Nanotechnology **13** (2002) 357–362 PII: S0957-4484(02)31135-8

Fault-tolerant techniques for nanocomputers

K Nikolić, A Sadek and M Forshaw

Department of Physics and Astronomy, University College London, London WC1E 6BT, UK

E-mail: k.nikolic@ucl.ac.uk

Received 26 March 2002, in final form 24 April 2002
Published 23 May 2002
Online at stacks.iop.org/Nano/13/357

Abstract
The proposed nanometre-sized electronic devices are generally expected to show an increased probability of errors both in manufacturing and in service. Hence, there is a need to use fault-tolerant techniques in order to make reliable information processing systems out of those devices. Here we examine and compare four fault-tolerant techniques: R-fold multiple redundancy; cascaded triple modular redundancy; von Neumann's multiplexing method; and a reconfigurable computer technique. It is shown that the reconfiguration technique is the most effective technique, able to cope with manufacturing defect rates of the order of 0.01–0.1, but the technique requires enormous amounts of redundancy, of the order of 10^3–10^5. However, in the case of transient errors, multiple modular redundancy and multiplexing are the only feasible options.

1. Introduction

The invention of nanometre-scale devices should eventually permit extremely large scales of integration, of the order of 10^{12} devices per chip. At the present time, only a handful of truly nanoscale or molecular scale logic or memory devices exists, and the question of how to assemble 10^{12} such devices on a working chip seems academic. Nevertheless, it is almost certain that it will be very difficult to make even small nanoscale circuits—for example having one hundred devices—with any degree of certainty. Furthermore, it is probable that the proposed nanoelectronic devices will be more fragile than conventional devices, and will be sensitive to external influences such as radiation related effects (radioactive decay or cosmic rays), high temperature, electromagnetic interference, parameter fluctuations, etc. Hence, if progress is to be made in nanoelectronics, fault-tolerant architectures will certainly be necessary in order to produce reliable systems that are immune to manufacturing defects and to transient errors.

In general, errors can be split into permanent and transient errors. Permanent ('hard') errors may occur during manufacture or during the lifetime of the computer. The current manufacturing strategy is mainly geared to improving the reliability of the manufacturing process, so that only a modest proportion of chips fail during test. This requires that the manufacturing failure rate of individual devices (including transistors and wiring connections) be very low. For example, the manufacturing failure rate per device for present-day CMOS is approximately 10^{-7}–10^{-6}, while the failure rate for highly experimental devices such as molecular transistors is currently just below unity. The increasing miniaturization of CMOS technology is causing the chip failure rate to increase as both the number of devices and the individual device failure rates increase. Nobody knows what the eventual probability of failure during manufacturing, p_f, of nanoscale or molecular scale devices will be, but it is widely acknowledged that it will be significantly poorer than that of present-day transistors. Since it should eventually be possible to put more than 10^{12} molecular-sized devices on a 1 cm^2 chip, it is evident that advanced fault-tolerant strategies will have to be devised.

Several techniques exist for overcoming the effects of inoperative devices. All of them use the concept of redundancy (in resources or in time); most of them rely in some way on the availability of multiple copies of devices or circuits. There are no conceptual differences in the analysis of transient errors and manufacturing (or permanent) defects. However, from a practical point of view some techniques are, in general, more efficient for dealing with transient errors and some are for permanent defects.

Here we are mainly concerned with resource redundancy (temporal redundancy can be shown to be formally equivalent to resource redundancy). We extend the theory of fault-tolerance to nanocomputers, which are likely to suffer from large numbers of defects in manufacturing, and which may

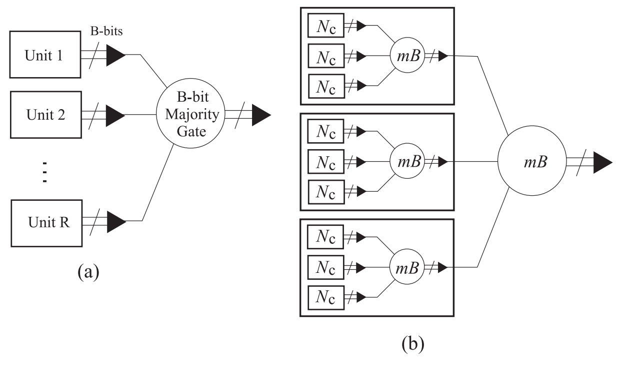

Figure 1. The basic elements of fault-tolerant techniques: (a) RMR; (b) CTMR.

also be subject to frequent errors in operation. We analyse four techniques:

- R-fold modular redundancy [1] (RMR, where $R = 3, 5, 7, 9, \ldots$);
- cascaded triple modular redundancy (CTMR, of the ith order) [2];
- NAND multiplexing [3];
- reconfiguration [4, 5].

The first two techniques are generalizations of the well-known triple modular redundancy (TMR) method [1, 2]. At present this technique is mainly used to correct for transient errors while a computer is being used. The third technique, which we call the NAND multiplexing method, was originally proposed by von Neumann [3] at the time when early computers were introduced which were notoriously unreliable. Finally, we develop a simplified theory of reconfigurable computers and we present derived results. This technique represent a vastly more sophisticated version of the 'test–find–replace' concept and it is mainly suitable for tackling manufacturing defects rather than transient errors. The effectiveness of this idea was successfully demonstrated a few years ago on a massively parallel computer ('Teramac' custom configurable computer) built at Hewlett-Packard Laboratories [4], but no theoretical analysis of this system has been published. However, if the circuit complexity is too high—for example, in the logic circuitry found in chips such as workstations or PCs—then even this strategy starts to become unworkable if there are too many defects.

2. *R*-fold modular redundancy

The concept of TMR is to have three units working in parallel, and to compare their outputs with a majority gate. Then TMR can provide an assemblage that behaves like one of its constituent components, but with an improved probability of working. The trade-off is that instead of n devices, at least $3n$ devices plus a majority gate are needed to make this new 'unit'. RMR is a generalization of TMR where instead of three we have R units working in parallel (see figure 1(a)).

In our analysis we have assumed a chip with N devices, with the probability p_f of an individual device failing. The probability P_{fail} of a complete chip failing during the working lifetime is minimized for some optimum cluster size (N_c), under the condition $N_c p_f \ll 1$. A cluster (or module)

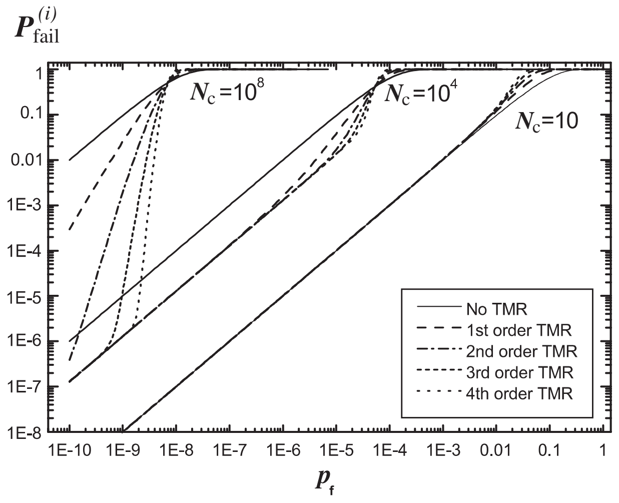

Figure 2. The probability of obtaining a defective (TMR/CTMR) unit ($P^{(i)}_{fail}$) (with $3^i N_c$ devices) with B-bit outputs, as a function of the individual device failure probability p_f, using imperfect majority gates. The groups of curves are for: $N_c = 10$, $B = 1$, $m = 10$ (right); $N_c = 10^4$, $B = 64$, $m = 20$ (centre); and $N_c = 10^8$, $B = 64$, $m = 20$ (left).

represents the unit (with N_c devices) which is replicated R times; outputs from each of the units are compared in a majority gate and the output is determined. It would be a more realistic approach to fix the logical depth of the system (to say $D = 10$) and then have determined values for N_c for each R (since the total number of devices must be $N = RN_cD$). However, in the proposed model we calculate the maximum theoretical improvement offered by this technique. Imperfect majority gates have B outputs and mB devices.

First we assume that a module of N_c devices works only if every single device in the module works:

$$P^{module}_{works} = (1 - p_f)^{N_c} \approx \mathrm{e}^{-N_c p_f} \qquad (p_f \ll 1). \tag{1}$$

Now the probability that a module fails, if $N_c p_f \ll 1$, is $P^{module}_{fails} = 1 - P^{module}_{works} \approx N_c p_f$. A group consisting of R modules and a majority gate works correctly when at least $(R+1)/2$ modules work correctly and when the majority gate also works (the probability is $P_{mg,w} \approx \mathrm{e}^{-mBp_f}$). The number of devices in a group is $RN_c + mB$, so the total number of groups is $N_{groups} = N/(RN_c + mB)$. The chip fails if any of the groups fail, hence the probability that the whole chip with N devices fails is (again when $P^{group}_f \ll 1$) approximately

$$P^{chip}_{fail} = \frac{N}{RN_c + mB}[C(N_c p_f)^{(R+1)/2} + mBp_f] \tag{2}$$

where C is the binomial factor

$$C = \binom{R}{(R-1)/2}. \tag{3}$$

Here we assume that the errors in each module are uncorrelated, i.e. that common-mode (or common cause) failures [6] are not present in the redundant system. The equation $\mathrm{d}P^{chip}_{fail}/\mathrm{d}N_c = 0$ gives the optimum module size (N_c) for a given p_f, which is substituted in equation (2), yielding the minimum chip failure probability. Table 1 gives the results for some values of R.

Table 1. Efficiency of RMR at reducing chip failure rates ($N_c p_f \ll 1$).

R	$P_{chip}^{fail} = \epsilon$	$p_{f,max}$	Optimum module size N_c
1	$N_{total} p_f$	ϵ / N_{total}	N_{total}
3	$N_{total} p_f \times 1.15(mBp_f)^{1/2}$	$\frac{0.9}{(mB)^{1/3}}(\epsilon/N_{total})^{2/3}$	$\left(\frac{mB}{3}\right)^{1/2}(p_f)^{-1/2}$
5	$N_{total} p_f \times 0.8(mBp_f)^{2/3}$	$\frac{1.1}{(mB)^{2/5}}(\epsilon/N_{total})^{3/5}$	$\left(\frac{mB}{20}\right)^{1/3}(p_f)^{-2/3}$
7	$N_{total} p_f \times 0.6(mBp_f)^{3/4}$	$\frac{1.3}{(mB)^{3/7}}(\epsilon/N_{total})^{4/7}$	$\left(\frac{mB}{105}\right)^{1/4}(p_f)^{-3/4}$
9	$N_{total} p_f \times 0.5(mBp_f)^{4/5}$	$\frac{1.5}{(mB)^{4/9}}(\epsilon/N_{total})^{5/9}$	$\left(\frac{mB}{500}\right)^{1/5}(p_f)^{-4/5}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
Limit	$\sim N_{total} p_f \times (mBp_f)$	$\sim \frac{1}{(mB)^{1/2}}(\epsilon/N_{total})^{1/2}$	$\sim 1/p_f$

3. Cascaded triple-modular redundancy

The TMR process can be repeated by combining three of the TMR 'units' with another majority gate to form a 'second-order' TMR unit with even higher reliability (a technique called CTMR, see figure 1(b)). If all three of the units work independently, then the probability of the assemblage (three units plus majority gate) working, $P_w^{(1)}$, is given by

$$P_w^{(1)} = (1 - p_f)^{mB}[P_{works}^3 + 3P_{works}^2(1 - P_{works})] \quad (4)$$

where $P_{working} \equiv P_{works}^{module}$ is given by equation (1). The performance with additional stages of TMR is obtained by repeated application of this formula and the probability that a CTMR configuration of ith order works is

$$P_w^{(i)} = (1 - p_f)^{mB}[(P_w^{(i-1)})^3 + 3(P_w^{(i-1)})^2(1 - P_w^{(i-1)})] \quad (5)$$

Figure 2 demonstrates the effectiveness of the CTMR technique. It shows that there is no advantage in using CTMR for units containing a small number of devices, when the majority gates are made from the same devices as the units that they are monitoring. However, at least in principle, improvement is possible for units with large N_c.

There are three regions in each set of curves:

(a) $N_c p_f > \ln 2$, where redundancy affords no advantage;
(b) $10^{-3} < N_c p_f < \ln 2$, where redundancy is most effective; and
(c) $N_c p_f < 10^{-3}$, where only first-order redundancy offers an advantage.

In case (b), the effectiveness of redundancy scales as a power law with the order of CTMR. The failure probability is

$$P_{fail}^{(i)} \propto (N_c p_f)^{2i}.$$

For case (c) the effectiveness of redundancy depends on the ratio mB/N_c. Starting from equation (5), it can be shown that in region (c) the failure probabilities are

$$P_f^{(0)} \equiv P_{fail}^{module} \approx N_c p_f,$$
$$P_f^{(i)} \approx \frac{mB}{N_c} N_c p_f = \frac{mB}{N_c} P_f^{(0)}, \qquad i = 1, 2, \ldots. \quad (6)$$

4. NAND multiplexing

Fifty years ago, von Neumann was the first person to consider the use of redundant components to overcome the effects of defective devices [3]. He described the now well-known technique of multiple redundancy (see above), but he also described another method, which we have called NAND multiplexing for brevity. Von Neumann showed that this method could in principle enable a circuit to work, even if the individual devices had a failure rate of ~ 0.01. The problem with this technique was that it required enormous levels of redundancy ($\sim 10^3$–10^4). Although this seems very unrealistic, very high levels of redundancy may be needed for nanocomputers, since it may be very hard to make huge numbers of nanoscale devices with good reliability. We have examined whether this technique can be used with smaller levels of redundancy, and we have also examined its performance for different circuit sizes [7].

In essence the basic technique of multiplexing is similar to RMR, but instead of having a majority gate to decide on the proper output, the output is carried on a bundle of wires, e.g. for a single bit output one would have R wires (or N_{bundle} if we use von Neumann's notation) in a bundle which carries the output to the next stage. Therefore, in this method, processing units of any size are replaced by multiplexed units containing N_{bundle} number of lines for every single input and output. Essentially, a multiplex unit consists of two stages. The first, the executive stage, performs the basic function of the processing unit in parallel. The second, the restorative stage, reduces the degradation caused by the executive stage and thus acts as a non-linear 'amplifier' of the output, see figure 3. An example for the executive stage given in figure 3 is a simple NAND (two-input gate), but it could be a unit with an arbitrary number of gates. Now the signals to and from units are not carried in single lines but in bundles ($N_{bundle} = 4$ in the example in figure 3) and the unit is replicated the appropriate number of times (there are four NAND gates in figure 3). If the inputs and processing units are perfectly reliable then the lines comprising each output should be identically stimulated (1) or unstimulated (0). However, due to errors in the input data as well as errors occurring in the processing of the inputs from faulty devices, not all of the

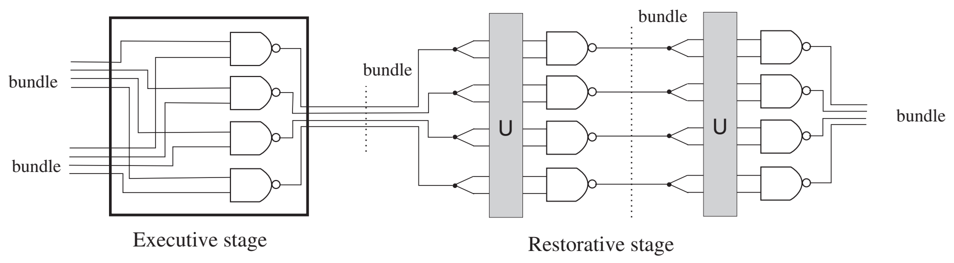

Figure 3. The basic elements of NAND multiplexing (after von Neumann [2]) fault-tolerant techniques. The specific logic gates shown are only for the purpose of illustration.

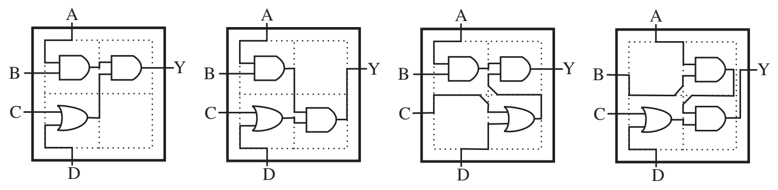

Figure 4. The basic structure for the reconfiguration technique theory (after Lach *et al* [3]). The specific logic gates shown are only for the purpose of illustration.

output lines in each output will be identically stimulated. Thus, for multiplexed networks, the final outputs are considered to be 1 if more than $(1 - \Delta)N_{bundle}$ lines are stimulated and 0 if less than ΔN_{bundle} lines are stimulated, where Δ is a critical level that is pre-defined ($0 < \Delta < 0.5$). Stimulation levels in between are considered to be undecided (consequently resulting in malfunction).

For the sake of simplifying calculations, von Neumann assumed the most basic logic gate in a chip to be the NAND gate. This is a universal logic gate and can be used to build the basic logic gates NOT and NOR (containing one and four NAND gates respectively). Thus, a NAND network equivalent can replace any conventional architecture. The use of only NAND gates in extremely large-scale integration (XLSI) architectures may also be justified in that construction is simplified through the use of identical repeating sub-units.

It is possible to develop a quantitative theory for NAND multiplexing whereby the probability distribution of the output stimulation fraction ψ, of a NAND multiplex can be calculated as a function of the input stimulation fractions, ξ and η (see figure 3), taking into account a probability of failure ε in each NAND gate. If ζ and ω are the stimulation fractions of the executive stage and intermediate restorative stage outputs, then probability distributions Φ_ζ, Φ_ω and Φ_ψ can be derived for ζ, ω and ψ, respectively. For large N_{bundle}, these distributions are approximately Gaussian and indeed the theory is only valid for large N_{bundle} due to this assumption. For details of the theory developed by von Neumann and the formulae for Φ_ζ, Φ_ω and Φ_ψ see [3]. Additional details are given in [7].

5. Reconfiguration

In 1998, a paper was published in *Science* on the 'Teramac' reconfigurable computer [4], with the proposal that this technique would be useful for overcoming manufacturing defects in nanocomputers. Unfortunately, no details of the theory are available, so that it is not possible to make quantitative estimates of its performance in general. We have therefore taken an analysis by Lach *et al* [5] as a theoretical basis, and extended the theory contained in that paper to find approximate upper bounds to the maximum allowable manufacturing defect rate, for different sizes of circuit.

Devices ('transistors'), each having a probability p_f of being defective during manufacture, are assembled in groups of N_{trans} to form a configurable logic block (CLB, shown as a sub-unit in figure 4). A number of these CLBs are grouped together, N_{CLB} at a time, to form an atomic fault-tolerant block (AFTB, larger units in figure 4). It is assumed that the AFTB can be configured to perform some basic set of operations, even though any one of its constituent CLBs may be faulty. In general, different types of AFTBs can be designed to carry out different functions, and each type may incorporate different numbers of CLBs. However, it is assumed here that all AFTBs contain the same number of CLBs.

The AFTBs are then grouped together, N_A at a time, to form a cluster which then performs some desired function. In the present context we suppose that a quite high-level function is implemented. For example, the cluster might operate as if it contained a large block of memory, or a 64-bit full adder, or a processing element for an artificial retina, or even the equivalent of a present-day workstation CPU chip. On the other hand, the cluster function could be much simpler.

One further stage is needed. It is assumed that the chip is completely filled with identical, independent copies of the cluster. Suppose that we would like to be able to manufacture chips, so that after fault-detection and reconfiguration, they have a certain probability—for example 90%—of working. We now apply a higher level of redundancy, by grouping the clusters together, R at a time, to form a supercluster. For simplicity we have assumed that it is possible to detect which, if any, of the R clusters work, and then reconfigure the supercluster so that it gives a valid output. The supercluster is then considered to be acceptable if at least one of the R clusters works. This model is greatly simplified, and other models are possible, but it is very general. Fuller results are given in [7].

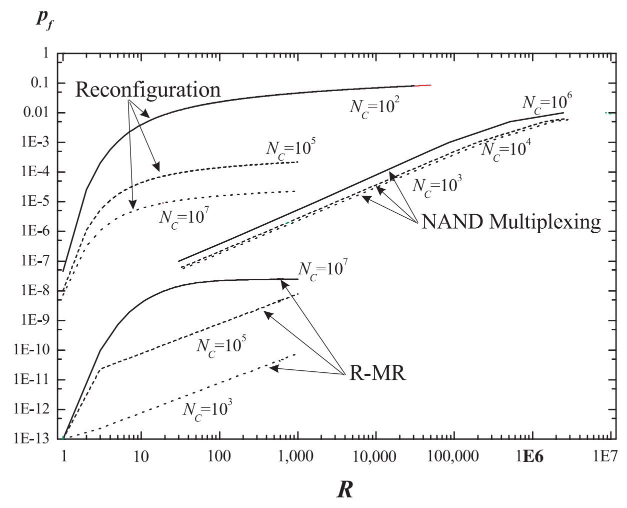

Figure 5. Allowable failure rate per device, p_f, as a function of the amount of redundancy, R, for different numbers of effective devices in a working unit (N_c). The total number of devices on the chip is $N = 10^{12}$ and the chip must work with 90% probability ($B = 64$ and $m = 20$).

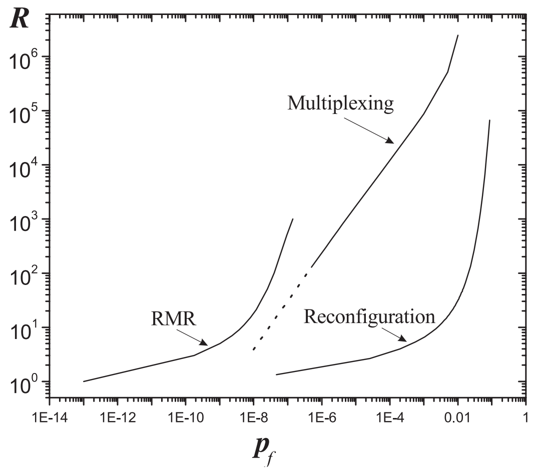

Figure 6. These curves compare three different fault-tolerant strategies (RMR, von Neumann multiplexing and a reconfigurable computer technique), applied to a hypothetical chip with 10^{12} devices (perhaps the ultimate number for a 1 cm^2 chip). The curves show the necessary level of redundancy R, as a function of the failure rate per device p_f, which ensures that the whole chip will work with a 90% probability. Starting points are at $R = 1$ (RMR), $R = 100$ (multiplexing) and $R = 1.33$ (reconfiguration). Results for the multiplexing are extrapolated for small R ($R < 100$) and presented with a broken curve, as the von Neumann's formula is not accurate in this region.

6. Results and discussion

Figure 5 illustrates the relative performance of three techniques (RMR, multiplexing and reconfigurability), applied to the rectification of manufacturing faults. The horizontal axis in figure 5 represents the redundancy R, which is effectively the number of times that a circuit has to be replicated before it becomes useful. The vertical axis is p_f, the probability of an individual device ('transistor') being defective in manufacture. The curves in figure 5 are based on the assumption that a chip with 10^{12} devices has to be made with a 90% probability of working. The numbers alongside each curve represent the number of devices in the basic circuit (that is assumed to be replicated in various ways). Thus $N_c = 100$ denotes a very small circuit (for example an elementary part of a field programmable gate array [8]). The value $N_c = 10^4$ or 10^5 might represent a moderately powerful 'neuron' or one of many small digital processors in an image processing array. Finally, $N_c = 10^7$ is intended to be representative of the number of devices on the main chips in present-day workstations.

The curves in figure 6 are representative of the approximate upper bound for each technique. For example, although the RMR technique is the least effective, with the level of redundancy of $R = 5$ we can achieve the same level of chip reliability, but with devices which are four orders of magnitude less reliable. The price for this improvement is that the effective number of devices is reduced to $N/5$ (and the p_f for each device must be smaller than 10^{-9} for $N = 10^{12}$ devices). On the other hand, the curve for the reconfigurable computer shows that this technique can in principle handle extremely large manufacturing defect rates—in the limit, even approaching unity—but only at the expense of colossal amounts of redundancy, for example, manufacturing defect rates of 0.1 (i.e. 10%). If one wishes to fabricate a chip containing the equivalent of many present-day workstations, then the device failure rate during manufacturing must be smaller than 10^{-5}. This may be difficult to achieve.

It can be seen that RMR and NAND multiplexing are in general worse than reconfiguration. However, if the dead devices cannot be located during manufacture, then a fault tolerant strategy must be adopted, which allows a chip to work, even with many faulty (either temporarily or permanently) devices. Furthermore, reconfiguration might be very time consuming for protecting against transient errors that may occur in service, and therefore demand temporary shutdown of the system until reconfiguration is performed. It may also be necessary to use NAND multiplexing if reconfiguration methods are impractical or if the probability of transient errors is very high.

RMR provides some benefits, but these are unlikely to be useful for chips with 10^{12} devices, once the manufacturing defect rate is greater than about 10^{-8} (see figure 6). Such values are almost impossible to achieve with present-day CMOS. The NAND multiplexing technique in principle would allow chips with 10^{12} devices to work, even if the fault rate is as high as 10^{-3} per device. However, as illustrated by the curve labelled 'multiplexing' in figure 6, this needs even more redundancy than the reconfiguration technique.

The implications of these results are that the future usefulness of various nanoelectronic devices may be seriously limited if they cannot be made in large quantities with a high degree of reliability. The results shows that it is theoretically possible to make very large functional circuits, even with one dead device in ten, but only if the dead devices can be located and the circuit reconfigured to avoid them. Even so, this technique would require a redundancy factor of ~10 000, i.e., a chip with 10^{12} non-perfect devices would perform as if it had only 10^8 perfect devices. If it is not possible to locate the dead devices, then one of the other two techniques would have to be used. These would require the manufacturing and lifetime failure rate for $R = 1000$ to be between 10^{-7} and 10^{-6}, which is comparable with the present-day manufacturing failure rates for CMOS transistors.

References

[1] Depledge P G 1981 Fault-tolerant computer systems *IEE Proc.* **128** 257–72
[2] Spagocci S and Fountain T 1999 Fault rates in nanochip devices *Electrochem. Soc. Proc.* **99** 354–68
[3] Von Neumann J 1955 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
[4] 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
[5] Lach J, Mangione-Smith W H and Potkonjak M 1998 Low overhead fault-tolerant FPGA systems *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **6** 212–21
[6] Mitra S, Saxena N R and McCluskey E J 2000 Common-mode failures in redundant VLSI systems: a survey *IEEE Trans. Reliab.* **49** 285–95
[7] Forshaw M, Nikolic K and Sadek A 2001 EC ANSWERS project (MELARI 28667) third year report, webpage http://ipga.phys.ucl.ac.uk/research/answers
[8] Xilinx Advanced Product Specification for Virtex-II FPGAs: April 2001 www.xilinx.com