INSTITUTE OF PHYSICS PUBLISHING NANOTECHNOLOGY
Nanotechnology **14** (2003) 224–230 PII: S0957-4484(03)53255-X

A defect- and fault-tolerant architecture for nanocomputers

Jie Han and Pieter Jonker

Pattern Recognition Group, Faculty of Applied Sciences, Delft University of Technology, The Netherlands

E-mail: jie@ph.tn.tudelft.nl and pieter@ph.tn.tudelft.nl

Received 6 September 2002, in final form 17 December 2002
Published 16 January 2003
Online at stacks.iop.org/Nano/14/224

Abstract
Both von Neumann's NAND multiplexing, based on a massive duplication of imperfect devices and randomized imperfect interconnects, and reconfigurable architectures have been investigated to come up with solutions for integrations of highly unreliable nanometre-scale devices. In this paper, we review these two techniques, and present a defect- and fault-tolerant architecture in which von Neumann's NAND multiplexing is combined with a massively reconfigurable architecture. The system performance of this architecture is evaluated by studying its reliability, i.e. the probability of system survival. Our evaluation shows that the suggested architecture can tolerate a device error rate of up to 10^{-2}, with multiple redundant components; the structure is efficiently robust against both permanent and transient faults for an ultra-large integration of highly unreliable nanometre-scale devices.

1. Introduction

Nanometre-scale electronics have been greatly developed in recent years. Besides the developments of various single nanoelectronic devices some research has advanced to the logic circuit level, such as single-electron tunnelling (SET) technology [1, 2], carbon nanotubes [3], semiconductor nanowires [4], chemically assembled electronic nanocomputers (CAENs) [5, 6] etc. The very small sizes of nanometre-scale devices make it possible to build a trillion (10^{12}) devices in a square centimetre [7]. However, for such a densely integrated circuit to perform a useful computation, it has to deal with the inaccuracies and instabilities introduced by fabrication processes and the tiny devices themselves. Permanent faults may emerge during the manufacturing process, while transient ones may spontaneously occur during computers' lifetimes. Future nanoelectronic architectures have to be able to tolerate an extremely large number of defects and faults. The design of fault-tolerant architectures for the ultra-large integration of highly unreliable nanometre devices is therefore inevitable.

In 1952, von Neumann initiated the study of using redundant components to obtain reliable synthesis from unreliable components, namely, the multiplexing technique [8]. It was then theoretically demonstrated that with an extremely high degree of redundancy, the integration of unreliable logic units could be made reliable. In his construction, von Neumann considered two sets of basic logic circuits, the majority voting and NAND logic, and assumed that they are not completely reliable, i.e., each of them fails with constant probability. By using a bundle of unreliable gates functioning as an ideally reliable one, von Neumann proved that if the failure probabilities of the gates are sufficiently small and the failures are statistically independent, computations may be done reliably with a high probability. However, the construction requires a large number of redundant components, which was seen as a major shortcoming of this method.

In 1965, the work by von Neumann and his contemporaries on fault tolerant logic was generalized by Pierce to a theory termed interwoven redundant logic [9]. In 1977 Dobrushin and Ortyukov theoretically improved von Neumann's result [10], showing that logarithmic redundancy is actually sufficient for any Boolean functions [10] and, at least for certain Boolean functions, necessary [11]. In the 1980s, it was proven by Pippenger that a variety of Boolean functions may be computed reliably by noisy networks requiring only constant multiplicative redundancy [12, 13]. Hence, it is theoretically demonstrated that the multiplexing technique may work effectively with a practically acceptable redundancy overhead. More recently, von Neumann's NAND multiplexing has been studied as an effective fault-tolerant technique for protection

against the increasing transient faults in nanoelectronic circuits [14, 15], while it is believed to be less efficient against manufacturing defects or permanent faults.

A reconfigurable architecture is a computer architecture which can be configured or programmed after fabrication to implement desired computations. Faulty components are detected during testing and excluded during reconfiguration. Reconfigurable architectures have been investigated as well for the solution of integration of highly unreliable nanometre-scale devices, in particular as defect-tolerant architectures against manufacturing errors. Teramac [16], built in HP laboratories, is such an extremely defect-tolerant reconfigurable machine. The basic components in Teramac are programmable switches (memory) and redundant interconnections. The high communication bandwidth is critical for both parallel computation and defect tolerance. With about 10% of logic cells and 3% of total resources defective, Teramac could still operate 100 times faster than a high-end single-processor workstation for some of its configurations. The embryonics architecture [17] is inspired by the biological growth and operation of all living beings. It is based on four hierarchical levels: a molecule (a multiplexer-based element of a programmable circuit), a cell (a small processor with an associated memory), an organism (an application-specific multiprocessor system) and the population of identical organisms. Each cell contains complete sets of instructions, the genomes, which make each cell universal and potentially apt for self-repair and self-replication. The objective of developing highly robust integrated circuits capable of self-repair and self-replication makes the embryonics architecture a potential paradigm for future nanometre-scale computation systems.

In this paper we seek fault-tolerant architectures for unreliable nanoelectronic devices, by extending the study of NAND multiplexing to a rather low degree of redundancy and implementing it into a massively reconfigurable architecture. The system performance of the architecture is evaluated by studying its reliability, which can be defined as the probability of system survival. Our evaluation shows that the suggested system is efficiently robust against both permanent and transient faults for an ultra-large integration of highly unreliable nanometre-scale devices.

The paper is organized as follows. In section 2 von Neumann's NAND multiplexing technique is briefly reviewed and developed. Section 3 gives the reliability analysis of reconfigurable architectures. In section 4 we present the implementation of a defect- and fault-tolerant architecture based on NAND multiplexing and reconfigurable architectures. Section 5 concludes the paper.

2. The NAND multiplexing technique

2.1. von Neumann's theory

Consider a NAND gate. Replace each input of the NAND gate as well as its output by a bundle of N lines, and duplicate the NAND N times, as shown in figure 1. The rectangle U is supposed to perform a 'random permutation' of the input signals in the sense that each signal from the first input bundle is randomly paired with a signal from the second input bundle to form the input pair of one of the duplicated NANDs.

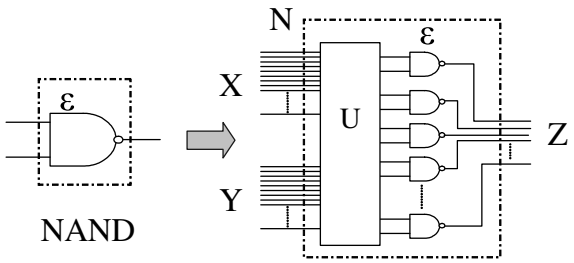

Figure 1. A NAND multiplexing unit.

Let X be the set of lines in the first input bundle being stimulated (a logic TRUE or '1'). Consequently, $(N - X)$ lines are not stimulated (they have the value FALSE or '0'). Let Y be the corresponding set for the second input bundle; and let Z be the corresponding set for the output bundle.

Assume that the failure probability of a NAND gate is a constant ε and assume that the type of fault the NAND makes is that it inverts its output; i.e. acts as an AND gate (a von Neumann fault). Let (X, Y, Z) have $(\bar{x}N, \bar{y}N, \bar{z}N)$ elements. Clearly $(\bar{x}, \bar{y}, \bar{z})$ are relative levels of excitation of the two input bundles and of the output bundle, respectively. The question is then: what is the distribution of the stochastic variable $\bar{z}$ in terms of the given $\bar{x}$ and $\bar{y}$? With a large N, von Neumann had concluded that $\bar{z}$ is a stochastic variable, approximately normally distributed [8].

2.2. Error distributions in a multiplexing unit

The NAND multiplexing unit was constructed as in figure 1. Von Neumann's theory applies when the bundle size N is large. If N is large, however, the theory is unrealistic in practice because of the huge amount of redundancy. In this section we study the error distributions in a multiplexing unit with a fairly low degree of redundancy.

Let us consider a single NAND gate in the multiplexing scheme. We still assume that there are $\bar{x}N$ and $\bar{y}N$ input lines stimulated. If the two inputs are independent, the probability of the output of the NAND gate that was found to be non-stimulated (by both stimulated inputs) is $\bar{r}' = \bar{x}\bar{y}$ (assuming that the NAND gate is fault free). If each NAND gate has a probability ε of making a von Neumann error, the probability of its output being non-stimulated is

$$\bar{r}_v = \bar{x}\bar{y} + \varepsilon(1 - 2\bar{x}\bar{y}); \tag{1}$$

for more common fault models stuck-at-0 and stuck-at-1, the probabilities become

$$\bar{r}_0 = \varepsilon + (1 - \varepsilon)\bar{x}\bar{y} \tag{2}$$

and

$$\bar{r}_1 = (1 - \varepsilon)\bar{x}\bar{y}. \tag{3}$$

For each single NAND gate, thus, the probability of the output being non-stimulated (event 0) is $\bar{r}$, $\bar{r} \in [\bar{r}_v, \bar{r}_0, \bar{r}_1]$, and the probability of being stimulated (event 1) is $1 - \bar{r}$. If the N NAND gates function independently, the probability of exactly k outputs being non-stimulated is given by the binomial distribution

$$R(k) = \binom{N}{k} \bar{r}^k (1 - \bar{r})^{N-k}. \tag{4}$$

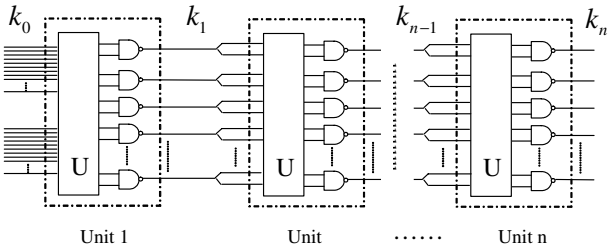

Figure 2. The multi-stage NAND multiplexing system.

If both inputs of the NAND gates are expected to be in stimulated states, the non-stimulated outputs are then considered as reliable ones. If the faulty devices in the multiplexing circuits are independent and uniformly distributed, the formula (4) could be easily used to calculate the output reliability. This may be reasonable when the dominant faults are transient ones. For manufacturing defects and permanent faults, however, the binomial distribution model is not sufficient to describe the actual manufacturing imperfections. The device components are not statistically independent but rather correlated since defects tend to cluster on a chip [18]. The formula (4) is therefore not appropriate for reliability calculation. (Although it is not yet clear what the future nanocomputers will be based on and how they will be built, it might be helpful to learn from present manufacturing processes.)

Variability of the manufacturing defects can be modelled with a continuous probability distribution function $f(r)$ of estimated component reliability r. Compounding the formula (4) with respect to this distribution function results in

$$R(k) = \int_0^1 \binom{N}{k} \bar{r}^k (1-\bar{r})^{N-k} f(r)\,\mathrm{d}r. \tag{5}$$

The success of the approach depends on finding appropriate parameters for the formula. Here we follow Stapper's beta distribution model [19], which gives

$$R(k) = \binom{N}{k} \bar{r}^k \left(\prod_{\mathrm{i}=0}^{k-1} \frac{\mu + \mathrm{i}}{\mu + \mathrm{i}\bar{r}} \right) (1-\bar{r})^{N-k} \times \left(\prod_{\mathrm{j}=0}^{N-k-1} \frac{\mu + \mathrm{j}\bar{r}/(1-\bar{r})}{\mu + k\bar{r} + \mathrm{j}\bar{r}} \right), \tag{6}$$

where μ is a variable parameter and $\bar{r}$ is the average or expected single output reliability. The formula calculates the probability that exactly k out of N identical NANDs give reliable outputs. The parameter μ is a measure of the amount of fault clustering. Small values of μ indicate high levels of clustering. As μ approaches infinity the formula becomes the case of independently distributed faults.

2.3. Error distributions in a multi-stage system

If the outputs of a NAND multiplexing unit are duplicated as the inputs of the succeeding one, a multi-stage system can be built as depicted in figure 2. In such a system the number of stimulated (or non-stimulated) outputs of each NAND multiplexing stage is actually a stochastic variable; it evolves as a Markov process (chain) because the outputs of one stage are totally determined by the inputs and device error distribution of the same stage. The characteristic of a Markov chain can be described by an initial probability distribution and transition probabilities.

If there are k_{l-1} of the N incoming lines stimulated for both inputs of the lth unit and each NAND gate has a fixed probability ε of making an error, according to formula (6), the probability of having k'_l non-stimulated outputs in the case of the corresponding k_{l-1} stimulated inputs is given by

$$R(k'_l|k_{l-1}) = \binom{N}{k'_l} \bar{r}^{k'_l}(k_{l-1}) \left(\prod_{i=0}^{k'_l-1} \frac{\mu + \mathrm{i}}{\mu + \mathrm{i}\bar{r}(k_{l-1})} \right) \times (1-\bar{r}(k_{l-1}))^{N-k'_l} \left(\prod_{j=0}^{N-k'_l-1} \frac{\mu + \mathrm{j}\bar{r}(k_{l-1})/(1-\bar{r}(k_{l-1}))}{\mu + k'_l\bar{r}(k_{l-1}) + \mathrm{j}\bar{r}(k_{l-1})} \right) \tag{7}$$

where $\bar{r}(k_{l-1})$ is a variation of equation (1), (2) or (3) with $\bar{x} = \bar{y} = \frac{k_{l-1}}{N}$.

If we are interested in the outputs which give faulty signals, then the probability of having k_l stimulated outputs, i.e., $k_l = N - k'_l$, is given by

$$P(k_l|k_{l-1}) = R((N-k_l)|k_{l-1}). \tag{8}$$

Noting the stochastic nature of k_{l-1}, the probability of k_l outputs being stimulated in all cases is obtained by

$$P(k_l) = \sum_{k_{l-1}=0}^{N} P(k_l|k_{l-1})P(k_{l-1}). \tag{9}$$

Formula (9) is inductive in the sense that, given an initial probability distribution and conditional probabilities, the output probability at any stage can be obtained. In the Markov chain an $(N+1) \times (N+1)$ transition probability matrix $\mathbf{\Psi}$, whose elements are $P(k_l|k_{l-1}), k_l, k_{l-1} \in [0, 1, 2, \ldots, N]$, can be made as (10), so that all conditional probabilities for any set of (k_l, k_{l-1}) are included.

$$\mathbf{\Psi} = \begin{bmatrix} P(0|0) & P(1|0) & P(2|0) & \cdots & P(N|0) \\ P(0|1) & P(1|1) & P(2|1) & \cdots & P(N|1) \\ P(0|2) & P(1|2) & P(2|2) & \cdots & P(N|2) \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ P(0|N) & P(1|N) & P(2|N) & \cdots & P(N|N) \end{bmatrix}. \tag{10}$$

Since the transition probability matrix $\mathbf{\Psi}$ for each stage is identical and irrelevant with regard to l, this is a homogeneous Markov chain. With the transition probability matrix and a fixed input distribution,

$$\boldsymbol{P}_0 = [p_0, p_1, p_2 \ldots p_N] \tag{11}$$

where p_i is the probability of i inputs being stimulated, the stimulated output distribution of a NAND multiplexing system with n stages is

$$\boldsymbol{P}_n = \boldsymbol{P}_0 \mathbf{\Psi}^n. \tag{12}$$

When n gets large, $\mathbf{\Psi}^n$ approaches a constant matrix $\boldsymbol{\pi}$, i.e.,

$$\lim_{n\to\infty} \mathbf{\Psi}^n = \boldsymbol{\pi}. \tag{13}$$

This indicates that, as n becomes extremely large, the system output distribution will become stable and independent of the number of multiplexing stages.

3. Reliability analysis of reconfigurable architecture

The key idea behind reconfigurable architectures is that the defects due to manufacture can be detected, located and then avoided. The reconfigurable computer concept is greatly assisted by the use of field programmable gate arrays (FPGAs) [20]. Fundamentally an FPGA contains a regular array of logic units, which are called configurable logic blocks (CLBs). Each CLB can communicate with its neighbours, and the CLBs are further grouped in blocks, then clusters of blocks. The CLBs can be individually reprogrammed so that a wide variety of logic or memory structures can be mapped onto the array of CLBs. When a part (or all) of a CLB is not working, the defective components are easy to locate and exclude from the working components. The Teramac machine [16], as a successful example of the reconfiguration concept, uses 864 identical FPGA chips, among which 75% (647) are partially defective. The first task of Teramac after it was built was to run self-diagnostic software, by which the defects were detected and located, and a defect database was generated. By reading the database, user applications are mapped onto good resources. Teramac 'has been successfully configured into a number of parallel architectures and used for extremely demanding computations'.

In processor arrays, the basic logic circuit blocks are referred to as processing elements (PEs), which are sometimes associated with local memories. In very large chips, the reliability can be improved by adding spare PEs to the design. Clearly, the more spares are added, the higher the resulting reliability will be. Instead of trying to achieve complete fault tolerance, defined as survival of a number of faults equal to the number of spares, most research aims at optimizing probability of survival, defined as the percentage of fault configurations that can be successfully overcome by the reconfiguration approach [21]. Reconfiguration approaches may be local or global. In local approaches, arrays are divided into subarrays. Spare elements are added to each individual subarray and reconfiguration is performed internally to each subarray. In global approaches, a set of spare elements is added to the whole array (usually as spare rows and columns along the edges of the array). Global approaches usually involve far more complex reconfiguration algorithms than local solutions [21].

For simplicity, we refer to logic blocks, clusters or PEs as modules and assume that all modules in the array are identical, so that any spare module can substitute any failed one, provided there exists a sufficient number of interconnection paths. If in an array there are n identical modules, out of which r are spares, then at least $n - r$ must be fault free for proper operation. We define R_{mn} as the probability of exactly m out of the n modules being fault free; then the reliability of the array is given by

$$R_n = \sum_{m=n-r}^{n} R_{mn}. \tag{14}$$

If each module has the same failure rate, or the same reliability R_0, and modules are statistically independent, we obtain the following binomial probability for the number of fault-free modules m:

$$R_{mn} = \binom{n}{m} R_0^m (1 - R_0)^{n-m}. \tag{15}$$

Once again the defective modules in an array are not uniformly distributed but rather correlated, therefore the binomial distribution formula (15) is not sufficient for reliability evaluation. Stapper's model can be used to improve the reliability calculation of correlated modules [19]:

$$\bar{R}_{mn} = \binom{n}{m} \bar{R}_0^n \left(\prod_{i=0}^{m-1} \frac{\mu + \mathrm{i}}{\mu + \mathrm{i}\bar{R}_0} \right) \times (1 - \bar{R}_0)^{n-m} \left(\prod_{j=0}^{n-m-1} \frac{\mu + \mathrm{j}\bar{R}_0/(1 - \bar{R}_0)}{\mu + m\bar{R}_0 + \mathrm{j}\bar{R}_0} \right), \tag{16}$$

where μ is a variable parameter indicating the amount of fault clustering and $\bar{R}_0$ is the average or expected single-module reliability.

The formula calculates the probability that exactly m out of n identical modules operate correctly. It can be applied to the reliability analysis of parallel processors with redundancy and fault-tolerant very large scale integration (VLSI) systems.

4. The defect- and fault-tolerant architecture

4.1. The basic circuits implemented with NAND multiplexing

Within a digital computer, the bulk of the logic gates is spent on memory and caches. The processor itself is made from a number of functional units, each of which can be separated into function blocks. Let us assume that the function block on the most refined level evaluates its inputs and produces a stable output within one clock cycle. Within this function block, many logic circuits may be cascaded; however, to avoid timing problems (hazard) usually the number of circuits cascaded and hence the possible paths from inputs to outputs through the various logic circuits is kept within bounds, and hence their path lengths are similar. Such function blocks are found everywhere in the processor and in memory. The function blocks or processors can be composed of arithmetic and logic units (ALUs), look-up tables (memories) or simply multiplexers. The fact that a NAND gate is a universal logic device makes it possible to use the NAND multiplexing technique on any logic operations, even though the NAND multiplexing can be easily extended to other specific logics. In this section we make an abstraction of such a function block and assume, to be able to make a statistical analysis, that it is made entirely out of stages of parallel NAND gates.

If the processors are implemented with the NAND multiplexing, then the obtained structure will be a NAND multiplexing system with a redundancy factor of N, as all the components are duplicated N times. The performance of a multiplexing system can be evaluated by investigating the probability that the number of faulty outputs is or is not beyond a threshold level. In other words those outputs with errors less than this threshold will be seen as reliable and their complementaries will be unreliable. The threshold level, together with N, may have an impact on the maximum tolerable value of the device failure rate ε. Since we are concerned with the minimum redundancy required to achieve fault tolerance, we take $N = 3$ and then the threshold $\frac{1}{3}$.

If the redundancy is as small as $N = 3$, the random interconnections of logic layers can be substituted in practical implementations by systematic ones which have specific

routes. Systematic interconnections are even likely to be superior in terms of error correction to random ones [9]. Since the NAND multiplexing uses redundant components to mask the effect of defective ones (no error detection is needed), the system is actually capable of tolerating both permanent and transient faults upon their occurrences.

If each processor has a logical depth of 11, which is sufficient for general computation tasks, then the reliability of the one-bit NAND multiplexing output after 11 stages can be studied with various device error rates ε, using the NAND multiplexing theory. With perfectly fault-free inputs, the probability distributions of output errors (unreliability distribution) are computed against the error rate of individual NANDs with $\mu = 1, 2, 5, 20$ and *infinity*, shown in tables 1, 2 and 3 for different fault models.

From the tables we can see that the von Neumann fault model brings the largest system performance degradation while the influence from μ, i.e., the amount of fault clustering, is insignificant. Since we are interested in the maximum device error rate that can be tolerated in general, we will take $\varepsilon = 10^{-2}$ and $\mu = 5$; then the reliability of the one-bit output can be obtained from table 1 as $R_0 = 0.9012$. In the following section we will take $R_0 = 0.9012$ as the average reliability of a one-bit multiplexing circuit.

4.2. The reconfigurable structures on bit, processor and cluster levels

We further assume that each processor has 32-bit processing capacity. For a 32-bit processor, if no redundant circuits are applied, it is only reliable when all of the output bits are reliable. Instead of exactly making a 32-bit circuit we build in the processor some redundant processing circuits, so that the spares can be configured to replace defective ones (figure 3(c)). If each one-bit circuit has a similar structure, the reliability of the 32-bit processor with redundant circuits can be evaluated against the number of spare bit circuits, using formulae (14) and (16), as plotted in figure 4. The left-most data in the figure indicate the reliability of a processor with no redundancy. The effect of the variability parameter μ here is rather significant. The improvement of reliability by using redundancy is explicit, in particular when μ is large. Assuming that errors are not strongly correlated in processor and upper levels, we take $\mu = 20$ for further evaluations. Thus a processor with 16 redundant bit circuits will have a reliability of 0.9927.

The development of nanotechnology makes it eventually possible to realize extremely large scales of integration, of the order of 10^{12} devices per chip. If on such a chip each processor has about 10^6 devices (logic, memory, communications etc), the number of processors on the chip will be about 10^6 ($2^{10} \times 2^{10}$). Instead of being connected globally, the processors can be assembled into 1024 (2^{10}) processing clusters, each containing 1024 (2^{10}) processors and calculating tasks independently. The clusters further compose the chip. Both clusters and processors can be connected in a two-dimensional (32×32) array, in which some columns are redundant, as in figures 3(a) and (b). The reconfigurable strategy therefore can be implemented on both cluster and processor levels.

Similarly, the performance of a cluster (chip) with redundant processors (clusters) can be evaluated using the formulae (14) and (16). The reliability of a cluster against the number of spare columns is plotted as figure 5, with $\mu = 20$. The left-most point indicates the reliability of a cluster with no spares. By using four columns of processors as redundant, the reliability of the cluster is elevated from 0.5589 to 0.9959, i.e., a cluster having 128 (4×32) redundant processors has a reliability of 0.9959. Further, the reliability of a chip with 1024 clusters is plotted against the number of spare columns of clusters with $\mu = 20$ as figure 6. It is shown that the reliability of a chip will be greatly improved by using redundant components. If 128 of the 1024 total clusters (four out of 32

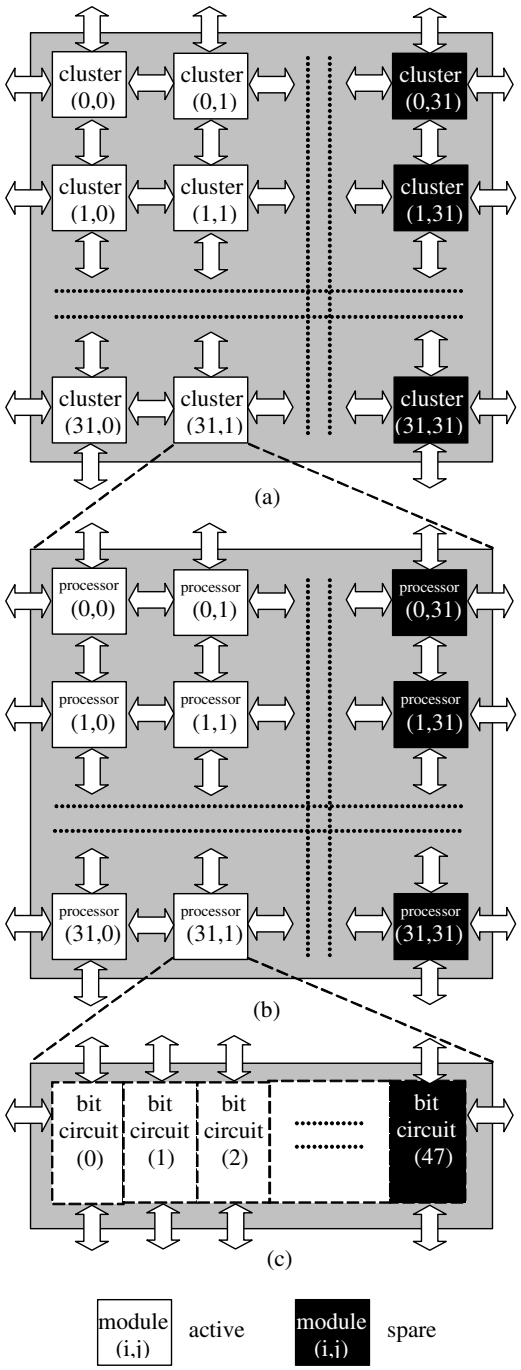

Figure 3. The architecture: (a) chip, (b) cluster, (c) processor, all with columns of spares.

Table 1. Output unreliabilities of a one-bit NAND multiplexing circuit (von Neumann fault).

ε	$\mu = 1$	$\mu = 2$	$\mu = 5$	$\mu = 20$	$\mu = \infty$
10^{-6}	1.207×10^{-5}	1.144×10^{-5}	1.070×10^{-5}	1.004×10^{-5}	9.735×10^{-6}
10^{-5}	1.207×10^{-4}	1.144×10^{-4}	1.070×10^{-4}	1.004×10^{-4}	9.734×10^{-5}
10^{-4}	1.206×10^{-3}	1.143×10^{-3}	1.069×10^{-3}	1.003×10^{-3}	9.728×10^{-4}
10^{-3}	1.196×10^{-2}	1.134×10^{-2}	1.061×10^{-2}	9.970×10^{-3}	9.669×10^{-3}
10^{-2}	0.1096	0.1047	0.09879	0.09346	0.09096
0.05	0.3797	0.3724	0.3616	0.3510	0.3459

Table 2. Output unreliabilities of a one-bit NAND multiplexing circuit (stuck-at-0 fault).

ε	$\mu = 1$	$\mu = 2$	$\mu = 5$	$\mu = 20$	$\mu = \infty$
10^{-6}	7.447×10^{-6}	7.374×10^{-6}	7.344×10^{-6}	7.341×10^{-6}	7.343×10^{-6}
10^{-5}	7.447×10^{-5}	7.374×10^{-5}	7.344×10^{-5}	7.341×10^{-5}	7.343×10^{-5}
10^{-4}	7.442×10^{-4}	7.369×10^{-4}	7.340×10^{-4}	7.337×10^{-4}	7.340×10^{-4}
10^{-3}	7.400×10^{-3}	7.329×10^{-3}	7.300×10^{-3}	7.298×10^{-3}	7.300×10^{-3}
10^{-2}	6.992×10^{-2}	6.938×10^{-2}	6.917×10^{-2}	6.918×10^{-2}	6.921×10^{-2}
0.05	0.2728	0.2730	0.2735	0.2741	0.2744

Table 3. Output unreliabilities of a one-bit NAND multiplexing circuit (stuck-at-1 fault).

ε	$\mu = 1$	$\mu = 2$	$\mu = 5$	$\mu = 20$	$\mu = \infty$
10^{-6}	4.627×10^{-6}	4.071×10^{-6}	3.352×10^{-6}	2.697×10^{-6}	2.391×10^{-6}
10^{-5}	4.628×10^{-5}	4.069×10^{-5}	3.355×10^{-5}	2.700×10^{-5}	2.391×10^{-5}
10^{-4}	4.627×10^{-4}	4.068×10^{-4}	3.355×10^{-4}	2.700×10^{-4}	2.391×10^{-4}
10^{-3}	4.613×10^{-3}	4.058×10^{-3}	3.350×10^{-3}	2.700×10^{-3}	2.393×10^{-3}
10^{-2}	4.472×10^{-2}	3.960×10^{-2}	3.302×10^{-2}	2.696×10^{-2}	2.409×10^{-2}
0.05	0.1956	0.1778	0.1544	0.1325	0.1221

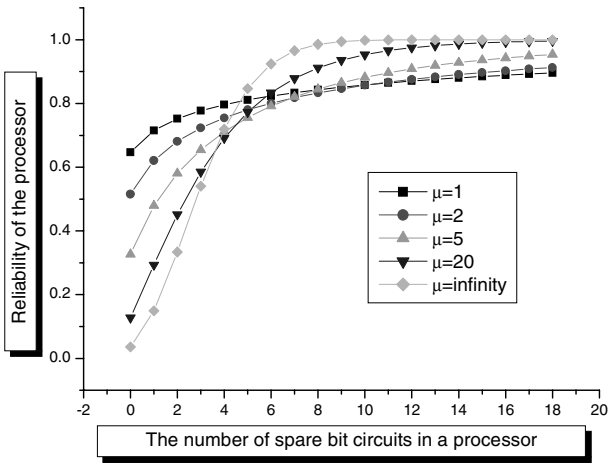

Figure 4. The reliability of a processor with spare bits.

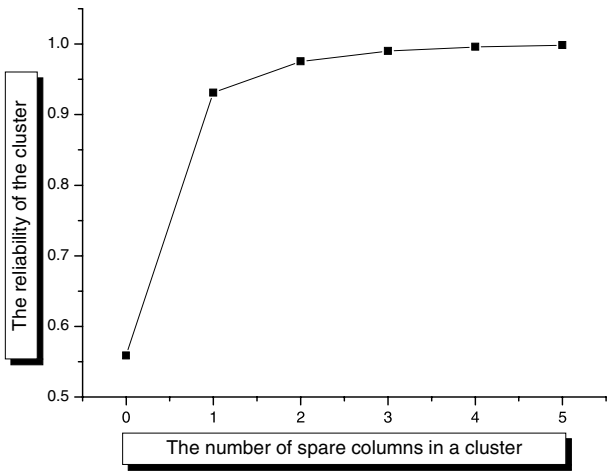

Figure 5. The reliability of a cluster with spare processors.

columns) are used as spare ones, then the reliability of the chip will be better than 99% (with a failure rate of 0.2%), provided that faulty components can be effectively substituted by spare ones.

In summary, we have discussed a set-up of a massively parallel fault-tolerant computer architecture. The NAND multiplexing technique is implemented in the fundamental circuits and reconfigurable structures are mapped to the bit, processor and cluster level. Containing up to 10^{12} devices, the conceived chip can have about 10^6 medium-sized processors and tolerate a device error rate up to 10^{-2}, which is generally unacceptable for any current VLSI systems. Redundant components are used at various levels and, as our evaluation shows, they are critical for the survival of the architecture. In contrast with [15], where plain NAND multiplexing was used to recover from transient errors, resulting in a massive redundancy, we now accept a higher error rate on the lowest level with considerably less redundancy, but compensate for this using a hierarchical reconfigurability. This leads to an acceptable failure rate for transient errors for the entire system (online error detection might be needed), and simultaneously forms a protection against permanent defects. The error detection problem remains open for further research. The system is expected to have a total redundancy factor

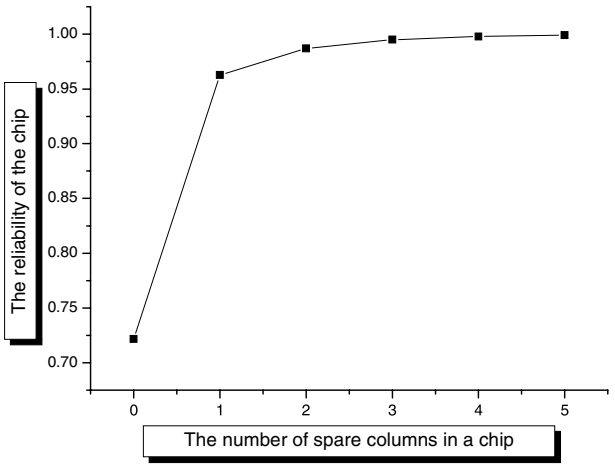

Figure 6. The reliability of a chip with spare clusters.

of $(3 \times \frac{3}{2} \times \frac{8}{7} \times \frac{8}{7} \times$ (*the fraction of other necessary spare components*)) <10. This indicates that future nanochips with 10^{12} devices might be working at an acceptable reliability level, virtually having about 10^{11} effective devices.

5. Conclusions

Due to the manufacturing process, the shrinking of electronic devices will inevitably introduce a growing number of defects and even make these devices more sensitive to external influences such as cosmic radiation, electromagnetic interference, thermal fluctuations etc. It is therefore likely that the emerging nanometre-scale devices will eventually suffer from more errors than classical silicon devices in large scale integrated circuits. In order to make future systems based on nanometre-scale devices reliable, the design of fault-tolerant architectures will be necessary. This paper can be seen as a part of the endeavour devoted to this work.

We have presented a defect- and fault-tolerant architecture, in which von Neumann's NAND multiplexing is implemented in basic circuits and reconfigurable architectures are mapped to the overall system. The system is expected to be working at an acceptable reliability level at the expense of having multiple redundant components. The architecture is potentially effective in protection against both permanent defects and transient faults for systems based on unreliable nanometre-scale devices.

Acknowledgments

We would like to thank Mike Forshaw of University College London, UK, for fruitful discussions. This work is supported by Delft University of Technology in its DIRC project 'Novel computation structures based on quantum devices'.

References

[1] Likharev K K 1999 Single-electron devices and their applications *Proc. IEEE* **87** 606–32
[2] Heij C P, Hadley P and Mooij J E 2001 Single-electron inverter *Appl. Phys. Lett.* **78** 1140–2
[3] Bachtold A, Hadley P, Nakanishi T and Dekker C 2001 Logic circuits with carbon nanotube transistors *Science* **294** 1317–20
[4] Huang Y, Duan X, Cui Y, Lauhon L J, Kim K and Lieber C M 2001 Logic gates and computation from assembled nanowire building blocks *Science* **294** 1313–17
[5] Collier C P, Wong E W, Belohradsky M, Raymo F M, Stoddart J F, Kuekes P J, Williams R S and Heath J R 1999 Electronically configurable molecular-based logic gates *Science* **285** 391–4
[6] Collier C P, Mattersteig G, Wong E W, Luo Y, Beverly K, Sampaio J, Raymo F M, Stoddart J F and Heath J R 2000 A [2]catenane-based solid state electronically reconfigurable switch *Science* **280** 1172–5
[7] Tseng G Y and Ellenbogen J C 2001 Toward nanocomputers *Science* **294** 1293–4
[8] von Neumann 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
[9] Pierce W H 1965 *Failure-Tolerant Computer Design* (New York: Academic)
[10] Dobrushin R L and Ortyukov S I 1977 Upper bound on the redundancy of self-correcting arrangements of unreliable functional elements *Prob. Inf. Trans.* **13** 203–18
[11] Dobrushin R L and Ortyukov S I 1977 Lower bound for the redundancy of self-correcting arrangements of unreliable functional elements *Prob. Inf. Trans.* **13** 59–65
[12] Pippenger N 1985 On networks of noisy gates *Proc. 26th Annu. Symp. on Foundations Comput. Sci.* (Los Alamitos, CA, USA: IEEE Computer Society Press) pp 30–8
[13] Pippenger N 1989 Invariance of complexity measures for networks with unreliable gates *J. ACM* **36** 531–9
[14] Nikolic K, Sadek A and Forshaw M 2002 Fault-tolerant techniques for nanocomputers *Nanotechnology* **13** 357–62
[15] Han J and Jonker P 2002 A system architecture solution for unreliable nanoelectronic devices *IEEE Trans. Nano.* **1** (4)
[16] 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
[17] Mange D, Sipper M, Stauffer A and Tempesti G 2000 Toward robust integrated circuits: the embryonics approach *Proc. IEEE* **88** 516–41
[18] Koren I and Koren Z 1998 Defect tolerance in VLSI circuits: techniques and yield analysis *Proc. IEEE* **86** 1819–36
[19] Stapper C H 1992 A new statistical approach for fault-tolerant VLSI systems *Proc. 22nd Int. Symp. on Fault-Tolerant Computing* (Los Alamitos, CA, USA: IEEE Computer Society Press) pp 356–65
[20] Lach J, Mangione-Smith W H and Potkonjak M 1998 Low overhead fault-tolerant FPGA systems *IEEE Trans. Very Large Scale Integr. Syst.* **6** 212–21
[21] Distante F, Sami M G and Stefanelli R 1995 Fault-tolerance and reconfigurability issues in massively parallel architectures *Proc. CAMP, Computer Architecture for Machine Perception (1995)* (Los Alamitos, CA, USA: IEEE Computer Society Press) pp 340–9