INSTITUTE OF PHYSICS PUBLISHING NANOTECHNOLOGY
Nanotechnology **16** (2005) 137–148 doi:10.1088/0957-4484/16/1/028

Prospects for terabit-scale nanoelectronic memories

Dmitri B Strukov and Konstantin K Likharev

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

Received 19 June 2004, in final form 18 November 2004
Published 10 December 2004
Online at stacks.iop.org/Nano/16/137

Abstract
We have calculated the minimum chip area overhead, and hence the bit density reduction, that may be achieved by memory array reconfiguration (bad bit exclusion), combined with error correction code techniques, in prospective terabit-scale hybrid semiconductor/nanodevice memories, as a function of the nanodevice fabrication yield and the micro-to-nano pitch ratio. The results show that by using the best (but hardly practicable) reconfiguration and block size optimization, hybrid memories with a pitch ratio of 10 may overcome purely semiconductor memories in useful bit density if the fraction of bad nanodevices is below ∼15%, while in order to get an order-of-magnitude advantage in density, the number of bad devices has to be decreased to ∼2%. For the simple 'Repair Most' technique of bad bit exclusion, complemented with the Hamming-code error correction, these numbers are close to 2% and 0.1%, respectively. When applied to purely semiconductor memories, the same technique allows us to reduce the chip area 'swelling' to just 40% at as many as 0.1% of bad devices. We have also estimated the power and speed of the hybrid memories and have found that, at a reasonable choice of nanodevice resistance, both the additional power and speed loss due to the nanodevice subsystem may be negligible.

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

1. Introduction

The recent spectacular advances in molecular electronics, in particular the demonstration of single-molecule single-electron transistors by several groups [1–5], offer the hope for a practical introduction of hybrid semiconductor/nanodevice circuits, first of all for terabit-scale memory applications [6–8]. In such memories, nanodevices (e.g., single molecules) would be used as single-bit memory cells, while the semiconductor transistor subsystem would perform all the peripheral (input/output, coding/decoding, line driving, and sense amplification) functions that require relatively smaller number of devices (scaling as $N^{1/2}$, where N is the memory size in bits). The first experimental steps toward the implementation of the hybrid memories have already been made; see, for example, [9–11].

The main architectural challenge faced by the hybrid memories is the anticipated substantial fraction of 'bad' nanodevices, limited by both the integration technique (e.g., the chemically-directed molecular self-assembly; see, for example, [12, 13]), and the vulnerability of nanoscale devices to random charged defects [7]. The main approach to addressing this problem in semiconductor memory technology [14, 15] is reconfiguration, i.e. the replacement of memory array lines (rows or columns) containing bad cells by spare lines. The effectiveness of the replacement depends on how good its algorithm is [15, 16]. The Exhaustive Search approach (trying all possible combinations) finds the best repair solution, though it is not practicable because of the exponentially large execution time. A more acceptable choice is the 'Repair Most' method that allows a simple hardware implementation and an execution time scaling linearly with the number of bits. In this approach, the number of faults in each line of a memory block (matrix) is counted, and the lines having the largest number of defects are replaced with the spare lines.

For a larger fraction of bad bits, better results may be achieved [17] by combining the bad line exclusion with error-correction code (ECC) techniques. In semiconductor memories the ECC is reserved mainly for the suppression of soft rather than the hard errors, i.e., for insuring the memory fault tolerance rather than defect tolerance [15, 18]. However, for the prospective hybrid memories the defect tolerance is expected to be a much more acute problem, and ECC may

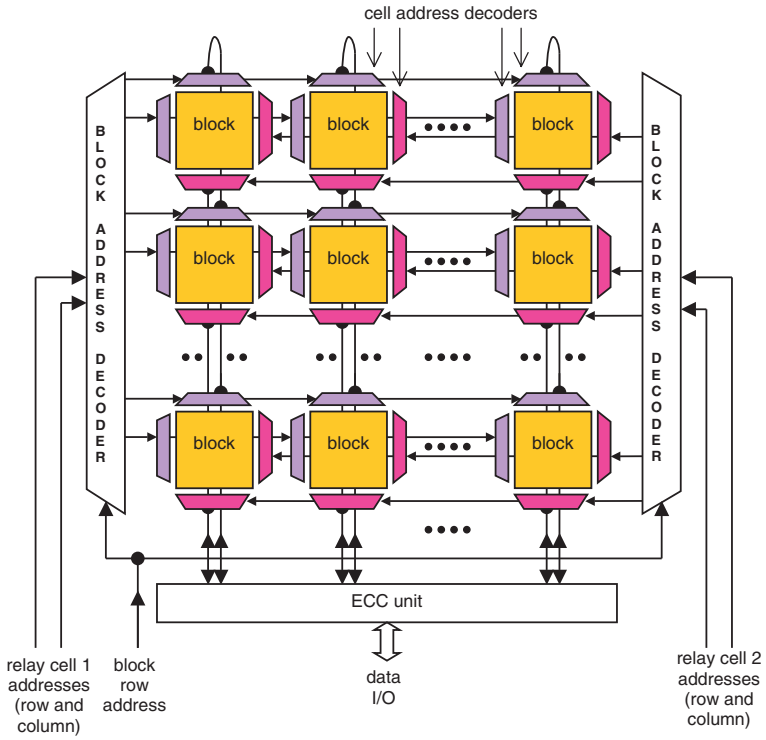

Figure 1. Top structure of the analysed hybrid memory. At each instance, block address decoders allow one to send the cell row and column addresses to a single row of blocks. The cell addresses are then processed by decoders of each block.

need to be involved at the initial repair process as well. The technical analysis of the opportunity, carried out in the pioneering work [17], has been limited to the exclusion of just a few spare lines.

The objective of our work is a study of the bad component exclusion techniques extended to an arbitrary number of redundant rows and columns of the matrix blocks, applied both with and without the error correction. We have calculated the chip real estate overhead necessary for the fabrication of spare rows, columns, and auxiliary circuits, as a function of the bad bit fraction q and the ratio R of critical dimensions of the semiconductor transistors and nanodevice components. The results for $R = 1$ are also presented, since we believe that they are important for the evaluation of scaling prospects of purely semiconductor memories.

2. Model

2.1. Memory structure

Figure 1 shows the assumed general structure of the hybrid memory. It is essentially a matrix of L memory blocks, while each block is in turn a rectangular array of $(n + a) \times (m + b)$ memory cells. Here a and b present redundant resources that are being used for bad bit replacement at the initial test and repair stage, so that the final number of used memory cells is $L \times n \times m$.[1] The good cells, addressed at each particular time step, form a row with one cell per block. They have the same external word and bit addresses in each block, though due to the internal re-routing during the initial reconfiguration process, the real physical location of the used cell may be different in each block.

In contrast to the memory top structure (figure 1), the block architecture (figure 2) is substantially different from the traditional 'uniform' (semiconductor) memories. At each elementary operation, the block decoders address two vertical and two horizontal lines implemented in the CMOS layers of the integrated circuit, thus selecting a pair of 'relay' CMOS cells. The reason for such doubling is straightforward: the number of nanodevice memory cells may be much higher than the product of the number of CMOS-level wires going in each direction, and hence the usual procedure of selection of one cell (at the crosspoint of one horizontal and one vertical line) does not allow the user to address every nanodevice cell. Figure 2 shows how such addressing may be accomplished using the 'CMOL' circuit concept [7, 19, 20]. In this approach, just as in virtually all other hybrid circuit proposals [6, 8–10, 21, 22], the nanodevices are formed at each crosspoint of two layers of a 'crossbar' array, consisting of two levels of parallel nanowires. Such parallel nanowire arrays may be fabricated by several advanced patterning technologies, such as nanoimprint [24] or interference lithography [25], which may provide much better resolution (in future, down to a few nanometres) than the standard photolithography. These novel technologies cannot be used for patterning of arbitrary integrated circuits, in particular because they lack adequate layer alignment accuracy; however, the fabrication of a crossbar array does not require accurate alignment. Such an approach, of course, implies that nanodevice formation at nanowire crosspoints is also accomplished without lithographic patterning. (The chemically directed self-assembly of pre-synthesized molecular devices from

[1] With the account of an ECC overhead, the number of useful bits is slightly lower; see equation (22).

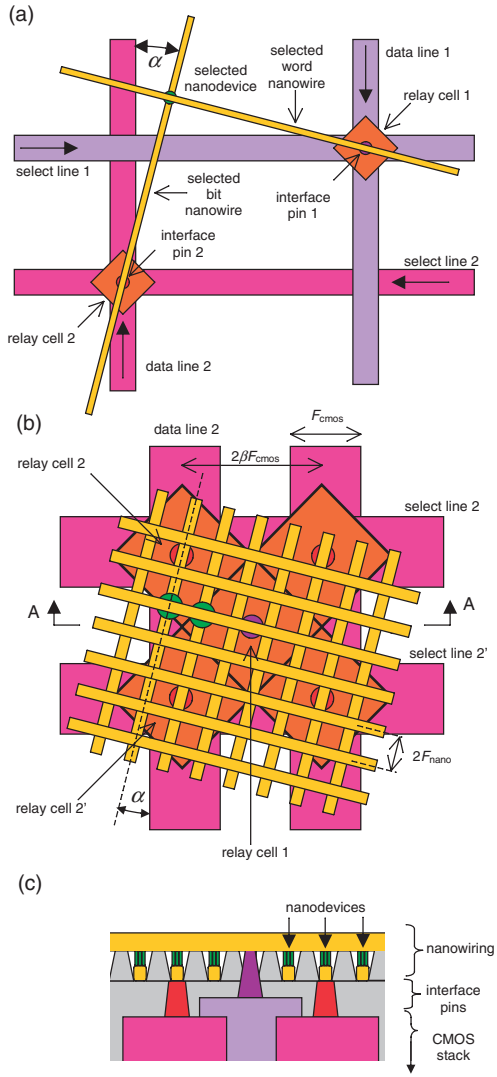

Figure 2. Low-level CMOL memory structure. (a) General scheme of addressing a particular nanodevice. Only the activated CMOS lines and nanowires are shown. (b) Zoom-in showing several neighbouring relay cells. For the sake of clarity, the panel shows only the two nanodevices discussed in the text; in reality, the devices are formed at any nanowire crosspoint. Also disguised are the CMOS-level wires going to the selected cell 1. (c) Side view on a CMOL circuit (schematically). It may be understood as a cross-section of panel (b) along line A–A.

solution [12, 13] is probably the most evident, but not the only possible, example of such a process.)

The difference between the CMOL approach and the earlier suggestions [21, 23] (which seem hardly feasible from the fabrication point of view [20]) is in the interfacing between the nanowires and the underlying CMOS-level wires: in CMOL it is provided by sharp-pointed pins[2] that are distributed all over the circuit area. Somewhat counter-intuitively, this approach allows individual access to each nanowire even if the half-pitch $F_{\rm nano}$ of the nanowire crossbar is much less than that ($F_{\rm CMOS}$) of semiconductor-level wiring. Figures 2(a), (b)

[2] The technology for fabrication of a-few-nm-sharp points has been already developed in the context of field-emission array applications; see, for example, [26].

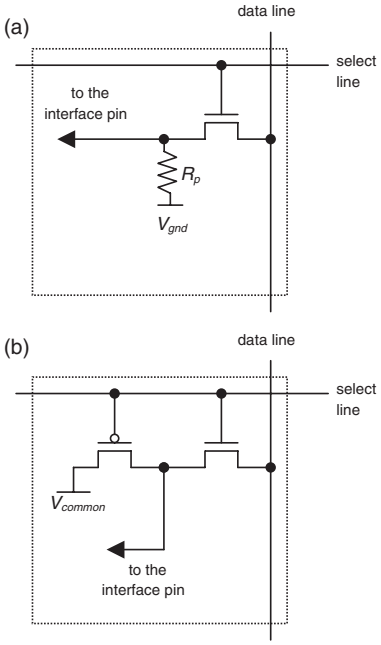

Figure 3. Two possible implementations of a CMOS relay cell.

explain the trick[3] that allows such access: the nanowire crossbar is turned by angle $\alpha = \arcsin(F_{\rm nano}/\beta F_{\rm CMOS})$ relative to the CMOS-level wiring matrix. (Here parameter β is defined by the area $2\beta^2 F_{\rm CMOS}^2$ of the CMOS relay cell. Since the cell structure should incorporate at least one transistor—see figure 3 and its discussion below—β may be substantially larger than unity.) For example, the selection of relay cells 1 and 2 (figure 2(b)) enables contacts to the nanowires leading to the left one of the two shown nanodevices. Now, if we keep selecting cell 1, and instead of cell 2 select cell 2′ (using the next CMOS wiring row), we contact the nanowires going to the right nanodevice instead. It is easy to understand that this trick allows addressing each nanodevice via a pair of relay cells located at the Manhattan distance $|\Delta x| + |\Delta y| < R + 1$, where $R \equiv 1/\sin\alpha$ is just the ratio of the CMOS wiring pitch $2\beta F_{\rm CMOS}$ to the nanowiring pitch $2F_{\rm nano}$.

The simplest and the most compact design of the relay cell may be based on just one pass transistor and one pull-down resistor (figure 3(a)). Another, complementary version of relay cell (figure 3(b)) may have a somewhat larger footprint, but ensures lower power consumption. In any version, the selected relay cell connects the CMOS data line to the corresponding nanowire.

Figure 4 shows (schematically) the assumed I–V curve of the nanodevice that is typical, in particular, for the single-electron latching switch [7, 27–29][4]. In one of the states (binary '0') the device passes almost no current until the applied voltage V reaches a certain threshold value $V_{\rm t}^{(+)}$. At this point, the device switches to state 1 with a finite current. The reciprocal switching takes place at voltage $V_{\rm t}^{(-)}$; for the

[3] Borrowed from our previous work on 'InBar'-type neuromorphic networks ('CrossNets') [27, 28].

[4] Such a switch is essentially a two-terminal version of a four-terminal device that had been discussed in brief earlier [30]. Two-terminal devices are much more promising for 'bottom-up' fabrication, in particular, the chemically-directed molecular self-assembly [12, 13].

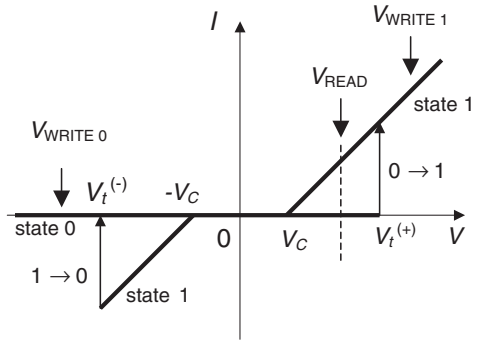

Figure 4. The assumed I–V curve of the nanodevice used as a memory cell (schematically).

sake of simplicity we have accepted that $V_t^{(-)} = -V_t^{(+)} \equiv -V_t$, but in practice this condition is not important. The exact shape of the I–V curves in the open state 1 is also not very important; the only requirement is that if the device has a Coulomb-blockade (or Coulomb-blockade-like) range $[-V_C, +V_C]$ (figure 4), it should be well within the $[V_t^{(-)}, V_t^{(+)}]$ interval.

With this assumption, the read and write operations are very similar to those in semiconductor memories [14], especially floating-gate nonvolatile memories (NVM [31]). For READ operation, one of the relay cells (e.g., cell 1 in figure 2(a)) applies voltage $V_{READ}/2$ to the corresponding word nanowire, while the complementary relay cell (cell 2 in figure 2(a)) connects the selected device, via the bit nanowire, to the CMOS data line biased by voltage $-V_{READ}/2$, and then to the input of a CMOS sense amplifier. As a result, the selected nanodevice becomes biased by voltage $V_{READ} > V_C$ (figure 4), so that if this device is in the open state, it passes current $I \approx (V_{READ} - V_C)/R_{nano}$ to the sense amplifier. Similarly, for WRITE 1 and WRITE 0 operations, each of the relay cells applies half of the necessary voltage ($V_{WRITE\,1}$ or $V_{WRITE\,0}$; see figure 4) to the corresponding nanowire, so that only one nanodevice, located at the crosspoint of these wires, is switched into the opposite state.

2.2. Defect model

The following assumptions have been made about the memory defects:

(i) Defective nanodevice cells are randomly distributed, with probability q, among the block matrices.
(ii) The density of defective interface pins, nanowires, and CMOS components is negligibly small[5].
(iii) The bad memory cells correspond to 'stuck-on-open' defects and do not affect (i.e. neither shorten nor interrupt) the nanowires. This assumption seems reasonable at least for molecular self-assembly, where most defects are expected to result from a failure to have a molecular device assembled at a certain nanowire crosspoint.

[5] Concerning the pins and nanowires, this assumption is made more plausible by the following fact. The hybrid memory structure (figure 2) has an inherent redundancy: nominally, each nanowire is contacted by several pins (spaced by distance $D = 2\beta F_{CMOS}/\tan\alpha \approx 2(\beta F_{CMOS})^2/F_{nano} = 2F_{nano}(\beta R)^2$). This means that even if the nanowires have a limited number of breaks (with the average distance between them much larger than D) and/or a small fraction (well below $F_{nano}/\beta F_{CMOS} = 1/R$) of pins do not provide good contacts, these defects may be excluded from the circuit operation without any area overhead.

2.3. Memory 'repair' (reconfiguration)

At the initial testing of each memory block, bad nanodevices are detected, and their physical location is used to compile a special table (additional memory) that maps the continuous external address space into the space of physical addresses of relay cells leading to good nanodevices. While the temporal latency of the mapping procedure can be overlapped with block decoding, the area of the mapping table, implemented in the CMOS subsystem, may be substantial and is included in our area calculations (see below).

2.4. ECC

The ECC circuitry is assumed to be basically the same as in traditional memories, where it usually takes no more than 10% of the chip area; see, for example, [15, 32]. For the most interesting cases such real estate overhead is insignificant, and we will ignore it. In contrast, the area occupied by redundant bits involved in the error correction may be substantial, and we are including it in our calculations (section 4).

3. Yield calculations

We have calculated the necessary redundant resources for two methods of bad bit exclusion: a simple 'Repair Most' approach and the absolutely best 'Exhaustive Search' method, both with and without the error-correction code enhancement. For that, we first calculate the full memory yield Y as a function of the bad bit probability $q = 1 - p$ and other parameters (L, m, n, a, b, R), and then determine the real estate overhead by requiring the yield to be equal to a realistic number.

3.1. Repair Most

In some versions of the Repair Most approach the direction of excluded lines is arbitrary. In this study we use a slightly simpler version of this method, in which lines are first excluded in one dimension (e.g., rows) and then in the other (columns).

With our assumption of defect independence, the probability to have k bad bits in an initial row of length $(m+b)$ obeys the binomial distribution: $P(k) = \binom{m+b}{k} q^k (1-q)^{m+b-k}$. For all reasonable parameters (see section 5 below), both $(m+b)$ and $1/q$ are large in comparison with the average number $d_i = \langle k \rangle = q(m+b)$ of defects in the initial row, so that we can use for $P(k)$ the Poisson approximation:

$$P(k) \approx \frac{d_i^k e^{-d_i}}{k!}. \qquad (1)$$

Now, considering all rows in a block matrix, the statistical distribution of rows having k defects follows the same equation (1). The exclusion of a rows with the largest number of defects of the initial $(n+a)$ rows is equivalent to dropping a tail, with the area $a/(n+a)$ underneath, of that distribution.

Finding the average number d_r of bad bits per row after such an exclusion is easiest if the number is large. In this case,

the distribution $P(k)$ may be treated as a continuous one, so that

$$d_{\mathrm{r}} = \frac{\int_0^x k P(k)\,\mathrm{d}k}{\int_0^x P(k)\,\mathrm{d}k}, \tag{2a}$$

where x, the maximum number of defects per line after the row exclusion, can be found from the solution to the equation

$$\int_0^x P(k)\,\mathrm{d}k = \frac{n}{n+a}. \tag{2b}$$

In the case when d_{r} is comparable with, or equal to, unity, it may be found numerically from the discrete version of equations (2):

$$d_{\mathrm{r}} = \frac{\sum_{k=0}^{x-1} k P(k) + x P_{\mathrm{r}}(x)}{\sum_{k=0}^{x-1} P(k) + P_{\mathrm{r}}(x)}, \qquad \text{for } x \geqslant 1, \tag{3a}$$

where $P_{\mathrm{r}}(x)$ is the part of the probability $P(x)$ which reflects the number of retained rows with the threshold number (x) of bad bits. The integer x and the fractional number $P_{\mathrm{r}}(x)$ are now defined by the normalization condition similar to equation (2b):

$$\sum_{k=0}^{x-1} P(k) + P_{\mathrm{r}}(x) = \frac{n}{n+a}. \tag{3b}$$

Finally, if $x = 0$, then in an average array all defective lines will be excluded. To account for deviations from this average, d_i can be expressed as a sum $d_i = \sum_{j=0}^{(n+a)(m+b)} e_j P(e_j)$, where $e_j = j/(n+a)$ plays the role of a local d_i and

$$P(e_j) = \binom{(n+a)(m+b)}{j} q^j (1-q)^{(n+a)(m+b)-j} \tag{4a}$$

is its probability. Indeed, for a large number L of arrays with the same probability q of defective cells, $L_j = LP(e_j)$ arrays will have j defects, which translates to the local e_j. In that case d_{r} can be found as a sum of local $d_{\mathrm{r}}(e_j)$ with the corresponding weights[6]:

$$d_{\mathrm{r}} = \sum_{j=0}^{(n+a)(m+b)} d_{\mathrm{r}}(e_j) P(e_j). \tag{4b}$$

After the row exclusion, the remaining defects are distributed randomly over the columns (no correlation between the defect's vertical and horizontal positions), and can be characterized by the new probability

$$q_{\mathrm{r}} = \frac{d_{\mathrm{r}}}{m+b} < q. \tag{5a}$$

From here, the probability of a column of height n to be perfect is $(1-q_{\mathrm{r}})^n$, so that the probability to have an imperfect column is

$$w_{\mathrm{c}} = 1 - \left(1 - \frac{d_{\mathrm{r}}}{(m+b)}\right)^n. \tag{5b}$$

The statistics of the number j of defective columns obeys the binomial distribution, and the probability y of finding not more than b defect-free columns which can be excluded at the second stage of repair (i.e. a completely good sub-block with the final size $n \times m$) can be expressed through this distribution:

$$y = \sum_{j=0}^{b} \binom{m+b}{j} w_{\mathrm{c}}^j (1-w_{\mathrm{c}})^{(m+b-j)}. \tag{6}$$

This formula gives the final (post-repair) yield of single blocks. The yield of the whole memory consisting of L blocks is simply

$$Y = y^L. \tag{7}$$

3.2. Exhaustive Search

The Exhaustive Search looks for the absolutely best way of excluding a spare lines and b spare rows. Though the practical hardware implementation of such a repair technique is formidable, an estimate of its potential results is important for the evaluation of quality of practicable algorithms (such as the Repair Most method). In this context, we may restrict our task to finding an upper bound $Y_{\max}$ for the yield after the Exhaustive Search repair[7].

The total number t of ways to choose a subarray $n \times m$ from the initial $(n+a) \times (m+b)$ array, and the total number h of ways to put d defects into the initial array, are, respectively:

$$t = \binom{n+a}{a}\binom{m+b}{b}, \qquad h = \binom{(n+a)(m+b)}{d}. \tag{8}$$

Consider the $t \times h$ matrix W shown in figure 5(a), with elements $w_{i,j}$ equal to unity if the subarray number i (ranging from 1 to t) is defect-free for the defect pattern number j (ranging from 1 to h), and equal to zero otherwise. The real yield $y(d)$ after the Exhaustive Search repair of a block array with d defects may be expressed as

$$y(d) = \frac{1}{h}\sum_{j=1}^{h} W_j, \qquad \text{where } W_j \equiv \begin{cases} 1, & \text{if } \sum_{i=1}^{t} w_{i,j} > 0 \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

because W_j shows whether a particular (jth) configuration of d defects may be completely excluded by some choice of subarray $n \times m$. An upper bound $y_{\max}(d) \geqslant y(d)$ can be found by replacing W_j with the larger or equal amount $\sum_{i=1}^{t} w_{i,j}$. This replacement yields

$$y_{\max}(d) = \max\left[\frac{1}{h}\sum_{j=1}^{h}\sum_{i=1}^{t} w_{i,j}, 1\right] = \max\left[\frac{t}{h}\binom{s}{d}, 1\right], \tag{10}$$

because the sum $\sum_{j=1}^{h} w_{i,j}$ (i.e. the number of binary units in any column of matrix W), by the definition of coefficients $w_{i,j}$, is equal to the number of ways in which $d \leqslant s$ defects may be distributed within the excluded area $s \equiv (n+a) \times (m+b) - n \times$

[6] Practical calculations of d_{r} from equations (3), (4) are facilitated by the fact that the Poisson distribution obeys the formula $\sum_{k=0}^{x} P(k) = \frac{\Gamma(x+1,d_i)}{\Gamma(d_i)}$, where the Γs in the nominator and denominator are, respectively, the incomplete and complete gamma functions [33].

[7] Finding the exact value of y is closely related to the calculation of the so-called Zarankiewicz number [34], which is a still unsolved fundamental mathematical problem [35].

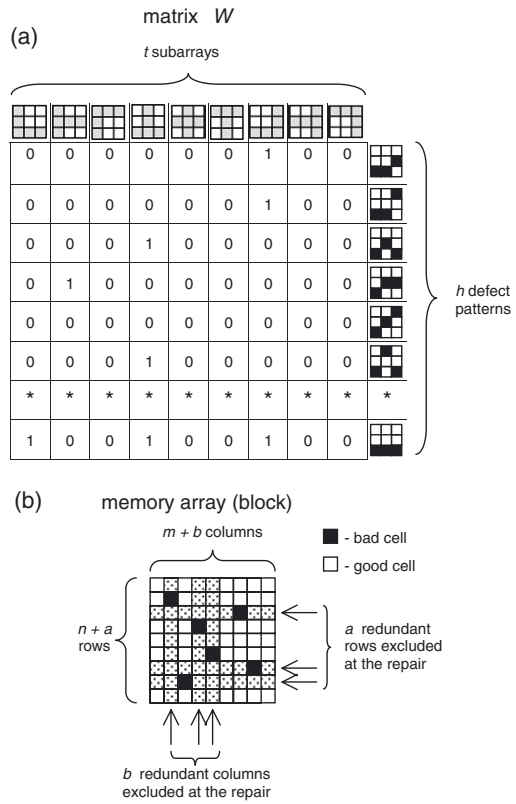

Figure 5. (a) Table W for the case of a 3×3 memory array with one redundant row and one redundant column ($m = n = 2$, $a = b = 1$) and three defects, and (b) the notation accepted for marking the rows and columns of this table. Binary 1 in a cell of matrix W means that this particular choice of excluded rows and columns of the memory array, at this particular defect pattern, makes the repaired array defect-free.

$m \geqslant d$. (If $d > s$, this cannot be done, i.e. all $w_{i,j} = 0$, and $y_{\max}(d) = 0$.) Summing $y_{\max}(d)$ over all the possible values of d (from 0 to s) with the weights equal to the probability of finding d defects in the initial array, we finally get

$$y_{\max} = \sum_{d=0}^{s} y_{\max}(d) \left(\begin{matrix} (n+a)(m+b) \\ d \end{matrix} \right) q^d (1-q)^{(n+a)(m+b)-d}. \tag{11}$$

Note that $y_{\max}$ does not depend on the array geometry (i.e. the particular values of n, m and a, b) but is rather a function of its total area $(n + a)(m + b)$ and the total number s of redundant cells.

Figure 6 shows, with dotted lines, typical results for $y_{\max}$ given by equation (11) for square block arrays with several values of the final array dimensions ($n = m$) and redundant rows and columns numbers ($a = b$). For comparison, the solid lines on that figure show the results for yield y obtained with the Repair Most algorithm (see equation (6)), while the dashed lines show yield $y_0 = q^{mn}$ without repair. One can see that the difference between the two repair methods is not too large, especially if the number of redundant lines is not too high (below, or of the order of the final memory size)[8]. The difference, however, becomes larger if we use the array reconfiguration together with the ECC technique; see below.

[8] For example, it is easy to comprehend that at $a = b = 1$ the Exhaustive Search can eliminate at most only one additional defect.

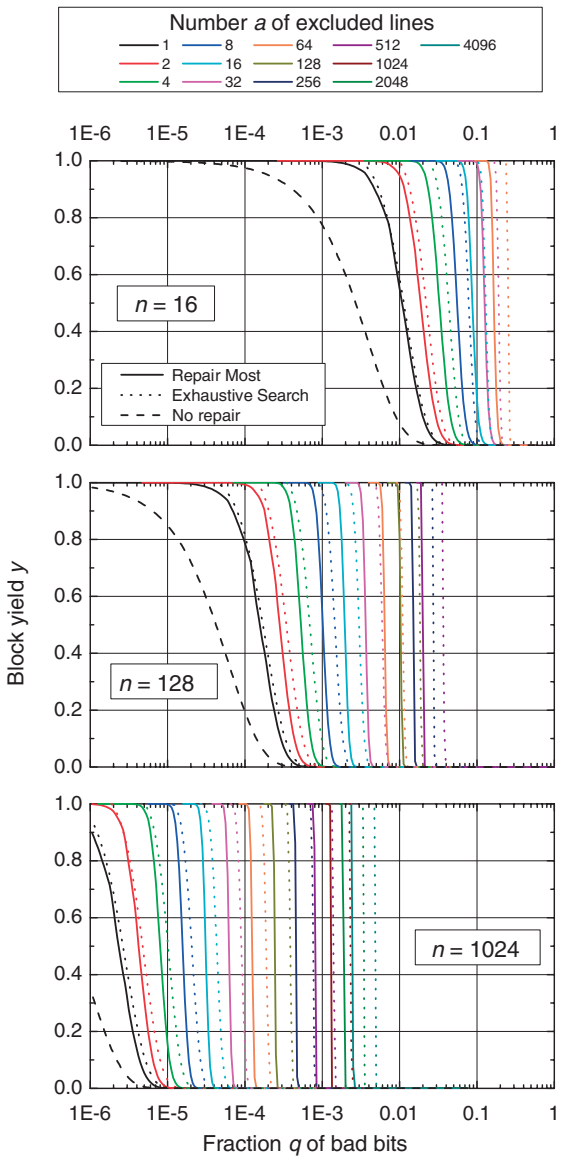

Figure 6. Comparison of the defect tolerance provided by the two reconfiguration techniques, Repair Most (solid curves) and Exclusive Search (dotted curves), analysed in this work, without the error correction. As a reference, the dashed curves show the results without the bad bit exclusion.

3.3. Hamming-code ECC alone

We have restricted our analysis to the most popular ECC technique based on SEC-DED (single error correction, double error detection) Hamming codes (c, k) [15, 17, 18, 32, 36, 37], where c and k are the code and the data word lengths, respectively. (The number $(c - k)$ of parity bits should satisfy the condition $\sum_{i=1;i=\text{odd}}^{c-k} \left(\begin{matrix} c-k \\ i \end{matrix} \right) \geqslant c$ [37], i.e. $(c - k) \gtrsim 2 + \log_2 c$). The probability Q that the decoded word will not have an error is the sum of two probabilities: that the initial code has no errors, and that it has one error. Both probabilities follow the binomial distribution, therefore $Q = (1-q)^c + cq(1-q)^{c-1}$. Since the memory consisting of L blocks, with $n \times m$ bits each, can accommodate Lnm/c Hamming codes, its effective yield is

$$Y = Q^{Lnm/c} = [(1-q)^c + cq(1-q)^{c-1}]^{Lnm/c}. \tag{12}$$

3.4. Hamming-code ECC on top of the Repair Most exclusion

Since the Repair Most exclusion is performed for each block individually, the defect distribution over c cells used to keep each Hamming code remains random, and we can use the formula similar to equation (12)

$$Y = Q^{Lnm/c} = [(1-q_f)^c + cq_f(1-q_f)^{c-1}]^{Lnm/c}, \quad (13)$$

where q_f is the probability of a single cell to be bad after the whole exclusion (of rows and columns) has been completed. This probability can be calculated from q_r exactly as q_r itself has been calculated from the initial probability q, i.e. using equations (1)–(5) with the substitutions

$$m+b \to n, \qquad n+a \to m+b, \qquad n \to m, \quad (14)$$

reflecting the swapping of rows and columns at the second stage of the repair process.

3.5. Exhaustive Search combined with Hamming-code EEC

The Hamming-code error correction could be also performed 'on top of' (after) the Exhaustive Search repair. Since such a repair is hardly practicable, and we are mostly interested in it as the upper bound for the yield, it is more logical to consider an even more perfect strategy in which the bad bit exclusion and error correction are intertwined. In this approach, each possible combination of $n \times m$ subarrays, with arbitrary renumbered rows and columns, in each of c blocks used by the Hamming code, is evaluated for the best error correction outcome, and the best found combination is kept in the mapping table[9].

The number of ways to exclude a rows and b columns from each of c blocks is simply $\binom{n+a}{a}^c \binom{m+b}{b}^c$. Therefore, the full number t of ways to choose a set of c subarrays with renumbered n rows and m columns (i.e. a certain combination of code words) is

$$t = \binom{n+a}{a}^c \binom{m+b}{b}^c (n!m!)^{c-1}, \quad (15)$$

where the subtraction of 1 from c in the last term cancels the counting of cases when all the bits in the Hamming code words are simply exchanged. Let us introduce matrix W similar to that described in section 3.2 above, although now it describes all c blocks, i.e. t is now given by equation (15), while

$$h = \binom{(n+a)(m+b)c}{d}. \quad (16)$$

The upper bound $y_{\max}(d)$ is given by the first part of equation (10); however, the value of $\sum_{j=1}^{h} w_{i,j}$ has to be recalculated.

Let $l \leqslant d$ defects be located in the selected set of subarrays, of the total size nmc. In order to be fixed with a Hamming code, only one defect is allowed in a code word; such a defect can be located in any of c bit positions. Hence, for l defects in l words (with one defect per word), the number of ECC-correctable bit patterns is c^l. Now, accounting for all possible positions of l code words inside the set of subarrays (with the total number of code words nm) the number of ECC-correctable patterns is $\binom{nm}{l}c^l$. Note that there is a total of $\binom{nmc}{l}$ defect patterns. Since for all of these correctable patterns the remaining $(d-l)$ defects outside the set of subarrays can be in any bit positions, the total number of perfect or ECC-correctable patterns is

$$N(l) = \binom{cs}{d-l}\binom{nm}{l}c^l, \qquad \text{for } d-l \leqslant sc, \quad l \leqslant d, \quad (17)$$

where s, as above, equals $(n+a)\times(m+b) - n\times m$. Summing the number of binary 1s in any column of matrix W over the range of allowed l by equation (17), we get

$$\sum_{j=1}^{h} w_{i,j} = \sum_{l=\max[0,d-sc]}^{\min[nm,d]} N(l). \quad (18)$$

Since this sum does not depend on index i (just as in equation (10) above),

$$y_{\max}(d) = \max\left[\frac{t}{h}\sum_{j=1}^{h} w_{i,j},\, 1\right], \quad (19)$$

and, similarly to equation (11), the upper bound for yield of the whole memory is

$$Y_{\max} = (y_{\max})^{L/c} = \left[\sum_{d=0}^{sc} y_{\max}(d)\binom{(n+a)(m+b)c}{d} \times q^d(1-q)^{(n+a)(m+b)c-d}\right]^{L/c}. \quad (20)$$

4. Chip area assumptions

We have calculated the hybrid memory chip area necessary to reach a fixed chip yield Y, using the following assumptions:

(i) The total chip area is the sum

$$A = A_{\text{block decoders}} + L \times [A_{\text{array}} + A_{\text{cell decoders}} + A_{\text{drive/sense}} + A_{\text{mapping table}}], \quad (21)$$

while its useful bit capacity is

$$N = Lnm(k/c). \quad (22)$$

The factor (k/c) in the last equation implies that the cells keeping the ECC parity bits are considered (together with the cells of excluded rows and columns) as an overhead rather than the useful capacity.

(ii) Memory blocks are symmetric ($n = m$ and $a = b$). Such simple organization is preferable from the system architecture standpoint, while giving nearly the best density results.

(iii) The block array area is

$$A_{\text{array}} = \max[A_{\text{cells}}, A_{\text{interface}}] = \max[(n+a)^2(2F_{\text{nano}})^2, (n+a)(2\beta F_{\text{CMOS}})^2], \quad (23)$$

where the second term is the total area of $2(n+a)$ relay cells and pins necessary for contacting $(n+a)$ word and $(n+a)$

[9] The necessary size of this auxiliary mapping memory is relatively small, close to $(n+a)\log_2(n+a)+(m+b)\log_2(m+b)$ bits per each $(n+a)(m+b)$-bit block it serves. In contrast, an arbitrary cell renumbering would require the mapping memory of size $\sim(n+a)(m+b)\log_2(n+a)\log_2(m+b)$, i.e. larger that the block itself.

bit nanowires. (As figure 2(b) shows, $(2\beta F_{\rm CMOS})^2$ gives the area of *two* such cells.) Note that $A_{\rm array}$ is determined by $A_{\rm cells}$ if $(n+a) > R^2$ (where $R \equiv \beta F_{\rm CMOS}/F_{\rm nano}$); in this case the CMOL interface is $(n+a)/R^2$-fold redundant; see footnote 3. In the opposite case, $(n+a) < R^2$, the array area is determined by that of the interface that allows one to address only a fraction of nanodevices formed on that area[10].

(iv) For the CMOS relay cell area, we have used an estimate of $5F_{\rm CMOS}^2$, resulting in $\beta = \sqrt{5/2} \approx 1.6$. We believe this is a fair estimate for the cells shown in figure 3(a) implemented in a style similar to the usual NAND flash memory cell [31, 38]. If this estimate seems too optimistic, note that it does not affect the results in the most realistic case $(n+a) > R^2$; see equation (23).

(v) The mapping table, necessary for the translation of external bit addresses to their new physical addresses after the block reconfiguration, is implemented in the CMOS subsystem, and takes area

$$A_{\rm mapping\ table} = N_{\rm mapping\ table}(2F_{\rm CMOS})^2, \qquad (24)$$

where the number of bits in the table is given by

$$N_{\rm mapping\ table} = 2n \begin{cases} \lceil \log_2 \sqrt{n+a} \rceil + \lceil \log_2 \sqrt{n+a} \rceil, \\ \quad \text{for } (n+a) \leqslant R^2, \\ \left\lceil \log_2 \dfrac{n+a}{R} \right\rceil + \left\lceil \log_2 \dfrac{n+a}{R} \right\rceil, \\ \quad \text{for } (n+a) > R^2, \end{cases} \approx 2n \log_2 (n+a). \qquad (25)$$

This formula reflects the fact that the memory should keep physical addresses of the CMOS relay cells serving both word-line and bit-line nanowires (figure 2), each with n entries.

(vi) Each block needs four cell address decoders (figure 1) for the selection of two relay cells. For keeping open the option of nanowire and pin repair (see footnote 3), each decoder should have the output width equal that of the array, i.e., $(n+a)/R$ wires. Assuming that the circuit is implemented as the tree decoder [38], it requires approximately $2(n+a)/R$ transistors, and the pitch $2\beta F_{\rm CMOS}$ of the CMOS nanowiring (figure 2(b)) should be sufficient to house the decoder output pitch. Another linear size ('depth') of the tree decoder with complementary wires may be estimated as

$$d_{\rm cell\ decoder} = 4F_{\rm CMOS} \times \log_2 \frac{n+a}{R} + d_{\rm drive/sense}, \qquad (26)$$

where the last term reflects the contribution from line drivers and sense amplifiers. This term may be estimated by assuming that the area $d_{\rm drive/sense} \times 2\beta F_{\rm CMOS}$ is that of a four-transistor logic gate (e.g., two inverters). The latter area is always close to 320 $(F_{\rm CMOS})^2$ [39]. Now, since the corner area between the adjacent cell decoders (figure 1) can hardly be used effectively, three contributions to equation (21) may be written together as

$$A_{\rm array} + A_{\rm cell\ decoders} + A_{\rm drive/sense} = \left(\sqrt{A_{\rm array}} + 2d_{\rm cell\ decoder} + d_{\rm drive/sense}\right)^2. \qquad (27)$$

(vii) Finally, the application of the decoder area estimates given above to block decoders and to mapping table decoders shows that their contribution to equation (21) is negligible for all examples we have studied.

5. Results

Requiring that the total yield Y, calculated as discussed in section 3, is fixed at a certain level, we can use the formulae of section 4 to calculate the chip area A necessary to achieve a certain useful bit capacity N, and hence the area per useful bit, A/N. The last number, normalized to the CMOS half-pitch area,

$$a \equiv \frac{A}{N(F_{\rm CMOS})^2}, \qquad (28)$$

is a very convenient figure of merit that depends only on the ratio $F_{\rm CMOS}/F_{\rm nano}$ rather than on the absolute parameters of the fabrication technology. Another parameter affecting a is the size n of a single memory block (or, in other words, the number of blocks L at a fixed useful memory capacity N; see equation (22)). Figure 7 shows, by the top orange line, this dependence for a typical case ($F_{\rm CMOS}/F_{\rm nano} = 10$, bad bit fraction $q = 3 \times 10^{-3}$).[11] The plot shows that as the block size n is increased, some overheads are reduced: most importantly, the cell address decoder area that is logarithmic in $(n+a)$; see equation (26). On the other hand, the redundancy overhead starts to grow quickly, since the probability to have a completely good bit row or column without repair drops exponentially, as $q^{(n+a)}$. As a result, there always exists an optimum block size (and hence an optimum number L of blocks at fixed N) for which the useful bit density is minimal. For example, in the case shown in figure 7, the optimal n is 512, and at $N = 4$ Tb the optimal L is close to 16 millions.

Figure 8 shows similar curves (for the total cell area only) for two realistic values of $F_{\rm CMOS}/F_{\rm nano}$ (10 and 3.3) and for several values of the single bit yield, while figure 9 shows the results optimized over the block size, as functions of bad bit fraction, for $c = 137$, $k = 128$.[12] The results show that hybrid memories can indeed be denser than purely semiconductor memories, but for that the single-bit yield has to exceed a certain minimum value. For example, in the case $F_{\rm CMOS}/F_{\rm nano} = 10$, the hybrid memories can overcome a perfect CMOS memory only if the fraction of bad bits is below ~15%, even using the Exhaustive Search algorithm of bad bits exclusion, which may require an impracticably long time. For

[10] Our final results (see figures 7–10) show that the latter case is not very realistic, since for small array size $(n+a)$ the cell decoders give too much area overhead per bit.

[11] The dependence of a on the memory capacity N, is relatively weak. For our illustrations (figures 7–11) we have selected the values of N that would allow fitting a perfect memory on a 1 cm^2 chip. The dependence on the final yield Y (within reasonable limits, e.g., between 10 and 90%) is also weak; see figures 6 and 10, and the numbers at the top horizontal axis of figure 7.

[12] Our calculations using various Hamming codes with $k \leqslant 128$ for the Repair Most algorithm have shown that (137, 128) is the best code in terms of redundancy usage. For example, the shorter code (12, 8) provides better error correction, but it has to be replicated 16 times to cover 128 data bits. This is why, from the real estate standpoint, it is beneficial to use a longer code and allocate the saved real estate for extra spare lines of memory blocks.

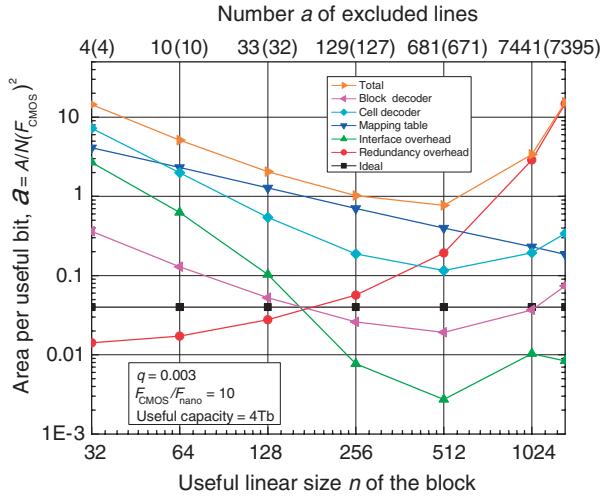

Figure 7. A typical dependence of the total chip area a per useful bit (top curve), and its major components, on the block size, providing the final memory yield Y of 90%. The horizontal line (square points) shows the 'ideal' case when the chip area $A = A_{\text{ideal}} \equiv N(2F_{\text{nano}})^2$. Most of the shown components of the total area are specified by equation (21). The redundancy overhead is defined as $A_{\text{red}} \equiv (M - N)(2F_{\text{nano}})^2$, while the interface overhead as $A_{\text{int}} \equiv A_{\text{array}} - M(2F_{\text{nano}})^2$, where $M = L(n + a)(m + b)$ is the total number of memory cells. The numbers at the top horizontal axis show the number a of excluded lines necessary to achieve the 90% yield (in parentheses, the 10% yield).

the simple and fast Repair Most algorithm, the bad bit fraction should be reduced to ∼2%. If one wants to obtain an order-of-magnitude density advantage from the transfer to hybrid memories (such a goal seems natural for the introduction of a novel technology), the numbers given above should be reduced to approximately 2% and 0.1%, respectively.

6. Uniform (semiconductor) memory

Since the bit yield requirements for the hybrid memories, cited in the previous section, do not look overly optimistic, it makes sense to carry out a similar estimate for a purely semiconductor memory with the traditional architecture (several code words stored in a single row of a memory block).

We assume the same defect model (section 2.2) and the same reconfiguration algorithms (section 3) as for the hybrid memory. For the Repair Most algorithm augmented with Hamming-code error correction, repeating the argumentation of section 3.4, we get

$$y = [(1 - q_f)^c + cq_f(1 - q_f)^{c-1}]^{nm/c}, \tag{29}$$

so that the total yield $Y = y^L$ is again expressed by equation (13).

For the ECC-enhanced Exhaustive Search, we repeat the calculation steps leading to equation (19), adjusted for the present case (each code word is located completely in one row). Since all permutations of subarrays and defect patterns are taken within one block, values of t and h are the same as in equation (8). Similarly to section 3.5, the total number of correctable defect patterns for a given subarray is derived by first finding the number of correctable defect patterns provided that the array of the size $n \times m$ has exactly

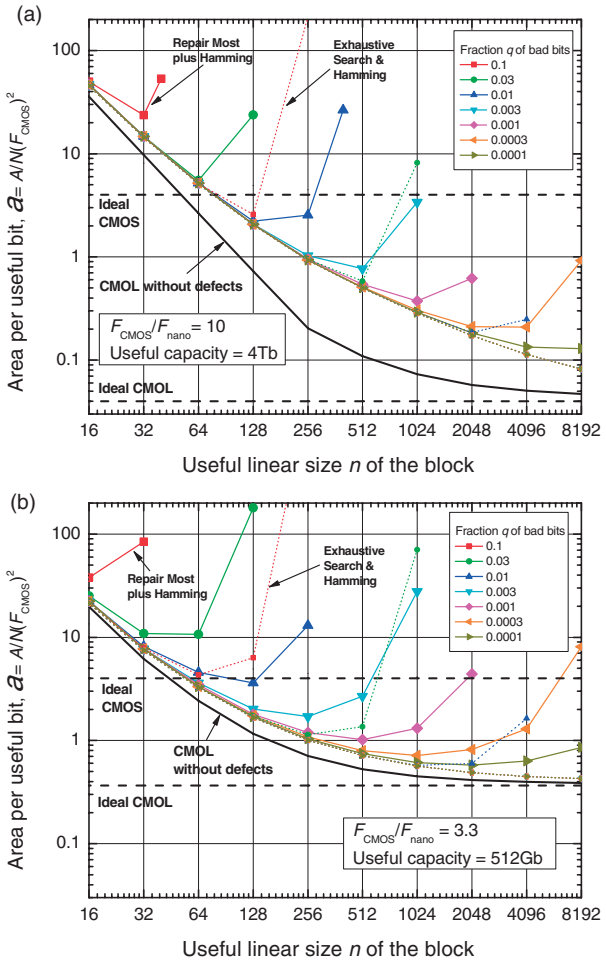

Figure 8. The reciprocal memory density (area per useful bit) for two values of the $F_{\text{CMOS}}/F_{\text{nano}}$ ratio, (a) 10, and (b) 3.3, at several values of the single device yield. The dashed lines show the single bit footprint, i.e., the reciprocal memory density in the ideal case (no bad devices, no peripheral circuits), for CMOS and nanodevice implementations.

l defects (equation (17)) and then summing over the range of l (equation (18)), i.e.

$$y_{\max}(d) = \max\left[\frac{t}{h}\sum_{i=1}^{t}\sum_{l=\max[0,d-s]}^{\min[nm/c,d]}\binom{s}{d-l}\binom{\frac{nm}{c}}{l}c^l, 1\right], \tag{30}$$

The upper bound on the yield of a whole memory is then given by

$$Y_{\max} = (y_{\max})^L = \left[\sum_{d=0}^{s} y_{\max}(d)\binom{(n+a)(m+b)}{d} \times q^d(1-q)^{(n+a)(m+b)-d}\right]^L. \tag{31}$$

Figure 10 shows the defect tolerance potentials of these two techniques, calculated from equations (13) and (31), respectively, for the arrays with $m = nc/k$ (giving n^2 useful bits in array) and $a = b$. Comparing these results to those shown in figure 6, one can clearly see the advantage of the array reconfiguration and Hamming-code error-correction synergy: the total yield is substantially better than in the case when these techniques are applied separately.

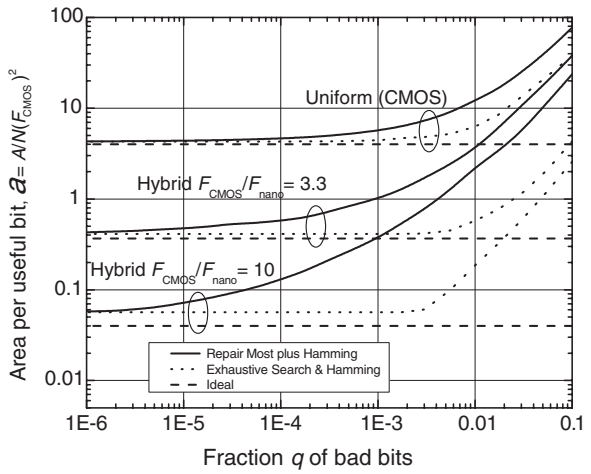

Figure 9. The area per useful cell after the block size optimization, as a function of single bit yield, for hybrid and purely semiconductor memories.

Proceeding to the memory area evaluation, we first of all note that each array requires only one row and one column decoder. Next, the uniform memory does not necessarily need a mapping table, since simple nanofuse-nanoshort circuitry [40, 41] enables a simple bypass of defective rows and columns. (The classic redundancy implementations [15] are not acceptable because they are limited to just a few spare lines.) By incorporating the line steering structure of the decoders, the total area of the uniform memory may be calculated as

$$A = A_{\text{block decoders}} + L \times [A_{\text{array}} + A_{\text{row decoder}} + A_{\text{column decoder}} + A_{\text{drive/sense}}]. \quad (32)$$

Here, the array area is just $A_{\text{array}} = (n + a)^2(2F_{\text{CMOS}})^2$. The decoders' area overhead (again assuming tree decoders) is determined by the products of the array's linear size and decoder depths

$$d_{\text{row decoder}} = 2F_{\text{CMOS}} \times (2\log_2 n + a),$$

$$d_{\text{column decoder}} = 2F_{\text{CMOS}} \times \left(2\log_2 \frac{m}{c} + b\right) + d_{\text{sense/driver}}.$$

This expression implies that the bypass implementation requires one additional lithographic wire for each redundant line.

Figure 11 and the top lines in figure 9 show the results of calculations using these formulae. They indicate that for the bit yield typical for present-day memory technologies ($q \sim 10^{-6}$), the area overhead is virtually negligible, because of the possibility to use large block sizes (e.g., 8 K × 8 K bits). However, as the fraction of bad memory cells is increased (as will certainly happen with scaling down their size), the purely semiconductor memories will experience virtually the same 'swelling' as the hybrid memories. For example, for the realistic Repair Most technique, augmented with the Hamming-code error correction, at $q = 0.1\%$ the effective cell area is about 40% larger than its physical area, while at $q = 1\%$ it becomes almost three times larger. Moreover, figure 9 clearly shows that at the same bit yield, the hybrid memories can clearly provide higher memory density.

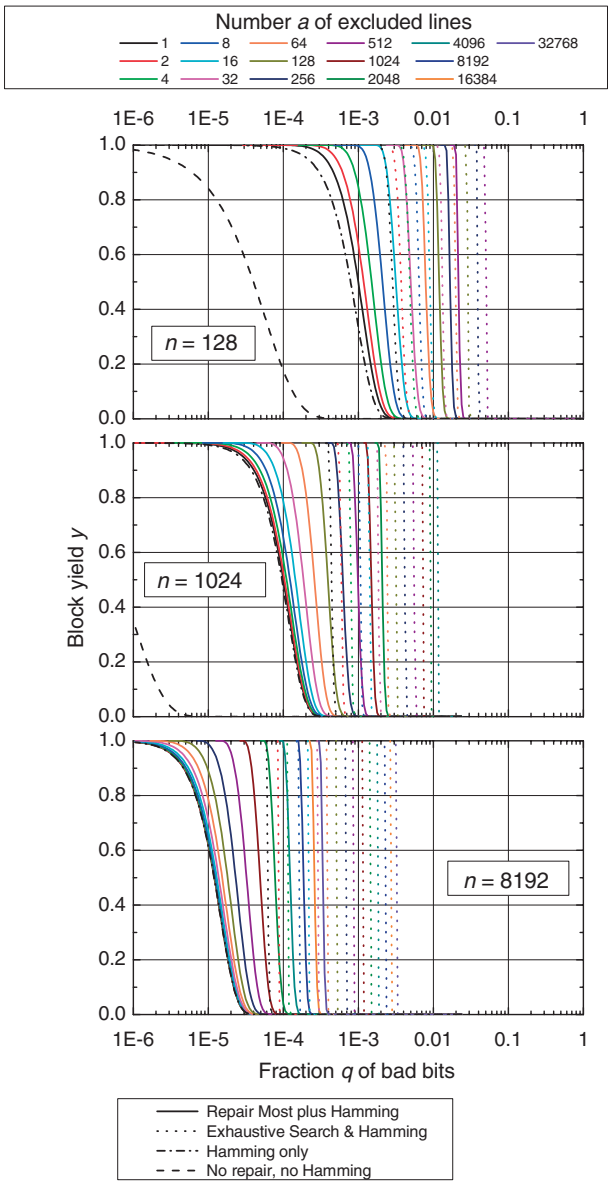

Figure 10. The plots similar to those of figure 6 (but both with the Hamming-code error correction), for a purely semiconductor memory.

7. Discussion

To summarize, this study shows a clear potential advantage of hybrid memories in useful density, provided that the nanodevice fabrication technology offers a reasonable yield of single cells. Moreover, some nanodevices (e.g., single-electron latching switches [19, 20, 28]) promise nonvolatile operation[13]. All these prospects would, however, be insubstantial if the addition of the nanowire/nanodevice layer to CMOS chips resulted in unacceptable power dissipation or memory access slowdown. A detailed analysis and optimization of the power-to-speed tradeoff are beyond the scope of the present work, so we will only provide a crude

[13] Chemical synthesis of single-molecule versions of these devices may enable the implementation of graded ('crested') barriers [42], which would allow combining a multi-year retention time with nanosecond-scale write/erase speed.

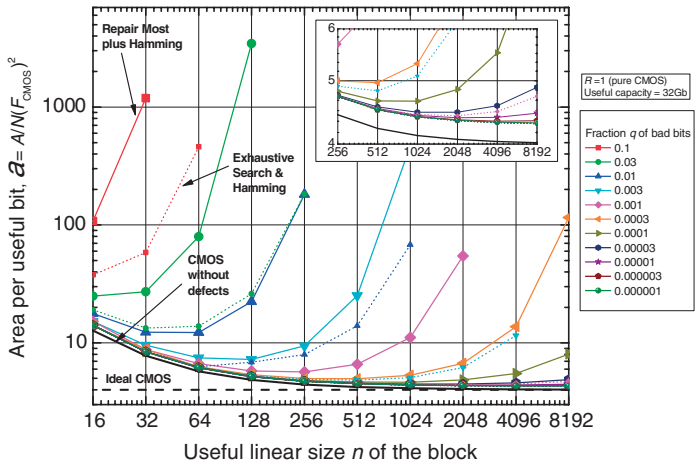

Figure 11. The plots similar to those of figure 8, for a purely semiconductor memory.

estimate showing that these challenges may be met by a proper choice of nanodevice resistance.

Let us consider, as an example, the case $q = 10^{-3}$ and $F_{\rm CMOS}/F_{\rm nano} = 10$. According to figure 8(b), the Hamming-code error correction on top of the simple Repair Most reconfiguration allows a 4 Tb memory using $L = 4$ M of 1 K × 1 K blocks. (This optimum is achieved at $a = b = 681$, so that $(n + a) \approx 1.7 \times 10^3$.) We have used the FastCap code [43] to calculate the capacitance of a nanowire fragment of length $2F_{\rm nano}$ for a simple geometric model of the nanodevice layer (figure 2), in which the both the width and the thickness of the wire, and both the vertical and the horizontal distances between the nanowires, are all equal to $F_{\rm nano}$. Such a calculation yields $C/F_{\rm nano} \approx 0.48 \times 10^{-10}\varepsilon$ (F m^{-1}), where ε is the relative dielectric constant of the insulating environment. (This number is in reasonable agreement with numerical and experimental results for isolated nanowires [44].) With $\varepsilon = 3.9$ (corresponding to SiO_2) and $F_{\rm nano} = 3$ nm (a substantially smaller half-pitch would result in unacceptable quantum tunnelling between nanowires), this calculation gives capacitance $C \approx 1.9$ fF for a nanowire passing through the whole block[14]. With the open nanodevice resistance $R_{\rm open} = 1$ MΩ, we find that the nanowire recharging through the resistance (the longest process in the nanodevice subsystem[15]) would take just a few nanoseconds, quite comparable with the typical access time of contemporary semiconductor memory chips.

In order to estimate the power dissipation in the nanodevices for the same value of $R_{\rm open}$, we should take into account at least three components. The first component, the dynamic power, may be estimated as $P_{\rm dyn} < 2L^{1/2}CV^2f$, where the factor $L^{1/2}$ is the upper bound for the number of blocks addressed simultaneously; see figure 1. (In practice, this number may be considerably less than $L^{1/2}$ because of the memory bus width limitations.) Assuming that the read/write voltages V are close to 1 V (a typical value for room-temperature single-electron devices [7]), we get $P_{\rm dyn} \lesssim 5$ mW even at a high access frequency f of 300 MHz. The static power $P_{\rm open} < 2L^{1/2}V^2/R_{\rm open}$, that is due to dissipation in simultaneously open nanodevices (one device per block), is equally negligible: with the parameters used above we get $P_{\rm sta} \lesssim 4$ mW. More substantial may be the static power dissipation $P_{\rm leak} < 2L^{1/2}(n + a)(V/2)^2/R_{\rm leak}$ due to current leakage through 'semi-selected' nanodevices connected to both (bit and word) addressed nanowires, where $R_{\rm leak}$ is the effective resistance of nanodevices in the closed (e.g., Coulomb blockade) state. Requiring that $P_{\rm leak} < 0.1$ W, for our case we get the condition $R_{\rm leak} \gtrsim 10$ MΩ, i.e. $R_{\rm leak}/R_{\rm open} \gtrsim 10$, that seems quite realistic, at least for molecular single-electron transistors [1–5].

To summarize, our results show that a simple combination of array reconfiguration (bad line exclusion) and Hamming-code error-correction techniques may make hybrid CMOS/nanodevice memories tolerant to a substantial fraction of bad nanodevices. However, the fabrication technology still has to reduce this fraction to below ∼0.1% before terabit-scale memory chips, exceeding purely semiconductor memories in bit density by an order of magnitude, become possible. Possibly, this threshold may be moved up by almost an order of magnitude by the development of novel reconfiguration algorithms that would combine the simplicity and speed of the Repair Most exclusion with a higher repair quality (comparable to that of the Exhaustive Search).

[14] Note that the linear size, $2F_{\rm nano} \times (n + a) \approx 10$ μm, of the 1 Mb block corresponds to useful memory density close to 1 Tb cm^{-2}.

[15] The internal switching of a single-electron transistor with such resistance is much shorter [7]. Also, resistance of a metallic nanowire of this cross-section ($F_{\rm nano}^2 \approx 10$ nm^2) and length ($2F_{\rm nano} \times (n + a) \approx 10$ μm) is of the order of 10^5Ω, i.e. much less than the accepted value of $R_{\rm nano}$. Note, however, that the use of molecular nanowires would make the access time unacceptably slow.

Acknowledgments

Fruitful discussions with A Pacelli are gratefully acknowledged. The work has been supported by the AFOSR, DOE, and NSF.

References

[1] Park H *et al* 2000 Nanomechanical oscillations in a single-C_{60} transistor *Nature* **407** 57–60

[2] Shorokhov V V, Johansson P and Soldatov E S 2002 Correlated electron tunnelling in the single-molecule nanosystems *J. Appl. Phys.* **91** 3049–53

[3] Zhitenev N B, Meng H and Bao Z 2002 Conductance of small molecular junctions *Phys. Rev. Lett.* **88** 226801
[4] Park J *et al* 2002 Coulomb blockade and the Kondo effect in single-atom transistors *Nature* **417** 722–5
[5] Kubatkin S *et al* 2003 Single-electron transistor of a single organic molecule with access to several redox states *Nature* **425** 698–701
[6] Kuekes P *et al* 2000 Molecular wire crossbar memory *US Patent Specification* #6 128 214
[7] Likharev K 2003 Electronics below 10 nm *Nano and Giga Challenges in Microelectronics* ed J Greer *et al* (Amsterdam: Elsevier) pp 27–68
[8] Stan M *et al* 2003 Molecular electronics: from devices and interconnect to circuits and architecture *Proc. IEEE* **91** 1940–57
[9] Chen Y *et al* 2003 Nanoscale molecular-switch crossbar circuits *Nanotechnology* **14** 462–8
[10] Zhong Z *et al* 2003 Nanowire crossbar arrays as address decoders for integrated nanosystems *Science* **302** 1377–9
[11] Li C *et al* 2004 Multilevel memory based on molecular devices *Appl. Phys. Lett.* **84** 1949–51
[12] Fendler J H 2001 Chemical self-assembly for electronics applications *Chem. Mater.* **13** 3196–10
[13] Tour J 2003 *Molecular Electronics* (Singapore: World Scientific)
[14] Prince B 1991 *Semiconductor Memories* 2nd edn (Chichester: Wiley)
[15] Chakraborty K and Mazumder P 2002 *Fault-Tolerance and Reliability Techniques for High-Density Random-Access Memories* (Upper Saddle River, NJ: Prentice-Hall)
[16] Huang C *et al* 2003 A built-in redundancy analysis for memory yield improvement *IEEE Trans. Reliab.* **52** 386–99
[17] Stapper C and Lee H 1992 Synergistic fault-tolerance for memory chips *IEEE Trans. Comput.* **41** 1078–87
[18] Koren I and Singh A D 1990 Fault tolerance in VLSI circuits *IEEE Comput.* **23** 73–83
[19] Likharev K 2004 CMOL: a new concept for nanoelectronics *Invited Talk at the 12th Int. Symp. on Nanostructures: Physics and Technology (St Petersburg, Russia, July 2004)* extended abstract available online at http://rsfq1.physics.sunysb.edu/~likharev/nano/SPb.pdf
[20] Likharev K and Strukov D 2004 CMOL: devices, circuits, and architectures *Introducing Molecular Electronics* ed G Cuniberti *et al* (Springer: Berlin) at press
[21] Kuekes P and Williams S 2001 Demultiplexer for a molecular wire crossbar network (MWCN DEMUX) *US Patent Specification* #6 256 767
[22] Luyken R and Hofmann F 2003 Concept for hybrid CMOS-molecular non-volatile memories *Nanotechnology* **14** 273–6
[23] DeHon A *et al* 2003 Stochastic assembly of sublithographic nanoscale interfaces *IEEE Trans. Nanotechnol.* **2** 165–74
[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 Optical Engineering Press) pp 85–109
[26] Jensen K L 1999 Field emitter arrays for plasma and microwave source applications *Phys. Plasmas* **6** 2241–53
[27] Türel Ö and Likharev K 2003 CrossNets: possible neuromorphic networks based on nanoscale components *Int. J. Circuit Theory Appl.* **31** 37–53
[28] Türel Ö, Lee J-H, Ma X and Likharev K 2004 Neuromorphic architectures for nanoelectronic circuits *Int. J. Circuit Theory Appl.* **32** 277–302 http://rsfq1.physics.sunysb.edu/%7Elikharev/nano/IJCTA04.pdf
[29] Fölling S, Türel Ö and Likharev K K 2001 Single-electron latching switches as nanoscale synapses *Proc. 2001 Int. Joint Conf. on Neural Networks* (Mount Royal, NY: Int. Neural Network Soc.) pp 216–21
[30] Heath J R, Kuekes P K, Snider G S and Williams R S 1998 A defect-tolerant computer architecture: opportunities for nanotechnology *Science* **280** 1716–21
[31] Brown W D and Brewer J E (ed) 1998 *Nonvolatile Semiconductor Memory Technology* (Piscataway, NJ: IEEE)
[32] Fifield J A and Stapper C H 1991 High-speed on-chip ECC for synergetic fault-tolerant memory chips *IEEE J. Solid-State Circuits* **26** 1449–52
[33] Abramowitz M and Stegun I A (ed) 1972 *Handbook of Mathematical Functions* (New York: Dover) p 260, 941
[34] Zarankiewicz K 1951 Problem 101 *Colloq. Math.* **2** 301
[35] Ballobás B 1998 *Modern Graph Theory* (Berlin: Springer) p 112
[36] Hamming R W 1950 Error detecting and error correcting codes *Bell Syst. Tech. J.* **26** 147–60
[37] Hsiao M Y 1970 A class of optimal minimum odd-weight-column SEC-DED codes *IBM J. Res. Dev.* **14** 395–401
[38] Rabaey J, Chandrakasan A and Nikolic B 2002 *Digital Integrated Circuits* 2nd edn (Upper Saddle River, NJ: Prentice-Hall)
[39] *International Technology Roadmap for Semiconductors (ITRS)* 2003 Edition, available online at public.itrs.net/
[40] Korotkov A N 2002 Analysis of integrated single-electron memory operation *J. Appl. Phys.* **92** 7291–5
[41] Kothandaraman C, Iyer S K and Iyer S S 2002 Electrically programmable fuse (eFUSE) using electromigration in silicides *IEEE Electron Device Lett.* **23** 523–5
[42] Likharev K K 2000 NOVORAM: a new concept for fast, bit-addressable nonvolatile memory based on crested barriers *IEEE Circuits Devices* **16** 16–21
[43] Nabors K, Kim S and White J 1992 Fast capacitance extraction of general three-dimensional structures *IEEE Trans. Microw. Theory Tech.* **40** 1496–506 See also http://www.rle.mit.edu/cpg/research_codes.htm
[44] Lu J G, Hergenrother J M and Tinkham M 1997 Effect of island length on the Coulomb modulation in single-electron transistors *Phys. Rev.* B **57** 4591–8