equal to $F_{\text{nano}}$, while the vertical distance between two layers was varied from 2 to 4 nm.[6] The insulator between and around the wires is assumed to have a dielectric constant of 3.9 (corresponding to $SiO_2$); the use of a low-$\kappa$ dielectric would give the corresponding increase of the circuit operation speed cited below. The result of the calculation is shown in figure 13.

In order to calculate the specific resistance $R_{\text{wire}}/L$ of a metallic nanowire with the assumed square-shaped cross-section $F_{\text{nano}} \times F_{\text{nano}}$, the usual formula $\rho/(F_{\text{nano}})^2$ has to be generalized to include the increase of resistivity $\rho$ due to possible diffusive surface scattering of electrons. (This effect becomes substantial when $F_{\text{nano}}$ is decreased below the electron mean free path $l$ due to scattering on phonons.) A reasonable approximation for $\rho$ is given by the Matthiessen rule [33] in the form

$$\rho \approx \rho_0 \times (1 + l/F_{\text{nano}}), \tag{5}$$

where $\rho_0$ is the table (bulk) resistivity. We will accept values $\rho_0 = 2\ \mu\Omega$ cm and $l = 10$ nm which are typical for good metals at room temperature[7].

[5] General physical arguments show [20] that estimates for electron nanodevices using other transport control mechanisms would be of the same order, while the single-electron option seems most attractive in view of the possible molecular implementation [21].

[6] This is the length range for single-electron latching switches designed for room-temperature operation; see, for example, figure 1(c) of [21]. For practical estimates we took the most plausible value of 3 nm.

[7] A more precise estimate of $R_{\text{wire}}$ is unnecessary, since it gives a noticeable effect on our results only at the extreme (and rather artificial) combination of the largest $F_{\text{CMOS}}$ with smallest $F_{\text{nano}}$. However, the use of semiconductor or molecular nanowires would change the situation dramatically and severely suppress the performance of CMOL FPGAs (or any other realistic hybrid nanoelectronic circuit).

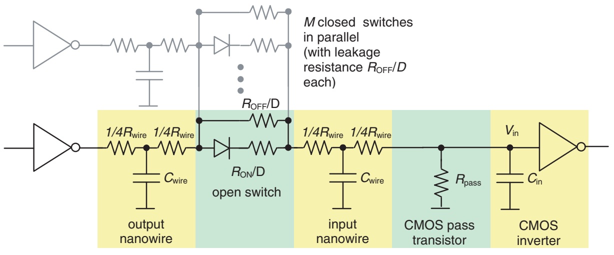

**Figure 14.** The equivalent circuit of a CMOL logic stage with unit fan-in and fan-out.

Applying these formulae, one should remember that while charged capacitance always corresponds to the full nanowire segment length $L = 2\beta^2 F_{\mathrm{CMOS}}^2/F_{\mathrm{nano}}$, only a part of the fragment (from the crosspoint nanodevice to the interface pin) contributes to its resistance $R_{\mathrm{wire}}$. In order to keep our estimates on the conservative side, we have assumed the worst configuration case when the length of this part is largest ($L/2$).

## 6.3. Circuit

In order to speed up the CMOL FPGA circuit, it is beneficial to reduce the signal swing of CMOS inverter's input voltage $V_{\mathrm{in}}$ by decreasing the effective parallel resistance $R_{\mathrm{par}}$ defined as (figure 14)

$$\frac{1}{R_{\mathrm{par}}} \equiv \frac{1}{R_{\mathrm{pass}}} + \frac{M}{R_{\mathrm{OFF}}/D}. \tag{6}$$

The limit to this reduction is set up by the requirement for the swing $V_{\mathrm{in}}$ to be larger than the possible total noise swing at the inverter input. Two most important components of the noise are the thermal fluctuations and digital noise of other gates.

At $M \gg 1$, the thermal noise is typically Gaussian, with the rms value

$$V_{\mathrm{T}} = (k_{\mathrm{B}}T/C_{\mathrm{wire}})^{1/2}, \tag{7}$$

which is of the order of a few millivolts for our parameters (see below). With the very strict requirement for the bit error rate to be below $q_{\mathrm{gate}} = 10^{-28}$ (corresponding, for example, to a mean time between failures of at least 10 000 h [4] for a CMOL FPGA chip with as many as $10^{10}$ gates operating with a 0.1 ns clock cycle), the maximum swing $\Delta V_{\mathrm{T}}$ of this noise, calculated from the equation $1 - \mathrm{erf}(\Delta V_{\mathrm{T}}/2\sqrt{2}V_{\mathrm{T}}) = 2q_{\mathrm{gate}}$, is close to 23 $V_{\mathrm{T}}$.

The digital noise is created mostly by coupling of output signals of $M$ other gates (with swing equal to $V_{\mathrm{DD}}$ each) through the $M$ parallel resistances of latching switches turned OFF (figure 14). Though for $M \gg 1$ the statistics of this noise is usually also close to Gaussian, one cannot exclude the possibility of strong correlation of signals processed by neighbouring gates. To play it safe, we have assumed the worst case scenario when all digital noise sources are fully correlated, resulting in the maximum swing $MV_{\mathrm{DD}}/(R_{\mathrm{OFF}}/D)$ of the current flowing to the inverter input.

Summing these two noise contributions, we get the following condition on $V_{\mathrm{in}}$:

$$V_{\mathrm{in}} > \Delta V_{\mathrm{T}} + MV_{\mathrm{DD}}\frac{R_{\mathrm{pass}}}{R_{\mathrm{OFF}}/D}, \tag{8}$$

where the simplification is due to the fact that for all considered cases $R_{\mathrm{pass}} \ll R_{\mathrm{OFF}}/(DM)$, i.e., $R_{\mathrm{par}} \approx R_{\mathrm{pass}}$, and $V_{\mathrm{DD}} \ll V_{\mathrm{in}}$.

Indeed, with the parameters considered below, this condition allows one to reduce $V_{\mathrm{in}}$ well below 100 mV, i.e., make it much lower than $V_{\mathrm{DD}}$. This means that the CMOL FPGA circuit speed is limited by the relatively slow recharging of a few-fF 'input' (post-latch) nanowire capacitance $C_{\mathrm{wire}}$ shunted by a relatively low parallel resistance $R_{\mathrm{pass}}$ given by equation (6), through a much higher series resistance $R_{\mathrm{ser}} \sim R_{\mathrm{ON}}/D + 2R_{\mathrm{wire}}$.[8] This is why the full equivalent circuit of one logic stage (figure 14) yields the elementary formula for the signal delay per logic stage:

$$\tau_0 \approx \log(2I)R_{\mathrm{pass}}C_{\mathrm{wire}}, \tag{9}$$

where $I$ is the gate fan-in, while the necessary value of $R_{\mathrm{pass}}$ may be calculated as

$$R_{\mathrm{pass}} = V_{\mathrm{in}}/DI_{\mathrm{ON}}. \tag{10}$$

The ON current of the nanodevice should be generally calculated from the $I$–$V$ curve (figure 2(a)), with $D$ parallel nanodevices connected in series with the Ohmic resistance $R_{\mathrm{wire}}$, driven by voltage $V_{\mathrm{DD}}$. However, since the only lower bound on the suppressed Coulomb blockade threshold $V_{\mathrm{t}}$ is to be larger than $V_{\mathrm{in}}$ (in order to prevent current leakage through ON-state nanodevices fed by 0-level output of CMOS inverters), $V_{\mathrm{t}}$ may be substantially less than $V_{\mathrm{DD}}$. Hence, we may consider the nanodevice $I$–$V$ curve linear, and find $I_{\mathrm{ON}}$ as

$$I_{\mathrm{ON}} \approx \frac{V_{\mathrm{DD}}}{R_{\mathrm{ser}}} = \frac{V_{\mathrm{DD}}}{R_{\mathrm{ON}}/D + 2R_{\mathrm{wire}}}. \tag{11}$$

## 6.4. Power

The average total power consumption of a CMOL gate may be estimated as a sum of the static power $P_{\mathrm{ON}}$ due to currents $I_{\mathrm{ON}}$, static power $P_{\mathrm{leak}}$ due to current leakage through nanodevices in their OFF state, and dynamic power $P_{\mathrm{dyn}}$ due to recharging of nanowire capacitances. The above estimate $V_{\mathrm{in}} \ll V_{\mathrm{DD}}$ allows one to calculate these contributions using simple formulae:

$$\begin{gathered} P_{\mathrm{ON}} \approx \frac{V_{\mathrm{DD}}^2}{2R_{\mathrm{ser}}}, \qquad P_{\mathrm{leak}} = \frac{MV_{\mathrm{DD}}^2}{2R_{\mathrm{OFF}}/D}, \\ P_{\mathrm{dyn}} = \frac{C_{\mathrm{wire}}V_{\mathrm{DD}}^2}{4\tau}, \end{gathered} \tag{12}$$

where $\tau$ is the total circuit delay, taken to be the product of the delay $\tau_0$ per logic stage (with $I = I_{\mathrm{max}}$) by the logic depth of the circuit (figure 9). The factors 1/2 reflect the natural assumption that on average there is an equal number of CMOS inverters with Boolean 1 and 0; the dynamic power has an additional factor 1/2 describing the energy loss at capacitance recharging.

[8] The output dynamic impedance of the CMOL inverter and its input capacitance $C_{\mathrm{in}}$ give negligible contributions to $\tau_0$. For example, $C_{\mathrm{in}}$ of the 22 nm minimum-width inverter is of the order of 0.02 fF, i.e., much less than $C_{\mathrm{wire}}$.

### 6.5. Area

To estimate the circuit density, we need parameter $\beta$, the linear size of the CMOS cell in units of CMOS pitch $2F_{\mathrm{CMOS}}$ (figure 3). We will show below that at acceptable power density the ON current necessary for driving one nanodevice is of the order of 1 $\mu$A. With a linear current density of the order of 1000 $\mu$A $\mu$m$^{-1}$, typical for the long-term CMOS projections [4], such current may be provided with a MOSFET channel as narrow as 1 nm. Hence, we can assume that all four transistors of the CMOS cell are of the minimum width. Using the SCMOS design rules [34], we may estimate the cell area $A_{\mathrm{cell}}$ as $64(F_{\mathrm{CMOS}})^2$, i.e., $\beta \approx 4$. For each combination of $F_{\mathrm{CMOS}}$ and $F_{\mathrm{nano}}$, we have selected a $\beta$ larger than, but closest to, $\beta_{\min} = 4$ from the possible spectrum given by equation (1), giving us the corresponding value of the connectivity radius $r$. This rule leads to jumps of $r$ (and hence $r'$ and all circuit parameters) as a function of $F_{\mathrm{nano}}$; see figure 15 below. (We have accepted a modest connectivity domain confinement, $r - r' = 2$, which is sufficient for high defect tolerance; see figure 11.)

### 6.6. Optimization

In order to evaluate the CMOL FPGA performance, we have limited the total power $P = P_{\mathrm{ON}} + P_{\mathrm{leak}} + P_{\mathrm{dyn}}$ per unit circuit area at the level $P/A = 200$ W cm$^{-2}$ planned by the ITRS [4] for the next decade. With the sum fixed, the power supply voltage $V_{\mathrm{DD}}$ may be optimized to minimize the total logic delay $\tau = d\tau_0$ of the circuit.

In order to do this, for each pair of $F_{\mathrm{CMOL}}$ and $F_{\mathrm{nano}}$ (and hence for parameters $\beta$, $r, r'$, $L$, $C_{\mathrm{wire}}$, and $R_{\mathrm{wire}}$ calculated as described above), we vary $V_{\mathrm{DD}}$, each time adjusting the ratio $R_{\mathrm{ON}}/D$ (and hence the product $DI_{\mathrm{ON}}$ calculated from equation (11)) so that the total power calculated from equation (12) equalled the specified level. At this procedure, $R_{\mathrm{OFF}}$ is also adjusted to keep the thermal stability requirement, expressed by the left part of equation (4), satisfied[9].

### 6.7. Results

Figure 15 shows the results of such optimization for three long-term CMOS technology nodes specified by the ITRS [4]. First of all, figure 15(a) indicates that the largest contribution to power consumption is given by $P_{\mathrm{ON}}$; this is very typical for diode-logic circuits like ours. Static power is not too sensitive to $F_{\mathrm{nano}}$, but the dynamic power drops with increased nanowire pitch, together with $V_{\mathrm{DD}}$.

Figure 15(b) shows that the nanowire segment capacitance $C_{\mathrm{wire}}$ increases if the nanodevice pitch is scaled down, due to the increase of the segment length $L = 2\beta^2 F_{\mathrm{CMOS}}^2/F_{\mathrm{nano}}$. However, the circuit delay follows this trend only at lower values of $F_{\mathrm{nano}}$, because at larger values of the nanowire pitch (and hence lower $L$) the connectivity domain radius $r$ decreases and results in an increase of the logic depth $d$ of the circuit (figure 9) and hence of the circuit delay $\tau = d\tau_0$. The same

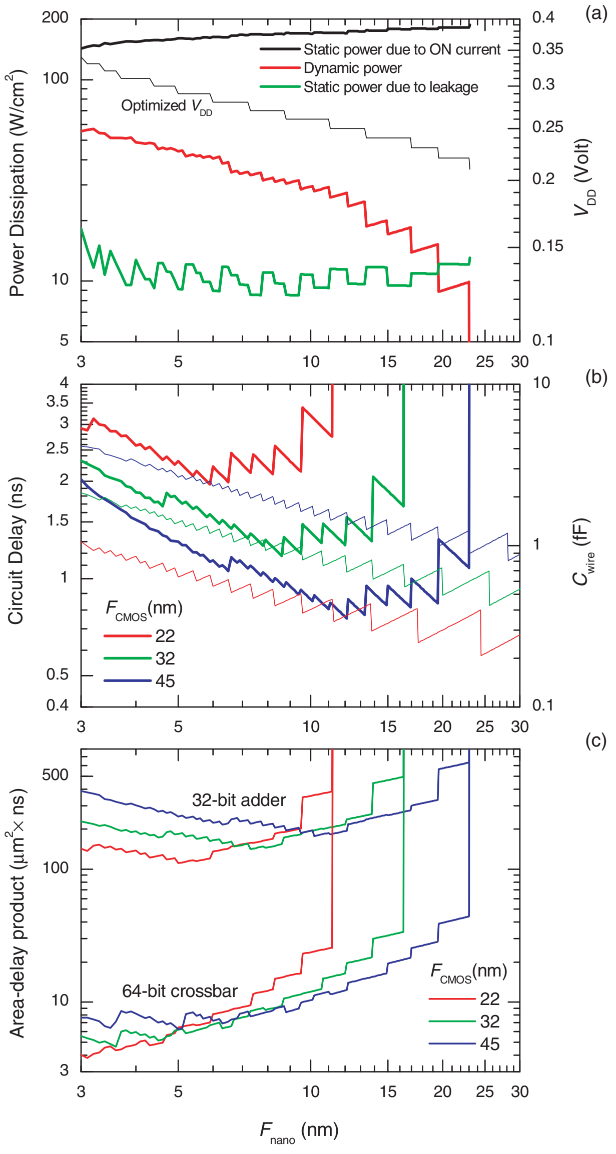

**Figure 15.** CMOL FPGA optimization results as functions of nanowire half-pitch $F_{\mathrm{nano}}$: (a) three components of the total power (fixed at 200 W cm$^{-2}$), and the optimum value of the power supply voltage $V_{\mathrm{DD}}$, for the 32-bit adder with $F_{\mathrm{CMOS}} = 45$ nm; (b) nanowire segment capacitance (thin lines) and the total logic delay of the circuit (bold lines); and (c) area-delay product $A\tau$ of the two CMOL FPGA circuits under analysis, for three ITRS long-term CMOS technology nodes. The (formal) jump of the $A\tau$ product to infinity at some $(F_{\mathrm{nano}})_{\max}$ reflects the fact that our procedure of initial circuit mapping may only be implemented for $F_{\mathrm{nano}}$ below this value; see figure 9 and its discussion. The finite sharp jumps of the curves are due to the transfers between adjacent integer values of $r$ that would satisfy equations (1) and provide the smallest $\beta > \beta_{\min} = 4$. All results are for $r' = r - 2$.

effect is clearly visible in figure 15(c), which shows our results for a popular figure-of-merit of integrated circuits, the circuit area-by-delay product $A\tau$.[10] This product increases both at very small $F_{\mathrm{nano}}$ (due to the increase of $C_{\mathrm{wire}}$) and at larger $F_{\mathrm{nano}}$

[9] The quantum-fluctuation part of that requirement is only used to check that the minimum number of elementary devices in each crosspoint, $D_{\min} \approx R_{\mathrm{Q}}(R_{\mathrm{OFF}}/D)/(R_{\mathrm{ON}}/D)^2$, is above one. Within the range of parameters shown in figure 15, $D_{\min}$ varies between 7 and 100, numbers which are very convenient for the molecular self-assembly [6, 7] of such devices.

[10] The growth of $A\tau_{\mathrm{total}}$ with $F_{\mathrm{nano}}$ is expressed even more strongly in the crossbar where $d$ grows approximately as $n^2/(r-2)$ starting already from low values of the connectivity domain radius; see figure 10 and section 5.1 above.

(due to the growing $d$). As a result, the area-delay product as a function of $F_{\text{nano}}$ features a minimum, indicating the existence of the optimum nanotechnology for each $F_{\text{CMOS}}$. Note that for the most realistic values of $F_{\text{CMOS}}$ (45 and 32 nm), the optimum value of $F_{\text{nano}}$ is not very low, at least for the integer adder.

Finally, as could be expected, the best performance improves with better CMOS subsystem technology, though not too quickly: $(A\tau_{\text{total}})_{\text{min}}$ is approximately proportional to $F_{\text{CMOS}}$.

## 7. Discussion

The reader has to agree that the absolute numbers shown in figure 15 are very impressive. For example, for the apparently realistic values $F_{\text{CMOS}} = 32$ nm and $F_{\text{nano}} = 8$ nm, the 32-bit CMOL FPGA adder could have an area about 110 $\mu\text{m}^2$ and total logic delay 1.3 ns, at acceptable power dissipation. (The 64-bit crossbar performance is an order of magnitude better.)

In order to compare these numbers with purely CMOS FPGAs, we have used the Xilinx ISE WebPack package (see www.xilinx.com) to simulate the similar 32-bit Kogge–Stone adder for the commercially available 90 nm Xilinx Spartan-3 technology. (The basic unit of such an FPGA is a slice consisting of two 4-input LUTs.) The total delay of the adder, excluding the pin-to-slice propagation delay, has turned out to be about 5.1 ns. Assuming the $1/s$ delay scaling [28], this corresponds to 1.7 ns for the 32 nm technology. The circuit area of the CMOS circuit could be calculated from the known number of its cells (‘tiles’), equal to 139, and the tile area estimate of approximately 2100 $\mu\text{m}^2$, which follows from the data cited in [35]. With the usual $1/s^2$ area scaling, for $F_{\text{CMOS}} = 32$ nm this gives a 280 $\mu\text{m}^2$ tile area, i.e., a total adder area of about 39 000 $\mu\text{m}^2$. (This estimate is close to the one given by DeHon [36].) Thus the delay-area product would be about 70 000 ns $\mu\text{m}^2$, i.e., about 500 times larger than in a CMOL FPGA with the same $F_{\text{CMOS}}$.

Though the performance advantage of CMOLs, obtained using our (very conservative) estimates, seems overwhelming, we need to make the following two reservations. First, the estimated CMOL circuits did not include latches (besides the relatively slowly switching nanodevices used for circuit mapping and reconfiguration), while a typical CMOS FPGA has flip-flops after each logic stage, which allows pipelined design for operation at clock frequencies of the order of $1/2\tau_0$. In CMOL FPGAs, pipelining may be readily achieved by interleaving CMOL arrays with clocked CMOS registers (figure 16(a)). Since each CMOL array may be configured in its own way, such interleaved structures may be used to accommodate complex hierarchical computing structures. For example, figure 16(b) shows a possible implementation of the PLASMA chip [15], the fundamental unit of the Teramac computer [5]. A crude estimate[11] has shown that even accounting for the latches, and sufficient block redundancy (see footnote 4), the whole computer could be mapped on a CMOL chip (with $F_{\text{CMOS}} = 32$ nm and $F_{\text{nano}} = 8$ nm) with area well below 1 $\text{cm}^2$.

[11] For this estimate we have assumed that a $16 \times 16$ CMOL array is functionally close to the Teramac ‘hextant’ block with 16 [6:2] LUTs. Also, in the CMOL implementation the crossbar sizes have been adjusted to keep the hierarchy and the Rent rule exponent of 2/3 of the original Teramac.

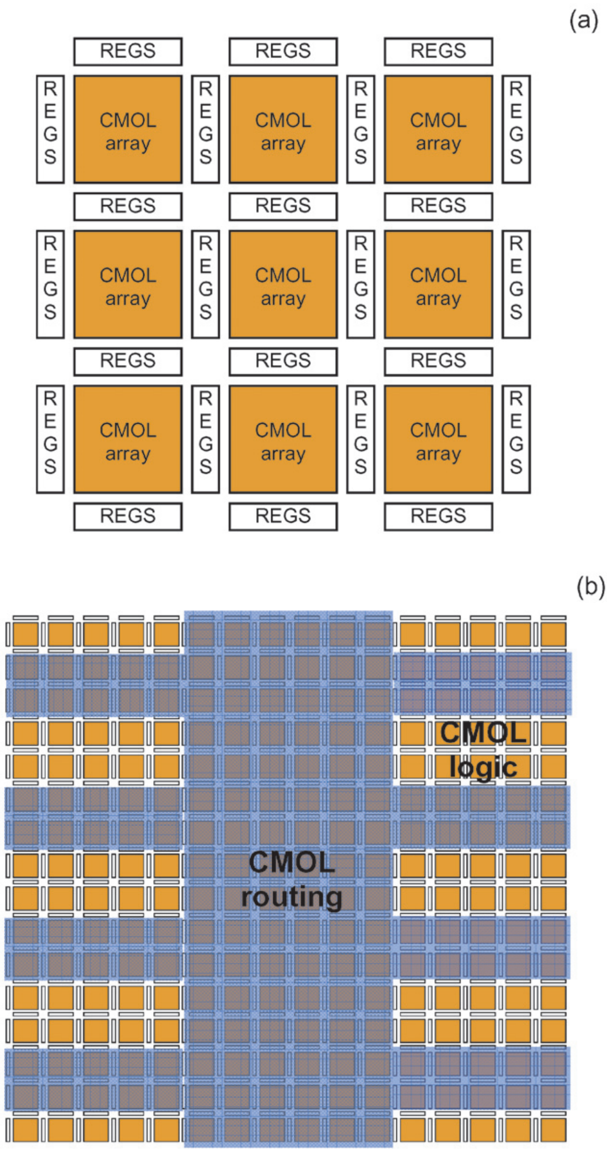

**Figure 16.** (a) A macro-array of CMOL FPGA arrays interleaved with CMOS registers, and (b) its use for implementation of the PLASMA chip architecture [15].

The additional power consumption of CMOS registers will certainly increase power consumption of the circuit as a whole. On the other hand, it would also increase its area, so it is not quite clear whether the circuit performance (at fixed power management limitations) would increase or decrease. However, the number of the register cells may be low (of the order of $2n$ per array with $n \times n$ cells), so that any change should be relatively small. More exact evaluation may be carried out only after a substantial number of various functional units and other circuits necessary for digital signal processing and/or general-purpose computing have been mapped on the CMOL fabric. (This will probably require a modification of existing CAD tools.) Eventually, CMOL FPGA systems should be evaluated on generally accepted computing benchmarks. However, we believe that even the preliminary estimates described in this paper give a strong evidence that the CMOL FPGA approach may far outperform CMOS FPGAs in virtually all areas of their application.

The comparison between CMOL FPGA and custom CMOS chips is a more complex issue[12]. Indeed, in our current design each CMOS cell uses just a few nanodevices for actual operation, while most nanodevices are used just for circuit reconfiguration. However, the functional density of our approach can be substantially improved if gates with higher fan-in are allowed. (Note that the current hardware implementation is already suited for such gates.) Multi-fan-in-gate implementation may allow CMOL FPGAs to compete with custom CMOS chips while hopefully decreasing only slightly the defect tolerance[13]. Exploring these advanced capabilities of CMOL FPGAs is our next goal.

## Acknowledgments

Useful discussions with V Beiu, S Das, J Ellenbogen, and M Stan are gratefully acknowledged. The work has been supported by the AFOSR and NSF.

## References

[1] Brown S D *et al* 1992 *Field-Programmable Gate Arrays* (Norwell, MA: Kluwer)

[2] Tessier R and Schmit H (ed) 2004 *FPGA 2004: Proc. ACM/SIGDA 12th Int. Symp. on Field Programmable Gate Arrays (Monterey, CA, USA, Feb. 2004)*

[3] *FCCM 2004: Proc. 12th IEEE Symp. on Field-Programmable Custom Computing Machines (Napa, CA, April 2004)* (Piscataway, NJ: IEEE)

[4] International Technology Roadmap for Semiconductors (ITRS), 2004 Update available online at public.itrs.net/

[5] Heath J R, Kuekes P J, Snider G S and Williams R S 1998 A defect-tolerant computer architecture: Opportunities for nanotechnology *Science* **280** 1716–21

[6] Tour J 2003 *Molecular Electronics* (Singapore: World Scientific)

[7] Reimers J R, Picconnatto C A, Ellenbogen J C and Shashidhar R (ed) 2003 *Molecular Electronics III*; *Ann. New York Acad. Sci.* **1006** 1–33

[8] Stan M *et al* 2003 Molecular electronics: from devices and interconnect to circuits and architecture *Proc. IEEE* **91** 1940–57

[9] von Neuman J 1956 Probabilistic logics and the synthesis of reliable organisms from unreliable components *Automata Studies* ed C E Shannon and J McCarthy (Princeton, NJ: Princeton University Press) pp 43–98

[10] Nikolic K, Sadek A and Forshaw M 2002 Fault-tolerant techniques for nanocomputers *Nanotechnology* **13** 357–62

[11] Roy S and Beiu V 2005 Multiplexing schemes for cost effective fault-tolerance *IEEE-NANO'04 (Munich, Germany, Aug. 2004)*; *IEEE Trans. Nanotechnol.* at press

[12] DeHon A 2005 Design of programmable interconnect for sublithographic programmable logic arrays *Proc. FPGA'05 (Monterey, CA, Feb. 2005)* pp 127–37

[13] Snider G *et al* 2004 CMOS-like logic in defective, nanoscale crossbars *Nanotechnology* **15** 881–91

[14] Rose J *et al* 1990 Architecture of field-programmable gate arrays—the effect of logic block functionality and efficiency *IEEE J. Solid-State Circuits* **25** 1217

[15] Amerson R *et al* 1996 Plasma: an FPGA for million gate systems *Proc. FPGA'96 (Monterey, CA, Feb. 1996)* pp 10–6

[16] Strukov D and Likharev K 2005 Prospects for terabit-scale nanoelectronic memories *Nanotechnology* **16** 137–48

[17] Ahmed E and Rose J 2004 The effect of LUT and cluster size on deep-submicron FPGA performance and density *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **12** 288–98

[18] Kouloheris J and Gamal A E 1992 PLA-based FPGA versus cell granularity *Proc. Custom Integrated Circuits Conf.* (Piscataway, NJ: IEEE) pp 4.3.1–4

[19] Sverdlov V A, Walls T J and Likharev K K 2003 Nanoscale silicon MOSFETs: a theoretical study *IEEE Trans. Electron Devices* **50** 1926–33

[20] Likharev K 2003 Electronics below 10 nm *Nano and Giga Challenges in Microelectronics* ed J Greer *et al* (Amsterdam: Elsevier) pp 27–68

[21] Likharev K and Strukov D 2005 CMOL: devices, circuits, and architectures *Introducing Molecular Electronics* ed G Cuniberti *et al* (Berlin: Springer) at press (preprint available online at http://rsfq1.physics.sunysb.edu/~likharev/nano/Springer04.pdf)

[22] Fölling S, Türel Ö and Likharev K K 2001 Single-electron latching switches as nanoscale synapses *Proc. 2001 Int. Joint Conf. on Neural Networks (Washington, DC, July 2001)* pp 216–21

[23] Jensen K L 1999 Field emitter arrays for plasma and microwave source applications *Phys. Plasmas* **6** 2241–53

[24] Zankovych S *et al* 2001 Nanoimprint lithography: challenges and prospects *Nanotechnology* **12** 91–5

[25] Brueck S R J *et al* 2002 There are no limits to optical lithography *International Trends in Optics* (Bellingham, WA: SPIE Press) pp 85–109

[26] Türel Ö, Lee J H, Ma X and Likharev K 2004 Neuromorphic architectures for nanoelectronic circuits *Int. J. Circuit Theory Appl.* **32** 277–302

[27] Betz V *et al* 1999 *Architecture and CAD for Deep-Submicron FPGAs* (Dordrecht: Kluwer)

[28] Rabaey J, Chandrakasan A and Nikolic B 2002 *Digital Integrated Circuits* 2nd edn (Upper Saddle River, NJ: Prentice-Hall)

[29] Sentovich E M *et al* 1992 SIS: A system for sequential circuit synthesis *UCB/ERL Report* #M92/41 University of California, Berkley (available online at ftp://ic.eecs.berkeley.edu)

[30] Averin D V, Korotkov A N and Likharev K K 1991 Theory of single-electron charging of quantum wells and dots *Phys. Rev.* B **44** 6199–211

[31] Glazman L I and Matveev K A 1990 Residual quantum conductivity under Coulomb blockade conditions *JETP Lett.* **51** 484–7

[32] Nabors K, Kim S and White J 1992 Fast capacitance extraction of general three-dimensional structures *IEEE Trans. Microw. Theory Tech.* **40** 1496–506

[33] Kittel C 1995 *Introduction to Solid State Physics* 7th edn (New York: Wiley)

[34] Mead C and Conway L 1980 *Introduction to VLSI Systems* (Reading, MA: Addison-Wesley)

[35] Padalia K *et al* 2003 Automatic transistor and physical design of FPGA tiles from an architectural specification *Proc. FPGA'03 (Monterey, CA, Feb. 2003)* pp 164–72

[36] DeHon A 1996 Reconfigurable architectures for general-purpose computing *AI Technical Report* 1586 MIT Artificial Intelligence Laboratory

[37] Kogge P M and Stone H S 1973 Parallel algorithm for efficient solution of a general class of recurrence equations *IEEE Trans. Comput.* **22** 786–93

[38] Betz V and Rose J 1999 FPGA place-and-route challenge U. Toronto (available online at http://www.eecg.toronto.edu/~vaughn/challenge/challenge.html)

[39] Naeimi H and DeHon A 2004 A greedy algorithm for tolerating defective crosspoints in NanoPLA design *Proc. ICFPT'2004 (Brisbane, Australia, Dec. 2004)* pp 49–56

[12] Forgetting for a minute that the FPGA approach helps to bypass the present-day IC design bottleneck.

[13] Indeed, our preliminary estimates for several benchmark circuits [38] have shown that the increase of maximum gate fan-in from 2 to 4 would allow one to decrease the area-delay product by approximately twice at the same power density. The further fan-in increase (especially above 6) does not lead to noticeable performance increase. Thus, the optimal fan-in for CMOL FPGAs seems to be close to that for the traditional LUT-based FPGAs [14].