Fault Tolerance Issues in Nanoelectronics

S.M. Spagocci

A dissertation submitted in partial fulfilment
of the requirements for the degree of
Doctor of Philosophy
of the
University of London

Image Processing Group
Department of Physics and Astronomy
University College London

2008

Abstract

The astonishing success story of microelectronics cannot go on indefinitely. In fact, once devices reach the few-atom scale (nanoelectronics), transient quantum effects are expected to impair their behaviour. Fault tolerant techniques will then be required. The aim of this thesis is to investigate the problem of transient errors in nanoelectronic devices. Transient error rates for a selection of nanoelectronic gates, based upon quantum cellular automata and single electron devices, in which the electrostatic interaction between electrons is used to create Boolean circuits, are estimated. On the bases of such results, various fault tolerant solutions are proposed, for both logic and memory nanochips. As for logic chips, traditional techniques are found to be unsuitable. A new technique, in which the voting approach of triple modular redundancy (TMR) is extended by cascading TMR units composed of nanogate clusters, is proposed and generalised to other voting approaches. For memory chips, an error correcting code approach is found to be suitable. Various codes are considered and a lookup table approach is proposed for encoding and decoding. We are then able to give estimations for the redundancy level to be provided on nanochips, so as to make their mean time between failures acceptable. It is found that, for logic chips, space redundancies up to a few tens are required, if mean times between failures have to be of the order of a few years. Space redundancy can also be traded for time redundancy. As for memory chips, mean times between failures of the order of a few years are found to imply both space and time redundancies of the order of ten.

Acknowledgements

I would like to express my deepest thanks to Dr. Terry Fountain, my former supervisor who, after his retirement and my return to Milan, kept on giving me invaluable advice during the writing up process. My deepest thanks to Dr. Mike Forshaw, as well. After Terry's retirement, he also followed me during the writing up process, unofficially at the start and then as my new supervisor. Equally fundamental was the contribution of Prof. Peter Jonker and Prof. Andrew Fisher, my Viva examiners, who kindly assisted me during the writing up of version 2. My deepest thanks to them, as well. I will never forget the moral and, at times, financial help that I received from my parents during my stay in London. To them I dedicate this thesis. Last, but not at all least, my warmest gratitude goes to my Lombard friends here in London, and especially to Alex and Moira Severi and Claudio and Rita Rigolli. They gave me friendship and encouragement and offered me food and shelter during the many visits to London I had to pay while writing up. Finally, the official part. This thesis and associated research were partly funded in the framework of the DARPA ULTRA program, grant number N00014-96-1-0850.

Contents

Chapter 5 Nanodevice Error Rates

Chapter 6 Nanochip Error Rates

List of Figures and Tables

Chapter 1 Introduction

1.1 The Realm of Nanoelectronics

The astonishing success story of microelectronics since World War II cannot go on indefinitely. In fact, the minimum size of a device is fixed by atomic dimensions, though molecular dimensions are a more realistic limit [1]. As device dimensions shrink, the predominance of quantum effects will tend to impair the on-off behaviour of logic gates. This is likely to happen around 2015 [1,2].

When the length scale of a device is smaller than the De Broglie wavelength of electrons travelling across it, the wavelike nature of electrons dominates. In particular the limiting device length scale, l_d, is [2]:

$$l_d = \frac{h}{m_e v_e} \tag{1.1}$$

with $h = 6.6 \cdot 10^{-34}$ J·s = Planck's constant, $m_e = 9 \cdot 10^{-31}$ Kg = electron mass, v_e = electron velocity. By assuming $v_e = 10^5$ m/s [1], one gets $l_d = 7.3$ nm, of the same order of magnitude as the molecular length scale.

1.2 Nanoelectronics

Devices with a length scale of a few to a few tens of nanometres (nanoelectronic devices) have been proposed [1,2]. An overview of nanoelectronics is given in Chap. 2. Nanoelectronics seems to promise a big improvement in packing density. In fact, since 10 nm = 10^{-6} cm, 10^{12} devices cm^{-2} can be predicted. If we allow a factor of ~10 for interconnections, a packing density of ~10^{11} devices cm^{-2} can be anticipated [3,4], as compared to the 10^{6}-10^{7} devices cm^{-2} of present-day (2007) chips [2].

Nanoelectronic devices turn quantum mechanics to their advantage by exploiting unconventional approaches. In particular, quantum-dominated versions of the transistor (e.g. the resonant tunnelling transistor [5]) have been proposed. In other devices (based on the electron pump [6]), electrons act as the bits and their repulsion as a means of performing logic calculations. Other devices are cellular automata, where the interacting cells contain pairs of electrons or other entities (quantum cellular automata [7]). Finally, nanotubes or similar molecular devices can be used as logic gates [8].

1.3 Fault Tolerance

In the last four decades, a number of techniques for increasing system reliability have been proposed [9]. An overview of fault tolerance is given in Chap. 3. In order to make a technological system fault tolerant, a certain degree of redundancy has to be provided. Spare systems components have to be provided (space redundancy) or their operation has to be repeated for a number of times (time redundancy).

In space redundancy, voting is performed among replicas of the fault prone subsystems, so that errors can be masked. Alternatively, if faults are permanent, the system can be reconfigured so as to bypass the faulty subsystems. In time redundancy, the operation of each fault prone subsystem is repeated for a number of times and voting is performed. Another kind of fault tolerance is given by error correcting codes [10,11] that allow a corrupted message to be reconstructed (within limits), if suitable check bits are added to the information bits. This implies both time redundancy (the decoding process) and space redundancy (the check bits).

1.4 Fault Tolerant Architectures

With packing densities of 10^{11} or more devices cm^{-2}, nanochips are unlikely to work without fault tolerance [1,2]. To see this, let us assume that a processor is unreliable if, on average, it produces more than one error every few years, or $\sim 10^{-8}$ errors s^{-1}. This corresponds to less than 10^{-19} gate errors s^{-1}. Supposing, conservatively, that the processor is driven to 1 GHz, this gives 10^{-28} gate errors per clock cycle. A typical nanodevice error rate is 10^{-8} gate errors per clock cycle [12,13]. In order to make the chip reliable, then, one has to gain 20 orders of magnitude!

1.5 Achievements

Stimulated by the previous considerations, we decided to investigate transient errors for memory and logic nanochips composed of 10^{11} gates and operated at 1 GHz. We rejected configurations with space redundancy > 100 and/or time redundancy > 10. In this way, we got chips with linear dimensions of the order of a few cm, without loosing more than an order of magnitude in clock frequency. Manufacturing errors, and errors due to cosmic rays and radioactivity, do exist, of course. However, recent developments [3] imply that the redundancy level provided for transient errors may be enough to cope with manufacturing defects.

1.5.1 Nanodevice Transient Errors

To the best of our knowledge, the problem of transient error rates in nanodevices has not been investigated in great detail. There are qualitative considerations, and there are simulations or experimental measurements referring to certain devices of well-defined sizes, but no indications or data are given with respect to the problem of rescaling such simulations, or experimental data, to the nanometre size.

Starting from the literature data, and rescaling them, we estimated transient error rates for a number of nanodevices. In particular, we considered the electron pump and related devices [6], and QCAs and similar devices [7]. It turned out that, in devices based upon the electron pump, the main transient error source comes from quantum tunnelling, which make the electron go the wrong way. In QCA-based devices, transient errors mainly come from the fact that the device may absorb thermal energy from the environment and make a transition to a wrong state.

1.5.2 Fault Tolerant Techniques on Nanochips

In this research, we first considered logic chips. The main technique of choice, here, seems to be triple modular redundancy (TMR) [9]. In triple modular redundancy, potentially faulty units are tripled, so that errors are masked by majority voting. Voting can also be performed on any odd number of device replicas and/or in different ways. One then has n-modular redundancy (NMR) [9] and general modular redundancy (GMR) (the latter including TMR and NMR) [9].

The calculated nanodevice transient error rates were such that applying TMR, NMR or GMR to the single gates, in addition to being an impractical solution, in many cases would not suffice. A novel fault tolerant technique, cascaded triple modular redundancy (CTMR) and its generalization, cascaded general modular redundancy (CGMR), was therefore devised [12,13].

We then considered memory chips. We found that error correcting codes are the only feasible solution in this case. The problem was then to choose suitable codes, so as to achieve a fair trade-off between error correcting capability and redundancy. Good error correcting capability, in fact, tends to imply more redundancy, since more check bits are required. We considered two codes with opposite properties. Hamming codes [11] have a low error correcting capability and a low redundancy. Reed-Muller codes [10] have a high error correcting capability and a high redundancy.

1.5.3 Nanochip Transient Error Rates

It was then possible to calculate the space and time redundancies needed for making nanochips work in an acceptable way (in the sense defined above). It turned out that logic chips require either space or time redundancies of a few tens at worst, and space can be traded for time. Memory chips, on the other hand, can be made to work if both time and space redundancies of the order of ten at worst are provided.

1.5.4 Publications

The work described in this thesis has been publicly disseminated through two publications [12,13]. These are included as Appendix 2 and 3.

1.6 Structure of the Thesis

- In **Chapter 2: Nanoelectronics**, we give a review of the field of nanoelectronics, presenting some important devices and circuits based on them.

- In **Chapter 3: Fault Tolerance**, we review the field of fault tolerance, giving an overview of the available techniques.

- In **Chapter 4: A Model for Chip Error Rates**, we present a model of chip error rates and set up the terminology used in the thesis.

- In **Chapter 5: Nanodevice Transient Error Rates,** we examine error mechanisms leading to transient errors in some nanoelectronic devices. We concentrate on devices based upon the electron pump and quantum cellular automata.

- **In Chapter 6: Nanochip Transient Error Rates**, we first present the theory of cascaded general modular redundancy. We are then able to give redundancy estimations for logic nanochips. We then propose error correcting codes for taking care of transient errors in memory nanochips and estimate the resulting redundancy.

- In **Chapter 7: Conclusions,** we summarise the results obtained, place them in the context of nanoelectronics research and give suggestions for future work.

- In **Appendix 1 and Appendix 2** we present the two papers that arose from the present thesis.

- **In Appendix 3: Parallel Computing**, we give an overview of parallel computing, in view of the fact that nanoscale integration is particularly interesting for single-chip implementation of massively parallel systems [2].

1.7 Final Remarks

Since this is a highly interdisciplinary thesis, we had to concentrate on those aspects of each involved field which are relevant to the overall study. Given the present state of nanoelectronics, where only a few devices have been built at the nanometre scale [1,2], we were not able to give detailed error rate predictions. Order-of-magnitude predictions, however, turned out to be feasible.

The bulk of this thesis was completed early in 2001, but personal reasons prevented its full completion at that time. Some additional references to post-2000 have been included to indicate where significant developments have occurred. When the research described here was embarked upon, in mid 1997, few nanodevices had been described and data on the effects of error rates were almost non-existent. It has now become clear that manufacturing faults and transient errors might be the biggest stumbling block to the implementation of nanodevices [3]. The author believes that the work described here was (and is) among the first to tackle this serious problem.

Finally, we point out that there are other potential approaches to obtaining fault tolerant computation at a nanometre scale. We are referring, in particular, to such fields as DNA computing [14] and quantum computing [15]. Here, we will not consider these issues in any more detail.

Chapter 2 Nanoelectronics

2.1 Introduction

The limits of downscaling microelectronic devices are set by the fact that, at a length scale of ~0.1 μm, quantum effects start to dominate. Nanoelectronics is aimed at exploring the possibility of building quantum devices, particularly attractive because of their small dimensions. As shown in Chap. 1, for a typical device dimension of ~ 1 nm, a density of ~10^{11} devices/cm^2 can be expected.

Our treatment will mainly draw on a recent review [2]. Our analysis focuses on resonant tunnelling devices (RTDs), single electron devices (SEDs), quantum cellular automata (QCAs) and molecular nanoelectronic devices. The first ones are continuous devices, in which bits are defined by voltages. The second and third ones are granular devices, in the sense that they use single electrons to represent bits. The last ones are probably the most basic nanoelectronic devices. We think that these devices are representative of the whole spectrum of nanoelectronics.

2.2 Nanodevice Types

Here is a list of the nanoelectronic devices proposed up to now [2]:

- Devices that control the flow of a large number of electrons, using the voltage applied to a control gate, such as organic transistors [16] and nanotriodes [17]. These devices act in the same way as conventional microelectronic devices, though at a smaller length-scale.

- Devices that control the flow of a large number of electrons and exploit the tunnelling effect. Examples are the resonant tunnelling diode [18,19] and resonant tunnelling transistor [20].

- Devices that control the flow of one or a small number of electrons through a gate voltage. An example is the single electron transistor [21,22].

- Devices where a magnetic field is used to control the current flowing through a superconducting circuit, toggling between a superconducting and an ordinary state, i.e. between the on and off states [23,24].

- Molecular nanoelectronic devices such as nanowires, nanotubes or molecules [25], in which a gate voltage is used for electron flow control.

- Devices that control the electron flow by changing the properties of a material with an electric field, so that it turns from conducting to insulating [26].

- Memory devices that store information by e.g. trapping single electrons in an energy well [27,28] or changing electron spin [29-31].

- Devices that use quantum interference to control currents [32,33].

- Nanoscale cellular automata, in which various physical objects interact with each other without moving in space [34]. Examples are quantum cellular automata [35,36]. Other examples include devices where the interacting cells are magnetic domains (magnetic QCAs [37,38]) or superconducting loops [39-42].

Most of these devices are only at the state of single or few-device demonstration, often not even at the nanoelectronic scale, but rather at the 10-100 nm scale [1,2].

2.3 Tunnelling Devices

In the tunnelling effect, a particle of energy E is able to go through an energy barrier higher than E. This would be classically forbidden. Due to its wave-like nature, the particle is instead transmitted with probability [2,43]:

$$P(E) = \exp[-\frac{2d}{\hbar}\int_0^d dx\sqrt{2m(W(x)-E)}] \tag{2.1}$$

where d is the barrier thickness, $\hbar$ is Planck's constant, m is the particle mass and W(x) is the energy profile of the barrier.

2.3.1 Resonant Tunnelling Diodes

The above-mentioned effect can be exploited in a number of devices. Resonant tunnelling diodes (RTDs) are the most elementary examples. A resonant tunnelling diode [18,19] consists of a heterostructure (e.g. AlInAs/GaInAs [18] or Si/SiGe [19]) arranged in such a way as to create a double quantum barrier (3-10 nm thick), containing a quantum well between the emitter and the collector, see Fig. (2.1).

The quantum well has a discrete set of energy levels. The energy level of the electrons in the emitter and collector regions are lower than the first allowable well energy that, in turn, is much larger than kT, so that under normal conditions there is only a small tunnelling probability through the double barrier.

When a suitable voltage is applied between the emitter and collector regions, the energy band of the electrons in the emitter region may be brought to be aligned with that of the well. The tunnelling probability is than greatly enhanced and current flows through the double junction, see Fig. (2.1). Fig. (2.2) explains the resulting I-V curve.

Advantages of RTDs are the very short transit times (1.5 ps [44]) and switching speeds, due to the barrier's thickness and the possibility to operate at low voltages (0.5-1V). The main advantage of RTDs would be the possibility to scale them down to ~1 nm. However, the devices proposed up to now make a hybrid use of RTDs and conventional transistors, so that the advantage of down-scalability is lost [2].

It has to be mentioned that the older versions of the RTD, the so-called tunnelling diodes, or Esaki diodes, only had one energy barrier. These diodes, invented in 1968 [2], never gained any importance in microelectronics. RTDs, due to the double quantum barrier, and the consequent larger number of degrees of freedom, are more flexible than Esaki diodes.

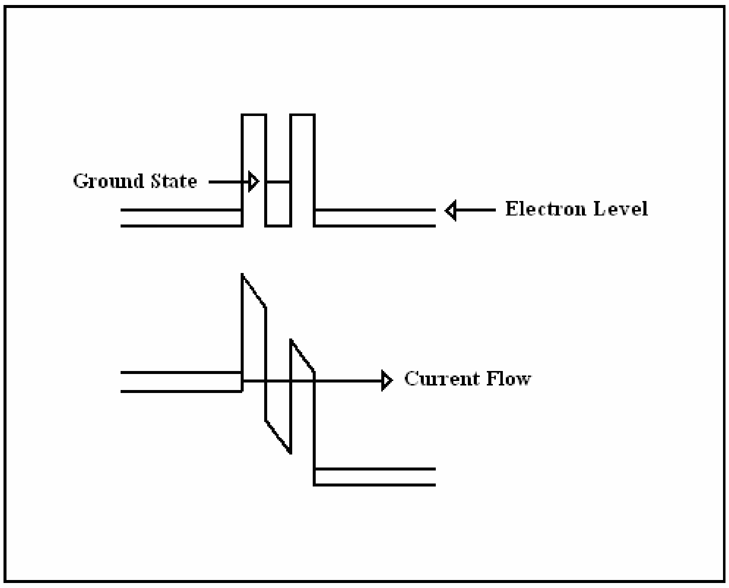

Fig. 2.1 Conduction band structure of an RTD. In the upper diagram, we see that no emitter/collector bias is applied. The electron band is lying below the well's ground state and no current flows, due to the very low tunnelling probability. In the lower diagram, an emitter/collector bias is applied. The well's ground state becomes aligned with the electron band and the current can flow by resonant tunnelling.

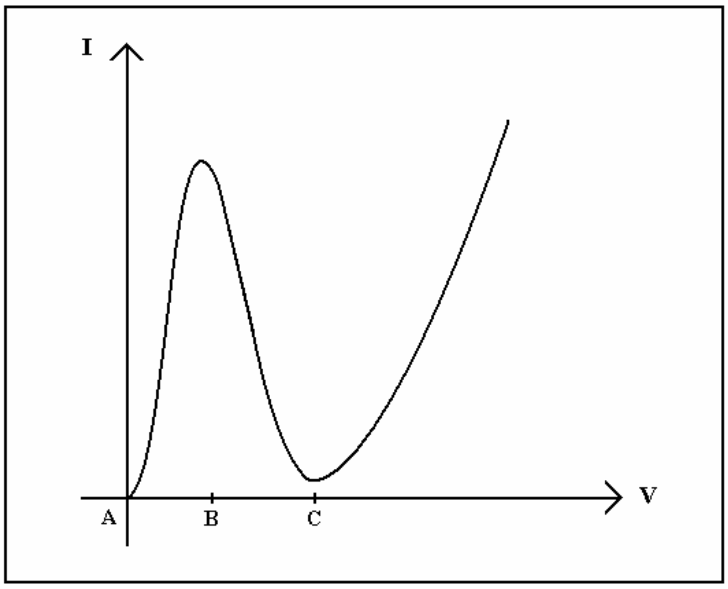

Fig. 2.2 I-V curve of an RTD. Between points A and B, the well's energy level becomes progressively nearer to the electron band, so that the current increases, since the electron band is more populated towards its bottom. In point B, the peak current is reached. The well's energy level is in fact aligned to the electron band's bottom line. Between points B and C, one has negative differential resistance, i.e. the current is decreasing with the increasing bias voltage. In point C, the current reaches its minimum and the wall's energy level is below the electron band's bottom line. Beyond point C, the current starts to rise again, because the electrons are no longer constrained by the quantum wells. Redrawn from [2]

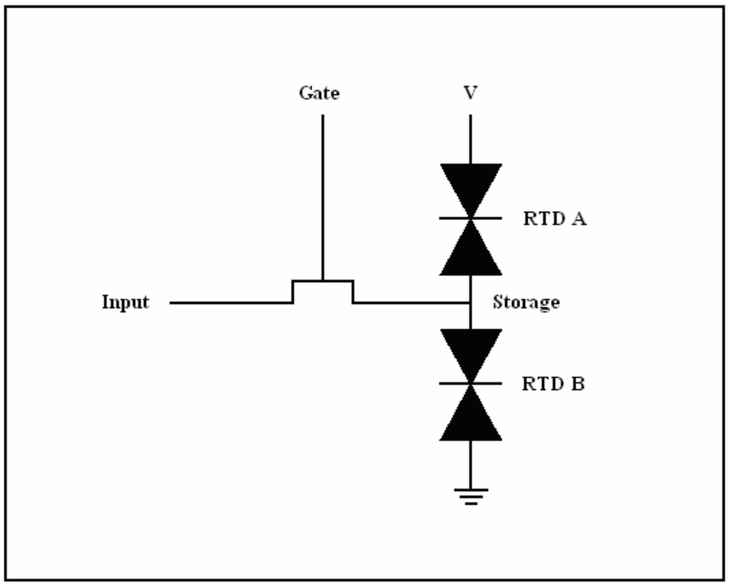

Fig. 2.3 A TSRAM cell. Two RTDs (A and B) are put in a series. As explained in Fig. (2.4), the system has two stable points, representing the 0 and 1 states. Which minimum the electron chooses is determined by the transistor that controls the voltage level at the storage node and allows, or denies, access to the memory cell through its gate. Redrawn from [2].

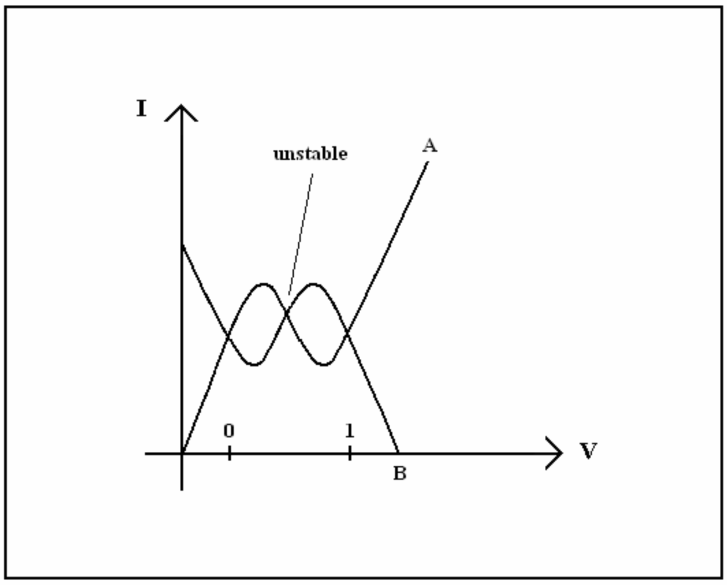

Fig. 2.4 A TSRAM cell, working principle. By applying Ohm's law to RTDs A and B, see Fig (2.3), the I-V curves in the storage point are A and B, respectively The storage point current and potential will then be defined by the intersection of curves A and B. The intersection point in the middle is unstable and there are two stable intersection points, the 0 and 1 logic states, respectively. Redrawn from [2].

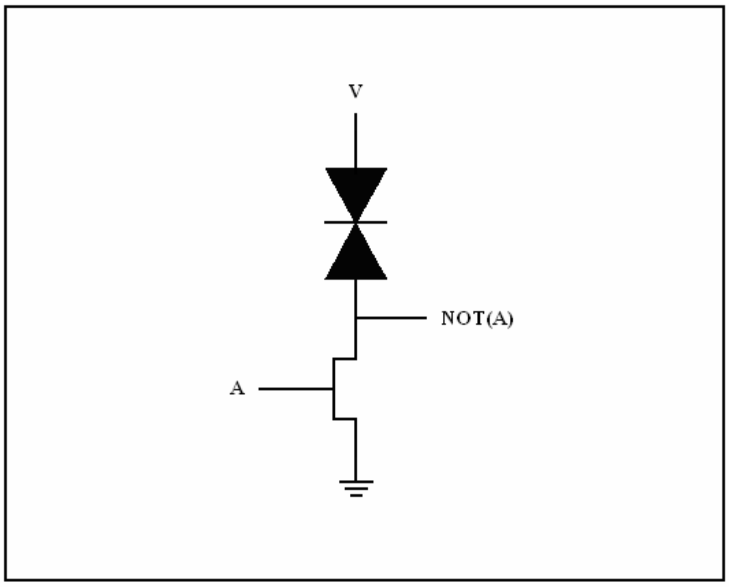

Fig. 2.5 A RTD-based NOT gate. When the input is low, the transistor is switched off. The output is then at voltage V (high). When the input is high, the transistor is switched on and the output is at the same potential as ground (i.e. low). The RTD, here, plays the same role played by a resistor in conventional NOT gate. Redrawn from [2].

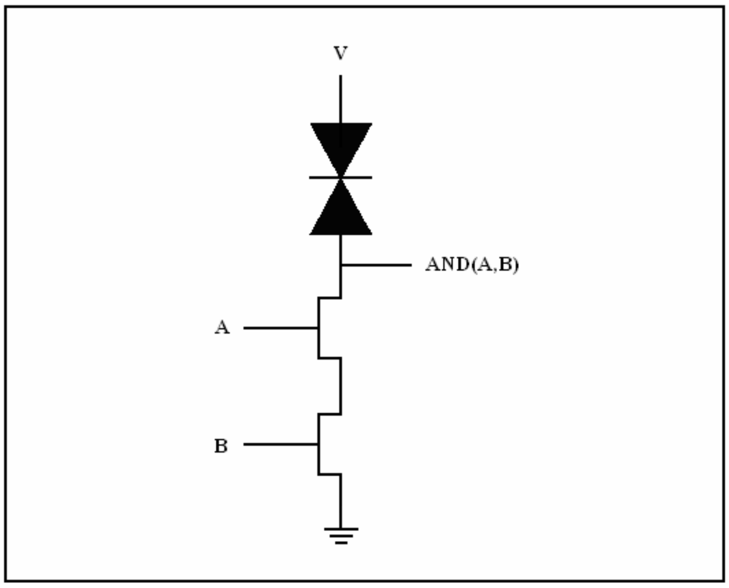

Fig. 2.6 A RTD-based NAND gate. When A and/or B are low, at least one of the transistors is switched off and there is no current. The output is then at voltage V (high). When both A and B are high, the transistors are switched on, and the output is at the same potential as ground (i.e. low). The RTD, here, plays the same role played by a resistor in conventional NAND gates. Redrawn from [2].

2.3.2 Resonant Tunnelling Transistors

The chip design principles prescribe that any device must have its input and output electrically separated, i.e it must be a three-terminal device [2]. The RTD is, instead, a two-terminal device and is then not suitable for chip integration.

The resonant tunnelling transistor (RTT) [20] consist in an RTD inserted into the emitter of a conventional transistor. According to the transistor type, one has RTD-FETs or RTBTs (Resonant Tunnelling Bipolar Transistor) [2]. These are three-terminal devices, suitable for chip integration. RTTs at present suffer from surface noise effects, due to the presence of the gate [2].

2.3.3 RTD Memory Devices

RTD memory devices are potentially interesting for their low power consumption, high speed and dense packing. This interest would be greater if they could be scaled down to the nanometre level. Fig. (2.3) shows the structure of a tunnelling-based static RAM (TSRAM), composed of two RTDs and a FET transistor [28,45]. The two RTDs in series create two stable operating points, defined as 0 and 1. See Fig. (2.4) for details. Small arrays of such memories (4x4 bits) have been demonstrated [28].

2.3.4 RTD Logic Devices

Boolean logic-based RTD logic devices have been proposed, in which the conventional resistors are replaced by RTDs [2], see Figs. (2.5) and (2.6). Threshold logic gates have also been proposed [46] (e.g. the MOBILE device [47]). Parallel adder units, each one containing 20 RTD/FET transistor devices, have been demonstrated [48]. Technological difficulties were found when trying to build a 1-bit full adder [49].

2.4 Single Electron Devices

In single electron devices (SEDs), single electrons are used as bits. The electrons are controlled through the so-called Coulomb blockade [2]. When an electron tunnels through a junction, the final result is charging the capacitor formed by the junction's faces by one charge unit. For the charge transfer to take place, the Coulomb energy of the charged capacitor must be smaller than that of the neutral one. In particular, the energy of a capacitor, storing n elemental charges, is:

$$E = \frac{e^2}{2C} \cdot n^2 \qquad (2.2)$$

where e is the electron charge and C is the junction capacitance. If the junction is kept at potential V, the condition for tunnelling is then:

$$e \cdot V + \frac{e^2}{2C} \cdot n^2 \leq \frac{e^2}{2C} \cdot (n+1)^2 \qquad (2.3)$$

$$V \leq \frac{e}{C} \cdot (n + \frac{1}{2}) \qquad (2.4)$$

Eq. (2.4) shows the condition to be satisfied for tunnelling to take place. If the junction is initially uncharged, so that n=0, we can simplify this condition.

In particular, one has:

$$V \leq \frac{e}{2C} \qquad (2.5)$$

Tunnelling, then, can only happen if the junction capacitor is at a voltage at least corresponding to a charge bias e/2. This condition can be exploited in a number of devices.

2.4.1 Single Electron Transistors

An important SED is the single electron transistor (SET) [21,22]. A SET is composed of two junctions and a gate electrode in between, see Fig. (2.7). An electron is driven through the two junctions. This is only allowed if the gate voltage has the right value, according to Eq. (2.5).

Coulomb blockade, and a SED, can only work if the thermal energy is considerably lower than the electrostatic energy E_t [2]. This gives:

$$E_t = \frac{e^2}{2C} >> k \cdot T \tag{2.6}$$

If the capacitance can be scaled down to ~10^{-18} F, room temperature operation will become feasible. This capacitance is envisaged for nanometre-scale SETs.

It is interesting to derive the minimum resistance of a tunnelling junction in a SED [2] (this result will also be used in Chap. 5). First of all, we notice that the uncertainty principle imposes that:

$$E_t \cdot \tau_t \geq h \tag{2.7}$$

where h is Planck's constant and τ_t is the tunnelling time constant:

$$\tau_t = R_t \cdot C \tag{2.8}$$

By inserting E_t, defined by Eq. (2.6), and Eq. (2.8), into Eq. (2.6), one has:

$$R_t \geq 2R_k \tag{2.9}$$

where:

$$R_k = \frac{h}{e^2} = 25.8\,k\Omega \tag{2.10}$$

is the so-called Von Klitzing resistance (the resistance quantum).

At a first glance, it would seem difficult to fabricate SETs consisting of two tunnelling elements and a small island. However, three approaches have been devised up to now. The challenge is now to find the appropriate way to bring one of these concepts into mass production [2].

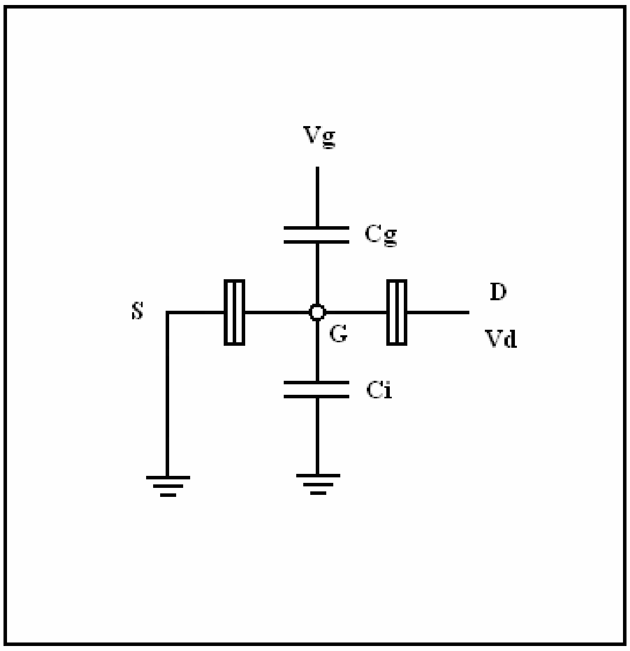

Fig. 2.7 A single electron transistor. The gate voltage V_g controls the tunnelling of an electron from the source S. Depending on the value of the gate voltage, the same value of the drain voltage V_d may or may not cause the electron to tunnel to the island G and reach the drain D. The device, then, acts as a switch. Redrawn from [2].

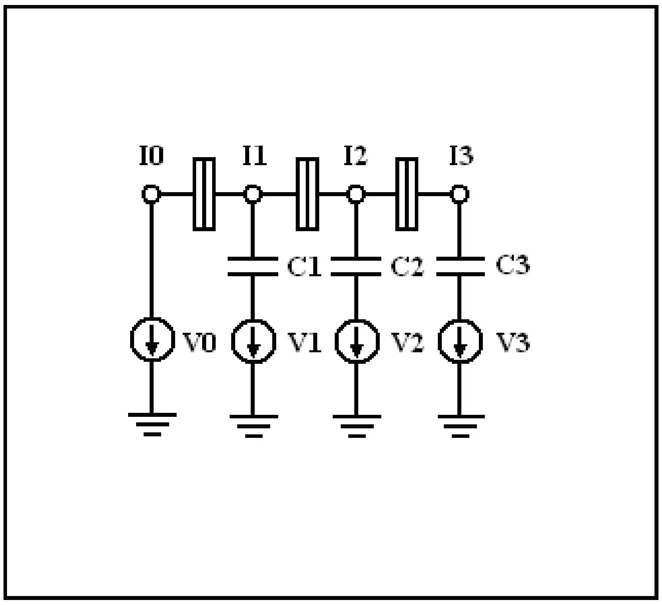

Fig. 2.8 A 3-junction single electron pump. When a suitable pulsing scheme is applied to the islands I, an electron can sequentially tunnel from I_0 to I_3. A fixed number of electrons can then be driven through the structure, which can e.g. be used as a metrological standard. Redrawn from [2].

Here is a list of the three approaches [2]. The first approach uses a batch process to coat small metal balls with an insulator. Such a ball is placed on top of an interrupted conductor. In the second approach, the structure is realized with molecular beam epitaxy, using different deposition angles to create overlapping junctions. In the third approach, the junction plus island structure is realized in Si on a SiO_2 insulating layer. The gate electrode is given by the Si substrate.

It has also to be noticed that SETs can be used as ultra-sensitive (a fraction of e) electrometers [50]. SET-based memory devices are presented in Sect.2.4.3. SET-based logic gates are presented in Sect. 2.4.4.

2.4.2 Electron Pumps

The electron pump [51], see Fig. (2.8), is an array of metallic islands, separated by nanometre-scale junctions, through which an electron is made to tunnel sequentially. It is then possible to control single electrons, so as to obtain a metrological current standard with a precision of $\sim 10^{-8}$ [50,52].

Typically, a triangular pulsing scheme [50] is applied to the islands. This charge bias insures that before tunnelling starts, and after tunnelling has happened, the junction's faces have no charge while, in the middle of the pulsing cycle, the junction is given the right *e*/2 bias for tunnelling, see Eq. (2.5).

A single-electron switch [6], based upon the electron pump, is shown and explained in Fig. (2.9). Once a switch has been built, it is possible to conceive of logic gates based upon it, see Sect. 2.4.4. Other SEDs include the electron turnstile [53], which is basically an electron pump with only one gate electrode at the center of the array.

2.4.3 SED Memory Devices

Different kinds of SED memory devices have been proposed, based on the fact that, by Coulomb blockade, an electron can be taken in and out of an island between two tunnelling junctions [2].

Hitachi built a 128 Mbit SED memory chip [54], in which conventional CMOS circuits were also present. The memory cell size was 0.15 μm^2 and the chip could be operated at room temperature. The chip had reliability problems: only about half of the chip's cells turned out to be operational. A group from Cambridge proposed and demonstrated a 3x3 bit memory array, including conventional CMOS circuitry [55].

2.4.4 SED Logic Devices

Single electron gates, based on the electron switch, have been proposed [6]. Figs. (2.10) and (2.11) show and explain an AND/OR and a NOT gate. Logic gates whose action is similar to the conventional CMOS gates have also been proposed [2].

Single-electron majority gates [56] and NAND/NOR gates [57] have been proposed and simulated. Two SET inverters have also been built. One of these, based upon a GaAs SET (with the junction built out of a two-dimensional electron gas kept confined by a metal gate), had a working temperature of 1.9 K [58]. Another SED inverter [59], based upon $Al/Al_2O_3/Al$ tunnel junctions, could only work below 0.14 K. A NAND logic gate has also been built [60], but it showed severe problems, due to the difficulty in controlling the effect of background charges in the substrate.

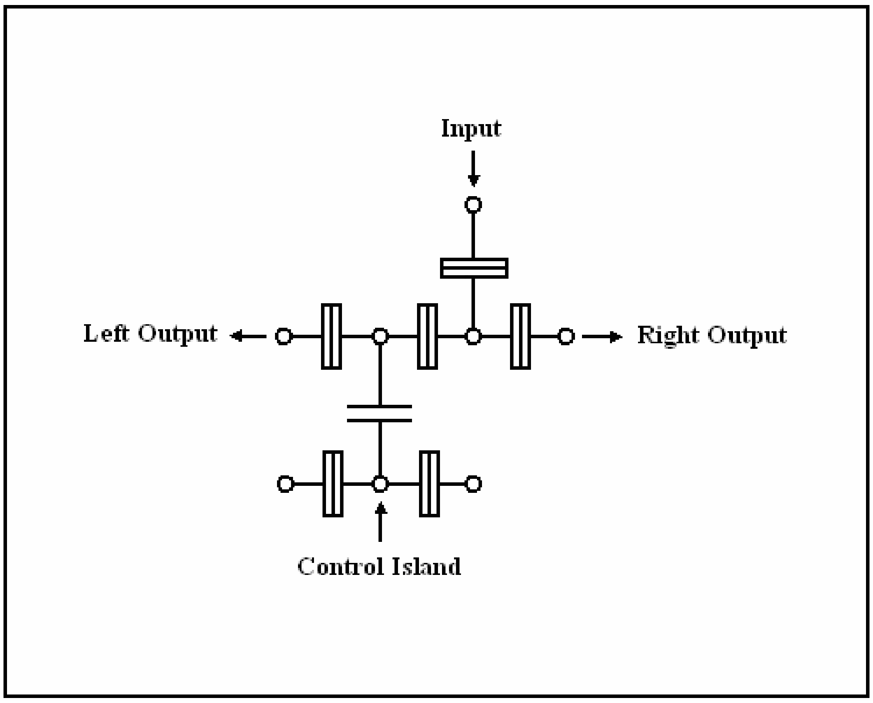

Fig. 2.9 A single electron switch, based upon the electron pump principle. The input electron is made to sequentially tunnel through the electron pump, formed by the junctions joining the metallic islands. When the control island is free, the input electron turns left. When the control island is occupied by an electron, the input electron turns right, since it is repelled by the control electron. Redrawn from [6].

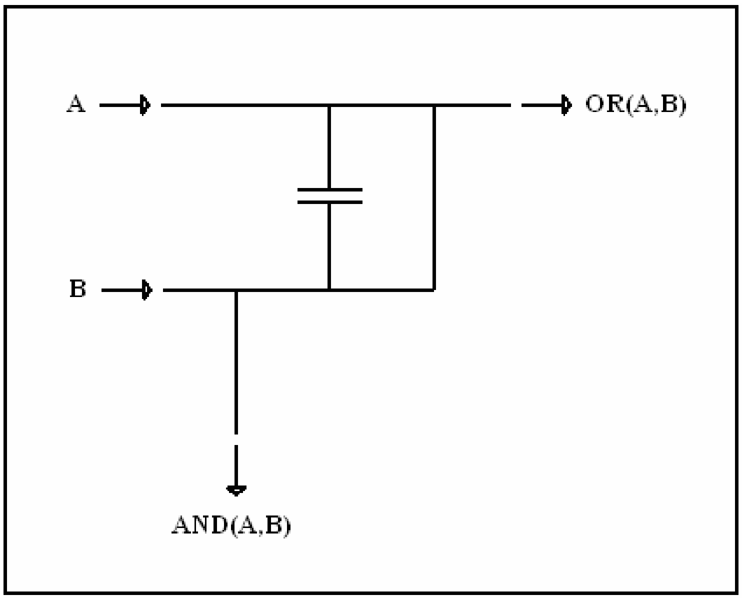

Fig. 2.10 An AND/OR gate, based on a single-electron switch. Any electron appearing at input A is driven to the OR (A,B) output. Any electron appearing at input B is either switched to the AND (A,B) or the OR (A,B) outputs, depending on the presence or absence of the electron from input A. Driving happens through the principle of electron pumps. Electron pumps are represented as solid lines.

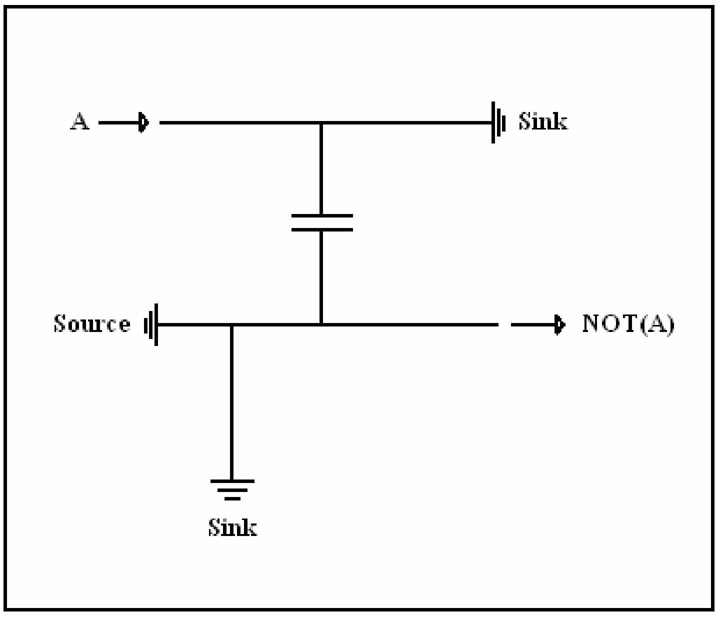

Fig. 2.11 A NOT gate, based on single electron switching. Any electron appearing at input A is driven to a sink. One electron per clock cycle is taken from a source and either switched to a sink or to the NOT (A) output, depending on the presence or absence of the electron coming from input A. Electrons are driven through the principle of electron pumps, which we schematize as solid lines.

2.5 Quantum Cellular Automata

Quantum cellular automata (QCA) systems consist of arrays of interacting cells. Each cell affects its neighbours and it is possible to propagate information and compute logic functions without physical signal propagation. The best-known examples are electronic quantum cellular automata [35,36]. The cells may also consist of magnetic domains (magnetic quantum cellular automata [37,38]) or superconducting loops [39-42].

Here we concentrate on electronic QCAs. For the sake of simplicity, these will be referred to as QCAs. A QCA cell is a square array of four quantum dots, occupied by two electrons, see Fig. (2.13) and Fig. (2.14). The cell has two stable states, corresponding to electrons sitting at the ends of each diagonal. Due to electrostatic repulsion, each cell is influenced by its neighbours. Logic and memory gates based on this principle can then be built [7], see Sect. 2.5.1 and 2.5.2.

Such a system can be described by the Hamiltonian [61]:

$$H = \sum_{i,\sigma} E_0 n_{i,\sigma} + \sum_{i \neq j} t \cdot (a_{i,\sigma}{}^{+} a_{j,\sigma}) + \sum_{i} E_q n_{i,\uparrow} n_{i,\downarrow} + \sum_{i>j,\sigma,\sigma'} V_q \frac{n_{i,\sigma} n_{j,\sigma'}}{|R_i - R_j|} \qquad (2.11)$$

In Eq. (2.11), E_0, t and E_q are energy terms, estimated with an experimentally reasonable one-dimensional model, $V_q = (4\pi\varepsilon)^{-1}$, where ε is the dielectric constant (a relative dielectric constant of 10 is assumed), $n_{i,\sigma}$ is the number operator describing an electron at site i (i = 0,1,2,3), with spin σ (σ = ↑,↓) , $a_{i,\sigma}{}^{+}$ is the creation operator that creates an electron at site i with spin σ, $a_{i,\sigma}$ is the corresponding annihilation operator, $|R_i-R_j|$ is the distance between the centers of dots i and j.

The various terms of Eq. (2.11) have the following meaning:

- The first term describes the ground states of the dots holding 0 or 1 electrons each one. The ground state of a 10 nm quantum dot diametre, with an electron of effective mass 0.067 m_e (m_e is the electron mass), is assumed.

- The second term describes the tunnelling process between all the couples of dots. t = 0.3 meV is assumed, for a distance between dot centers of 20 nm.

- The third term describes the fact that, due to the Pauli principle, two electrons can only stay in the same dot if they have opposite spins. Due to the Coulomb repulsion, it is assumed that $E_q = V_q/(D/3)$ (D is the dot diameter).

- The fourth term describes the Coulomb repulsion of couples of electrons in different quantum dots.

The above-mentioned Hamiltonian allows the (numerical) calculation of the energy level and polarization of a system of cells.

The simulations have shown that nearly full polarization can be obtained, provided a QCA chain is driven by adiabatic switching [2]. In adiabatic switching, the height of the potential barrier of the dots is controlled by an external electric field. An external electric field also causes a cell to switch its polarization. Before changing the polarization state of a cell, the barriers are lowered and the electrons reach a delocalized state. The input signal is then changed and the barriers raised again. In adiabatic switching, the clocking time must be significantly greater than the characteristic transition time of the system.

In order to give directionality to QCA computations, QCA-based devices have to be clocked [62-65]. This involves currents. Clocked single-electron switching in QCAs has been demonstrated [64,65]. QCA signals have to be sensed, e.g. using SETs as probes [66]. This, too, involves currents. However, a theoretical analysis has shown that in QCA circuits ultralow levels of power dissipation (e.g. 10^{-9}-10^{-11} W/device) should be achievable [67].

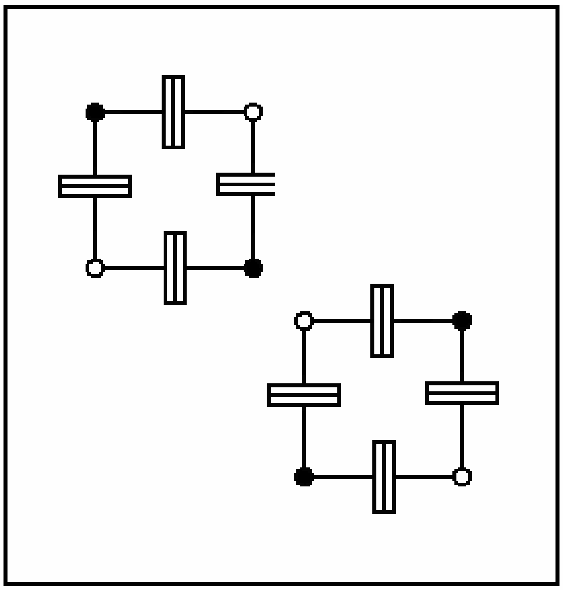

Fig. 2.12 A QCA cell, implemented with tunnelling junctions. In this kind of implementation, there are two electrons in the structure and each electron is constrained between two tunnelling junctions. The electrons tunnel from island to island. The two polarization states, due to Coulomb repulsion, are shown. Redrawn from [2].

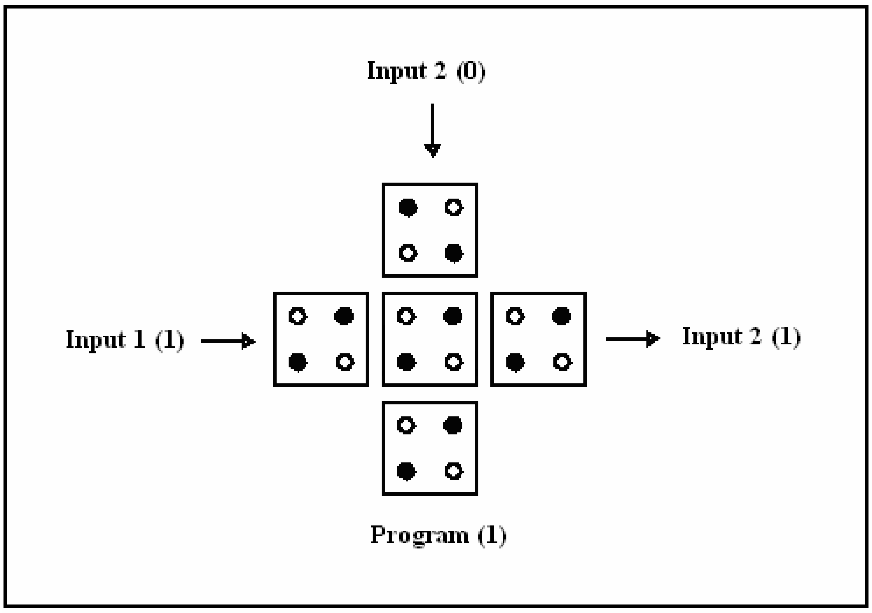

Fig. 2.13 A programmable AND/OR (majority) gate, based on QCAs. The central cell performs majority voting among the two input cells and the control cell. If the control cell is set to 0 the device works as an AND gate, if it is set to 1 it works as an OR gate. In the example, the control cell is set to 1, and we have an OR gate. Redrawn from [2].

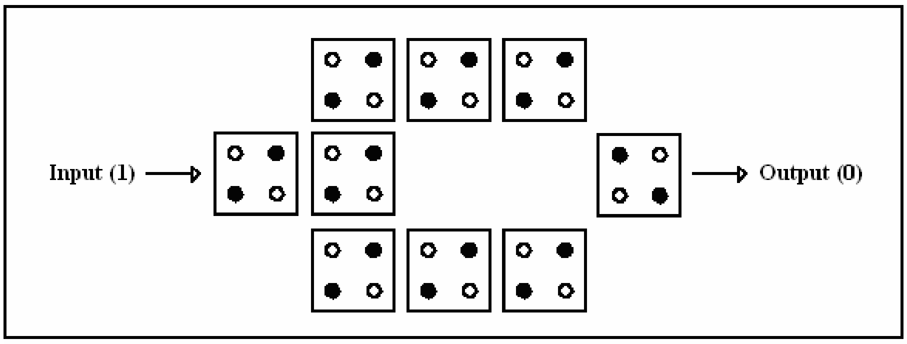

Fig. 2.14 A NOT gate, based on QCAs. The input line extends one cell beyond the beginning of the two circuit branches. The input signal is propagated unaltered through the branches, due to electrostatic repulsion. The two branches, then, converge onto the output line. The electrostatic repulsion causes the input signal to be inverted. Redrawn from [2].

Different approaches have been proposed for the implementation of QCAs [2]. In particular: Si quantum dots on an insulating SiO_2 layer, potential wells (islands) coupled with tunnelling elements, see Fig (2.12), molecular implementation.

It has also to be mentioned that simulation work made in Pisa [68] has shown that QCA devices will be extremely sensitive to manufacturing imperfections. This is, of course, a good reason to implement fault and defect-tolerant solutions.

2.5.1 QCA Logic Devices

Logic gates based upon the QCA principle have been proposed [7]. Figs. (2.13) and (2.14) show and explain a programmable AND/OR and a NOT gate.

A logic gate of the above-mentioned kind has been built. However, the device consisted of one only cell, with the adjacent cells simulated by suitable gate electrodes [66]. An isolated QCA cell, operated below 50 mK, has been built [69]. A two-stage QCA shift register, operated at ~15 mK, has also been demonstrated [62].

2.5.2 QCA Memory Devices

At the conceptual level, some designs have been proposed for QCA units. In particular, a QCA memory design has been proposed at UCL [63]. This memory unit is built out of standardized QCA sub-units, implementing logic gates and wires. An adder design has also been proposed [35].

2.6 Molecular Devices

Molecules are presumably the smallest nanoelectronic device that can ever be implemented. It might seem strange, at first, that molecules can be employed as computational devices. However, this seems less strange when one considers the structure of a typical organic molecule, with a cloud of de-localized electrons [25]. One might think of controlling the conductance of a molecule by an external electric field. This would implement a molecular switch.

2.6.1 Molecular Memory Circuits

A scheme for a molecular memory, based on the influence of the position of chemical side-groups attached to aromatic molecules on the electrons propagating through these molecules has been proposed [70].

An interesting approach to macromolecular memories makes use of molecules that change their configuration state according to the frequency of the light they absorb. For example, bacteriorhodopsin changes its state according to whether it absorbs green or red light. The absorption of blue light, instead, erases the memory. One trillion bits/cm^2 should be achievable with this approach [2].

2.6.2 Molecular Logic Circuits

Molecular wires have already been demonstrated [71]. A scheme for designing molecule-based logic circuits has also been suggested [72]. The basic building blocks of wires, resistances, diodes, implemented with macromolecules, are shown in Figs. (2.15), (2.16) and (2.17), respectively. Logic gates, built out of these building blocks, are in turn shown in Fig. (2.18).

2.6.3 Carbon Nanotubes

Carbon nanotubes can be considered as molecules, and carbon nanotube transistors have already been built [73-75]. Molecules (or carbon nanotubes) can be mechanically compressed, typically by means of a scanning tunnelling microscope tip, so as to vary their electrical properties. In this way, transistor-like structures, and logic gates, can be built. A few of these structures have already been fabricated, using C_{60} molecules [76] or carbon nanotubes [77]. An adder composed of 464 C_{60} transistors [78,79] and various nanotube-based logic circuits [80,81] have also been proposed.

2.6.4 Semiconductor Nanowires

Finally, semiconductor nanowires also offer interesting perspectives to nanoelectronics. By using n and p-doped semiconductor nanowires, in fact, it is possible to assemble the analogous of conventional microelectronic devices. In fact, diodes, bipolar transistors and inverters based upon crossed nanowires have already been built [82-84].

2.7 Conclusions

Nanoelectronics offers exciting perspectives for the future. Packing densities up to ten thousand times greater than today's densities could be theoretically expected. However, we have seen that most of nanoelectronic devices are at present only theoretical proposals. Those which have actually been built are not exactly "nanodevices", since their typical length scales are rather in the 10-100 nm range, though some features (e.g. tunnelling junctions) are already at the nanometre scale. Moreover, as we have seen, in most cases only simple logic gates, or even wires, have been built.

Nanoelectronics has other drawbacks [2]. In fact, the huge number of devices on nanochips will make mass-scale production a challenging task. In this context, self-assembly techniques might provide a solution. However, this techniques are suited to non conventional computing architectures such as neural networks. These architectures, in turn, being able to work with uncertain information, might be suited to cope, together with more conventional techniques, with the high failure rates that, as shown in this thesis, derive from the quantum nature of nanoelectronic devices and the huge density of devices.

It is expected that nanoelectronic devices will suffer from wiring limitations. For instance, 20 nm wide wires, due to distortion problems, will only be able to propagate information for a few microns [2]. QCAs are also expected to suffer from wiring problems. Due to their geometry, in fact, classical wiring schemes would imply low packing densities. In this context, the above-mentioned non conventional architectures might be a solution. Three-dimensional integration might be a solution, as well, since it would require shorter wires [2].

In conclusion, it seems [2] that such concepts as self organization, fuzzy information, soft computing, fault tolerance, three-dimensional integration will be crucial for coping with the quantum nature and packing density of the future nanochips. The aim of this thesis is to show how traditional fault tolerant architecture might alleviate some of the above-mentioned problems. All these issues will hopefully be a stimulus for further technological research.

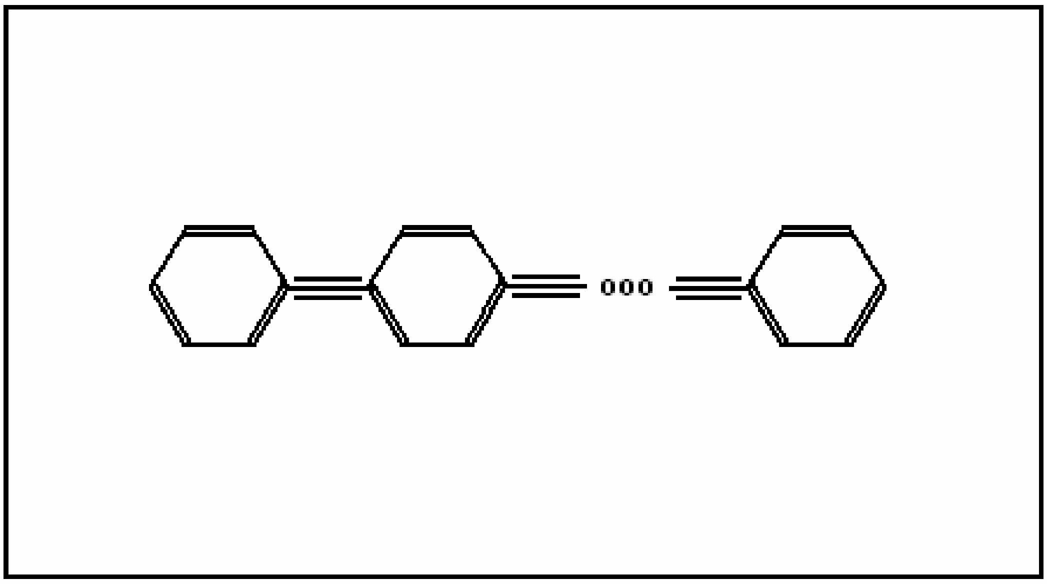

Fig. 2.15 A molecular wire. The polyphenylene-based molecule shown here acts as a wire. In fact, the delocalized electrons in it are able to conduct a current. The molecule is contacted with gold electrodes. Redrawn from [2].

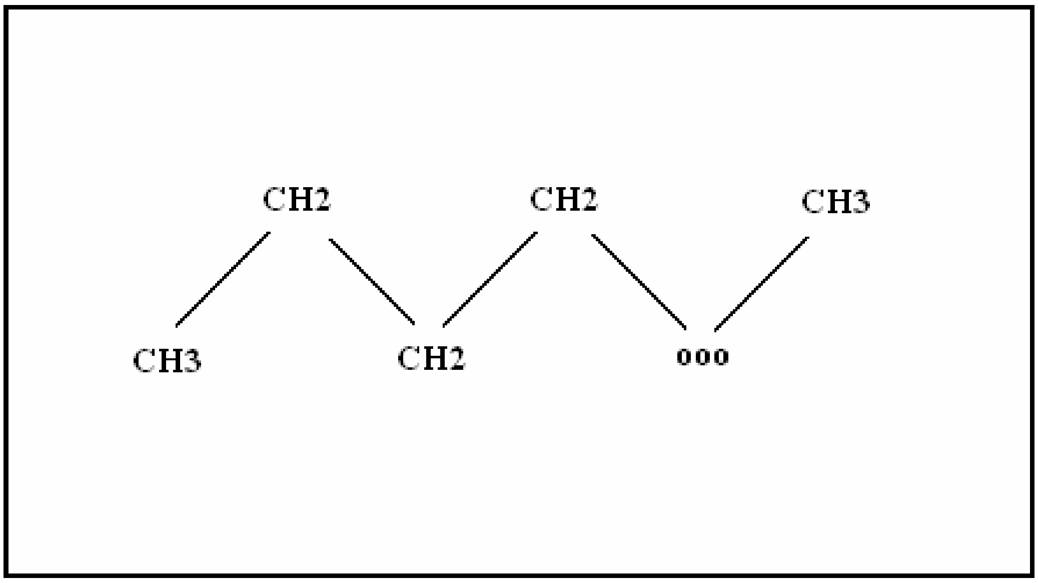

Fig. 2.16 A molecular insulator/resistor. The polymethylene-based molecule shown here acts as a insulator, i.e. a resistor with high resistance. In fact, the bound electrons in it are unable to conduct a current. The molecule is contacted with gold electrodes. Redrawn from [2]

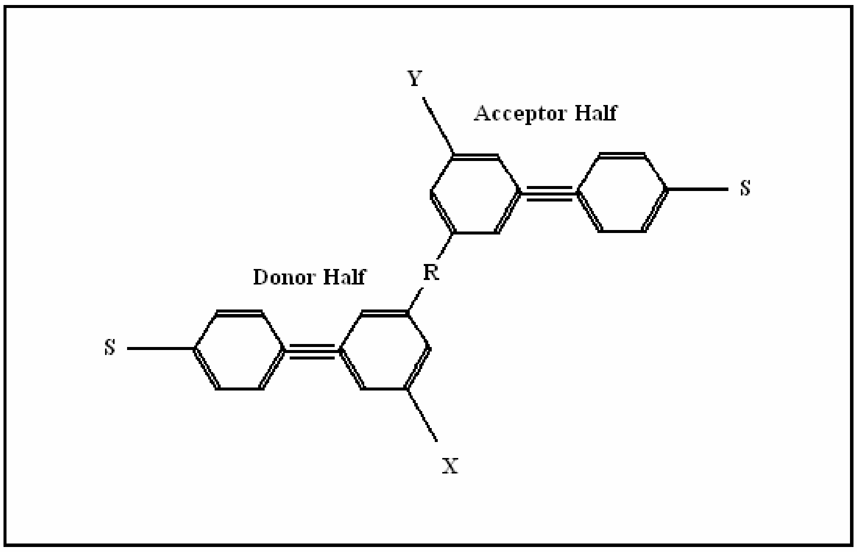

Fig. 2.17 A molecular diode. The molecule shown here has a donor and an acceptor halves. Due to its delocalized electrons, it is able to conduct a current. However, the electron current can only flow from the donor to the acceptor half. The molecule is contacted with gold electrodes. Redrawn from [2].

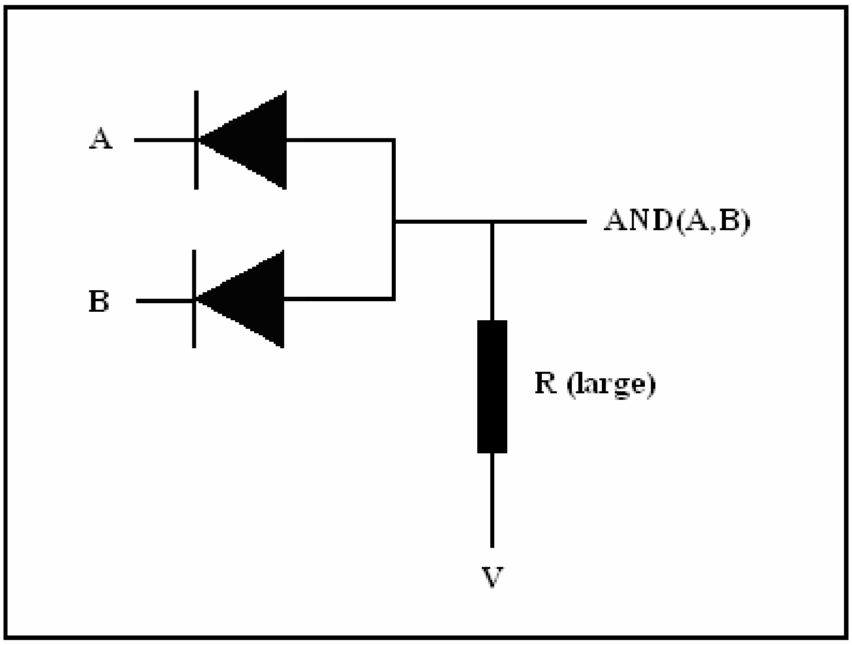

Fig. 2.18 A molecular AND gate. The molecule shown here is composed of two molecular diodes and one molecular resistor, see Figs. (2.15) to (2.17). When A and B are high, both diodes do not conduct and there is no current. The output is then at potential V, i.e. at 1. When A and/or B are low, at least one diode conducts. Due to the large value of R, the output is then low, i.e. at 0. The molecule is contacted with gold electrodes. By reversing the diode polarity and applying a negative voltage, an OR gate is obtained. Redrawn from [2].

Chapter 3 Fault Tolerance

3.1 Introduction

The introduction of nanometre-scale components should make it possible to conceive of chips containing 10^{11} or more logic gates [2]. Such an assembly of components will most probably require the introduction of fault-tolerant techniques, because the huge number of devices is expected to make a chip unreliable (in a well-defined technical sense, see Chap. 5 and 6), even if its devices are intrinsically highly reliable. In this chapter, we give an account of the basic principles of fault tolerance.

Before proceeding, it has to be said that, after this chapter was first drafted, more recent work on fault tolerance has appeared [85-87]. In particular, the NAND multiplexing technique [88], introduced by Von Neumann and based on a massive duplication of imperfect devices and randomized imperfect connections, has been re-discovered and applied to the problem of fault tolerance in nanodevices. In our opinion, such developments seem not to affect the significance of this work, which presents useful fault tolerant techniques, with similar redundancy levels.

3.2 The Basic Principles

An introduction to the basic principles of fault tolerance is given in Ref. [89]. The first relevant concept is that of a system. A system is a collection of components, which are coordinated so as to perform a given function. When one of the components fails to perform the function it was designed for, we have a fault. If the system's performance is affected by this fault, we say that the system had a failure. Fault tolerance consists in designing the components of a system so that their faults are not turned into failures.

Faults can either be transient or permanent. Permanent faults can either be due to manufacturing defects or occur during the lifetime of a system. Transient and permanent faults require different fault tolerant strategies, all of which exploit space, time or information redundancy.

3.3 Space-Redundant Techniques

In space-redundant techniques [9], each potentially faulty unit is replaced by a number of replicated units so that, by majority voting or other means, faults can be masked to a certain extent. Space-redundant techniques are collectively known as general modular redundancy (GMR) [90-97] and reconfiguration [98].

The basic idea behind GMR [92] is that the place of a potentially faulty unit is taken by a number of online units, among which some kind of a voting process is performed. Whenever an online unit is faulty, a spare unit is switched in to replace it. Should spare units run out, the system exhibits graceful degradation, until the number of working units falls below a certain level.

In reconfiguration strategies [98], if a functional unit is faulty its links to the other units are reconfigured with various algorithms, so as to bypass it. Spare units are needed, of course.

As we will see in the next chapters, the errors we will mainly consider are of a transient nature. Reconfiguration, of course, is not applicable to such errors. For these reasons, here we do not describe reconfiguration algorithms in any more detail. The reader is referred to Appendix 3, where reconfiguration is presented in the context of parallel computing.

3.3.1 N-Modular Redundancy

To our knowledge, the only available space-redundant techniques for dealing with transient faults are triple modular redundancy (TMR) [93], Fig. (3.1), and its generalization, N-modular redundancy (NMR) [91]. In NMR, the place of each potentially faulty unit is taken by a block of N identical units and majority voting is performed. N is odd, of course. NMR and TMR are particular cases of GMR, in which there are no spare units and majority voting is performed

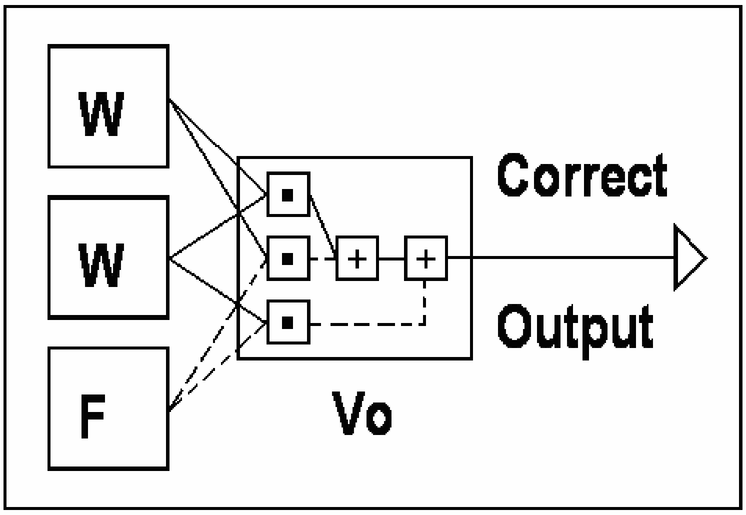

Fig. 3.1 A triple modular redundancy (TMR) unit. Three copies of the potentially faulty devices send their output to a voter Vo, which performs majority voting. The majority voting unit is made up of AND (·) and OR (+) gates. Its answer is taken to be the correct output. W = working device, F = failing device. Zero outputs are marked with a dashed line.

A NMR unit cannot tolerate (N+1)/2 or more faults. If the intrinsic failing probability per clock cycle of an element, $P_{f;u}$, is small, the failing probability of an NMR unit, $P_{f/nmr}$, can be calculated using binomial statistics, with a simple result. For binary units and perfectly reliable voting circuitry, one has [99,100]:

$$P_{f/nmr} = \begin{pmatrix} N \\ \frac{N+1}{2} \end{pmatrix} \cdot (P_{f;u})^{\frac{N+1}{2}} \tag{3.1}$$

For the general case of an imperfect voting circuitry with (small) failing probability per clock cycle $P_{f;v}$, Eq. (4.8) holds and:

$$P_{f/nmr} = \begin{pmatrix} N \\ \frac{N+1}{2} \end{pmatrix} \cdot (P_{f;u})^{\frac{N+1}{2}} + P_{f;v} \tag{3.2}$$

A TMR voting unit can be implemented as:

$$\mathrm{MV}(x,y,z) = xy + xz + yz \tag{3.3}$$

where x,y,z are Boolean variables and MV is the majority voting function. A NMR unit can be implemented by summing over all the (N+1)/2-ples of variables.

In Chap. 6, we will introduce a generalization of GMR (and NMR), named cascaded general modular redundancy (CGMR) [12,13]. According to CGMR, the potentially faulty units are firstly clustered in a suitable way and modular redundancy is applied to the clusters. The clusters are then suitably clustered, as well, and modular redundancy is applied to each cluster.

3.4 Time-Redundant Techniques

In time-redundant techniques [9], processor instructions are suitably repeated, so as to achieve fault tolerance. In particular, one can either mask errors with modular redundancy or detect them and restart processor operation from the previous state (i.e. backward error recovery [89]). Instructions can be repeated at any level, from the single-bit to software level (i.e. software redundancy [9]). Here we will only consider instructions at the single-bit level. Any software instruction is finally translated into bit exchanges within the processor. Our approach, then, can implicitly give indications on the possible effectiveness of software-redundant strategies.

3.4.1 Time-Domain N-Modular Redundancy

An application of error-masking is suggested in Ref. [101] in the form of a voting unit accumulating results, so as to perform TMR (or, more generally, NMR) in the time domain. In Chap. 6, we will present a cascaded version of time-domain NMR we devised, taking inspiration from Ref. [101].

3.4.2 Backward Error Recovery

In backward error recovery (BER) [89], the outputs of N copies of suitable functional units are compared through a logical OR between all the possible terms of the form $x_i \oplus x_j$. A disagreement signal from any of the N-ples of replicated units causes the system to step back to the previous state. For example, triplication would lead to the following error detection function, ED:

$$\mathrm{ED}(x,y,z) = x \oplus y + x \oplus z + y \oplus z \tag{3.4}$$

where $\oplus$ is the XOR function, see Fig. (3.2).

An arrangement like that of Fig. (3.2) is able to detect faults occurring in the units, unless all the N units fail. For a unit with failing probability $P_{f;u}$ and imperfect voting circuitry with failing probability $P_{f;v}$, under the hypothesis that both probabilities are very small (we will see in Chap. 6 and 7 that this is the case), one has:

$$P_{f/ber} = (P_{f;u})^N + P_{f;v} \tag{3.5}$$

In Chap. 6, we show that the CGMR can also be applied to BER units.

3.5 Information-Redundant Techniques

In information-redundant strategies [9], the information stored in a chip is made redundant through the use of error-correcting codes [10]. In an error-correcting code, a suitable encoding process is applied to the bit words one wants to protect, so that errors can be located and corrected. Among the codes with practical applications, the most widely known is the parity check code, widely used in computer systems [10]. This code allows detection, but not correction, of one error. Hamming codes [11], in turn, are also based upon parity checking and are able to detect and correct one error. On the other hand, Reed-Muller and Golay codes [10] have been extensively used in space applications.

It is beyond the scope of this thesis to give a detailed account of the theory of error-correcting codes and its many practical applications. The reader is referred to Ref. [10], or similar texts, for an extended treatment. Here we will only outline the basic principles of error correcting code theory by generalizing the examples of Hamming and Reed-Muller codes, since they are the codes used in Chap. 6.

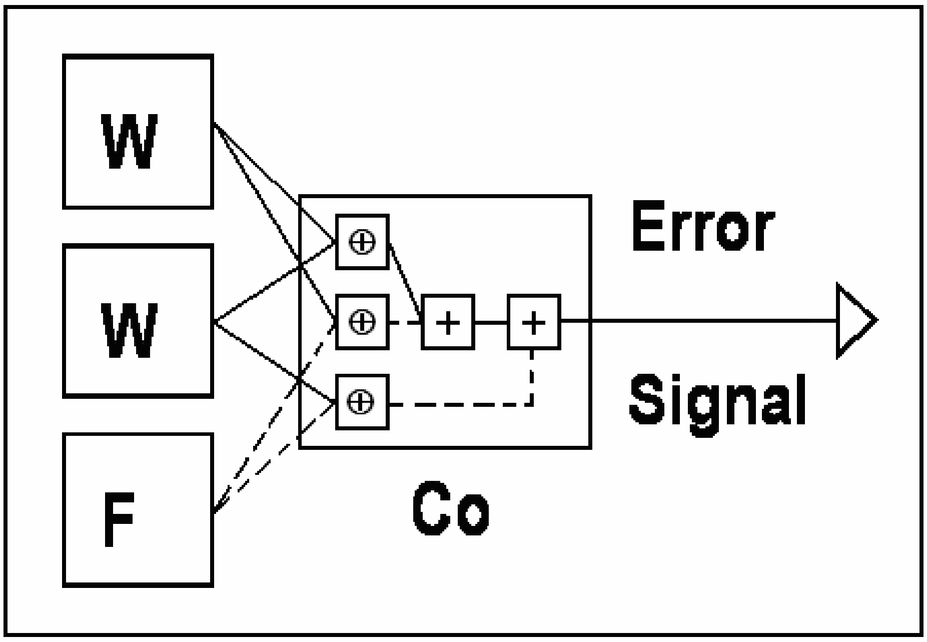

Fig. 3.2 A backward error recovery (BER) unit with triplication. Three copies of the potentially faulty devices send their output to a comparator Co. The comparator, made up of OR (+) and XOR (⊕) gates, detects any disagreement between the outputs and emits an error signal. W = working device, F = failing device. Zero outputs are marked with a dashed line. The error signal is given by a 1.

3.5.1 Hamming Codes

In Hamming codes [11], parity checking is used to locate and correct a maximum of one error. In order to illustrate the basic ideas of Hamming coding, we consider the case of 4-bit strings. This case is of limited practical interest but allows one to keep the discussion at a simple enough level, while being easy to generalize.

Let us then consider a 4-bit string. We will have bit 2^0 (i.e. bit 1), bit 2^1 (i.e. bit 2), bit 2^0+2^1 (i.e. bit 3) and bit 2^2 (i.e. bit 4). The bits whose associated number contains only one power of 2 (bits 1, 2 and 4, in our case) are defined as the check bits. The remaining ones are defined as the information bits. The information bits (bit 3, in our case) contain the information that has to be protected from errors.

How do we set the values of the check bits? The rule is that the check bit whose position is expressed by 2^i must be defined so that the set of all the bit positions containing 2^i in their binary representation contains an even number of ones. As a concrete example, let us suppose that the information bit is set to 1 and let us determine the values of the check bits:

- Let us first consider check bit number 1 (2^0). Bits number 1 (2^0) and 3 (2^0+2^1) are associated to it. Bit number 3 is set to 1 by hypothesis. We must then set bit number 1 to one, so that the set contains an even number (two) of ones.
- Let us then consider check bit number 2 (2^1). Bits number 2 (2^1) and 3 (2^0+2^1) are associated to it. Bit number 3 is set to 1 by hypothesis. We must then set bit number 2 to one, so that the set contains an even number (two) of ones.
- Let us finally consider check bit number 4 (2^2). There are no other bits associated to it. It must then be set to zero, so that the set has an even number (actually, zero) of bits.

Summarizing, we have to protect the (toy) string '1'. We express it as (xx1x), where the x's are check bits. Hamming coding sets the value of the 4-bit string to (1110).

In Hamming decoding [11], all the checksums are considered. Notice is taken of those checksums that are unbalanced and the sum of their positions gives the location of the error. Coming to our toy example, let us then suppose that the information bit number 3 has been flipped to zero. The corrupted string is then (1100) and we want to locate and correct the error. The following considerations can be made:

- Let us first consider the checksum identified by bit number 1 (2^0). Bits number 1 (2^0) and 3 (2^0+2^1) are associated to it. The total number of ones in the checksum is 1, so it is odd. This checksum is then unbalanced.
- Let us then consider check bit number 2 (2^1). Bits number 2 (2^1) and 3 (2^0+2^1) are associated to it. The total number of ones in the checksum is then 1, so it is odd. This checksum is then unbalanced.
- Let us finally consider check bit number 4 (2^2). There are no other bits associated to it. The total number of ones in the checksum is 0, so it is even. This checksum is then balanced.

The unbalanced checksums are then checksums number 1 and 2, so that the error is located in position 1 + 2 = 3, as it should be, and can be corrected.

3.5.2 Reed-Muller Codes

Reed-Muller codes [10] are best described by listing their codewords by induction. We start by defining the 0th order Reed-Muller code as the set:

$$R(0) = \{(00),(01),(10),(11)\} \tag{3.6}$$

The (n+1)th order Reed Muller code is then defined by induction as:

$$R(n+1) = \{ u\,u \mid u \in R(n) \} \cup \{ u\,u^c \mid u \in R(n) \} \tag{3.7}$$

In other words, R(n+1) is obtained by considering the set of the strings obtained by doubling all the strings of R(n) and the strings obtained by joining any string of R(n) to its binary complement. To make a specific example:

$$R(1) = \{ (0000),(0101),(1010),(1111),(0011),(0110),(1001),(1100) \} \tag{3.8}$$

Let us first define the Hamming distance [11] between two binary strings as the number of places where the corresponding figures differ, e.g. the Hamming distance between (0010) and (1000) is 2 and the Hamming distance between (0010) and (0000) is 1. The minimum distance between two string in the set (3.8) is 2. Suppose the 8 strings in (3.8) are broadcast in a noisy channel, so that some of the bits are flipped. As the (possibly corrupted) string goes out of the channel, we can associate it to the nearest string in (3.8). The error can then be located and corrected.

Let us be specific and consider the toy example of a 4-bit string that has to be protected against errors. We encode and decode the string according to the rule that the string whose decimal representation is n corresponds to the nth component of the list:

$$R(2) = \begin{pmatrix} (00000000), & (01010101), & (10101010), & (11111111), \\ (00110011), & (01100110), & (10011001), & (11001100), \\ (00001111), & (01011010), & (10100101), & (11110000), \\ (00111100), & (01101001), & (10010110), & (11000011) \end{pmatrix} \qquad (3.9)$$

where the 1st column has components 1 to 4, the 2nd column 5 to 8 etc. Suppose, then, that we broadcast the string (0101) through a noisy channel and bit number 3 of the coded message is flipped. The encoding/decoding sequence is:

- The string (0101) is coded as (00001111).
- The string (00001111) is broadcast through the noisy channel.
- The third bit of the coded string is flipped, so it becomes (00101111).
- The string (00101111) does not belong to R(2), so the error is detected.
- To decode, we look for the string of R(2) which is nearest to (00101111).
- The Hamming distances to (00101111) are (5,5,4,4|4,4,5,5|1,5,4,8|4,4,5,5).
- The corrupted string is then decoded as the 3rd component of R(2).
- We then recover the string (0101) and the error is corrected.

3.5.3 Error Rates

Let us then generalize what has been said in the previous section. In general [10], one has a set S of strings of a given length, taken from a certain alphabet. These strings are broadcast in a noisy channel. In order to detect and correct the potential errors, the set s is coded by putting it in a one-to-one correspondence with the elements of a suitable subset of a vector space, the code C. In this thesis, we only consider codes made up of suitable binary strings. A binary string of length n can, of course, be seen as a vector in a n-dimensional space with binary coefficients. In such a space, a sphere rather looks like a hypercube with unit sides. For the sake of clarity, however, in Figs. 3.3 to 3.6 we pretend we are dealing with ordinary spheres.

Since the code C has a finite number of elements (the codewords), there will be a minimum distance d_{min} between the codewords, see Fig. 3.3. Let us suppose that a string s of C is broadcast through a channel, where noise can corrupt it. If the corrupted string s_1 has a distance d from s such that $d \leq ½\cdot d_{min}$, then the errors due to noise can be corrected. To this purpose, it is enough to agree that all the strings belonging to the sphere of radius $½\cdot d_{min}$, and centred on s have to be identified with s itself, see Fig. 3.4. If the corrupted string s_1 goes out of the sphere of radius $½\cdot d_{min}$ and centred on s, while remaining out of any sphere centred on a codeword, the error can be detected but not corrected, see Fig. 3.5. Otherwise an error occurs, see Fig. 3.6.

From these considerations it follows that, if d_{cw} is the minimum Hamming distance (or any other suitable distance, in the technical sense of metric space theory), between two codewords in a code, the number of errors the code can correct, N_e, is [10]:

$$N_e = [\ \frac{d_{cw} - 1}{2}\ ] \tag{3.10}$$

where [·] is the integer part function.

Error-correcting codes can e.g. be applied to memory chips. We assume that every memory cell of the chip is refreshed at each clock cycle, with a finite failure probability. We then suppose that a certain error-correcting code has the capability of correcting N_e errors. The data word is written to the memory and stays there for N_s clock cycles. Once a bit error has been generated, the memory cell's feedback loop makes it permanent. If the memory cell's fault rate per clock cycle, $P_{f;m}$, satisfies the usual relationship $P_{f;m} << 1$, it is possible to give a formula for the probability that a N_b-bit memory word is corrupted by a number of errors $\geq N_e+1$, after being stored for N_s clock cycles [10]:

$$P_{f/ecc} = \begin{pmatrix} N_b \\ N_e + 1 \end{pmatrix} \cdot (N_s \cdot P_{f;m})^{N_e+1} \tag{3.11}$$

We will come back on the significance of Eq. (3.11) in Chap. 6, in relation to error rates in nanoelectronic memory chips.

3.6 Conclusions

In this chapter, we have presented the basic ideas behind fault tolerance. We have first presented some basic concepts of the field, dealing with transient and permanent errors, fault detection and error recovery, fault masking and redundancy. Having introduced these basic concepts, we have described the various approaches to fault tolerance. We have first examined space-redundant techniques, and in particular n-modular redundancy (NMR). We have then turned our attention to time-redundant techniques, and shown their relationship with space-redundant techniques. Finally, we have examined the basic principles underlying error-correcting codes.

Fault tolerance is achieved at the expense of redundancy. We have seen, in fact, that any fault tolerant system exhibits space, time or information redundancy. Moreover, the three kinds of redundancy are never mutually exclusive. Space redundant system are also partly time redundant and vice-versa and the implementation of error correcting codes implies space and time redundancy.

In the following chapters, we will see that logic nanochips are expected to require substantial amounts of either space or time redundancy and memory nanochips will probably require error correcting codes, with both space and time redundancy. Space can be traded for time, at least within limits, but redundancies up to one or two orders of magnitude will have to be accepted, if nanoelectronic devices are to work.

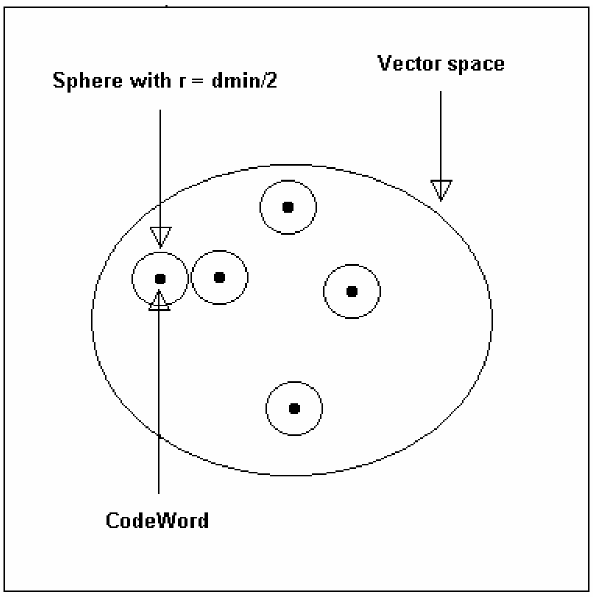

Fig. 3.3 The basic principle of error correcting codes. The message to broadcast in a noisy channel is encoded as a sequence of codewords. Each codeword can be identified with a vector in an n-dimensional vector space and is surrounded by a sphere, whose radius is half the minimum distance between the codewords.

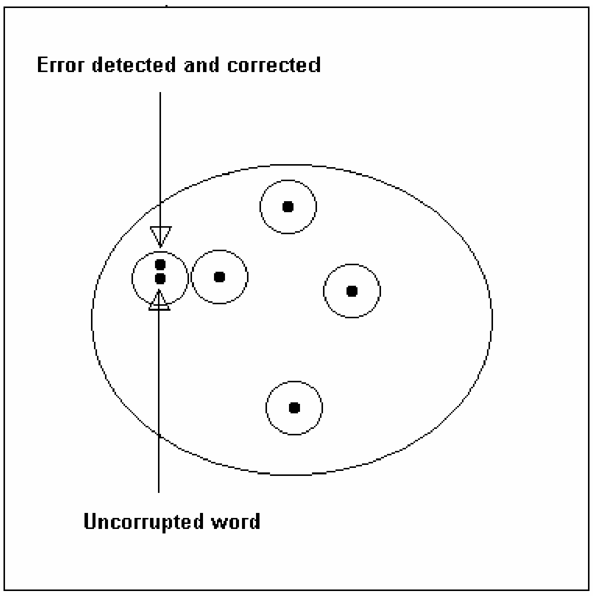

Fig. 3.4 An error that can be corrected. One of the words in the message, encoded in a codeword, is corrupted by noise. The corrupted vector still lies in the error correction sphere of the uncorrupted vector. The error can then be corrected.

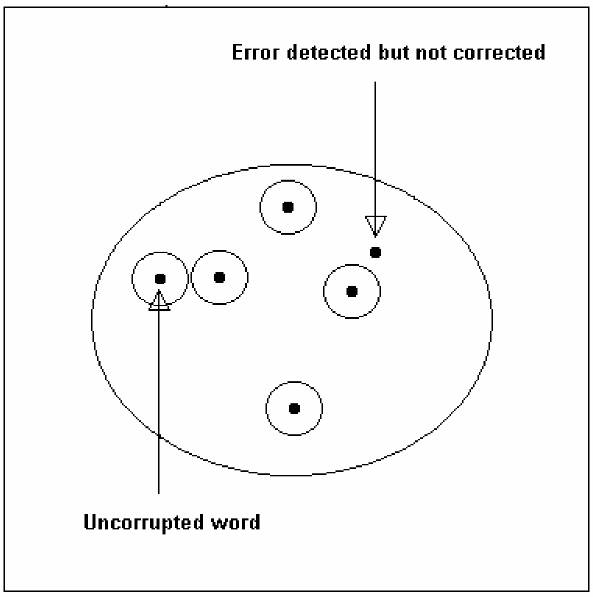

Fig. 3.5 An error that can be detected. One of the words in the message, encoded in a codeword, is corrupted by noise. The corrupted vector lies outside the error correction sphere of the uncorrupted vector, while not lying in any other sphere. The error can then be detected but nor corrected.

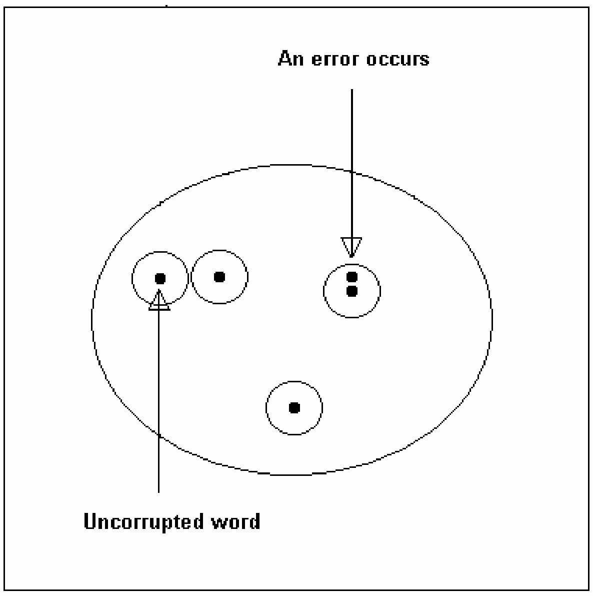

Fig. 3.6 An error that cannot be corrected. One of the words in the message, encoded in a codeword, is corrupted by noise. The corrupted vector lies outside the error correction sphere of the uncorrupted vector, while lying into another sphere. Despite the error correcting code, an error occurs.

Chapter 4 A Model for Chip Error Rates

4.1 Introduction

In this chapter, a model for chip error rates is presented. In the meantime, the terminology to be used in the next chapters is set up. In order to calculate the probability for a chip to work, three levels are considered: the device level, the circuit level and the chip level. The chip is composed of a number of circuits, which are composed of a number of devices. The chip is assumed to fail if any of its circuits fail. A circuit is assumed to fail if any of its devices fail. This approach is used in the next chapters, where the circuit level corresponds to the cluster level of Chap. 6.

4.2 Device Error Rates

The failing probability per clock cycle of a device d in a circuit c, $P_{f;dc,}$ is:

$$P_{f;dc} = \sum_{s} P_{f/s;dc} + P_{f/\mathrm{int};dc} \tag{4.1}$$

where $P_{f/s;dc}$ is the probability per clock cycle for a fault source s to strike the device and $P_{f/int;dc}$ is the probability per clock cycle of faults due to interference among different sources, typical of quantum mechanical systems. Eq. (4.1) can be simplified if, as shown in Chap. 5, $P_{f/s;dc}$ is small. In fact, considering the matrix element formalism in quantum mechanics [43], it is easy to see that $P_{f/int;dc}$ is given by a sum of terms of the form ($P_{f/s1;dc}^{m1}$

$P_{f/s2;dc}{}^{m2}$), where s1 and s2 are different fault sources and m1 and m2 are integer numbers, and similar interference terms among three or more sources. If $P_{f/s;dc}$ is small, the interference terms are also negligible. One then has:

$$P_{f;dc} \approx \sum_s P_{f/s;dc} \tag{4.2}$$

4.3 Circuit Error Rates

Under the assumption that the devices act independently, the working probability per clock cycle of a circuit c is the product of the working probabilities of its devices d. The failing probability per clock cycle of a circuit c, $P_{f;c}$, is then:

$$P_{f;c} = 1 - \prod_d (1 - P_{f;dc}) \tag{4.3}$$

where $P_{f;dc}$ is given by Eqs. (4.1). Eq. (4.3) can be simplified if, as in Chap. 5, $P_{f;dc}$ is small. In this case, in fact, Eq. (4.3) simplifies to:

$$P_{f;c} \approx \sum_d P_{f;dc} \tag{4.4}$$

4.4 Chip Error Rates

It is now possible to calculate the chip failure rate, including the effect of fault tolerant strategies. The functional unit to which fault tolerant techniques are applied (the cluster of gates of Chap. 6) is defined as a circuit c, composed of different devices d. Under the hypothesis that each circuit acts independently, the probability per clock cycle for the system to fail when no fault tolerant techniques are applied, P_f, is:

$$P_f = 1 - \prod_c (1 - P_{f;c}) \tag{4.5}$$

where $P_{f;c}$ is given by Eq. (4.3). If, as in Chap. 5, $P_{f;c}$ is small:

$$P_f \approx \sum_c P_{f;c} \tag{4.6}$$

The probability per clock cycle for the chip to fail with fault tolerance, $P_{f/ft}$, can then be calculated. A chip with fault tolerance provided, in fact, fails if either the number of errors is such that they can be masked, but the voter fails, or the number of errors is such that they cannot be masked and the voter works. So:

$$P_{f/ft} = P_{f;v} \cdot (1 - P_{f/idft}) + P_{f/idft} \cdot (1 - P_{f;v}) \tag{4.7}$$

where $P_{f;v}$ is the failing probability per clock cycle of the voter and $P_{f/idft}$ is the probability per clock cycle for the fault tolerant technique to be unsuccessful in an idealized, perfect voter, and is a non-linear function of P_f, as seen in Chap. 3. If, as in Chap. 5, the probabilities in Eq. (4.7) are small:

$$P_{f/ft} \approx P_{f/idft} + P_{f;v} \tag{4.8}$$

These failing probabilities are explicitly calculated in Chap. 5 and 6.

4.5 Conclusions

In this chapter, a model for chip error rates was built. In addition, the terminology to be used later in this thesis has been set up. The model shows that, provided the error rates for the various sources are small, the overall error rate (with no fault tolerance provided) is simply the sum of the error rates for the various sources. A (generally non-linear) function is then applied at the circuit and chip level, if fault tolerance has to be provided. The values of the various parameters are calculated in Chap. 5 and 6.

The model presented here can easily be generalised to more and/or different levels. For example, the probability per clock cycle for a fault source to be effective on a given device might vary according to the device area where the fault source happens to be located. In this case, a sub-device level should be added. Fault-tolerant solutions might also be applied to the device and/or chip levels. The model can equally be generalized to permanent (typically, manufacturing) errors.

In the next chapters, error rate mechanisms for two families of nanoelectronic devices will be considered and fault-tolerant strategies for both logic and memory chips will be proposed.

4.6 List of Symbols

The following terms, introduced in this chapter, are used throughout the thesis. We list them here below, in hierarchical order:

$P_{f/s;dc}$	Failing probability of device d in circuit c, given error source s
$P_{f/int;dc}$	Failing probability of device d in circuit c, interference term
$P_{f;dc}$	Overall failing probability of device d in circuit c
$P_{f;c}$	Failing probability of circuit c, no fault tolerance
P_{f}	Overall chip failing probability, no fault tolerance
$P_{f/idft}$	Overall chip failing probability, fault tolerance provided, perfect voter
$P_{f;v}$	Failing probability of voter v
$P_{f/ft}$	Overall chip failing probability, fault tolerance provided, imperfect voter

where all the probabilities are per clock cycle.

Chapter 5 Nanodevice Error Rates

5.1 Introduction

Faults in nanodevices can either be transient or permanent [89]. Faults due to the quantum nature of the devices (intrinsic faults) are transient. The device gives a wrong output at a certain clock cycle but what happens in the next cycle is unrelated to this event. The aim of this chapter is to give estimations of intrinsic error rates in nanodevices, conservatively operated at frequencies ~1 GHz.

In this chapter, the attention is focused on those nanodevices that use single electrons or holes as the bit unit (granular devices [2]). The effects of the various possible intrinsic fault sources on nanometre-scale devices are assessed. Order-of-magnitude estimations are only given, since detailed calculations would require a detailed knowledge of the shape, dimensions and composition of the future nanodevices. This is impossible to predict at present, beyond the qualitative level.

5.2 Methodology

The methodology employed has to differ between nanodevices based upon electron transport and electron repulsion.

In the former case, the error rates are experimentally known, but so far there has been no theory allowing one to rescale them to the nanometre regime. However, the non-dimensional parameters upon which they depend are known. The approach followed, then, was to consider the shrinking level necessary for making such devices work at room temperature and GHz frequency, while keeping error rates constant. Fortunately, shrinking the devices to the nanometre scale leads to the desired properties.

In devices based upon electron repulsion, expressions for error rates were calculated that turned out to be exponentially dependent on the energy jump necessary to create an error in the gate. As we will see, such energy jumps have only been calculated numerically for minimum feature sizes > 1 nm, so they had anyway to be rescaled to the nanometre level.

5.3 Electron Pumps

The electron pump [51] is an array of metallic islands, separated by nanometre-scale junctions, through which an electron is made to tunnel sequentially. Its main application is as a charge standard [52,102,103]. However, analyzing intrinsic fault rates in the electron pump allows one to calculate error rates for electron-pump based devices.

5.3.1 The Available Data on Error Rates

There are three kinds of fault sources affecting an electron pump [50]: frequency, thermal and cotunnelling errors. In frequency errors, the electron is pumped too fast as compared to the time scale for tunnelling, so that the desired tunnelling process is missed. In thermal errors, the electron goes the wrong way, acquiring the necessary energy through thermal exchange with the environment. In cotunnelling errors, the electron goes the wrong way by tunnelling through all the junctions in a single event. Experimentally, the main error source for electron pumps at the micron scale, operated at frequencies > ~5 MHz and temperatures < ~0.1 K, is frequency errors [52,102,103]. The corresponding error rate per clock cycle, $P_{f/f;ep}$, was calculated as [50]:

$$P_{f/f;ep} = \exp\left(-\frac{\alpha}{R\cdot C\cdot f}\right) \tag{5.1}$$

where R and C are the tunnelling junction's resistance and capacitance, respectively, and f is the clock frequency. Experimentally, R ≈ 0.35 MΩ and C ≈ 0.25 fF at the micron scale [52,102]. The constant α is given by:

$$\alpha = \frac{n-1}{8\cdot n^2} \tag{5.2}$$

where n is the number of junctions in the pump. If the tunnelling process is missed, the electron goes back through the pump by cotunnelling, in a small time scale as compared to the clocking time [50].

At frequencies < ~5 MHz and temperatures < ~ 0.1 K, the error rate approaches an asymptotic value [52,102,103], as shown in Fig. (5.1). The most likely candidate for this behaviour is photon-assisted cotunnelling [52,102-104]. The electron tunnels the wrong way by absorbing energy from the environment. The most likely noise candidate seems to be charge traps on the device substrate, which slowly relax with time [102]. An asymptotic fault rate per clock cycle of ~10^{-8} was observed for a 7-junction electron pump, with micron-scale islands [52,103]. An asymptotic fault rate per clock cycle of ~10^{-6} was observed for a 5-junction electron pump, with micron-scale islands [102]. As discussed above, rescaling to the nanometre regime is required.

5.3.2 Scaling to the Nanometre Regime

Error rates are controlled by non-dimensional parameters, so that rescaling to the nanometre regime becomes possible.

For frequency and cotunnelling errors, the relevant parameter is [50]:

$$\eta_f = R \cdot C \cdot f \tag{5.3}$$

which is essentially the ratio between the time scale for the tunnelling process and the time scale for the pumping process.

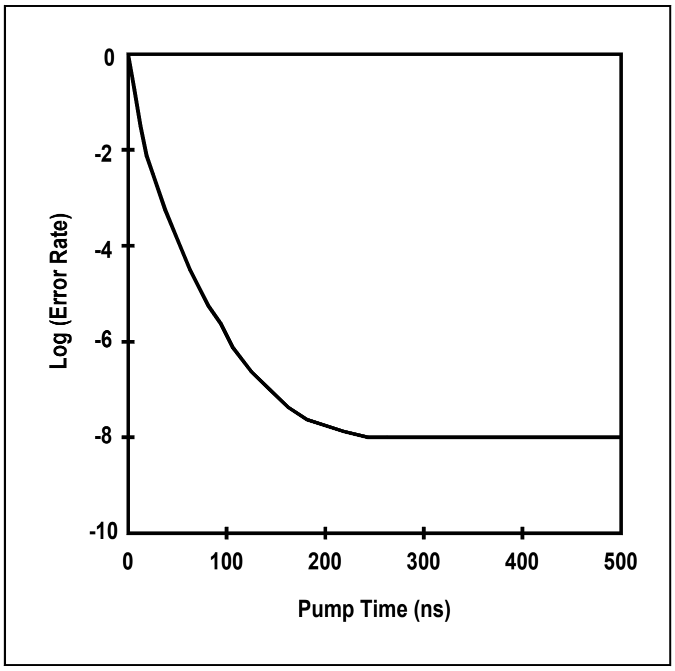

Fig. 5.1 Error rate vs. pumping time for a micron-scale 7-junction electron pump, operated at 35 mK. At high frequencies, the error rate is exponentially dependent on pump time. At low frequencies, the error rate approaches an asymptotic value of ~10^{-8}. The transition takes place at f ≈5 MHz. Redrawn from [52].

For thermal errors, the relevant parameter is the ratio between the energy jump due to a tunnelling event and the thermal energy at the operating temperature, i.e. [50]:

$$\eta_T = \frac{e^2}{C \cdot k \cdot T} \tag{5.4}$$

where e is the electron charge and k is Boltzmann's constant.

Analogously to what happens in CMOS circuits [2], the devices can be operated to higher frequencies and/or different minimum feature sizes, provided the device parameters are rescaled. The relevant parameters, according to Eq. (5.3) and (5.4), are the junction capacitance C and the junction resistance R.

The junction capacitance is given by the well-known formula:

$$C_j = \varepsilon \cdot \frac{A}{d} \tag{5.5}$$

where ε is the dielectric constant of the tunnelling medium. The tunnelling junction resistance is shown to be [105,106]:

$$R = R_k \cdot \exp\left(\frac{d}{d_0}\right) \cdot d_0 \cdot \frac{d}{A} \tag{5.6}$$

where R_k is Von Klinzing's resistance (i.e. the minimum resistance of a tunnelling junction [2], see Chap. 2) and d_0 has the dimensions of a length and can be expressed as a function of system parameters.

In particular, one has

$$R_k = \frac{h}{e^2} \approx 25.8\ k\Omega \tag{5.7}$$

and:

$$\frac{1}{d_0} = 2 \cdot \sqrt{\frac{2 \cdot m \cdot W}{\hbar^2}} \tag{5.8}$$

defines a characteristic length for the tunnelling resistance variation. In the equations shown above, ħ is Planck's (reduced) constant, m is the electron mass, W is the tunnel barrier's height.

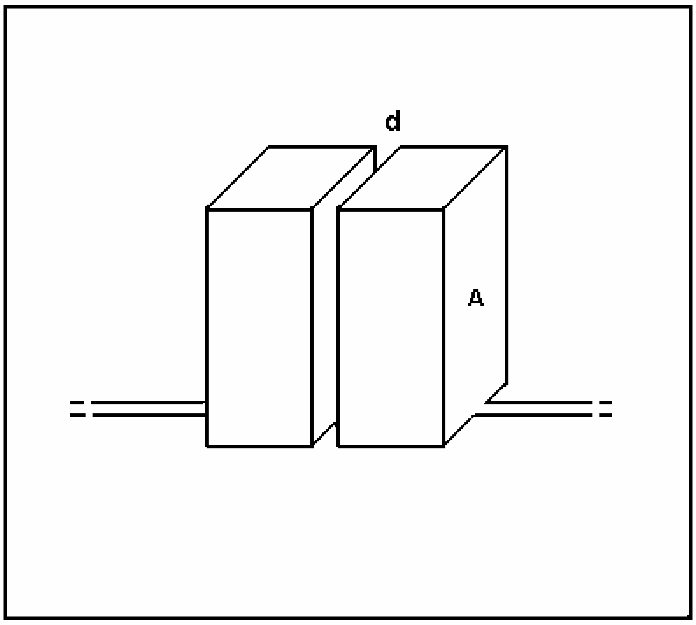

Fig. 5.2 A nanometre scale tunnelling junction. In this conceptual layout, tunnelling takes place in the gap d between the two metal islands. The junction behaves as a capacitor C with face area A and gap d, in parallel to a resistance R, whose expression depends on d and A.

The product R·C, then, is only fixed by the gap width d, see Eq. (5.5) and (5.6), which is already at the nanometre scale and does not need to be rescaled. For scaling purposes, then, we only have to worry about the product C·T.

5.3.3 Nanometre-Scale Operating Conditions

We now consider the shrinking level required for making the devices work at room temperature and GHz frequency, while keeping error rates constant.

Let us rescale the operating temperature. The data were obtained at a temperature of ~0.1 K and we want to work at ~300 K. The transformation is:

$$T \to 3\cdot 10^{3} \cdot T$$

If the product C·T in Eq. (5.4) is to be kept constant, C has also to undergo a transformation. In particular:

$$C \to 0.3\cdot 10^{-3} \cdot C$$

According to Eq. (5.5), this corresponds to shrinking the length scale by:

$$\lambda \to 0.3\cdot 10^{-3} \cdot \lambda$$

Since the characteristic length is ≈ 1 μm in the configuration of Ref. [107], rescaling brings to a characteristic length ≈ 0.3 nm, in the nanometre regime. A nanometre scale junction can then operate at room temperature. As shown above, we do not have to worry about the product R·C·f. As a consequence of this, the junction can be operated in the GHz region.

5.3.4 Error Rates at the Nanometre Scale

Let us now deal with error rates for a nanometre scale electron pump. It will be seen in Chap. 6 that the behaviour of the 9-junction electron pump is worth considering, besides the 5 and 7-junction pumps. To the best of our knowledge, there are no experimental data for the 9-junction electron pump in the nanometre-scale regime. An extrapolation process is then required. We have seen that errors in the asymptotic regime are mainly determined by cotunnelling. The cotunnelling probability for an n-stage electron pump can be calculated [50] and has the form:

$$Log(P_{f/ct;ep}) = \alpha(n) + \beta(n) \cdot n \tag{5.9}$$

where α and β are slowly varying (logarithmic) functions of n. Assuming α and β to be constant over the range considered, it possible to extrapolate linearly the experimental error rates per clock cycle for the 5 and 7-junction electron pumps (10^{-6} and 10^{-8}) to an error rate per clock cycle of ~10^{-10} for a 9-junction electron pump.

5.4 Single-Electron Switches

A single electron switch [6], based on the electron pump, is shown in Fig. (2.9). When the control island is free, the input electron turns left. When it is occupied by an electron, repulsion makes the previous path energetically unfavorable. The input electron, then, turns right. The single electron switch is the building block of a family of logic gates [6], shown in Figs. (2.10) and (2.11).

5.4.1 The Available Data on Error Rates

Single electron switches are affected by switching and pump errors. In fact, the electron can be switched the wrong way, or it can travel back through one of the electron pumps composing the gates, although correctly switched. Pumping errors can be made smaller by adding a number of junctions to the input and output lines. However, switching errors fix an upper limit for device dimensions, since above a certain number of islands they dominate over pumping errors, making any improvement in pumping accuracy pointless. Error probabilities of $\sim 10^{-8}$ and $\sim 10^{-4}$ per clock cycle are reported for right and left switching, respectively [108]. The overall switching error probability is therefore of the order of $\sim 10^{-4}$ per clock cycle.

5.4.2 Error Rates at the Nanometre Scale

Error rates in single-electron switches, $P_{f;es}$, can be estimated, at least when the error sources have different orders of magnitude, as:

$$P_{f;es} = Max\,(P_{f/s;es}, P_{f/ep;es}) \tag{5.10}$$

Here $P_{f/s;es}$ is the switching error rate and $P_{f/ep;es}$ is the error rate per clock cycle of an electron pump with a number of islands given by the minimum path an electron has to travel in the gate.

If one assumes that switching errors can be made negligible as compared to pumping errors, an error rate per clock cycle of $\sim 10^{-10}$ is predicted for a design based on the 9-junction electron pump.

On the other hand, if switching errors dominate, an error rate of $\sim 10^{-4}$ per clock cycle is predicted (see Section 6.4.1). As shown in Chap. 7, such an error rate is not good enough if fault tolerance with an acceptable redundancy is to be achieved. If pumping errors are to dominate the process, the electron gates need to be redesigned. This problem was not tackled in this thesis.

5.5 Korotkov’s Single-Electron Logic Gates

Koroktov’s single-electron logic gates (KSEGs) [109] are essentially arrays of electron pumps, placed at right angles to each other. The electron pumps are initially uncharged, so that both electron and holes are involved. The distance among the pumps is such that tunnelling cannot happen between them, but only within each pump.

When an electron (hole, respectively) reaches the end of an electron pump which is located orthogonal to another pump, it induces a potential difference on this pump. Without this potential difference, tunnelling in this pump would be possible. The potential difference makes tunnelling impossible by repulsion.

By suitably arranging the electron pumps, logic gates can be built, shown in Figs. (5.3) and (5.4). The 0 and 1 states of the output electron pump are represented by electron on the right / hole on the left and electron on the left / hole on the right.

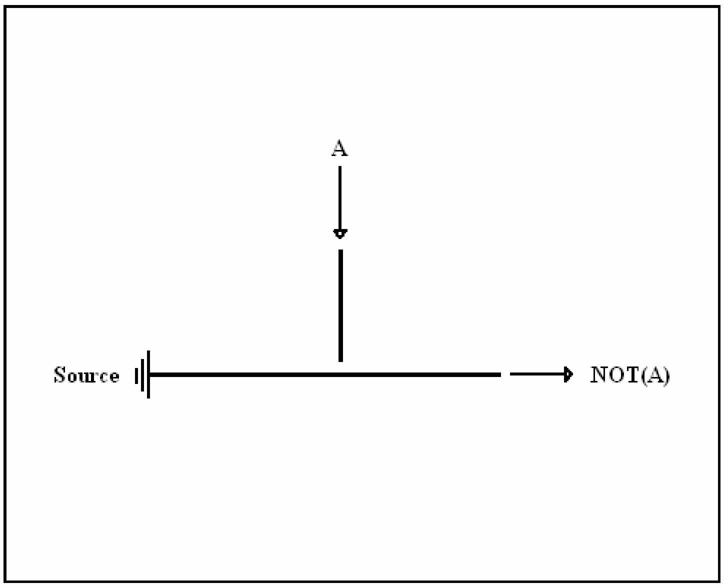

Fig. 5.3 A NOT gate based upon Koroktov's logic gates. The NOT(A) electron pump is only able to drive an electron to the output if there is no electron in the electron pump A. Otherwise, due to electrostatic repulsion the electron cannot tunnel. A NOT function is then implemented. Electron pumps in the circuit are represented as solid lines.

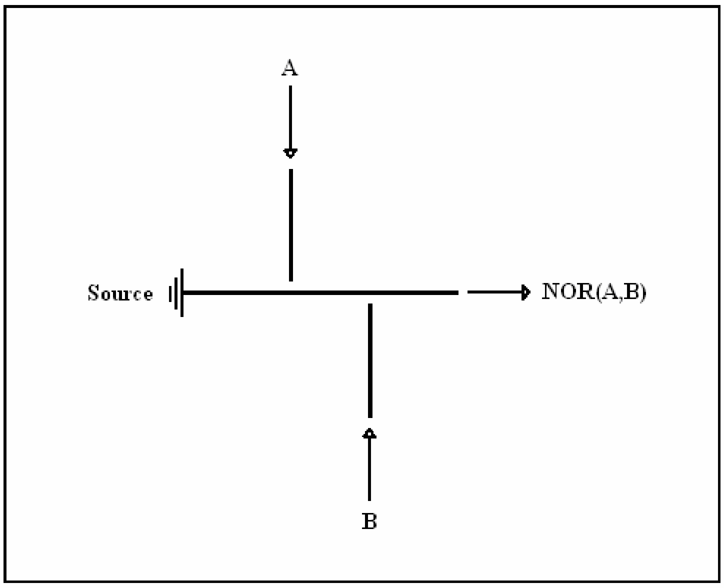

Fig. 5.4 A NOR gate based upon Koroktov's logic gates. The NOR(A,B) electron pump is only able to drive an electron to the output if there are no electrons in both the A and B electron pumps. Otherwise, due to electrostatic repulsion the electron cannot tunnel. Therefore a NOR function is implemented. With a NOR and a NOT function any Boolean function can be implemented. Electron pumps in the circuit are represented as solid lines.

We estimate the error rate per clock cycle of a KSEG, $P_{f;kseg}$, by schematizing it as a set of independent electron pumps and considering the N_p shortest ones, whose error rates dominate:

$$P_{f;kseg} = N_p \cdot P_{f;ep} \tag{5.11}$$

where $P_{f;ep}$ is the error rate per clock cycle of the shortest electron pumps. Since $N_p \sim 10$ and the shortest electron pumps have 8 junctions [109], Eq. (5.11) gives an error rate per clock cycle of $\sim 10^{-8}$.

5.6 The Parametron

The parametron [110], see Fig. (5.5), is a device composed of a number of cells, each cell being composed of three metallic islands, such that the middle island is slightly shifted with respect to the line passing through the centers of the edge islands.

The three islands are initially electrically neutral. A vertical electric field (the clock field) varies periodically between two extreme values. When the clock field is positive enough, the parametron cell remains neutral (i.e. it is switched off). When the clock field goes below a certain threshold, an electron from the upper island is induced to tunnel to one of the lower be determined. The cell is then polarized.

If the 0 and 1 states are taken to be electron on the right / hole on the left and electron on the left / hole on the right, the horizontal field determines the digital state in which the cell is to be found.

Logic gates can be built from parametron cells [110], see Fig. (5.5), by suitably distributing the clock signals over an array of cells and exploiting the electrostatic repulsion among neighbouring cells.

5.6.1 The Available Data on Error Rates

In Ref. [110], an analysis of error rates in parametrons is given. Errors are essentially the as in electron pumps: thermal, frequency and cotunnelling errors.

Thermal errors are calculated with a Boltzmann factor. In the calculation, the simultaneous occurrence of errors in different cells of a logical gate is neglected. The error probability per clock cycle, $P_{f/t;p}$, is [110]:

$$P_{f/t;p} = N_c \cdot \exp\left(-\frac{\Delta E}{k \cdot T}\right) \tag{5.12}$$

where N_c is the number of cells in a gate and ΔE is the energy jump experienced by the electron and must be calculated numerically.

Frequency errors are described by an expression similar to Eq. (5.1) [110], except that the parameter α has to be calculated numerically. To the author's best knowledge, such a process has not yet been performed. In the following, we show that frequency errors, in our conditions, can be neglected.

The error rate estimations presented in Ref. [110] assume that cotunnelling errors can be brought to a negligible level, as compared to the other error sources. Cotunnelling errors, in fact, can be brought to a negligible level by suitably adjusting the distance between the parametron cells [110], since tunnelling probabilities fall exponentially with distance. Therefore, such errors will be neglected.

5.6.2 Error Rates at the Nanometre Scale

We are now in the position to estimate KSEG error rates per clock cycle. The overall error rate per clock cycle, $P_{f/p}$, can in fact be estimated as the maximum value among the thermal and frequency errors. In particular, one has [110]:

$$P_{f;p} = Max\,(P_{f/f;p}, P_{f/t;p}) \tag{5.13}$$

In Ref. [110], spherical islands with a 5 nm diameter D (and radius r = 2.5 nm) are considered. The cell parameters were chosen as d/r = 3 (2d is the distance between the lower islands), b/r = 1 (b is the distance between the top island and the line joining the lower islands). See Fig. (5.5).

The proposed dimensions correspond to the limits set by present-day e-beam writing techniques. Using the above-mentioned parameters, the energy difference was calculated to be $\Delta E \approx 0.03$ eV [110]. According to the goals set in Chap. 1, a minimum feature size $l \approx$ 1nm is desired. A scaling law for ΔE is then needed.

In order to obtain an expression for the scaling law, we observe that when the electron tunnels to one of the lower islands, an electron/hole pair is created. In the absence of an external polarizing field, this process leads to a Coulomb energy difference proportional to $(\varepsilon \cdot \lambda)^{-1}$.

If the external polarizing field is present there is an additional energy term, due to the interaction of the electric dipole formed by the electron and the hole with the external field. The energy term is $2d \cdot E_h$, where E_h is the polarizing horizontal field and 2d is the distance between the two lower islands [110]. One has:

$$\Delta E - E_h \cdot 2d \propto \varepsilon^{-1} \cdot \lambda^{-1} \tag{5.14}$$

where E_h is the polarizing horizontal electric field and 2d is the distance between the two lower islands. If, as in Ref. [109], the polarizing field E_h is chosen to scale like $(\varepsilon\lambda)^{-1}$, ΔE itself is found to scale like $(\varepsilon \cdot \lambda)^{-1}$.

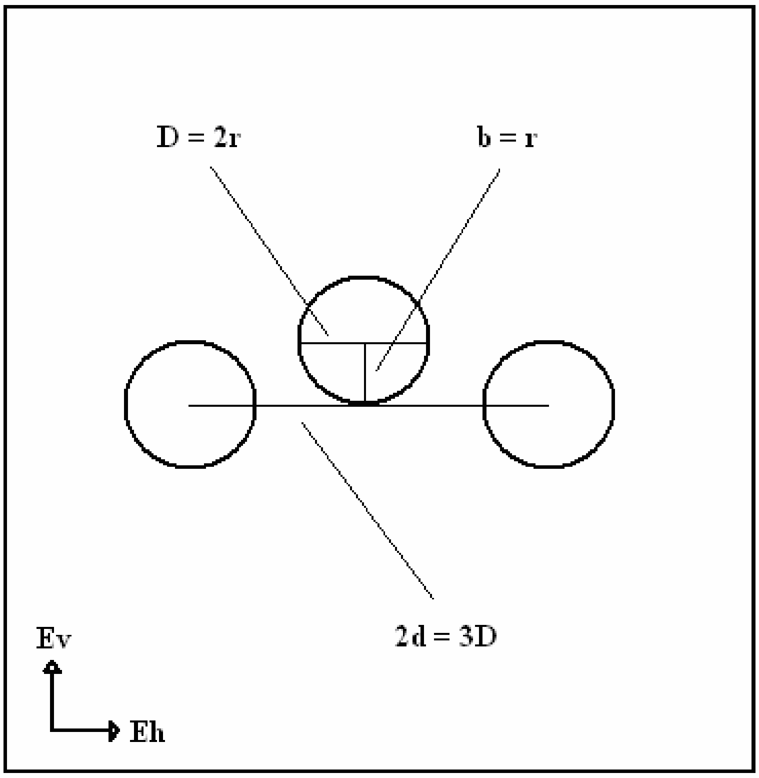

Fig. 5.5 A parametron cell. The transition from the upper island to the lower one is governed by a vertical electric field Ev. A horizontal electric field Eh determines the transition to the left or right lower islands. Parametron cells can be arranged so as to achieve logic gates, similarly to the QCA gates of Fig. (2.13) and (2.14).

5.6.2.1 Different Implementations

As suggested in Chap. 1, two implementations are considered: a semiconductor and a macromolecular implementation. In the semiconductor implementation, the minimum feature size is given by the diameter D of the island. In the macromolecular implementation, the minimum feature size is given by the distance between two atoms in the macromolecule implementing the device. In our case, Fig (5.5), this distance is 3/2·D.

From Eqs. (5.12) and (5.13), with the assumption that the number of cells in a gate is $N_c \sim 10$ [109], one can calculate error rates for the implementations. For the calculation, a minimum feature size = 1 nm is assumed. Furthermore, it is assumed that, at the nanometre level, screening effects are no longer present, so that $\varepsilon \approx 1$, while $\varepsilon \approx 2$ for the configuration of Ref. [109]. Room temperature ($T \approx 300$ K) operation and liquid nitrogen temperature ($T = 77$ K) operation, when needed, are considered.

5.6.2.2 Semiconductor Implementation

For a hypothetical semiconductor implementation, the minimum feature size is the island diameter D = 1 nm. Then, as shown in Sect. (5.6.2.1), one simply has to rescale the island diameter from 5 nm to 1 nm, taking the varying dielectric constant into account. In particular, one has:

$$\lambda \rightarrow \lambda / 5, \varepsilon \rightarrow \varepsilon / 2$$

Consequently, the energy difference ΔE of Eq. (5.14) is ≈ 0.3 eV.

Using Eqs. (5.12) and (5.13), an error rate per clock cycle $\sim 10^{-4}$ is then predicted at room temperature. As shown in Chap. 6, this is not acceptable. However, considering operation at liquid nitrogen temperature, one obtains an error rate per clock cycle $\sim 10^{-19}$. We will see in Chap. 6 that this is acceptable.

5.6.2.3 Macromolecular Implementation

For a hypothetical macromolecular implementation, the minimum feature size is the distance between two atoms in the molecule. As seen in Fig. (5.5), this is $3/2\cdot D$ (D is the island diameter). If the minimum feature size is 1 nm, the island diameter D is 2/3 nm. One has then to rescale the island diameter from 5 nm to 2/3 nm. Taking the varying dielectric constant into account, one has:

$\lambda \rightarrow 2\lambda / 15, \varepsilon \rightarrow \varepsilon / 2$

Consequently, the energy difference ΔE of Eq. (5.14) is ≈ 0.45 eV.

Using Eqs. (5.12) and (5.13), an error rate per clock cycle $\sim 10^{-7}$ is then predicted at room temperature. As shown in Chap. 6, this is acceptable.

5.6.2.4 Other Error Sources

If one approximates the parametron tunnelling junction as a planar one, therefore using Eq. (5.2) (with $n = 2$) for evaluating the constant α, the effect of frequency errors can be estimated. By using $R \approx 0.35$ MΩ, $C \approx 0.25$ aF, $f \approx 1$ GHz, one obtains a frequency error rate per clock cycle of $\sim 10^{-155}$. This is a negligible value.

5.7 Quantum Cellular Automata

A Quantum cellular Automata (QCA) cell [112] is a square array of four quantum dots, occupied by two electrons. The cell has two stable states, with the electrons sitting at the ends of each diagonal. Each cell is influenced by its neighbours through electrostatic repulsion. Logic gates can be built out of QCAs [112].

5.7.1 The Available Data on Error Rates

The main error source in QCA-based devices arises from thermal excitation [111], which may create kinks in a row of previously aligned cells and give a wrong output, as in Fig. (5.6). Here it is assumed that such rates can be estimated from thermodynamic considerations, following the remarks of Ref. [111].

The applicability of thermodynamic considerations is not a priori guaranteed, since the device might not have enough time to explore higher excited states during each clock cycle. However, we think that the use of thermodynamic arguments can be justified. In the so-called adiabatic switching regime [111,113], the clocking time t_c is chosen to be much longer than the time corresponding to the energy splitting between the ground state and the first excited state. One then has [114]:

$$t_c >> \frac{\hbar}{\Delta E} \qquad (5.15)$$

Condition (5.15) is verified even for the higher rank states, since their energy splitting is given by $\Delta E' > \Delta E$. The unit is then prevented from getting stuck on an excited metastable state [114]. In our opinion, condition (5.15) also gives the unit time to explore higher rank states through thermal excitation, therefore justifying the use of thermodynamics.

An order-of-magnitude estimation for fault rates in a QCA unit can then be obtained by making use of thermodynamic arguments and considering the case of a QCA infinite wire. This approach has been previously invoked for approximated considerations on QCA failure rates [111].

By applying Boltzmann's statistics, the transition probability to the nth excited state of an assembly of N_c QCA cells, $P_{f;qca}$, can then be calculated, for the case of an infinite wire [111]. In particular, one has the following Boltzmann-like result:

$$P_{f;qca} = g_n \cdot \frac{\exp(-\frac{\Delta E_n}{k \cdot T})}{\sum_n \exp(-\frac{\Delta E_n}{k \cdot T})} \tag{5.16}$$

In Eq. (5.16), g_n is the degeneracy of the nth excited level. In the case of a QCA wire, g_n is the number of ways in which n kinks can be chosen from N_c-1 positions.

In particular, one has:

$$g_n = \binom{N_c - 1}{n} \qquad (5.17)$$

while ΔE_n is the splitting of the nth excited level with respect to the ground state. In the case of an infinite wire, ΔE_n can then be calculated.

In particular, one has:

$$\Delta E_n = n \cdot \Delta E \qquad (5.18)$$

where ΔE is the splitting between the first excited level and the ground state. We have thus calculated the required transition probabilities.

Eqs. (5.15) to (5.18) show that the probabilities of thermal excitation of rank higher than one are exponentially damped. On the other hand, Eq. (5.18) is approximately valid even for a finite wire, see Ref. [111]. Therefore, in a first approximation, one can assume that errors arise from the thermal excitation of one kink, Fig. (5.6). The denominator of the right side of Eq. (5.16) (i.e. the partition function) can be likewise approximated to one. Furthermore, from Eq. (5.17), one has $g_n = N_c - 1 \approx N_c$, so that:

$$P_{f;qca} = N_c \cdot \exp\left(-\frac{\Delta E_n}{k \cdot T}\right) \qquad (5.19)$$

Eq. (5.19) is taken as an order-of-magnitude estimation of the faulting probability of a N_c-cell QCA unit (gate or wire).

5.7.2 Error Rates at the Nanometre Scale

We are now ready to estimate error rates for nanometre-scale QCA gates. To this purpose, one can observe that the energy splitting between the 0 and 1 states, ΔE, has been calculated to be ≈ 0.8 meV for an infinite QCA wire, with dots having a diameter of 10 nm, laid out along a square with 40 nm diagonals [112].

A scaling law is then needed. In order to derive such a law, let us examine the structure of the Hamiltonian for a QCA system [112], given in Eq. (2.11). The above-mentioned Hamiltonian contains:

- A ground state energy.
- A term describing the Coulomb repulsion of electrons sitting in the same dot.
- A term describing the Coulomb repulsion of electrons sitting in different dots.
- A tunnelling energy contribution, describing tunnelling processes between dots.

The interaction between different cells is described with a perturbative Coulomb term, while cell-to-cell tunnelling is neglected. The quantity ΔE is an energy difference. Therefore, the ground state term is not relevant.

ΔE measures the difference between two configurations in which electrons are sitting at the opposite extremes of one of the diagonals, see Fig. (5.6).

Then:

- Since in both configurations the electrons occupy different dots, the term describing electrons in the same dot is not relevant.
- For symmetry reasons the tunnelling energy contribution is the same for the two configurations, and it does not contribute to ΔE.

One is then only left with the Coulomb term describing the interaction of electrons sitting in different dots, plus the perturbative terms, that we neglect here. Then, as shown in the previous paragraph, the scaling law is determined by the Coulomb part of the Hamiltonian.

In particular, one has:

$$\Delta E \propto \varepsilon^{-1} \cdot \lambda^{-1} \qquad (5.20)$$

where ε is the relative dielectric constant of the device's substrate ($\varepsilon \approx 10$ [112]) and λ is the length scale of the devices. It is assumed here that scaling is performed by keeping the relative proportions of the QCA cell unchanged.

5.7.2.1 Different Implementations

A semiconductor implementation and a macromolecular implementation [114] are considered here. In the semiconductor implementation, the minimum feature size is given by the diameter D of the quantum dots. In the molecular implementation, the minimum feature size is fixed by the dimensions of the macromolecule and given by the length L of the square side of a cell.

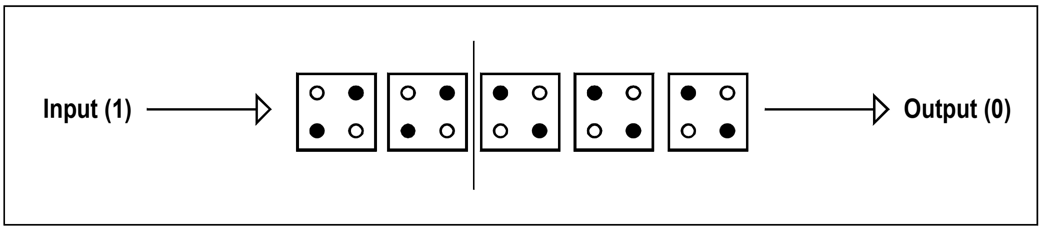

Fig. 5.6 A kink in a QCA wire. Due to thermal fluctuations, the device has absorbed a quantity of energy enough to create a kink in it. The input, which would normally be propagated unaltered through the wire, is then flipped. The wire then acts as an inverter and an error occurs.

From Eqs. (5.18) and (5.19), under the assumption that the number of cells is $N_c \sim 10$ [113], one can calculate error rates. A minimum feature size = 1 nm is assumed. It is also assumed that at ~1 nm screening effects are not present and $\varepsilon \approx 1$ [xxx], while $\varepsilon \approx 10$ for the configuration of Ref. [111]. As shown in Chap. 6, room temperature is not feasible for QCAs in the nanometre regime. Liquid nitrogen temperature operation (T = 77 K) is therefore considered.

5.7.2.2 Semiconductor Implementation

For a semiconductor implementation, D = 1 nm and one simply has to rescale the dot diameter from 10 nm to 1 nm.

Taking the varying dielectric constant into account, the scale transformation is:

$$\lambda \to \lambda / 10, \varepsilon \to \varepsilon / 10$$

Then the energy difference ΔE of Eq. (5.21) is ≈ 0.08 eV. An error rate per clock cycle ~ 1 (which is clearly unacceptable) is predicted from Eq. (5.19) at room temperature. Considering liquid nitrogen temperature operation, one has an error rate per clock cycle $\sim 10^{-4}$. As seen in Chap. 6, this is acceptable.

5.7.2.3 Macromolecular Implementation

For a macromolecular implementation, the 40 nm diagonal of Ref. [xxx] has to be rescaled. A 40 nm diagonal corresponds to a $40/\sqrt{2}$ nm side. We then have to rescale the cell size, see Fig. (5.6) from $40/\sqrt{2}$ nm to 1 nm. The scale transformation can then be worked out.

In particular, one has:

$$\lambda \to \lambda / 40\sqrt{2}, \quad \varepsilon \to \varepsilon / 10$$

Consequently, the energy difference ΔE of Eq. (5.19) is ≈ 0.23 eV. An error rate per clock cycle $\sim 10^{-3}$ is then predicted by Eq. (5.19) at room temperature. This is unacceptable, as shown in Chap. 6. Operation at liquid nitrogen temperature gives an error rate per clock cycle $\sim 10^{-14}$, which is acceptable, see Chap. 6.

5.8 Conclusions

In this chapter, error rates per clock cycle for various devices using single electrons as the bit (granular devices) were considered. In particular, gates based upon the single-electron switch, Koroktov's single-electron logic, the parametron cell and quantum cellular automata were examined. Models for error rates in such devices have been built or taken from the literature, when available. By establishing appropriate scaling laws, the parameters in such models were rescaled to the nanometre level, wherever they had been determined for other regimes.

The significance of the results obtained (summarized in Tab. 1) will be fully assessed in Chap. 6, where they will be applied to fault tolerance problems in nanochips. However, here we can point out that, according to our calculations, granular devices can be operated at room temperature although, as seen in Chap. 6, fault-tolerant techniques have to be applied to nanochips based on these devices. As for Boltzmann-type devices, where error rates are determined by thermal transition, liquid nitrogen temperature operation has to be considered if, as seen in Chap. 6, acceptable redundancy levels are to be achieved. In any case, Chap. 6 will show that redundancy levels (in time and/or space) of up to 2 orders of magnitude have to be provided.

Chapter 6 Nanochip Error Rates

6.1 Introduction

The aim of this thesis, which we tackle in this chapter, is to give estimations for error rates in nanochips, applying fault-tolerant techniques where needed. It was seen in Chap. 1 that nanochips are predicted to contain up to ~10^{12} devices per chip. Conservatively, we focus our attention to nanochips containing ~10^{11} devices [2], for which, as one can readily see, typical linear dimensions are ~1 cm.

We will see that fault-tolerant techniques are needed, if nanochips are to give acceptable performances. We will then introduce cascaded general modular redundancy (CGMR), a novel fault-tolerant technique. Using the results obtained in this respect, we will be able to give predictions on error rates in logic nanochips, based upon the previously calculated nanodevice error rates. Error correcting codes will also be introduced, to take care of error rates in memory nanochips. We will show that, if acceptable mean times between failures have to be obtained, redundancy levels up to ~100 may be required.

6.2 Two Chip Models

Somewhat artificially, we assume that memory and logic functions can be implemented in separate units, which we call "chips" for terminological simplicity. We will then give two separate models for logic chips and memory chips.

6.2.1 Logic Chips

For reasons that will be clear later, when we deal with CGMR, logic chips will be considered as an aggregate of N_t interconnected logic gates, partitioned into clusters. Each cluster consists of N_g logic gates. Each gate has a faulting probability per clock cycle $P_{f;g}$, with $P_{f;g} << 1$. By a suitable partitioning process, see Fig. (6.1), we can obtain a linear chain of functional units, each one having a variable number of inputs and outputs.

We further assume that when an error occurs in a unit, it certainly emerges at one or more of its outputs and that errors in the various gates and at various times are statistically independent. The fact that logic gates have a certain degree of intrinsic tolerance to a faulty input (think of an OR gate) can be taken into account by multiplying the gate failing probability per clock cycle by a factor that turns out to be ~1. Since we deal with order-of-magnitude estimations, such a correction will be ignored in the following.

The previously mentioned assumptions imply that we only need to worry about one input and one output for each functional unit. Our model for a chip, then, will be that of a linear chain of functional units, each one having one input and one output. Each functional unit is in turn a cluster of N_g logic gates, having a faulting probability per clock cycle $P_{f;g}$.

We assume the system to have a "perfect" clock. In particular, we assume that the system clock is generated with micrometre-scale technology. We further assume that each cluster generates an answer (or answers) every clock cycle. This implies perfect pipelining, which is not always feasible.

6.2.2 Memory Chips

Memory chips will be modeled as an array of independent 1-bit memory cells, obtained by considering microelectronic static RAMs and putting a nanogate wherever a microgate would be present. Cells are organized into N_b—bit words. Each gate has a faulting probability per clock cycle $P_{f;g}$, with $P_{f;g} << 1$. The faulting probability per clock cycle of a memory cell is ~10 $P_{f;g}$, as it happens for static RAMs [115].

6.3 Cascaded General Modular Redundancy

We have seen in Chap. 2 that fault-tolerant techniques with transient errors go under the name of general modular redundancy (GMR) [92]. In GMR, voting is performed among different copies of a unit and spare units are switched in when a unit is deemed to be faulty. Specific embodiments of the GMR idea are triple modular redundancy (TMR) [93], see Fig. (3.1), and its generalization, N-modular redundancy (NMR) [91]. In NMR, the place of each potentially faulty unit is taken by a block of N identical units and majority voting is performed among them. N has to be odd, of course.

We will see in the chapter that the application of TMR or NMR alone is not sufficient to give nanochips a long enough mean time between failures. However, we devised a new fault-tolerant technique that generalizes GMR, through which the goal can be achieved. We named such technique as cascaded general modular redundancy (CGMR). Special cases of CGMR are cascaded triple modular redundancy (CTMR) and cascaded N-modular redundancy (CNMR). According to CGMR, the potentially faulty units are first clustered in a suitable way and modular redundancy is applied to the clusters. The clusters are then suitably clustered, as well, and modular redundancy is applied to each cluster. The process is iterated for a number of steps, as in Fig. (6.2). In this section, we present detailed calculations about the redundancy levels required by the application of CGMR.

6.3.1 The CGMR Formalism

For the CGMR formalism, we use the model of a logic chip given in Section 6.2.1 and Fig. (6.1), considering a logic chip as an aggregate of N_t interconnected logic gates, partitioned into clusters. Each cluster consists of N_g logic gates. Each gate has a faulting probability per clock cycle $P_{f;g}$, with $P_{f;g} << 1$. We also suppose that the voting circuitry has a failing probability per clock cycle $P_{f;v}$ where $P_{f;v} << 1$.

We now need an expression for the failure probability of N replicated units on which majority voting is performed. The failing probability per clock cycle of a cluster, $P_{f/cgmr;c}$, for negligible gate failing probabilities is given by:

$$P_{f/cgmr;c} = \alpha \cdot (N_g \cdot P_{f;g})^{\beta} + P_{f;v} \tag{6.1}$$

where α is a combinatorial factor and β is the minimum number of units whose output has to agree in their wrong answer, for the error not to be masked. For example, in TMR an error goes undetected if at least 2 units agree in their wrong answer, hence β=2. On the other hand, α=3 since there are 3 ways to choose 2 objects (the faulty units) out of 3. In the following section, we will give suitable values to α and β.

If every cluster contains N_g logic gates, the chip will contain N_t/N_g clusters, each one having a failing probability per clock cycle $P_{f/cgmr;c}$. The chip's failing probability per clock cycle, $P_{f/cgmr}(0)$, is then:

$$P_{f/cgmr}(0) = \frac{N_t}{N_g} \cdot P_{f/cgmr;c} \tag{6.2}$$

where N_t is the total number of gates in the chip, N_g is the number of gates in a cluster, $P_{f/cgmr;c}$ is the cluster's failing probability per clock cycle of Eq. (6.1).

A consequence of Eq. (6.2) is that the chip's failing probability per clock cycle is minimized if cluster size is kept as small as possible. This follows from the fact that $P_{f/cgmr}(0)$ is proportional to $N_g^{\beta-1}$, see Eqs. (6.1) and (6.2), and, to the best of our knowledge, $\beta > 1$. These equations also show that below a certain cluster size $P_{f;v}$ dominates. In particular, this happens when:

$$\alpha \cdot (N_g \cdot P_{f;g})^{\beta} \leq \frac{N_v}{\eta} \cdot P_{f;g} \tag{6.3}$$

where η~10 (one order of magnitude) and we have defined $P_{f;v} = N_v\, P_{f;g}$, so that N_v is an effective number of gates in the voter (the real number of gates, if the gates in the clusters and in the voting circuitry are of the same kind).

Eq. (6.3) has the consequence that, in order to minimize error rates, the clusters must have a maximum size given by:

$$N_{\max} = \sqrt[\beta]{\frac{N_v}{\alpha \cdot \eta \cdot (P_{f;g})^{\beta-1}}} \qquad (6.4)$$

It is in the designer's interest to keep cluster size as small as possible, so that the chip's failing probability per clock cycle is kept at the lowest level.

Taking cluster size as $N_{\max}$, the error probability per calculation of a cluster after i CGMR stages, $P_{f/cgmr}$ (i) is:

$$P_{f/cgmr}(i) = \frac{N_t}{(N_{\max})^i} \cdot N_v \cdot P_{f;g} \qquad (6.5)$$

with cluster size given by $N_{\max}$, as in Eq. (6.4).

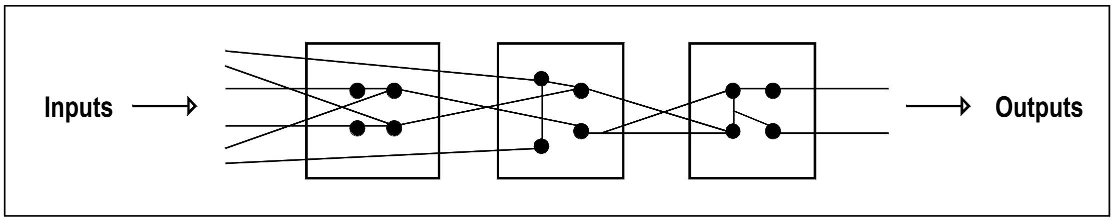

Fig. 6.1 A model for a logic chip. A logic chip is represented as a set of interconnected gates which is suitably partitioned, so that it can be considered as a linear array of functional units. The functional units have a variable number of inputs and outputs.

The function defined by Eq. (6.5) is monotonically decreasing with i. One therefore needs as many CGMR stages as possible. However, the iteration process cannot go on indefinitely. The maximum number of CGMR stages is defined by the condition:

$$\frac{N_t}{(N_{\max})^i} = 1 \qquad (6.6)$$

Using Eqs. (6.4) and (6.6), we are now able to define the maximum number of stages:

$$i_{\max} = [-\beta \cdot \frac{Log(N_t)}{Log(\alpha \cdot \eta \cdot (N_v)^{-1} \cdot (P_{f;g})^{\beta-1})}] + 1 \qquad (6.7)$$

where [.] is the integer part function. The chip's overall failing probability per clock cycle, $P_{f/cgmr}$ is then:

$$P_{f/cgmr} = N_v \cdot P_{f;g} \qquad (6.8)$$

6.3.2 Improved CNMR

The failing probability given by CNMR can be improved, if necessary. It is in fact possible to apply NMR to the chip's outputs, using a voting circuitry much more reliable than the chip's gates (micron-scale technology, in practice).

The system's failing probability per clock cycle, $P_{f/icgmr}$, can then be computed by using the formulae seen in Chap. 3. In particular, the chip's failing probability for improved CGMR is given by [99,100]:

$$P_{f/icgmr} = \begin{pmatrix} N \\ \frac{N+1}{2} \end{pmatrix} \cdot (P_{f/cgmr})^{\frac{N+1}{2}} \qquad (6.9)$$

An expression like Eq. (6.9), with $N_t \cdot P_{f;g}$ in place of $P_{f/cgmr}$, describes the chip's failing probability per clock cycle for NMR with micrometre-scale circuitry, when no nanometre-scale redundancy is used. This might seem to provide fault protection in a nanochip. Unfortunately, the condition $N_t\, P_{f;g} << 1$ has to be satisfied, thus limiting the applicability of the technique to the case $P_{f;g} << (N_t)^{-1}$.

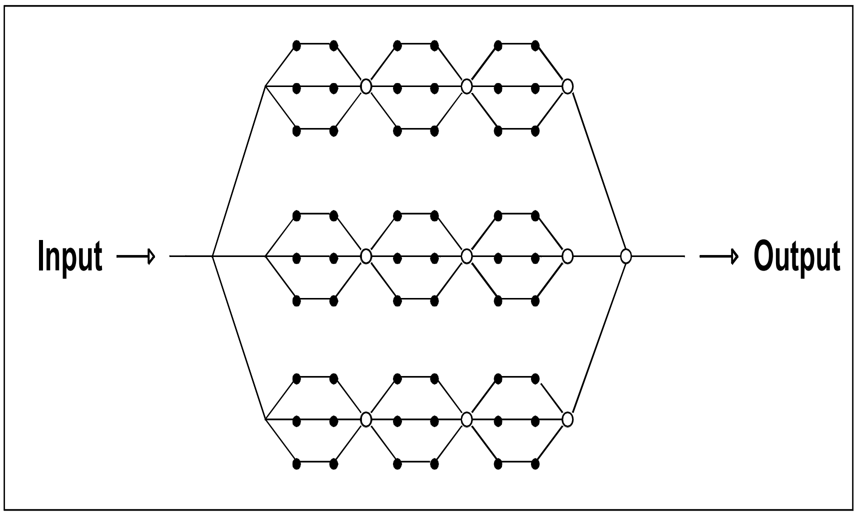

Fig. 6.2 A 2-stage CTMR arrangement. A linear array of six gates is partitioned into clusters of two gates. In the 1st stage, each cluster is tripled and majority voting is performed on each triplet. In the 2nd stage, the linear chain thus obtained is in turn tripled and majority voting is performed on the triplet.

6.3.3 CGMR in the Space Domain

As shown in Chap. 3, fault tolerance can be achieved by performing majority voting among N copies of a potentially faulty device. One then has N-modular redundancy (NMR). The probability for an NMR unit to fail is given by [99,100]:

$$P_{f\,/\,nmr} = \begin{pmatrix} N \\ \frac{N+1}{2} \end{pmatrix} \cdot (P_{f;g})^{\frac{N+1}{2}} \tag{6.10}$$

Eq. (6.10) describes a CGMR unit, see Eq. (6.1), with:

$$\alpha = \begin{pmatrix} N \\ \frac{N+1}{2} \end{pmatrix}$$
$$\beta = \frac{N+1}{2} \tag{6.11}$$

In our case, time redundancy is ≈ 1 (since $N_g >> 1$, the number of clock cycles taken by an error to reach the chip's output is on average much greater than the number of cycles taken for voting) and space redundancy is:

$$r_s = M^{i_{\max}} \cdot N \tag{6.12}$$

where i_{max} is defined by Eq. (6.7).

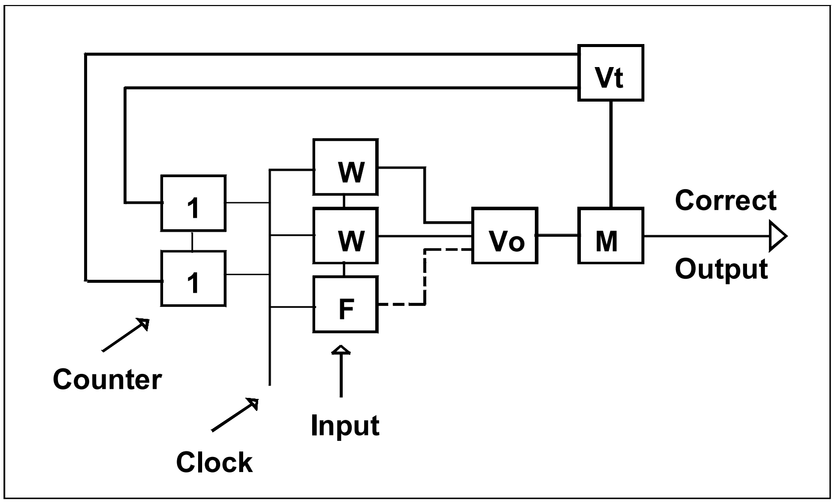

Fig. 6.3 A voting unit for time domain TMR. The input stream is accumulated in the shift register. The three stages of the shift register send their output to a majority voting unit. The answer of the voting unit is sent to a memory cell. A veto unit (an AND gate) prevents the memory cell from storing the input data coming from the voter unless the counter (as in the picture) gives a "11" output. This happens every 4 clock cycles. W = data coming from a working device, F = data coming from a failing device, Vo = voting circuit, Vt = veto unit, M = memory cell. Wrong outputs are marked with a dashed line.

6.3.4 CGMR in the Time Domain

In Chap. 3, it has been shown that GMR (and NMR) can also be implemented in the time domain. In particular, for intrinsic errors in the time domain GMR can either take the form of time-domain NMR [101] or backward error recovery (BER) [89]. We propose to extend the ideas of CGMR to the time domain, as well.

6.3.4.1 Time-Domain CNMR

By reading Fig. (6.2) as a time diagram, we have an idea of how to implement CNMR in the time domain [101]. In particular, in Fig. (6.3) we show a possible solution for a time-domain TMR voting unit (such ideas, however, can be easily extended to NMR). If the clock frequency of a unit at level i is de-multiplied by a factor $(N+1)^i$ with respect to the master clock frequency, and the voting units are connected like in Fig. (6.3), time-domain CNMR can be performed (under the hypothesis of perfectly reliable clock, in practice made up of microelectronic components).

As for redundancy, the same results apply for both space-domain and time-domain CNMR, Eq. (6.12) except, of course, that the roles of space and time are exchanged.

6.3.4.2 Backward Error Recovery

In Chap. 3, we presented the principles of backward error recovery (BER). In BER [89], the outputs of N copies of potentially faulty units are compared and, should a disagreement be found, the system steps back and the computation is repeated. The error rate per calculation for N replicated N_g-gate units is [89]:

$$P_{f/ber} = (N_g \cdot P_{f;g})^N \tag{6.13}$$

One can cascade the BER units and apply CGMR, Eq. (6.1), with:

$$\begin{aligned} \alpha &= 1 \\ \beta &= N \end{aligned} \tag{6.14}$$

and gate counting gives $N_v \sim 10$ for duplication [115].

If BER has to make any sense, the intrinsic probability per clock cycle for a fault to occur anywhere in the chip must be << 1, because otherwise there would be no point in repeating processor operation. In other words, we must have $P_{f,g} << (N_t)^{-1}$. Time redundancy is then ~1. Space redundancy, instead, is given by Eq. (6.12). It is interesting to notice that BER, although usually defined as a time-redundant technique, in our cascaded version implies a non-negligible amount of space redundancy.

Alternatively, the comparison process can be performed by accumulating results in time. An arrangement similar to Fig. (6.3) has to be employed. In this case, the roles of space and time redundancies are exchanged. As for space redundancy, gate counting on the version of the time-voting device of Fig. (6.3) suitable for duplication gives $N_v \sim 100$ [115].

6.4 Error Correcting Codes

We saw in Chap. 3 that, in information-redundant strategies, the information stored in a chip is made redundant through the use of error correcting codes [10]. In an error correcting code, suitable check bits are added to the information bits one wants to protect, so that errors can be located and/or corrected.

What has just been said suggests that error correcting codes can be applied to memory chips, for which the above-mentioned clustering process would not be feasible. We adopt the model suggested in Section 6.2.2 for a memory chip. Namely, we consider a memory chip as an array of independent memory cells, organized into memory words. We assume that every memory cell, with a given fault probability per clock cycle, is periodically refreshed. We finally suppose that a certain error correcting code has the capability to correct N_e errors. The data word is written to the memory and stays there for N_s clock cycles. Once a bit error has been generated, the memory cell's feedback loop [115] makes it permanent, even though the error cause is of a transient nature.

If the memory cell's fault rate per clock cycle, $P_{f;m}$, satisfies the relationship $P_{f;m} << (N_s)^{-1}$, we can neglect the process through which a bit error can be flipped back to its correct value. The probability that a N_b-bit memory word (where N_b includes the contribution of the check bits) is corrupted by a number of errors $\geq N_e+1$, after being stored for N_s clock cycles, is then given by (see Chap. 3):

$$P_{f/ecc} = \begin{pmatrix} N_b \\ N_e + 1 \end{pmatrix} \cdot (N_s \cdot P_{f;m})^{N_e+1} \qquad (6.15)$$

where, due to gate counting, it can be shown that $P_{f;m} \sim 10\ P_{f;g}$ [115].

Eq. (6.15) requires that $N_s \cdot P_{f;m} << 1$. This limits the applicability of the proposed technique to error probabilities such that $P_{f;m} << (N_s)^{-1}$. Whenever possible, it would be advisable to choose N_s as the number of clock cycles corresponding to the mean time between failures we want to guarantee. Otherwise a storage time has to be chosen and, of course, periodic correcting operations have to be performed.

In an array of $\sim 10^{11}$ logic gates (and $\sim 10^{10}$ memory cells [115]), one has $N_d \sim 10^{10}\ (N_b)^{-1}$ data words. Then the storage time (in clock cycles) N_s cannot be $\leq N_d$, since this is the minimum number of clock cycles between two correcting and re-encoding operations on the same data word, if these operations are to be performed sequentially. This is a further constraint on the applicability of Eq. (6.15). However, if a shorter storage time is required for probability reasons, one can split the nanochip (and the lookup table) into a number N_u of sub-units that are refreshed in parallel. The resulting circuitry should not have a significant impact on space redundancy, at least at an order-of-magnitude level.

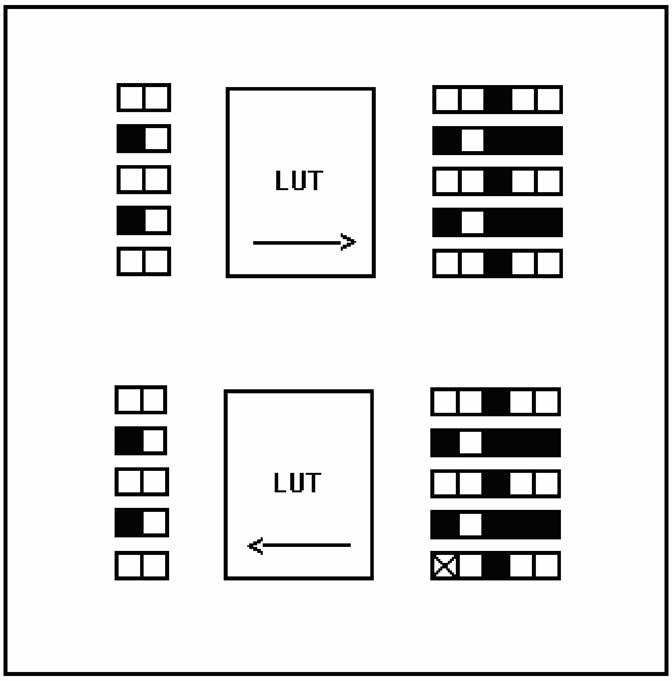

Fig. 6.4 Fault tolerance through error correcting codes. In this memory nanochip, fault tolerance is achieved by an error correcting code, implemented in a lookup table (LUT). The picture above shows how the memory input is codified by the LUT. In the picture below, an error occurs and a bit is flipped. The LUT is able to correct the error.

6.4.1 The Lookup Table Approach

The space redundancy level depends on the chosen code. On the other hand, time redundancy can be a potentially serious problem. However, if encoding and decoding are performed through a lookup table operation [10] the problem can be overcome.

More specifically, when a dataword has to be written to memory, it is first fed to a pre-computed lookup table, which in turn produces an encoded input. The encoded input is then fed to the memory and stored. When a dataword has to be read from memory, it is extracted from it and fed to a pre-computed lookup table, which provides decoding and error correction. The lookup tables and correcting circuitry have to be much more reliable (e.g. built with micrometre-scale technology) than the memory they protect.

In the lookup table approach, one can easily estimate the time redundancy due to the encoding/decoding process. Let us suppose, in fact, that for each memory access operation (reading or writing) N_C dataword correction operations are needed for memory refreshing. Time redundancy is then:

$$r_t = 1 + N_c \tag{6.16}$$

where the reading and writing operations take the same amount of clock cycles.

6.4.2 Error Correcting Code Implementation

There are many different kinds of error correcting codes. We suggest the use of two quite different codes. Hamming codes [11] are characterized by having a low redundancy (1.5 for an 8-bit information word) and a low error-correcting capability (1 error). Reed-Muller codes [10], on the other hand, are characterized by quite a high redundancy (for example, 16 for an 8-bit information word) and a high error-correcting capability (for example, 31 errors for an 8-bit information word).

A straightforward argument [10], based upon metric space theory and reported in detail in Chap. 3, shows that if d_{cw} is the minimum Hamming distance (or any other suitable distance, in the technical sense of metric space theory), between two codewords in a given code, then the number of errors the code can correct, N_e, is given by:

$$N_e = \left[\frac{d_{cw} - 1}{2} \right] \tag{6.17}$$

where $[\cdot]$ is the integer part function.

Error rates in nanomemories are expressed by Eq. (6.17). The outcome of such an equation, however, critically depends on the number of bits (information and parity check bits) in the codeword. We suppose to quote our results for a 16-bit dataword. However, redundancy reasons can prevent one from applying error correction to the whole dataword. Datawords may have to be split into sub-words, to which error correction applies.

Our encoding/decoding approach implies a lookup table dimension of 2^{N_b}, where N_b is the total number of bits. If we assume a ratio of $\sim 10^4$ between the typical areas of our devices and the conventional micrometre-scale devices, we find that the lookup table cannot contain more than $\sim 10^6$ memory cells, if its area has to be at worst of the same order as the area occupied by the nanoscale devices.

The above-mentioned considerations limit dataword size, including redundancy, to less than 20 bits. Any long word can be split into N_{bl} smaller blocks, separately encoded and decoded.

The failure probability for a storage time of N_s clock cycles is:

$$P_{f/ecc} = N_{bl} \cdot \begin{pmatrix} N_b \\ N_e + 1 \end{pmatrix} \cdot (N_s \cdot P_{f;m})^{N_e + 1} \tag{6.18}$$

For Hamming codes, we can afford to use the whole 16-bit dataword for error correction. The redundancy level is 1.5 and one only error can be corrected [10,11].

For Reed-Muller codes, we found that a solution is to use a 3rd order Reed-Muller code. This implies $N_{cw} = 32 = 2^5$ [10], see Chap. 3, and we have to split the dataword into four 5-bit blocks (the last block containing four fictitious bits). Since $l_{cw,}$ the codeword length, is 16 for each block, the redundancy is 16/5 for the single blocks. The code can correct a maximum of 3 errors on a 5-bit block.

6.5 Nanodevice Error Rates

We are now able to present the different fault-tolerant solutions we devised. As explained in Chap. 1, we required a mean time between failures, t_f, of ~1 y (10^8 sec) at a frequency f = 1 GHz for 10^{11} gates, with a redundancy level $\leq$ 100 in space and/or $\leq$ 10 in time. Using the device error probabilities summarized above, we calculate the mean time between failures, t_f.

In particular, one has [2]:

$$t_f = \frac{1}{f \cdot P_{f/ft}} \tag{6.19}$$

where f is the clock frequency (1 GHz, in our case) and $P_{f/ft}$ is the chip's failure probability per clock cycle when fault tolerance is provided.

6.5.1 Logic Chips

In Tab. 2, 3, 4 and 5, we list the solutions found for logic chips based on SEDs, Koroktov's devices, QCAs and parametrons, with a redundancy level $\leq$ 100 in the space domain and/or $\leq$ 10 in the time domain. Liquid nitrogen is only considered when room temperature operation would not give results within these limits. In particular:

- In column 1, we describe the devices and their operating temperature. If, for instance, we mention "SED, 300 K, 5 islands", we are dealing with an SED gate in which the minimum path of an electron is 5 islands, operated at 300 K.
- In column 2, we describe the fault tolerance approach proposed. CGMR = cascaded general modular redundancy and BER = backward error recovery.
- In column 3, we describe the redundancy domain involved (spatial or temporal).
- In column 4, we give the number of replicas of a unit in the proposed approach (e.g. 3 for triple modular redundancy) at the nanometre scale.
- In column 5, we give the number of CGMR cascading levels at the nanometre scale.

- In columns 6 and 7, we give the same quantities as columns 4 and 5, except that they are referred to the micrometre-scale level (when not implemented, a 0 is used).
- In column 8, we give the space redundancy level required by the proposed solution.
- In column 9, we give the time redundancy level required by the same solution.

From the analysis of Tab. 2 to 5, a number of conclusions can be drawn. First of all, we observe that SED-based and Koroktov gates give similar results in terms of the required fault tolerance level. The same holds for QCA and parametron-based gates. As for SED-like and QCA-like gates, our conclusions can be summarized as follows:

- SED-like gates can be operated at room temperature.
- SED-like gates require space redundancy in the region 10-100 and no time redundancy.
- QCA-like gates have to be operated at liquid nitrogen temperature.
- QCA-like gates require space or time redundancy in the region 1-10.
- In QCA-like gates, time redundancy can be traded for space redundancy.

SED-like gates, then, require more redundancy but can be operated at room temperature. QCA-like gates require less redundancy and are more flexible, since space can be traded for time redundancy, but have to be operated at liquid nitrogen temperature.

6.5.2 Memory Chips

We list here the alternative solutions found for memory chips. As for Hamming codes, we applied error correction to the whole dataword. As for Reed-Muller codes, we had to split the dataword into four 5-bit blocks (with four fictitious bits in the last block).

We have seen that the faulting probability per clock cycle of a memory cell, $P_{f;m}$, must be such that $P_{f;m} << (N_s)^{-1}$, where N_s is the storage time in clock cycles. On the other hand, N_s must exceed the number ($\sim 10^8$, in our case) of datawords in the chip, given by $\sim 10^{10}$ memory cells, grouped into datawords with $\sim 10^2$ memory cells. This implies $P_{f;m} << 10^{-8}$ or, in terms of gate fault probabilities per clock cycle, $P_{f;g} \leq \sim 10^{-10}$.

The above-mentioned figure is quite compatible with Tab. 1 but, conservatively, we chose a storage time of 10^7 clock cycles. This involves splitting the memory chip (and the lookup table) into ten independent blocks, refreshed in parallel. One has then to refresh (on average) a dataword every memory access, so that $r_t = 2$ in Eq. (6.16). In the case of Reed-Muller codes, the space redundancies have been multiplied by a factor 2, as an estimate for the space taken by the lookup table. In the case of Hamming codes, due to the low redundancy level, the space taken by the lookup table is negligible.

In Tab. 6, we list the various solutions found for memory chips based on SEDs, Koroktov's devices, QCAs and parametrons, respectively, with a redundancy level $\leq$ 100 in the space domain and/or $\leq$ 10 in the time domain. Liquid nitrogen is only considered when room temperature operation would not give results within these redundancy limits. The structure of the table is:

- In column 1, we describe the devices and their operating temperature. If, for instance, we mention "SED, 300 K, 5 islands", we are dealing with an SED gate in which the minimum path of an electron is 5 islands, operated at 300 K.
- In column 2, we describe the error-correcting code used (Hamming or Reed-Muller).
- In column 3, we give the space redundancy level required by the proposed solution.
- In column 4, we give the time redundancy level required by the proposed solution.

From the analysis of Table 6, a number of conclusions can be drawn:

- SED-like gates can be operated at room temperature.
- SED-like gates require space redundancy ~10 and time redundancy ~1.
- SED-like gates require codes with a high error-correcting capability.
- QCA-like gates have to be operated at liquid nitrogen temperature.
- QCA-like gates can be operated with space and time redundancies ~1.
- QCA-like gates do not require codes with a high error-correcting capability.

SED-like gates, then, require more space and time redundancy but can be operated at room temperature and require codes with high error-correcting capability (Reed-Muller codes are suggested). QCA-like gates, in contrast, require less space and time redundancy and are more flexible, since error-correcting codes with low error-correcting capability can be employed. However, they have to be operated at liquid nitrogen temperature.

6.6 Conclusions

In this chapter, we addressed the problem of estimating nanochip fault rates. We considered SED-based, Koroktov, QCA and parametron-based logic gates and proposed a new fault-tolerant technique, named cascaded general modular redundancy (CGMR). We were able to adapt other standard techniques to our requirements. By using the calculated figures for the intrinsic fault rates, we were able to propose a number of fault-tolerant solutions for both logic and memory chips. These fault-tolerant solutions were evaluated on the basis of the redundancy level and operating temperature required.

We found that a mean time between failures of ~1 y at ~1 GHz can be guaranteed (and in some cases vastly exceeded) for logic chips, with a redundancy level of a few tens at worse, either in space or in time. For memory chips, a mean time between failures of ~1 y at ~1 GHz can be guaranteed with a redundancy level of ~10 at worse, both in space and in time. However, we had already seen in Chap. 5 that a careful tuning of the devices' operating conditions and design solutions is required.

We comment on these results in Chap. 7. However, in our opinion, the results obtained clearly show that the forthcoming nanochip will have to implement levels of fault tolerance of one and even two orders of magnitude, in space and/or time. Some parts of the chip, especially the user interface [2], will have to be made up in microelectronics. In our opinion, we were among the first researchers to underline these facts.

Chapter 7 Conclusions

In this chapter, we give a summary and critical evaluation of our thesis. We then give suggestions for future research. We finally give some general remarks on our thesis, its achievements and limitations. In order not to make reading too heavy, we omit any references to the literature for the subjects that have been treated elsewhere in the thesis. Such references can be found in the relevant chapters. For the sake of clarity, we also duplicate here some of the information that is given, in more details, in the above-mentioned chapters.

7.1 Achievements

At this stage of development of nanoelectronics, it is not easy to anticipate how exactly a given device will be built. And, even if such data were known, there would be the architectural issues to consider. These, in turn, would depend on the way nanochips will be built, which is a problem far from being solved. With our order-of-magnitude approach, we were able to give sensible predictions on nanodevice (and nanochip) error rates, despite the above-mentioned problems, which may hopefully guide the direction of future efforts.

7.1.1 Single Electron Pump

The basic single electron device we considered was the electron pump. In an electron pump, an electron is made to jump through a series of electrodes by suitable potentials that trigger the tunnelling effect. The electron, then, is pumped down the array of junctions.

There are some error rate measurements for the electron pump, since this device was originally meant as a metrological charge standard. The main error cause is the fact that the electron can go the wrong way, after absorbing energy from the environment. In this work, the published experimental results were rescaled to the sizes and frequencies of interest to us and order-of-magnitude predictions were achieved.

An electron switch, exploiting the Coulomb interaction between two electrons driven by electron pumps, was also proposed in the literature. The problem of error rates for the electron switch was tackled by noticing that, once the electron is switched, it travels along an electron pump. The switching process may not work correctly but, once the electron is correctly switched, the error rates are that of an electron pump.

There are simulations on the probability of wrong switching. Such simulations are not encouraging since, in the proposed electron switches, switching errors would dominate and impose an unacceptable error rate. However we suggested that, if a manner can be found of redesigning electron switches so that switching errors become negligible and electron pump errors dominate, it might be possible to dimension the switch so as to make error rates acceptable, by making the electron pump long enough.

Starting from the electron switch, logic gates have been proposed in the literature. To the best of our knowledge, there are no experimental measurements or theoretical predictions on error rates in such devices. However, these devices are essentially electron switches. The error rate of the gate could be approximated by the error rate of an electron pump with a length equal to the shortest path an electron can travel in the device.

7.1.2 Koroktov's Devices

Koroktov's logic gates are essentially an array of interacting electron pumps. We found that, at an order-of-magnitude level, the error rate can be approximated as the error rate of the shortest electron pump in the gate, times the number of pumps.

7.1.3 Quantum Cellular Automata

Quantum cellular automata (QCAs), are cellular automata whose basic cell is composed of four quantum wells in a square, with two electrons in it. The electrons tend to sit in the opposite corners on the square. The resulting states are then taken to represent 0 and 1 and the cells interact by Coulomb interactions.

QCA-based gates have been proposed. For these devices, we could only find qualitative remarks on error rates. In particular, it seems that the main error cause is connected to the discrete structure exhibited by QCAs as quantum systems. A QCA gate can absorb thermal energy from the environment and make a transition to an upper energy state, in which its response is flipped.

An objection that could be raised against our approach to QCA error rates is that it assumes thermodynamic equilibrium at any stage of device operation. In fact, our error rate formulae are essentially the Boltzmann formula suitably rewritten. However, QCA operation has been proposed in the “adiabatic clocking regime”. Adiabatic clocking means that the clocking frequency is such that, at any stage, the system has the time to perform all the required transitions (included the unwanted ones).

Our approximate error formulae (that, strictly speaking, refer to an infinite QCA wire, an approximation which is used in the literature to estimate error rates for more complex QCA-based devices) predict that QCAs, at least in the operating regimes envisaged up to now, should be operated at liquid nitrogen temperature.

7.1.4 Parametrons

The parametron is an array of interacting cells. Each cell is made up of three electrodes, set as a triangle and containing an electron. By suitable driving potentials, the electron, sitting in the topmost electrode, can be made to tunnel to one of the two lower electrodes, taken to represent 0 and 1.

This operating principle is analogous to that of QCAs and error rates can be calculated in a similar way. The proponents of the parametron endorse this view. In particular, the most important error source is linked to the fact that the system, by absorbing energy from the environment, can make a transition to a wrong state.

7.1.5 Logic Nanochips

The fault tolerant technique of election for logic nanochips seems to be triple modular redundancy (TMR), where any potentially faulty unit is tripled and errors are masked by majority voting. However, TMR can only work provided the voting circuitry is much more reliable than the potentially faulty units. The voting circuit has then to be built with microelectronic technology. It is not possible to provide every nanogate with microelectronic voting circuitry, otherwise the system would be impossibly bulky (and, anyway, would be a microelectronic and not a nanoelectronic system).

We have shown that the solution to the above-mentioned dilemma is offered by a fault tolerance technique we devised, named cascaded triple modular redundancy (CTMR). In CTMR, TMR is applied to a cluster of gates. The clusters are then clustered and TMR applied to these cluster of clusters. The process goes on in a hierarchical way, till one reaches the level of the chip as a whole. If necessary, then, TMR can be applied to the whole chip, using microelectronic voting circuitry.

One might also think of using N-modular redundancy (NMR), a technique in which the voting process is performed among N replicas, in a cascaded fashion. One might also think of cascading backward error recovery (BER), in which the potentially faulty units is doubled and the outputs compared. In this way, an error can be detected rather then corrected. However, if the error is detected and the systems proceeds in discrete steps (as it happens for clocked systems), the computation that has just failed can be repeated. We devised cascaded versions of both techniques.

For permanent errors (which were not our concern), there are also a number of other techniques, based essentially on performing a voting operation on suitable copies of the potentially faulty units and keeping some spare units, so that a unit that is deemed to be permanently damaged can be switched off and replaced. All these techniques go under the name of general modular redundancy (GMR).

All the fault tolerant techniques we mentioned can also be implemented in the time domain. Instead of providing N copies of any potentially faulty units, the operation of each faulty unit can be repeated N times. The same formalism that holds in the space domain keeps holding in the time domain. We devised a voting circuitry that is able to work in the time domain.

The choice between space or time domain CGMR is a matter of convenience. As for many of the alternatives we presented in this thesis, the choice among them is not possible at this stage of development of nanoelectronics. When, if ever, nanochips reach the state of mass production, the different design alternatives can be considered and evaluated in much more detail than it is possible at present.

7.1.6 Memory Nanochips

Turning to memory nanochips, the clustering approach involved in CTMR cannot be applied. Every memory bit, in fact, has to be protected singularly. The solution in this case is given by error correcting codes. In error correcting codes, suitable check bits are added to the memory line to protect. Through suitable mathematical operations, this allows to detect and correct the faulty bits, if the number of errors is less than the error correcting capability of the code used.

There are a number of error correcting codes in the literature. We tried to reduce this arbitrariness by choosing two codes having opposite properties. The Hamming code has a low error correcting capability and a low redundancy. The Reed-Muller code has a high error correcting capability and a high redundancy.

More specialised choices would be possible but we reckon that, until nanochips are mass-produced, and the related design problems well understood, there will not be much of a point in fine-tuning the code choice.

Implementing encoding and decoding operations can be a serious problem. Encoding and decoding algorithms could be implemented in a hardware fashion. However, provided one is able to design the related circuitry, there is still the problem that such algorithms are quite time-consuming, since they have to be applied at each reading, writing or memory refreshing operation. Time redundancy might then become quite a serious limitation on nanochip operation.

The same algorithms could be implemented in a software way. At this stage of development of nanoelectronics, it does not make much sense to tackle this problem at the software level, since any high level software consideration would depend on how the future nanochips will be programmed, still a premature issue.

The solution we devised was to suggest an approach in which a lookup table is used for encoding and decoding. The lookup table has to be built with microtechnology, since it has to be much more reliable than the memory words it protects. Actually, for high redundancy codes, such as Reed-Muller's, the size of the lookup table might become impossibly large, if it has to be built with microtechnology.

We then had to consider the possibility of splitting every dataword into a number of subwords, so that all the subwords are encoded and decoded in parallel and a lookup table is assigned to each one of them. Once again, we did not delve into detailed design considerations, since such an exercise would be premature at this stage of development of nanoelectronics.

7.2 Suggestions for Future Research

What has been said when critically evaluating the work done in this thesis implicitly contains a number of suggestions for future research on nanochip fault tolerance. In this section, we summarise them by treating separately the cases of nanodevice error rates, chip error rates and technological issues.

7.2.1 Nanodevice Error Rates

We have already noticed that our error rate estimations were only meant to be valid as order-of-magnitude indications. This choice was imposed by the fact that our research involved a significant number of issues and it would not be possible to consider all of them in detail. Until people have a precise idea of how to build nanochips, detailed error rate predictions will not be very useful for predicting redundancy levels. However, sooner or later it will be fundamental to have a detailed error theory for nanodevice and, anyway, the problem is of great interest in itself.

7.2.2 Nanochip Error Rates

As for nanochip error rates, there are two directions that future research might take. First, one might like to find more sophisticated fault tolerant techniques for logic and/or memory chips. Second, one might want to refine the theory of cascaded general modular redundancy that was developed as a part of this thesis.

Concerning the search for more refined fault tolerant techniques, we feel that there may not be much room for improvement with respect to the techniques presented in this thesis and NAND multiplexing. This is particularly true for logic chips. It is also true that, though the basic fault tolerant techniques are described in easily accessible mainstream scientific journals, many results are buried in obscure proceeding papers. When (if ever) detailed implementation issues arise, we cannot rule out the possibility of improvements on logic nanochip fault tolerance implementations, based upon either these not easily accessible results and/or new developments.

As for memory nanochip fault tolerance, we have shown in the thesis that the technique of election is error correcting codes. We chose two different, and opposite, codes: Hamming and Reed-Muller. Once again, when (if ever) detailed implementation issues arise, one might find it convenient to consider other error correcting codes to reach a better trade-off between redundancy and error correcting capability. However, for many of the nanodevices we examined, a high error correcting capability, though requiring high redundancy, seems to be unavoidable.

7.2.3 Technological Issues

For what concerns fault tolerance in nanochips, many technological issues will have to be explored. In particular, one will first have to understand how to implement on logic nanochips the wiring scheme needed to implement cascaded modular redundancy, with a redundancy level of up to one hundred.

Second, one will have to understand how to implement the lookup table approach in memory chip encoding and decoding. In particular, the problem will be how to schedule the memory refreshing cycles and how to interface the lookup table (built with microtechnology) to the memory banks (built with nanotechnology).

Finally, the problem of permanent errors in nanoelectronics has not been tackled in this thesis, due to our time constraints. Anyway, this might be an important research field in itself. We have mentioned that works in defect tolerance in nanoelectronics appeared after the bulk of this thesis was completed.

7.3 Final Remarks

In this thesis, we considered the issue of fault tolerance in nanochips. We first calculated error rates for a number of nanogates and applied the results to nanochips with up to 10^{11} gates. For logical nanochips, the fault tolerant technique chosen was general modular redundancy (GMR). Since GMR turned out not to be enough for our purposes, we had to devise a cascaded version of GMR, cascaded general modular redundancy (CGMR). For memory chips, we considered the application of error correcting codes to protect the memory words and devised a lookup table approach for encoding and decoding.

Using the above-mentioned techniques, we were able to show that error rates of the order of one every few years can be achieved for a nanochip containing up to 10^{11} gates and operated at GHz frequencies, provided that redundancy levels of up to a few tens are provided. These results raise the question of how to implement such redundancy levels, especially in terms of wiring. However, it is not possible to tackle such issues until people understand if and how nanochips of various kinds can be produced. What can be said at this stage of development of nanoelectronics is that the high level of redundancy we envisage would anyway fit into typical chip dimensions of the order of a few centimetres.

References

[1] R. Waser, Ed., *Nanoelectronics and Information Technology*, Wiley-VCH, New York (2003).

[2] K. Goser, P. Glösekötter and J. Dienstuhl, *Nanoelectronics and Nanosystems*, Springer-Verlag, Berlin (2004).

[3] K. Nikolic, A. Sadek and M. Forshaw, Architectures for reliable computing from unreliable components, *Proc. IEEE-NANO 2001*, 254-259 (2001).

[4] *International Technology Roadmap for Semiconductors*, 2002 edition, http://public.itrs.net.

[5] J.N. Randall, M.A. Reed and G.A. Frazier, Nanoelectronics: Fanciful physics or real devices?, *J. Vac. Sci. Tech. B* **7**, 1398-1404 (1989).

[6] M.G. Ancona, Design of computationally useful single-electron digital circuits, *J. Appl. Phys.* **79**, 526-539 (1996).

[7] P.D. Tougaw and C.S. Lent, Logical devices implemented using quantum cellular automata*, J. Appl. Phys.* **75**, 1818-1825 (1994).

[8] M. Kruger, M.R. Buitelaar, T. Nussbaumer and C. Schonenbarger, Electrochemical carbon nanotube field-effect transistors, *Appl. Phys. Lett.* **78**, 1291-93 (2001).

[9] P.G. Depledge, Fault-tolerant computer systems, *IEE Proc. A* **128**, n. 4, 257-273 (1981).

[10] S. Roman, *Introduction to Coding and Information Theory*, Springer-Verlag, Berlin (1997).

[11] R.W. Hamming, Error detecting and error correcting codes*, Bell Syst. Tech. J.* **26**, n. 2, 147-160 (1950).

[12] S. Spagocci and T.J. Fountain, Fault rates in nanochips, *Proc. 5th Intl. Symp. on Quantum Confinement: Nanostructures*, 582-596 (1999).

[13] S. Spagocci and T.J. Fountain, Fault rates in nanochip devices, *Proc. Electrochemical Society* **99-22**, 354-368 (1999).

[14] L. M. Adleman, Computing with DNA, *Scientific American* **August**, 54-61 (1998).

[15] S. Lloyd, Quantum-mechanical computers, in: *Scientific American: The Solid State Century,* Scientific American, New York, 98-104 (1997).

[16] B. Crone, A. Dodabalapur, Y.Y. Lin, R.W. Filas, Z. Bao, A. LaDuca, R. Sarpeshkar, H.E. Katz and W. Lin, Large-scale complementary integrated circuits based on organic transistors, *Nature* **403**, 521-523 (2000).

[17] A.A.G. Driskill-Smith, D.G. Hasko and A.H. Ahmed, Fabrication and characterization of vacuum nanoelectronic devices, *Microelectronic Eng*. **53**, 179-182 (2000).

[18] T.P.E. Broekaert, W. Lee and C.G. Fonstad, Pseudomorphic $In_{0.53}Ga_{0.47}As/AlAs/InAs$ resonant tunnelling diodes with peak-to-valley current ratios of 30 at room temperature, *Appl. Phys. Lett.* **53**, 1545-1547 (1988).

[19] D.J. Paul, P. See, I.V. Zozoulenko, K.F. Berggren, B. Kabius, H. Holländer and S. Mantl, Si/SiGe electron resonant tunnelling diodes, *Appl. Phys. Lett.* **77**, 1653-1655 (2000).

[20] J. Stock, J. Malindretos, K.M. Indlekofer, M. Pottgens, A. Forster and H. Luth, A vertical resonant tunnelling transistor for application in digital logic, *IEEE Trans. Electron. Dev.* **48**, n.6, 1028-1032 (2001).

[21] M.H. Devoret and R.J. Schoelkopf, Amplifying quantum signals with the single-electron transistor, *Nature* **406**, 1039-1046 (2000).

[22] H.W.C. Postma, T. Teepen, Z. Yao, M. Grifoni and C. Dekker, Carbon nanotube single electron transistors at room temperature, *Science* **293**, 76-79 (2001).

[23] S.B. Lee, D.G. Hasko and H.M. Hamed, Thermally switched superconducting weak-link transistor with current gain, *Appl. Phys. Lett.* **76**, 2295-2297 (2000).

[24] S.B. Lee, G.D. Hutchinson, D.A. Williams, D.G. Hasko and H.M. Hamed, Superconducting nanotransistor based digital logic gates, *Nanotechnology* **14**, 188-191 (2003).

[25] C. Joachim, J.K. Gimzewski and A. Aviram, Electronics using hybrid-molecular and mono-molecular devices, *Nature* **208**, 541-545 (2000).

[26] J.A. Misewich and A.G. Schrott, Room-temperature oxide field-effect transistor with buried channel, *Appl. Phys. Lett.* **76**, 2632-2634 (2000).

[27] C. Wasshuber, H. Kosina and S. Selberherr, A comparative study of single-electron memories, *IEEE Trans. Electron Dev.* **45**, 2365-2371 (1998).

[28] J.P.A. van der Wagt, Tunnelling based SRAM, *Nanotechnology* **10**, 174-186 (1999).

[29] S.S.P. Parkin, K.P. Roche, M.G. Samant, P.M. Rice, R.B. Beyers, R.E. Scheuerlein, E.J. O'Sullivan, S.L. Brown, J. Bucchigano, D.W. Abraham, Y. Lu, M. Rooks, P.L. Trouilloud, R.A. Wanner and W.J. Gallagher, Exchange-biased magnetic tunnel junctions and application to non-volatile random access memory, *J. Appl. Phys.* **85**, 5828-5833 (1999).

[30] H. Boeve, C. Bruynserade, J. Das, K. Dessein, K. Borghs, J. De Boeck, R.C. Sousa, L.V. Melo and P.P. Freitas, Technology assessment for the implementation of magnetorestrictive elements with semiconductor components in magnetic random access memory (MRAM) architecture, *IEEE Trans. Magnetics* **35**, 2820-2825 (1999).

[31] R.P. Cowburn, The attractions of magnetism for nanoscale data storage, *Phil. Trans. R. Soc. Lond. A* **358**, 281-301 (2000).

[32] K. Hieke, J.O. Wesstrom, T. Palm, B. Stalnacke and B. Stoltz, Ballistic transport and gate control mechanisms in deeply etched electron-waveguide based devices, *Sol. State Electr.* **42**, 1115-1119 (1998).

[33] J.A. Del Alamo, C.C. Eugster, Q. Hu, M.R. Melloch and M.J. Rooks, Electron waveguide devices, *Superlat. & Nanostruc.* **23**, 121-137 (1998).

[34] W. Porod, G. Csaba and A. Csurgay, Field-coupled devices for nanoelectronic integrated circuits, in: *Proc. IEEE-NANO 2001*, 313-318 (2001).

[35] P.D. Tougaw and C.S. Lent, Dynamic behaviour of quantum cellular automata, *J. Appl. Phys.* **80**, n.8, 4722-4736 (1996).

[36] W. Porod, C.S. Lent, G.H. Bernstein, A.O. Orlov, I.H. Amlani, G.L. Snider and J.L. Merz, Quantum-dot cellular automata: Computing with coupled quantum dots, *Intl. J. Electronics* **86**, 549-590 (1999).

[37] R.P. Cowburn and M.E. Welland, Room temperature magnetic quantum cellular automata, *Science* **287**, 1466-1468 (2000).

[38] G. Csaba, W. Porod and A.I. Csurgay, A computing architecture composed of field-coupled single domain nanomagnets clocked by magnetic field, *Intl. J. Circuit Theor. Appl.* **31**, 67-82 (2003).

[39] K.K. Likharev and V.K. Semenov, RSFQ logic/memory family: A new Josephson-junction technology for sub-Terahertz-clock-frequency digital systems, *IEEE Trans. Appl. Supercond* **1**, 3-28 (1991).

[40] D.K. Brock, E.K. Track and J.M. Rowell, Superconductor ICs: The 100-GHz second generation, *IEEE Spectrum* **37**, 40-46 (2000).

[41] T. Orlando, J.E. Mooij, L. Tian, C.H. van der Wal, L. Levitov, S. Lloyd and J.J. Mazo, A superconducting persistent-current qubit, *Phys. Rev. B* **60**, 15398-15413 (1999).

[42] C.H. van der Wal, A.C.J. ter Haar, F.K. Wilhelm, R.N. Schouten, C.J.P.M. Harmans, T.P. Orlando, S. Lloyd and J.E. Mooij, Quantum superposition of persistent current states, *Science* **90**, 773-777 (2000).

[43] A. S. Davydov, *Quantum Mechanics*, Mir Editions, Moscow (1981).

[44] A. Seabaugh, B. Brar, T. Broekaert, F. Morris, J.P.A. van der Wagt and G. Frazier, Resonant tunnelling mixed-signal circuit technology, *Solid-State Elec.* **43**, 1355-1365 (1999).

[45] J.P.A. van der Wagt, A.C. Seabaugh and E.A. Beam, RTD/HFET low standby power SRAM gain cell, *IEEE Electron Device Lett.* **19**, 7-9 (1998).

[46] P. Mazumder, S. Kulkarni, M. Battacharya, J.P. Sun and G.I. Haddad, Digital circuit applications of resonant tunnelling devices, *Proc. IEEE* **86**, 664-686 (1998).

[47] K. Maezawa and T. Mizutani, A new resonant tunnelling logic gate employing monostable-bistable transition, *Jpn. J. Appl. Phys* **32**, 42-44 (1993).

[48] C. Pacha, U. Auer, C. Burwick, P. Glösekötter, K. Goser, W. Prost, A. Brennemann and F.J. Tegude, Threshold logic circuit design of parallel adders using resonant tunnelling devices, *IEEE Trans. VLSI Systems* **8**, 558-572 (2000).

[49] M. Forshaw, K. Nicolic and R. Compaño, The current status of nanoelectronic Devices, *Int. J. Nanoscience* **2**, (2003) 7-29.

[50] H.D. Jensen and J.M. Martinis, Accuracy of the electron pump, *Phys. Rev. B* **46**, 13407-13427 (1992).

[51] H. Pothier, P. Lafarge, P.F. Orfila, C. Urbina, D. Esteve and M.H. Devoret, Single electron pump fabricated with ultrasmall normal tunnell junctions, *Physica B* **169**, 573-574 (1991).

[52] M.W. Keller, J.M. Martinis, N.M. Zimmermann and A.H. Steinbach, Accuracy of electron counting using a 7-junction electron pump, *Appl. Phys. Lett.* **69**, n.12, 1804-1806 (1996).

[53] L.J. Geerligs, V.F. Anderegg, P.A. Holweg, J.E. Mooij, H. Pothier, D. Esteve, C. Urbina and M.H. Devoret, Frequency-locked turnstile device for single electrons, *Phys. Rev. Lett.* **64**, 2691-2694 (1990).

[54] K. Yano, T. Ishii, T. Sano, F. Murai and K. Seki, Single-electron-memory integrated circuit for Giga-to-Terabit storage, *Proc. IEEE Intl. Solid-State Circuits Conf.*, 266-267 (1996).

[55] Z.A.K. Durrani, A.C. Irvine and H. Ahmed, Coulomb blockade memory using integrated single-electron transistor/metal-oxide-semiconductor transistor gain cell, *IEEE Trans. Electron Devices* **47**, 2334-2339 (2000).

[56] T. Oya, T. Asay, T. Fukui and Y. Amemiya, A majority-logic device using an irreversible single-electron box, *IEEE Trans. Nanotechnology* **2**, n.1, 15-22 (2003).

[57] T. Oya, T. Asay and Y. Amemiya, Single-electron logic device with simple structure, *Electronics Lett.* **39**, n.13, 965-967 (2003).

[58] F. Nakajima, K. Kumamura, J. Motohisa and T. Fukui, GaAs single electron transistors fabricated by selective area metalorganic vapor phase epitaxy and their application to single electron logic circuits*, Jpn. J. Appl. Phys.* **38**, 415-417 (1999).

[59] C.P. Heij, P. Hadley and J.E. Mooij, Single-electron inverter, *Appl. Phys. Lett* **78**, 1140-1142 (2001).

[60] N.J. Stone and H. Ahmed, Logic circuit elements using single-electron tunnelling transistors, *Electronics Lett.* **35**, 1883-1884 (1999).

[61] C. S. Lent, P. D. Tougaw and W. Porod, Bistable saturation in coupled quantum dots for quantum cellular automata, *Appl. Phys. Lett*. **62**, 714-716 (1993).

[62] R.K. Kummamuru, A.O. Orlov, R. Ramasubramaniam, C.S. Lent, G.H. Bernstein and G.L. Snider, Experimental demonstration of a QCA shift register and analysis of errors, *Proc. IEDM 02*, 95-98 (2002).

[63] D. Berzon and T.J. Fountain, A memory design in QCAs using the SQUARES formalism, *Proc. 9th Great Lakes Symp. on VLSI,* 166-169 (1999).

[64] G. Toth and C.S. Lent, Quasiadiabatic switching for metal-island quantum-dot cellular automata, *J. Appl. Phys.* **85**, n.3. 2977-2984 (1999).

[65] A.O. Orlov, I. Amlani, R.K. Kummamuru, R. Ramasubramaniam, G. Toth, C.S. Lent, G.H. Bernstein and G.L. Snider, Experimental demonstration of clocked single-electron switching in quantum-dot cellular automata, *Appl. Phys. Lett.* **77**, n.2, 295-297 (2000).

[66] I. Amlani, A.O. Orlov, G. Toth, G.H. Bernstein, C.S. Lent and G.L. Snider, Digital logic gate using quantum-dot cellular automata, *Science* **284**, 289-291 (1999).

[67] J. Timler and C.S. Lent, Power gain and dissipation in quantum-dot cellular automata, *J. Appl. Phys.* **91**, n.2, 823-831 (2002).

[68] M. Governale, M. Macucci, G. Iannaccone, C. Ungarelli and J. Martorell, Modeling and manufacturability assessment of bistable quantum-dot cells, *J. Appl. Phys* **85**, 2962-2971 (1999).

[69] A.O. Orlov, I. Amlani, G.H. Bernstein, C.S. Lent and G.L. Snider, Realization of a functional cell for quantum-dot cellular automata, *Science* **277**, 928-930 (1997).

[70] R. Stadtler, M. Forshaw and C. Joachim, Modulation of electron transmission for molecular data storage, *Nanotechnology* **14**, 138-142 (2003).

[71] L.A. Bumm, J.J. Arnold, M.T. Cygan, T.D. Dunbar, T.P. Burgin, L. Jones II, D.L. Allara, J.M. Tour and P.S. Weiss, Are single molecular wires conducting?, *Science* **271**, 1705-1707 (1996).

[72] R. Stadtler, S. Ami, M. Forshaw and C. Joachim, A tight-binding study of a logic gate for adding numbers inside a molecule, *Nanotechnology* **13**, 424-428 (2002).

[73] P.G. Collins, M.S. Arnold and P. Avouris, Engineering carbon nanotubes and nanotube circuits using electrical breakdown, *Science* **292**, 706-709 (2001).

[74] Y. Huang, X. Duan, Q. Wei and C.M. Lieber, Direct assembly of one-dimensional nanostructures into functional networks, *Science* **191**, 630-633 (2001)

[75] M. Ahlskog, R. Tarkiainen, L. Roschier and P. Hakonen, Single-electron transistor made of two crossing multiwalled carbon nanotubes and its noise properties, *Appl. Phys. Lett.* **77**, n.24, 4037-4039 (2000).

[76] S. Ami and C. Joachim, Logic gates and memory cells based on single C_{60} electromechanical transistors, *Nanotechnology* **12**, 44-52 (2001).

[77] T.W. Tombler, C. Zhou, L. Alexseyev, J. Kong, H. Dai, L. Liu, C.S. Jayanti, M. Tang and S.Y. Wu, Reversible electromechanical characteristics of carbon nanotubes under local-probe manipulation, *Nature* **405**, 769-772 (2000).

[78] R. Stadtler, S. Ami, M. Forshaw and C. Joachim, A memory/adder model based on a single C_{60} molecular transistor, *Nanotechnology* **12**, 350-357 (2001).

[79] R. Stadtler and M. Forshaw, The performance of hybrid-molecular architectures with current CMOS technology as a reference, *Physica E* **13**, 930-933 (2002).

[80] A. Bachtold, P. Hadley, T. Nakanishi and C. Dekker, Logic circuits with carbon nanotube transistors, *Science* **294**, 1317-1320 (2001).

[81] R. Martel, V. Derycke, J. Appenzeller, S. Wind and P. Avouris, Carbon nanotube field-effect transistors and logic circuits, *Proc. DAC 2002*, 94-98 (2002).

[82] Y. Cui and C.M. Lieber, Functional nanoscale electronic devices assembled using silicon nanowire building blocks, *Science* **291**, 851-853 (2001).

[83] Y. Huang, X. Duan, Y. Cui, L.J. Lauhon, K.H. Kim and C.M. Lieber, Logic gates and computation from assembled nanowire building blocks, *Science* **294**, 1313-1317 (2001).

[84] Y. Cui, Z. Zhong, D. Wang, W.U. Wang and C.M. Lieber, High performance silicon nanowire field effect transistors, *Nano Lett.* **3**, n.2, 149-152 (2003) .

[85] J. Han and P. Yonker, A system architecture solution for unreliable nanoelectronic devices, *IEEE Trans. Nanotechnology* **1**, n.4, 201-208 (2002).

[86] J. Han and P. Yonker, A defect- and fault- tolerant architecture for nanocomputers, *Nanotechnology* **14**, 224-230 (2003).

[87] K. Nikolic, A. Sadek and M. Forshaw, Fault-tolerant techniques for nanocomputers, *Nanotechnology*, **13** 357-362 (2002).

[88] J. Von Neumann, Probabilistic logics and the synthesis of reliable organisms from unreliable components, in: C.E. Shannon and J. McCarthy, Eds., *Automata Studies*, Princeton University Press, 43-98 (1955).

[89] H. Schepers, Fault-tolerant systems, in: J. Vytopil, Ed., *Formal Techniques in Real-Time and Fault-Tolerant Systems*, Kluwer Academic Publishers, 3-31 (1993).

[90] F.P. Matur, On reliability modelling and analysis of ultrareliable fault-tolerant digital systems, *IEEE Trans. Comp.* **C 20**, 1376-1382 (1971).

[91] F.P Matur and P.T. de Sousa, Reliability models of NMR systems, *IEEE Trans. Reliability* **R 24**, n. 2, 108-113 (1975).

[92] F.P Matur and P.T. de Sousa, Reliability modelling and analysis of general modular redundant systems, *IEEE Trans. Reliability* **R 24**, n. 5, 296-299 (1975).

[93] J.F. Wakerly, Microcomputer reliability improved using triple-modular redundancy, *Proc. IEEE* **64**, n. 6, 889-895 (1976).

[94] J. Losq, A highly efficient redundancy scheme: self-purging redundancy, *IEEE Trans. Comp.* **C 25**, n. 6, 569-578 (1976).

[95] P.T. de Sousa and F.P Matur, Sift-out modular redundancy, *IEEE Trans. Comp.* **C 27**, n. 7, 624-627 (1978).

[96] H.Y. Lo, L.P. Ju and C.C. Su, General version of reconfiguration N modular redundancy system, *IEE Proc. G* **137**, n. 1, 1-4 (1990).

[97] K.W. Philp and N.D. Deans, Comparative redundancy, an alternative to triple modular redundant system design, *Microelectron. Reliab.* **37**, n. 4, 581-585 (1996).

[98] F. Distante, M. G. Sami and R. Stefanelli, Reconfiguration techniques in the presence of faulty interconnections, in: *Proc. 1st Intl. Conf. on Wafer Scale Integration*, 379-388 (1989).

[99] M. Radu, Assessing the reliability and safety of fault tolerant designs, in: *Proc. 24th Intl. Spring Seminar on Electronics Technology,* 56-58 (2001).

[100] M. Radu, D. Pitica, R. Munteanu and C. Posteuca, Complex reliability evaluation of voters for fault-tolerant design, in: *Proc. 24th Intl. Spring Seminar on Electronics Technology,* 331-336 (2001).

[101] M.G. Ancona, Systolic processor design using single-electron digital circuits, *Superlat. & Nanostruc.* **20**, n.4, 461-472 (1996).

[102] J.M. Martinis, M. Nahum and H.D. Jensen, Metrological accuracy of the electron pump, *Phys. Rev. Lett.* **72**, n.6, 904-907 (1994).

[103] M.W. Keller, J.M. Martinis and R.L. Kautz, Rare errors in a well-characterized electron pump: Comparison of experiment and theory, *Phys. Rev. Lett.* **80**, 4530-4533 (1998).

[104] J.M. Martinis and M. Nahum, Effect of environmental noise on the accuracy of Coulomb-blockade devices, *Phys. Rev. B* **48**, 18316-18319 (1993).

[105] M.I. Lutwyche and Y. Wada, Estimate of the ultimate performance of the single-electron transistor, *J. Appl. Phys.* **75**, 3654-3661 (1994).

[106] J.G. Simmons, Generalized formula for the electric tunnel effect between similar electrodes separated by a thin insulating film, J. *Appl. Phys.* **34**, 1793-1803 (1963).

[107] H.D. Jensen and J.M. Martinis, Performance of the electron pump with stray capacitances, *Physica B* **194-196**, 1255-1256 (1994).

[108] M.G. Ancona, Bit errors in single-electron digital circuits, in: *Proc. 3rd International Workshop on Quantum Functional Devices* (1997).

[109] A.N. Korotkov, Wireless single-electron logic biased by alternating electric field, *Appl. Phys. Lett.* **67**, n.16, 2412-2414 (1995).

[110] A.N. Koroktov and K. Likharev, Single-electron parametron-based logic devices, *J. Appl. Phys.* **84**, n.11, 6114-6126 (1998).

[111] C.S. Lent, P.D. Tougaw and W. Porod, Quantum cellular automata: The physics of computing with quantum dot molecules, in: *Proc. PhysComp '94,* 1-9 (1994).

[112] C.S. Lent, P.D. Tougaw, W. Porod and G.H. Bernstein, Quantum cellular automata, *Nanotechnology* **4**, 49-57 (1993).

[113] D. Berzon and T.J. Fountain, SQUARES – Standard quantum cellular automata array elements, http://ipga.phys.ucl.ac.uk/reports/rep98-1.pdf.

[114] C.S. Lent and P.D. Tougaw, A device architecture for computing with quantum dots, *Proc. IEEE* **85**, 541-557 (1997).

[115] B.S. Chalk, *Computer Organisation and Architecture*, Macmillan, London (1996).

Appendix 1 FAULT RATES IN NANOCHIPS

S. Spagocci, T. Fountain
University College London
Gower Street, London WC1E 6BT, England

ABSTRACT

In this paper we address the problem of estimating nanochip error rates, taking intrinsic error rates and fault-tolerant techniques into account. In particular we first describe cascaded triple modular redundancy (CTMR), an iterated version of triple modular redundancy that we believe might greatly improve the potentialities of the latter, while retaining its advantages. In particular CTMR is expected to be particularly suitable for online implementation. We then analyze intrinsic error rates for logic gates based on single electron devices and quantum cellular automata, respectively. Intrinsic error rates in such devices clearly show the necessity of fault-tolerant techniques. We then show that, through application of CTMR, chips containing ~ 10^{11} such devices can be made perfectly reliable, at least as far as intrinsic runtime errors are concerned, with a level of redundancy of ~ 100 at worst. This, however, requires carefully tuning the devices' operating conditions and design solutions.

INTRODUCTION

The proposed introduction of nanometer-scale components should make it possible to conceive chips containing up to 10^{12} logic gates. This would be particularly interesting for the implementation of parallel systems on a single chip. For such an assembly to work, the introduction of fault-tolerant techniques seems inevitable. The huge number of devices makes the chip unreliable, even if the devices are highly reliable in themselves.

We believe that triple modular redundancy (TMR) [1] is a fast and easily implemented fault tolerant technique. In TMR, the place of each device is taken by a block of three identical devices and majority voting is performed among them. When TMR is not enough, we propose the use of cascaded triple modular redundancy (CTMR). In this approach, the devices are first clustered and TMR is applied to the clusters. The voting circuits are then suitably clustered and TMR is applied again. The process is iterated for a number of steps.

We first considered single-electron devices, of which the electron pump [2] is a prototype. The electron pump is an array of metallic islands, separated by nanometer-scale junctions, through which an electron is made to tunnel sequentially. Logic gates based on the electron pump have been proposed [3]. At high frequencies, the main error source arises from pumping the electron too fast, so that the desired tunneling process is missed [4-6]. We predict complete reliability for a system with 10^{11} effective devices at GHz frequency and room temperature, with 4 TMR levels.

We also considered quantum cellular automata [7]. A QCA cell is a square array of 4 quantum dots, occupied by two electrons. The cell has two stable states, which are taken to mean 0 and 1. Due to electrostatic repulsion, each cell interacts with its neighbors. QCA logic gates based on this principle have been proposed [8]. The main error source arises from thermal excitation [9], which may create a kink in a row of previously aligned cells, thus giving a wrong answer. We predict complete reliability for a system with 10^{11} effective devices at GHz frequency and liquid nitrogen temperature, with 3 TMR levels.

2 CASCADED TRIPLE MODULAR REDUNDANCY

Since cascaded triple modular redundancy is an iterated version of the well known technique of triple modular redundancy [1], the latter will be briefly described here. In TMR the place of each potentially faulty unit is taken by a block of three identical units, and majority voting is performed among them. This works well provided one is dealing with binary units, under the assumption that a fault can only flip the unit's answer and the voting circuitry is perfectly reliable. If the working circuitry is not perfectly reliable, the formalism has to be suitably modified.

A TMR unit fails when either two or three units out of three fail. The probability of such an event to occur, $P_{f/tmr}$, can be calculated by a binomial distribution, as follows:

$$P_{f/tmr} = (P_f)^3 + 3 \cdot (P_f)^2 \cdot (1 - P_f) \qquad (1)$$

Eq. (1) takes an even simpler form if one considers the limiting case in which the intrinsic probability for an element to fail is small. As we see in the following sections, this will be the case in our work. By taking a first-order Taylor expansion of Eq. (1), one obtains:

$$P_{f/tmr} = 3 \cdot (P_f)^2 \qquad (2)$$

For the general case of an imperfect voting circuitry, with failing probability P_v, we have to consider that the TMR unit gives a wrong answer if either majority voting would give a wrong answer and the voting circuit works, or majority voting would give a right answer and the voting circuitry fails. If, furthermore, we consider a cluster of N_g gates, Eq. (2) generalises to:

$$P_{f/tmr} = 3 \cdot (N_g \cdot P_f)^2 + P_v \qquad (3)$$

We are now ready to discuss cascaded triple modular redundancy in detail. According to CTMR, the potentially faulty units are first clustered in a suitable way, and triple modular redundancy is applied to the clusters. The voting circuits are then suitably clustered, as well, and triple modular redundancy is applied to each cluster. The process is iterated for a number of steps, as in Fig. (1). The intrinsic reliability of both the functional units and the voting circuitry impose a bound on cluster size and the number of cascaded levels.

Let us consider a chip containing clusters of N_g identical logic gates. The gates have a faulting probability $P_{f;g}$. The voting circuits have a faulting probability $P_{f;v}$. Both probabilities are assumed to be small. We deal with two possible regimes. In the first regime, the voting circuitry can be considered as perfectly reliable in comparison to the TMR units. In the second regime, the TMR units can be considered as perfectly reliable in comparison to the voting circuits. The two regimes will be referred to as the perfect voting circuitry (PVC) regime and the imperfect voting circuitry (IVC) regime.

In the PVC regime, the voting circuitry is supposed to be much more reliable than the clusters.

We then require that, in Eq. (3):

$$3 \cdot (N_g \cdot P_{f;g})^2 \geq 10 \cdot P_{f;v} \tag{4}$$

By iterated use of Eq. (3), taking Eq. (4) into account, it is possible to calculate the failing probability of a cluster at the ith CTMR stage, $P_{f/tmr;c}(i)$, and its minimum size, $N_{min}(i)$, under the hypothesis that $P_v = P_{f;g}$. In particular, one obtains:

$$P_{f/tmr;c}(i) = 10^i \cdot P_{f;g} \tag{5}$$

and:

$$N_{\min}(i) = \sqrt{\frac{10^{2-i}}{3 \cdot P_{f;g}}} \tag{6}$$

The chip's failing probability after i CTMR stages, $P_{f/tmr}(i)$, can be written as:

$$P_{f/tmr}(i) = \frac{N_g}{\prod_{s=1}^{i} N_{\min}(s)} \cdot P_{f/tmr;c}(i) \tag{7}$$

where the product in Eq. (7) can be calculated by making use of decimal logarithms:

$$\prod_{s=1}^{i} N_{\min}(s) = 10^{\frac{i \cdot (3-i)}{4}} \cdot (3 \cdot P_{f;g})^{-\frac{i}{2}} \tag{8}$$

We can then give a final expression for the failing probability of a chip with i CTMR stages.

In particular, we have:

$$P_{f/tmr}(i) = 10^{\frac{i \cdot (1+1)}{4}} \cdot (3 \cdot P_{f;g})^{\frac{i}{2}} \cdot (N_g \cdot P_{f;g}) \tag{9}$$

Differentiation of Eq. (9) with respect to i shows that $P_{f/tmr}(i)$ is monotonically decreasing with i. It is therefore advisable to have as many CTMR levels as possible. The very nature of the iteration process, however, prevents it from going on forever.

In fact, the maximum number of stages is defined by the condition:

$$\frac{N_g}{\prod_{s=1}^{i} N_{\min}(s)} = 1 \tag{10}$$

Equivalently, the maximum number of stages is given by i$_{\max,}$ where i$_{\max}$ is the integer part of the number i satisfying Eq. (10) in conjunction with Eq. (8). If a number i$_{\max}$ of stages is not enough, it is further possible to apply TMR to the whole chip, by using micron-scale voting circuitry.

The chip's failing probability, then, becomes:

$$P_{f/tmr}(i) = 3 \cdot \{10^{\frac{i\cdot(1+1)}{4}} \cdot (3 \cdot P_{f;g})^{\frac{i}{2}} \cdot (N_g \cdot P_{f;g})\}^2 \tag{11}$$

where i = i$_{\max}$.

In the IVC regime, the clusters are supposed to be much more reliable than the voting circuitry. We will see that, by following a suitable clustering scheme, it is possible to bring the whole chip to have the same faulting probability as that of an individual gate. In order to implement this regime, we require that in Eq. (3):

$$3 \cdot (N_g \cdot P_{f;g})^2 \le \frac{1}{10} \cdot P_{f;v} \tag{12}$$

It follows than that the clusters must have a maximum size, at any stage and under the hypothesis that $P_{f;g} = P_v$, given by the following expression:

$$N_{\max} = \sqrt{\frac{1}{30 \cdot P_{f;g}}} \tag{13}$$

We can now calculate the faulting probability of a cluster after i CTMR stages.

In particular, we obtain:

$$P_{f/tmr}(i) = \frac{N_g}{(N_{\max})^i} \cdot P_{f;g} \tag{14}$$

By differentiating Eq. (14), used in conjunction with Eq. (13), with respect to i, $P_{f/tmr}$ (i) is seen to be monotonically decreasing with i. One therefore needs as many CTMR stages as possible. The iteration process, of course, cannot go on indefinitely.

The maximum number of stages is defined by the condition:

$$\frac{N_g}{(N_{\max})^i} = 1 \tag{15}$$

which translates into:

$$i_{\max} = [-2 \cdot \frac{Log(N_g)}{Log(30 \cdot P_{f;g})}] + 1 \tag{16}$$

where [.] is the integer part function.

We then obtain a surprising simple expression for the chip's overall failing probability:

$$P_{f/tmr} = P_{f;g} \tag{17}$$

If needed, it is then possible to apply TMR to the chip's output, using a voting circuitry much more reliable than the chip's gates (micron-scale technology, in practice). The system's failing probability then becomes:

$$P_{f/tmr} = 3 \cdot (P_{f;g})^2 \tag{18}$$

The previously derived formulae refer to an idealised clustering scheme. Real-world design solutions, presumably, can only approximate such a scheme.

3 ERROR RATES IN SEDs

The first class of devices we considered was single-electron devices (SEDs), of which the electron pump [2] is a prototype. The electron pump is an array of metallic islands, separated by nanometer-scale junctions, through which an electron is made to tunnel sequentially. A single-electron switch [3], based on the electron pump principle, is shown in Fig. (2). When the control island is free, the input electron is made to turn left. When it is occupied by an electron, repulsion makes the previous path energetically unfavorable. The input electron, then, turns right.

The single electron switch is the building block of a family of logic gates [3], which we show in Figs. (3) and (4). We first describe the error sources affecting electron pumps, since fault rates for electron pumps have been investigated both theoretically [4,10] and experimentally [5,6,11]. The results are then extended to the previously mentioned SED-based logic gates.

There are three kinds of fault sources affecting an electron pump [4]: frequency, thermal and cotunnelling errors. In frequency errors, the electron is pumped too fast as compared to the half life for the tunnelling process, so that the desired tunnelling process is missed. In thermal errors, the electron goes the wrong way, acquiring the necessary energy through thermal exchange with the environment. In cotunnelling errors, the electron goes the wrong way by simultaneously tunnelling through all its junctions.

Experimentally, the main fault source for electron pumps at the micron scale, operated at more than ~ 1 MHz, is frequency errors [5,6,11]. The corresponding error rate per clock cycle, $P_{f;g}$, is shown to have the form [4]:

$$P_{f;g} = \exp\left(-\frac{\alpha}{R \cdot C \cdot f}\right) \qquad (19)$$

with:

$$\alpha = \frac{n-1}{8 \cdot n^2} \qquad (20)$$

where R and C are the tunnelling junction’s resistance and capacitance, respectively, f is the clock frequency and n is the number of junctions in the pump. Eq. (19) simply represents the probability that the electron has not tunnelled through the junction after the pulsing cycle is completed. If the tunnelling process is missed, the electron goes back through the pump by cotunnelling, in a small time-scale as compared to the clocking time [4].

At frequencies below ~ 1 MHz the error rate approaches an asymptotic value [5,6,11], as shown in Fig. (5). The most likely candidate is photon-assisted cotunnelling [5,6,10,11]. The electron tunnels the wrong way, by absorbing energy from the environmental noise. The most likely noise candidate seems to be charge traps on the device substrate, which slowly relax with time [5]. A fault rate per cycle of ~ 10^{-8} was observed for a 7-junction electron pump operating at 35 mK, with micron-scale islands [6,11]. A fault rate per cycle of ~ 10^{-6} was instead observed for a 5-junction electron pump, under similar conditions [5].

Errors in electron pumps are controlled by suitable adimensional parameters. For frequency and cotunnelling errors the parameter is [4]:

$$R \cdot C \cdot f \qquad (21)$$

which is essentially the ratio between the pumping time and the time-scale for the tunnelling process. For thermal errors the relevant parameter is [4]:

$$\frac{e^2}{C \cdot k \cdot T} \qquad (22)$$

Eq. (22) essentially represents the ratio between the energy jump due to a tunnelling event and the thermal energy at the operating temperature.

The experimental or calculated data on failure rates we are aware of refer to micron-scale devices. We need them at the nanometer regime. We then introduce a parameter λ, the ratio between the target length scale and the length scale of available results.

As shown in Ref. [12], the effective junction capacitance can be calculated as:

$$C = C_j + C_i \tag{23}$$

where C_j is the junction capacitance and C_i is the island's capacitance to earth. The junction capacitance can be approximated by a parallel-plate capacitor of area A, separated by a gap d [13]. Since the tunnelling junction is already at the nanometer level, we do not need to vary the width d. The effective area A, instead, scales as λ^2. Then:

$$C_j \propto \lambda^2 \tag{24}$$

The island capacity to earth can be calculated by approximating it to a bidimensional metallic sheet [12]. So:

$$C_i \propto \lambda \tag{25}$$

Provided λ is much smaller than one ($\lambda \sim 10^{-3}$ in our case), the stray capacitance C_i will dominate in Eq. (23). The total capacitance, then, has the same scaling law as the stray capacitance, as given by Eq. (25).

Experimentally, C ~ 0.1 fF at the micron scale [5,6]. Consequently, we predict C ~ 0.1 aF at the nanometer scale. Incidentally, it has to be seen whether an electron pump dominated by its stray capacitance would still work.

An expression for the tunneling junction resistance is given in Ref. [14]. Using such an expression, it is possible to show that the effect of the scaling on R can be compensated by shortening the tunnelling barrier by δd, where δd ~ 0.1 nm. Experimentally, R ~ 0.1 MΩ for a micron scale pump [5,6]. Therefore, we assume a similar value for the nanometer scale.

Since the junction resistance R can be kept constant by displacing the tunnel junction, we only have to worry about the products $C\cdot f$ and $C\cdot T$ as long as scaling is concerned. The experimental data for fault rates in micron-scale electron pumps were obtained for a clock frequency of ~ 1 MHz and an operating temperature of ~ 0.3 K. We now need to scale such data to nanometer-scale pumps with a clock frequency of ~ 1 GHz and an operating temperature of ~ 300 K (room temperature). Our change of conditions can be described by the transformations $f \rightarrow 10^3 \cdot f$ and $T \rightarrow 10^3 \cdot T$. On the other hand, since we are passing from the micron to the nanometer scale, $\lambda \rightarrow 10^{-3} \cdot \lambda$. The products $C\cdot f$ and $C\cdot T$ are then kept constant and, according to Eqs. (21), (22) and (25), even the error rates and their relative importance.

To our knowledge, no quantitative treatment on error rates in SED-based logic gates has been proposed. Such gates are affected by two different kinds of errors: switching errors and pump errors. The electron, in fact, can be switched the wrong way, or it can travel back through one of the electron pumps composing the gates. In general, we can imagine two distinct regimes, in which either error dominates. Pumping errors can be made smaller by adding a suitable number of junctions to the input and output lines. However, switching errors fix an upper limit for device dimensions, since above a certain size they dominate over pumping errors, making any improvement in pumping accuracy pointless.

We considered an array of ~ 10^{11} SDE-based electron gates [3]. Simulation results on error rates in single electron switches are described in Ref. [15]. Error probabilities of ~ 10^{-8} and ~ 10^{-4} are reported for right and left switching, respectively. The overall switching error probability is therefore of ~ 10^{-4}. Such an error rate would imply a lifetime of ~ 10^{-2} sec, with 9+1 CTMR levels, which is clearly unacceptable. Therefore, the electron gates have to be redesigned, so as to reduce switching errors to an acceptable level. By adding a suitable number of junctions to the input and output lines, respectively, it is then possible to reduce pumping errors. In the following we assume that the gates have been suitably redesigned. We have preliminary indications that such a redesign process is possible. However, our design solution has to be further checked.

We can now give predictions on the lifetimes and redundancies of SED-based chips. For an array of ~ 10^{11} effective gates working at ~ 1 GHz, a design based on the 7-junction electron pump ($P_{f;g} \approx 10^{-8}$) implies a lifetime of ~ 1 month, with 4+1 CTMR levels. With a design based on the 9-junction electron pump ($P_{f;g} \approx 10^{-10}$, as extrapolated from the error rates of the 5 and 7-junction electron pumps, by using the cotunneling expression of Ref. [4]), this result can be greatly improved. An "infinite" lifetime is predicted, with 3+1 CTMR levels. The corresponding redundancy level is 81.

The linear dimensions of a SED-based chip with ~ 10^{11} effective gates (but ~ 10^{13} total gates) and 3+1 CTMR stages are estimated to be ~ 1 cm. From this point of view, the proposed solution seems to be realistic. However, power dissipation would pose serious difficulties. Powering the devices with nanometer-wide buses feeding rows, in fact, would imply a power dissipation of ~ 100 MW, which is a physically absurd result. On the other hand, widening the buses so as to bring power dissipation at the level of ~ 1 W would imply a chip having one side ~ 10 km long, which is again absurd. A power dissipation of ~ 1 W could be obtained by feeding each gate independently or, equivalently, by making pump islands stick out of a metallic base the size of the chip, which would carry the power. The proposed powering solution would be hardly feasible with the multiphase pulsing scheme proposed in Ref. [4]. Presumably, a single-phase design like that of Ref. [16] should be considered.

4 ERROR RATES IN QCAs

We also considered quantum cellular automata (QCAs) [7]. A QCA cell is a square array of four quantum dots, occupied by two electrons. The cell has two stable states, corresponding to electrons sitting at the ends of each diagonal. Each cell is influenced by its neighbors through electrostatic repulsion. A family of QCA logic gates based on this principle has been proposed [8]. We show them in Figs. (6) and (7).

The main error source in QCA-based devices arises from thermal excitation [9], which may create kinks in a row of previously aligned cells, thus giving a wrong output. See Fig. (8) for the one-kink case. To our knowledge, no detailed account of QCA fault rates has been given. We assume that such error rates can be estimated from thermodynamic considerations, following the qualitative remarks of Ref. [9].

The applicability of thermodynamic considerations is not a priori guaranteed, since the device might not have enough time to explore higher excited states during its clock cycle. However we think that, at least in one of the possible operating regimes for QCAs, the use of thermodynamic arguments can be justified. In the so-called adiabatic switching regime [9], in fact, the clocking time t_c is chosen to be much longer than the time corresponding to the energy splitting between the ground state and the first excited state of a QCA unit (a gate or a wire). If the first excited state has an energy splitting of ΔE with respect to the ground state, the previously mentioned condition translates into [17]:

$$t_c >> \frac{\hbar}{\Delta E} \qquad (26)$$

Condition (26) is verified even for the higher rank states, since their energy splitting is $\Delta E' > \Delta E$. The unit is then prevented from getting stuck on a metastable excited state [17]. We think that such a condition also gives the unit time to explore higher rank states through thermal excitation, therefore validating the use of thermodynamic arguments.

We obtained an order-of magnitude estimation for fault rates in a QCA unit by considering the case of a QCA wire. This approach has been invoked for approximated considerations on QCA fault rates [9]. By applying Boltzmann's statistics, the transition probability to the nth excited state of an assembly of N QCA cells is calculated to be:

$$P_{f;g} = g_n \cdot \frac{\exp\left(-\frac{\Delta E_n}{k \cdot T}\right)}{\sum_n \exp\left(-\frac{\Delta E_n}{k \cdot T}\right)} \qquad (27)$$

In Eq. (27), g_n is the degeneracy of the nth excited level. In the case of a QCA wire, g_n is the number of ways in which n kinks can be chosen from N-1 positions:

$$g_n = \binom{N-1}{n} \qquad (28)$$

while ΔE_n is the splitting of the nth excited level with respect to the ground state. In the case of an infinite wire, ΔE_n is given by the expression [9]:

$$\Delta E_n = n \cdot \Delta E \qquad (29)$$

where ΔE is the splitting between the first excited level and the ground state. Eqs. (27) and (29) show that the probabilities of thermal excitation of rank higher than one are exponentially dumped. On the other hand, Eq. (29) is approximately valid even for a finite wire [9]. Therefore, in a first approximation, we can assume that errors arise from the thermal excitation of one kink.

Eq. (27), then, takes the form:

$$P_{f;g} = N \cdot \exp\left(-\frac{\Delta E}{k \cdot T}\right) \tag{30}$$

where we have put N-1 ≈ N and approximated the partition function to 1.

Again, we introduce a parameter λ, the ratio between the target length scale and the length scale of available results and we assume the following Coulomb-like scaling law for ΔE:

$$\Delta E \propto \varepsilon^{-1} \cdot \lambda^{-1} \tag{31}$$

where ε is the relative dielectric constant of the device's substrate (ε ≈ 10 [7]).

The energy splitting ΔE has been calculated to be ≈ 0.8 meV for an infinite QCA wire with dot centers within a cell spaced by ~ 30 nm [18]. For cells spaced by ~ 1 nm, our change of scale can be expressed by the transformations $\lambda \rightarrow 3 \cdot 10^{-2} \cdot \lambda$, $\varepsilon \rightarrow 10^{-1} \cdot \varepsilon$. The scaling law for ε derives from the fact that, at the nanometer (i.e. molecular) scale, ε ≈ 1, since screening effects are no more present [9]. We estimate that ΔE ≈ 0.2 eV for nanometer-level QCA cells.

We can now predict lifetimes and redundancies of QCA-based chips. For a QCA-based gate working at a nanometer scale, room temperature, an error rate of ~ 10^{-4} is predicted. For an array of ~ 10^{11} effective gates, this implies a lifetime of ~ 10^{-2} sec at ~ 1 GHz, with 9+1 CTMR levels. This is clearly unacceptable. However, by operating at liquid Nitrogen temperature (77 K), a gate error rate of ~ 10^{-19} and an "infinite" lifetime might be expected, with 2+1 CTMR levels. The corresponding redundancy level would be 27. However these figures might be subject to revision, due to the doubts concerning the application of thermodynamic considerations to QCA fault rate estimations.

We estimate the linear dimensions of a QCA-based chip with ~ 10^{11} effective gates (but ~ 10^{12} total gates) and 3+1 CTMR stages to be ~ 1 cm. The proposed solution would then seem realistic. The problem of power dissipation in QCA-based chips has not yet been addressed.

5 CONCLUSIONS

CTMR, of course, is not the only possible approach to fault tolerance in nanochips. In particular, the following approaches [1] seem to be feasible:

- n-modular redundancy (plain or cascaded) in space
- n-modular redundancy (plain or cascaded) in time
- duplication (or n-fold replication) with error recovery
- error correcting codes

Apart from error correcting codes, all such ideas can be analyzed by following the same line of thought as TMR. Error correcting codes have still to be considered, though, and the preliminary results on the previously mentioned approaches have to be further checked. Reconfiguration, by its very nature, is unsuitable for taking care of the intrinsic faults of the nanodevices we considered, which are of a transient nature, though it may well be effective in dealing with manufacturing errors.

The intrinsic error rates we analyzed are of a transient nature. Impurities and dislocations, however, could cause permanents faults, which have not yet been considered in our device analysis. The fact that the typical dimensions of the devices we considered are ~ 10 nm, so that their area is ~ 10^4 times smaller than present-day devices suggests that single crystal errors will have more significant effects than in microcircuitry. On the other hand, the high level of redundancy required by the transient gate fault rates might provide protection against permanent faults, as well.

ACNOWLEDGEMENT

This paper and associated research were funded as part of the DARPA ULTRA program. Grant number N00014-96-1-0850. Principal investigator: T J Fountain.

REFERENCES

[1] H. Schepers, Terminology and Paradigms for Fault Tolerance, in: J. Vytopil, Ed., Formal Techniques in Real-Time and Fault-Tolerant Systems, pp. 3-31, Kluwer Academic Publishers, Boston, MA (1993).
[2] H. Pothier, P. Lafarge, C. Urbina, D. Esteve and M.H. Devoret, Europhys. Lett., **17**, 249 (1992).
[3] M.G. Ancona, J. Appl. Phys., **79**, 526 (1996).
[4] H.D. Jensen and J.M. Martinis, Phys. Rev. B, **46**, 13407 (1992).
[5] J.M. Martinis, M. Nahum and H.D. Jensen, Phys. Rev. Lett., **72**, 904 (1994).
[6] M.W. Keller, J.M. Martinis, N.M. Zimmermann and A.H. Steinbach, Appl. Phys. Lett., **69**, 1804 (1996).
[7] C.S. Lent, P.D. Tougaw, W. Porod and G.H. Bernstein, Nanotechnology, **4**, 49 (1993).
[8] P.D. Tougaw and C.S. Lent, J. Appl. Phys., **75**, 1818 (1994).
[9] C.S. Lent, P.D. Tougaw and W. Porod, in PhysComp '94: Proceedings of the Workshop on Physics and Computing, IEEE Computer Society Press, Los Alamitos, CA (1994).
[10] J.M. Martinis and M. Nahum, Phys. Rev. B, **48**, 18316 (1993).
[11] M.W. Keller, J.M. Martinis and R.L. Kautz, Phys. Rev. Lett., **80**, 4530 (1998).
[12] H.D. Jensen and J.M. Martinis, Physica B, **194-196**, 1255 (1994).
[13] L.J. Geerligs. V.F. Anderegg, C.A. van der Jeugd, J. Romijn and J.E. Mooij, Europhys. Lett, **10**, 79 (1989).
[14] M.I. Lutwyche and Y. Wada, J. Appl. Phys., **75**, 3654 (1994).
[15] M.G. Ancona, submitted to the 3rd International Workshop on Quantum Functional Devices, NIST, Gaithersburg MD (1997).
[16] M.G. Ancona, J. Appl. Phys., **81**, 3311 (1997).
[17] C.S. Lent and P.D. Tougaw, Proc. IEEE, **85**, 541 (1997).
[18] C.S. Lent and P.D.. Tougaw, J. Appl. Phys., **74**, 6227 (1993).

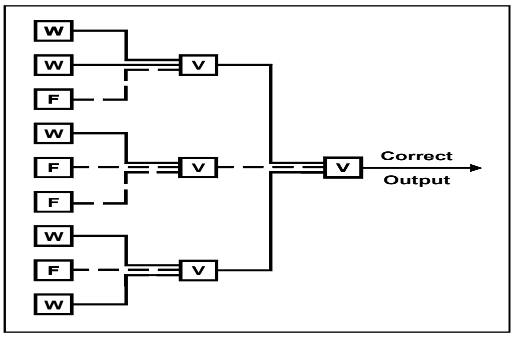

Fig. 1 A 3-stage CTMR unit In the 1st stage there are 9 copies of a device. In the 2nd stage majority voting is performed among triplets of devices of the 1st stage. In the 3rd stage majority voting is performed among the 3 voting units of the 2nd stage. W = working device, F = failing device, V = voting circuit. Wrong outputs are marked with a dashed line.

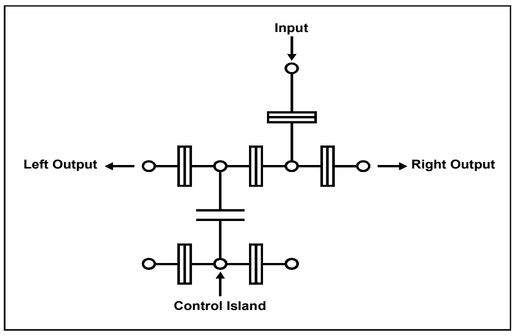

Fig. 2 A single electron switch The input electron is made to sequentially tunnel through the junctions joining the metallic islands. When the control island is free, the input electron turns left. When the control island is occupied by an electron, the input electron turns right. Metallic islands are marked as circles, tunnel junctions as rectangles.

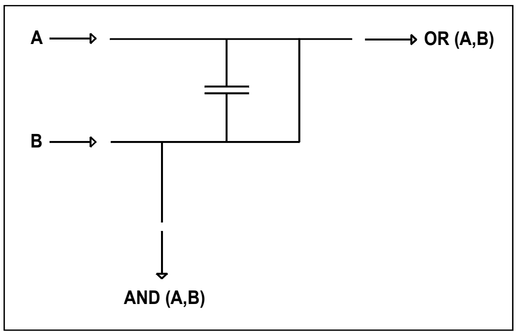

Fig. 3 An AND/OR gate, based on single-electron switching Any electron appearing at input A is driven to the OR (A,B) output. Any electron appearing at input B is either switched to the AND (A,B) output or to the OR (A,B) output, depending on the presence or absence of the electron coming from input A. Driving happens through the principle of electron pumps. Electron pumps in the circuit are represented as solid lines.

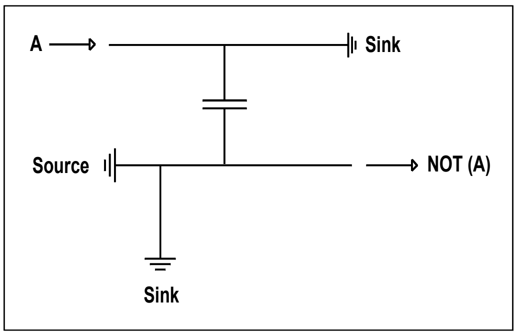

Fig. 4 A NOT gate, based on single electron switching Any electron appearing at input A is driven to a sink. One electron per clock cycle is taken from a source and either switched to a sink or to the NOT (A) output, depending on the presence or absence of the electron coming from input A. Electrons are driven through the principle of electron pumps, which we schematise as solid lines.

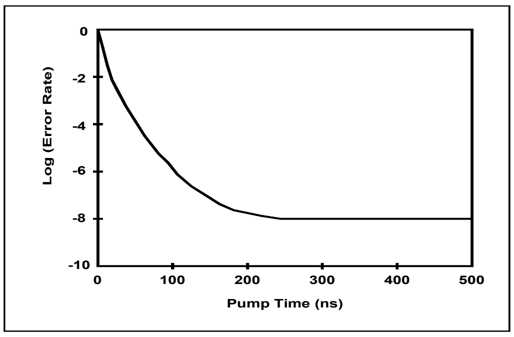

Fig. 5 Error rate vs. pump time for a micron-scale 7-junction electron pump, operating at 35 mK At high frequencies, the error rate is exponentially dependent on pump time. At low frequencies, the error rate approaches an asymptotic value of ~ 10^{-8}.

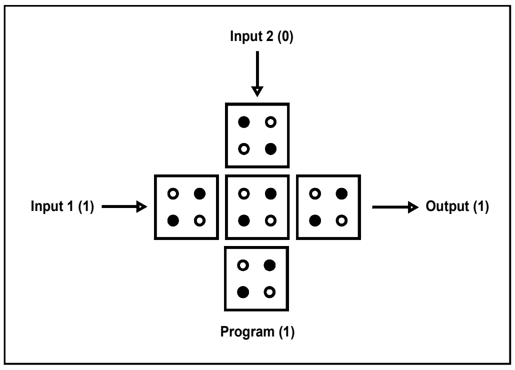

Fig. 6 A programmable AND/OR gate, based on QCAs The central cells performs majority voting among the two input cells and the control cell. Therefore, if the control cell is set to 0 the device works as an AND gate, if 1 as an OR gate. In the example shown the control cell is set to 1, so that we have an OR gate.

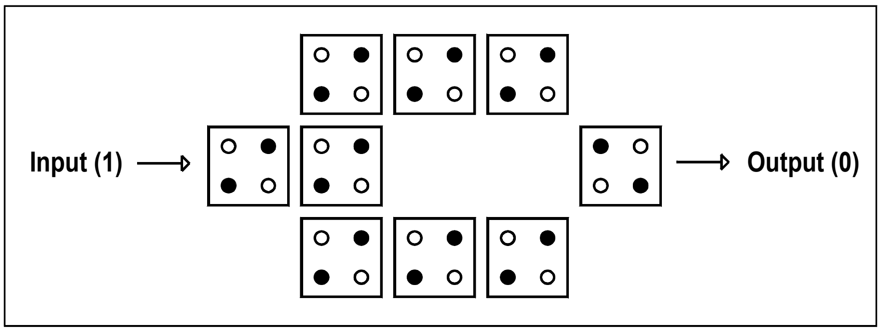

Fig. 7 A NOT gate, based on QCAs The input line extends one cell beyond the beginning of the two circuit branches. The input signal is propagated unaltered through the branches, due to electrostatic repulsion. The two branches, then, converge onto the output line. In this case there is diagonal alignment, so that electrostatic repulsion causes the input signal to be inverted.

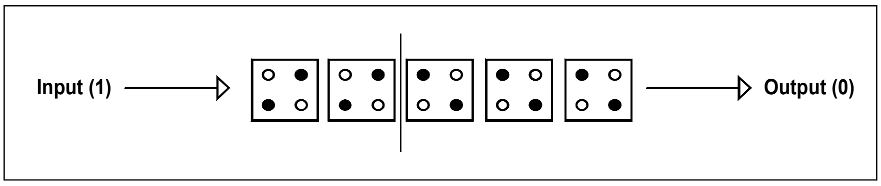

Fig. 8 A kink in a QCA wire Due to thermal fluctuations, the device has absorbed a quantity of energy enough to create a kink in it. The input, which would normally be propagated unaltered through the wire, is then flipped. The wire, then, acts as an inverter and an error arises.

Appendix 2 FAULT RATES IN NANOCHIP DEVICES

S. Spagocci, T. Fountain
University College London
Gower Street, London WC1E 6BT, England

ABSTRACT

This paper addresses the problem of estimating nanochip fault rates, taking intrinsic fault rates and fault-tolerant techniques into account. We considered single-electron device (SED) and quantum cellular automata (QCA) gates. After an analysis of the various fault sources, we surveyed the various space, time and information-redundant strategies available. For logic chips a new space-redundant technique, called cascaded general modular redundancy (CGMR), was proposed. For memory chips the use of error-correcting codes was proposed instead. Both SED and QCA-based chips, containing ~ 10^{11} gates and working at ~ 1 GHz, were considered. Our results suggest that a mean time between failures of ~ 1 y can be guaranteed for logic chips, with a redundancy level of a few tens at worse, either in space or in time. For memory chips, a mean time between failures of ~ 1 y can be guaranteed with a redundancy level of ~ 10 at worse, both in space and in time. This, however, requires carefully tuning the devices' operating conditions and design solutions.

INTRODUCTION

As pointed out in a previous paper [1], the proposed introduction of nanometer-scale components should make it possible to conceive chips containing up to 10^{12} logic gates. This would be particularly interesting for the implementation of parallel systems on a single chip. For such an assembly to work, the introduction of fault-tolerant techniques seems inevitable. The huge number of devices makes the chip unreliable, even if the devices are highly reliable in themselves.

We first considered single-electron devices, of which the electron pump [2] is a prototype. The electron pump is an array of metallic islands, separated by nanometer-scale junctions, through which an electron is made to tunnel sequentially. Logic gates based on the electron pump have been proposed [3], see Figs. (1) and (2). At high frequencies, the main fault source arises from pumping the electron too fast, so that the desired tunneling process is missed [4-6].

We also considered quantum cellular automata [7]. A QCA cell is a square array of four quantum dots, occupied by two electrons. The cell has two stable states, which are taken to mean 0 and 1. Due to electrostatic repulsion, each cell interacts with its neighbors. QCA logic gates based on this principle have been proposed [8], see Figs. (3) and (4). The main fault source arises from thermal excitation [9], which may create a kink in a row of previously aligned cells, thus giving a wrong answer.

In the previously cited work, we analyzed fault rates in SED and QCA-based devices. Extrapolation of the reported electron pump fault rates [4-6] to the nanometer scale led us to predict intrinsic fault rates per clock cycle of ~ 10^{-6}, ~ 10^{-8} and ~ 10^{-10}, for designs based on the 5, 7 and 9-stage electron pump, respectively, and room temperature operation [1]. Similarly, we predicted intrinsic QCA fault rates per clock cycle of ~ 10^{-4} and ~ 10^{-19}, for devices operated at room and liquid Nitrogen temperature (77 K), respectively [1].

We found that "perfect" reliability can be guaranteed for SED and QCA systems of ~ 10^{11} gates working at ~ 1 GHz. This was achieved through the use of a space-redundant technique named cascaded triple modular redundancy (CTMR) [1], see Figs. (6) and (7). The resulting redundancy level is ~ 100.

The work of Ref. [1] has now been extended by surveying the various space, time and information-redundant strategies available [10] and making a distinction between logic and memory chips. We required our fault-tolerant solutions to guarantee a mean time between failures of ~ 1 y at ~ 1 GHz, with a redundancy level $\leq$ 100 in space and/or $\leq$ 10 in time. The proposed techniques and their implications in terms of space and time redundancy are presented in this paper.

1 TWO CHIP MODELS

Somewhat artificially, we assume that memory and logic functions can be implemented in separate units, which we call "chips" for terminological simplicity.

For reasons which will be made clear later, a logic chip will be considered as an aggregate of N_t interconnected logic gates, partitioned into clusters. Each cluster consists of N_g logic gates. Each gate has a faulting probability per clock cycle $P_{f;g}$, with $P_{f;g} << 1$. By a suitable partitioning process, see Fig. (5), we can obtain a linear chain of functional units, each one having a variable number of inputs and outputs.

We further assume that whenever an error occurs within a unit, it will emerge at one or more of its outputs and that errors in the various gates and at various times are statistically independent. The fact that logic gates have a certain degree of intrinsic tolerance to a faulty input can be taken into account by multiplying the gate failing probability per clock cycle by a factor, which turns out to be of ~ 1. Since we deal with order-of-magnitude estimations, such a correction will be ignored in the following. The above-mentioned assumption has the consequence that we only need to worry about one input and one output for each functional unit. Our final model for a chip, then, will simply be that of a linear chain of functional units, each one having one input and one output. Each functional unit is in turn a cluster of N_g logic gates, having a faulting probability per clock cycle $P_{f;g}$.

We assume the system to have a "perfect" clock, which is generated with micrometer-scale technology. We further assume that each cluster generates an answer (or answers) every clock cycle. This implicitly assumes perfect pipelining, which is not always a feasible solution.

Memory chips will be modeled as an array of independent 1-bit memory cells, organized into N_b—bit words. As for logic chips, we suppose that each gate has a faulting probability per clock cycle $P_{f;g}$, with $P_{f;g} << 1$. It can be shown that the overall faulting probability per clock cycle of a memory cell is $3 \bullet P_{f;g}$.

2 SPACE-REDUNDANT TECHNIQUES

In space-redundant techniques [10], each potentially faulty unit is replaced by a number of replicated units so that, by majority voting or other means, faults can be masked to a certain extent. In particular, a line of distinction has to be drawn between permanent and transient faults [10]. A number of space-redundant techniques are available for dealing with permanent faults [11-17]. Such techniques are collectively known as general modular redundancy (GMR) [13] and reconfiguration [17].

The basic idea behind GMR is that the place of a potentially faulty unit is taken by a number of on-line units, among which some kind of a voting process is performed. Whenever an on-line unit is faulty, a spare unit is switched in to replace it. Should spare units run out, the system exhibits graceful degradation until the number of working on-line units falls below a certain level.

In reconfiguration strategies, which can be applied to processor arrays, whenever a processor is faulty its links to the other processors are reconfigured, so as to bypass it. Spare units are required, of course.

As we will see in the next sections, the errors we consider are of a transient nature. To our knowledge, the only available space-redundant techniques for dealing with transient faults are triple modular redundancy (TMR) [10,18,19], see Fig. (6), and its generalization, N-modular redundancy (NMR) [10]. In NMR, the place of each potentially faulty unit is taken by a block of N identical units and majority voting is performed among them. N has to be odd, of course. This works well provided one is dealing with binary units, under the assumption that a fault can only flip the unit's answer and the voting circuitry is perfectly reliable. If the voting circuitry is not perfectly reliable, the formalism has to be suitably modified. It has to be noticed that NMR and TMR are particular cases of GMR, with no spare units and majority voting.

Since a NMR unit performs majority voting, it cannot tolerate ½•(N+1) or more faults. In the limiting case in which the intrinsic failing probability per clock cycle for an element, P_f, is small, the probability per calculation for the NMR unit to fail, $P_{f/nmr}$, is given by:

$$P_{f/nmr} = \begin{pmatrix} N \\ \frac{N+1}{2} \end{pmatrix} \cdot (P_f)^{\frac{N+1}{2}} \tag{1}$$

For the general case of an imperfect voting circuitry with failing probability per clock cycle $P_{f;v}$, we have to consider that the NMR unit gives a wrong answer if either majority voting would give a wrong answer and the voting circuit works, or majority voting would give a right answer and the voting circuitry fails. If, furthermore, we consider a cluster of N_g gates having a failing probability per clock cycle $P_{f;g}$, Eq. (1) can be given a more general form.

In particular, if $P_{f;v} << 1$:

$$P_{f/nmr} = \binom{N}{\frac{N+1}{2}} \cdot (N_g \cdot P_{f;g})^{\frac{N+1}{2}} + P_{f;v} \tag{2}$$

In a previous work, we introduced a generalization of TMR, named cascaded modular redundancy (CTMR) [1]. According to CTMR, the potentially faulty units are first clustered in a suitable way and triple modular redundancy is applied to the clusters. The clusters are then suitably clustered, as well, and triple modular redundancy is applied to each cluster. The process is iterated for a number of steps, as in Fig. (7).

Here we propose a generalization of GMR (and NMR in particular), named cascaded general modular redundancy (CGMR), based on the same principles as CTMR. Generalizing Eq. (2), we suppose that:

$$P_{f/gmr} = \alpha \cdot (N_g \cdot P_{f;g})^{\beta} + P_{f;v} \tag{3}$$

where β is the minimum number of faulty units giving an error in the CGMR output and α is a combinatorial factor.

According to the model of section 1, the chip's failing probability per calculation turns out to be proportional to $(N_g)^{\beta-1}$. In fact, it can be expressed as $N_t \bullet (N_g)^{-1} \bullet P_{f/gmr}$, where $P_{f/gmr}$ is given by the first addendum of Eq. (3) and N_t is the total number of gates in the chip. A consequence of this is that at each CGMR stage the chip's failing probability per calculation is minimized if cluster size is kept as small as possible. Eq. (3), however, shows that below a certain cluster size $P_{f;v}$ dominates. We then require, in Eq. (3):

$$\alpha \cdot (N_g \cdot P_{f;g})^{\beta} \leq \frac{N_v}{\eta} \cdot P_{f;g} \tag{4}$$

where $\eta \approx 10$ and $P_{f;v} = N_v \bullet P_{f;g}$. It follows than that, at any stage, the clusters must have a maximum size given by the following expression:

$$N_{\max} = \sqrt[\beta]{\frac{N_v}{\alpha \cdot \eta \cdot (P_{f;g})^{\beta-1}}} \tag{5}$$

We can now calculate the faulting probability per calculation of a cluster:

$$P_{f/gmr}(i) = \frac{N_t}{(N_{\max})^i} \cdot N_v \cdot P_{f;g} \tag{6}$$

where, using Eq. (3), we have neglected the first addendum. The function defined by Eq. (6) is seen to be monotonically decreasing with i. One therefore needs as many CGMR stages as possible. The iteration process cannot go on indefinitely. The maximum number of stages is defined by the condition:

$$\frac{N_t}{(N_{\max})^i} = 1 \tag{7}$$

Using Eq. (7) in conjunction with Eq. (5), we are now able to define the maximum number of CGMR stages:

$$i_{\max} = [-\beta \cdot \frac{Log(N_t)}{Log(\alpha \cdot \eta \cdot (N_v)^{-1} \cdot (P_{f;g})^{\beta-1})}] + 1 \tag{8}$$

where [.] is the integer part function. We then obtain an expression for the chip's overall failing probability per calculation. Namely:

$$P_{f/gmr} = N_v \cdot P_{f;g} \tag{9}$$

In space-redundant techniques applied to transient faults (which is the case of interest here), we are concerned with cascaded N-modular redundancy (CNMR). The problem is, then, choosing a suitable N.

Eq. (9) shows that the chip's overall failing probability per calculation only depends on N through N_v. A TMR voting unit can be implemented in the following way:

$$MV(x,y,z) = xy + xz + yz \tag{10}$$

where x,y,z are Boolean variables and MV(.) is the majority voting function. A NMR unit can be similarly implemented by summing over all the possible ½•(N+1)-ples of variables. As a consequence, N_v is seen to increase monotonically with N and, according to Eq. (9), the same holds for the chip's overall failing probability per calculation.

On the other hand, under the hypothesis that $P_{f;g} << 1$, the maximum number of CGMR levels has the following asymptotic value:

$$i_{\max} = [-\frac{Log(N_t)}{Log(P_{f;g})}] + 1 \tag{11}$$

which is nearly reached for N = 3 (CTMR). Therefore if for N = 3 the asymptotic value of Eq. (11) is reached and/or the redundancy level, as described by Eq. (14), is satisfactory, CTMR is the best solution.

If needed, it is then possible to apply NMR to the chip's output, using a voting circuitry much more reliable than the chip's gates (micron-scale technology, in practice). The system's failing probability per calculation can then be computed.

In particular:

$$P_{f/nmr} = \binom{N}{\frac{N+1}{2}} \cdot (N_v \cdot P_{f;g})^{\frac{N+1}{2}} \qquad (12)$$

Furthermore, cluster size and the number of nanometer-scale CNMR levels are given by Eqs. (5) and (8), respectively, where:

$$\alpha = \binom{N}{\frac{N+1}{2}} \qquad (13)$$
$$\beta = \frac{N+1}{2}$$

An expression like that of Eq. (12), with N_t in place of N_v, describes the chip's failing probability per calculation for NMR with micrometer-scale circuitry, when no redundancy at the nanometer scale is used. This could seem a straightforward solution for providing fault protection in a nanochip. Unfortunately, the condition $N_t \bullet P_{f;g} << 1$ has to be satisfied, thus limiting the applicability of the technique to gate failing probabilities $P_{f;g} << (N_t)^{-1}$.

Anyway, under the hypothesis that M-modular redundancy is used at the nanometer-scale level and N-modular redundancy is used at the micron-scale level, the time redundancy level is ≈ 1 (since $N_g >> 1$, in fact, the number of clock cycles taken by an error to reach the unit's output is on average much greater than the number of clock cycles taken to process data in the voter) and space redundancy is given by:

$$r_s = M^{i_{\max}} \cdot N \qquad (14)$$

where $i_{\max}$ is defined by Eq. (8).

3 TIME-REDUNDANT TECHNIQUES

In time-redundant techniques [10] processor instructions are suitably repeated, so as to achieve fault tolerance. In particular one can either mask errors [20] or detect them and restart processor operation from the previous state (backward error recovery [21]). Instructions can be repeated at any level, from single-bit level to software (software redundancy [10]). It would be difficult, at this stage of development, to give any sensible estimations on the effects of software redundancy. Therefore, we will only consider instructions at the single-bit level. Any software instruction is finally translated into bit exchanges within the processor. Our approach then can implicitly give indications on the possible effectiveness of software-redundant strategies.

An application of error-masking is suggested in Ref. [20] in the form of a voting unit accumulating results, so as to perform TMR in the time domain. We propose to extend such a technique to CNMR. A voter unit, able to accumulate input data over N+1 clock cycles (through a shift register) and then perform voting upon them (driven by a counter), is proposed in Fig. (8). If Fig. (7) is read as a time-diagram, we have an idea of how to implement CNMR in the time domain. The clock frequency of a unit at level i has to be de-multiplied by a factor $(N+1)^i$ with respect to the master clock frequency. Note that in Fig. (8) the counter is assumed to be perfect. This can most easily implemented by supplying suitable signals generated from the master clock.

The same results as those given for CNMR in the space domain apply, except that the roles of space and time redundancies are exchanged. Micrometer-level redundancy can be added, if needed. In particular, cluster size and the number of CNMR levels are given by Eqs. (5) and (8), respectively, where α and β are defined by Eq. (13). The failing probability per calculation is given by Eqs. (9) and (12), where $N_v \approx 9$ for CTMR. Finally, space redundancy is ≈ 1 and time redundancy is given by an expression like Eq. (14).

In order to perform backward error recovery (BER), we propose the scheme shown in Fig. (9). The outputs of N copies of suitable functional units (clusters of N_g gates, as suggested in section 1) are compared through a logical OR between all the possible terms of the form $x_i \oplus x_j$. A disagreement signal from any of the N-ples of replicated units causes the system to step back to the previous state. For duplication, the task of implementing an error-detection function is straightforward. In fact, we can define:

$$ED(x,y) = x \oplus y \tag{15}$$

For triplication, on the other hand, Eq. (15) generalizes to:

$$ED(x,y,z) = x \oplus y + x \oplus z + y \oplus z \tag{16}$$

Eqs. (15) and (16) can be easily generalized, but we won't give a general formula here. An arrangement like that of Fig. (9) is able to detect faults occurring in the units, unless all the N units fail. The fault rate per calculation for N replicated N_g-gate units is then:

$$P_{f/ber} = (N_g \cdot P_{f;g})^N + P_{f;v} \tag{17}$$

Once again, we can cascade the BER units and apply the CGMR formalism of section 2. Analogously to section 2, duplicated units should usually be the most effective solution.

One further micron-scale level can be added, if necessary. More specifically, cluster size and the number of BER levels are given by Eqs. (5) and (8), respectively, where:

$$\begin{aligned} \alpha &= 1 \\ \beta &= N \end{aligned} \tag{18}$$

Furthermore, the failing probability per calculation is given by Eqs. (9) and (12), where $N_v = 5$ for duplication. If BER has to make any sense, the intrinsic probability per calculation for a fault to occur anywhere in the chip must be << 1, because otherwise there would be no point in repeating processor operation. In other words, we must have $P_{f;g} << (N_t)^{-1}$. Time redundancy is then ≈ 1. On the other hand, space redundancy is given by Eq. (14).

It is interesting to notice that BER, although usually defined as a time-redundant technique, in our cascaded version implies a non-negligible amount of space redundancy. Alternatively, the comparison process can be performed by accumulating results in time. An arrangement similar to the TMR unit of Fig. (8) has to be employed. In this case, we have $N_v \approx 13$ and the roles of space and time redundancies are exchanged.

4 INFORMATION-REDUNDANT TECHNIQUES

In information-redundant strategies [10] the information stored in a chip is made redundant through the use of error correcting codes [22]. In an error correcting code, suitable parity check bits are added to the information bits one wants to protect, so that errors can be located and/or corrected. An error-correcting code is characterized by the maximum number of errors it can detect and/or correct and its redundancy, i.e. the ratio between the total number of bits in a word and the number of information bits.

What has just been said suggests that error-correcting codes can be applied to memory chips, for which the above-mentioned clustering process would not be feasible. We will adopt the model suggested in section 1 for a memory chip. Namely, we will consider it as an array of independent memory cells, organized into memory words. We assume that every memory cell is refreshed at each clock cycle, with a finite probability of failure. We then suppose that a certain error-correcting code has the capability to correct N_e errors. The data word is written to the memory and stays there for N_s clock cycles. Once a bit error has been generated, the memory cell's feedback loop makes it permanent.

If the memory cell's fault rate per clock cycle, $P_{f;m}$, satisfies the relationship $P_{f;m} << 1$, we can neglect the process through which a bit error can be flipped back to its correct value. The probability that a N_b-bit memory word (where N_b includes the contribution of check bits) is corrupted by a number of errors $\geq N_e+1$, after being stored for N_s clock cycles, is then given by:

$$P_{f/ecc} = \binom{N_b}{N_e+1} \cdot (N_s \cdot P_{f;m})^{N_e+1} \qquad (19)$$

where, as underlined in section 1, it can be shown that $P_{f;m} = 3 \bullet P_{f;g}$. Eq. (19) requires that $N_s \bullet P_{f;m} << 1$, thus limiting the applicability of this technique to failing probabilities such that $P_{f;m} << (N_s)^{-1}$. Whenever possible, it would be advisable to choose N_s as the number of clock cycles corresponding to the mean time between failures we want to guarantee. Otherwise a storage time has to be chosen and periodic correcting and re-encoding operations have to be performed. In an array of ~ 10^{11} logic gates (and ~ 10^{10} memory cells), one has $N_d \approx 10^{10} \bullet (N_b)^{-1}$ data words. Then the storage time (in clock cycles) N_s cannot be $\leq N_d$, since this is the minimum number of clock cycles between two correcting and re-encoding operations on the same data word, if those operations are performed sequentially. This is a further constraint on the applicability of Eq. (19).

The space redundancy level depends on the chosen code. On the other hand, time redundancy can be a potentially serious problem. However, if encoding and decoding are performed through a lookup table operation [22] the problem can be overcome. More specifically, when a data word has to be written to memory it is first fed to a pre-computed lookup table, which in turn produces an encoded input. The encoded input in then fed to the memory and stored. When a data word has to be read from memory, the encoded form of it is first corrected by replacing it with the nearest neighbor in terms of Hamming distance [22], then it is fed to a pre-computed lookup table, which provides decoding. The lookup tables and correcting circuitry have to be much more reliable (e.g. built with micrometer-scale technology) than the memory they are meant to protect.

For memory-intensive operations, time redundancy is given by:

$$r_t = 2.5 + 5 \cdot N_c \tag{20}$$

where N_C is the number of data word correction and re-encoding operations per computation, under the hypothesis that access to both the lookup tables and memory and Hamming error correction take the same amount of time.

There are many different kinds of error-correcting codes. We suggest the use of two quite different codes. Hamming codes [23] are characterized by a low redundancy (1.5 for an 8-bit information word) and a low error-correcting capability (1 error). Reed-Muller codes [22], on the other hand, are characterized by quite a high redundancy (16 for an 8-bit information word) and a high error-correcting capability (31 errors for an 8-bit information word).

The choice of an 8-bit information word for both the Hamming and the Reed-Muller code requires some explanation. Our encoding/decoding approach implies a lookup table dimension of 2^{Nib}, where N_{ib} is information word size. If we assume a ratio of ~ 10^4 between the typical areas of our devices and the conventional micrometer-scale devices (see section 6), we find that the lookup table cannot contain more than ~ 10^6 memory cells, if its area has to be at worst of the same order as the area occupied by the nanoscale devices. This limits information word size to less than 20 bits. Keeping code redundancy into account, both the Hamming and the Reed-Muller codes require 8-bit information words. Any longer information word can be split into N_{bl} smaller blocks, which are separately encoded and decoded. The overall failure probability for a storage time of N_s clock cycles is then:

$$P_{f/ecc} = N_{bl} \cdot \binom{N_b}{N_e + 1} \cdot (N_s \cdot P_{f;m})^{N_e + 1} \tag{21}$$

so that a negligible price is paid (in order-of-magnitude terms) for information word sizes bigger than the chosen standard size.

5 FAULT RATES IN NANOCHIPS

As pointed out in the previous sections, logic chips require the use of either space or time-redundant strategies (see sections 2 and 3). On the other hand, error-correcting codes are suitable for memory chips (see section 4). We required a mean time between failures of ~ 1 y at ~ 1 GHz for ~ 10^{11} gates, with a redundancy level $\leq$ 100 in space and/or $\leq$ 10 in time (see section 1).

We present here the results obtained for both SED and QCA-based chips. In the rest of the section, by n-junction pump we mean a gate whose design is based on the n-junction electron pump [1], operated at room temperature. Furthermore, by r_s and r_t we mean space and time redundancy, respectively.

5.1 LOGIC CHIPS

We list here the alternative solutions found for logic chips. In the list that follows, the expression "x(n)+1(m) stages" is used to describe a CGMR arrangement consisting of x cascaded stages, with n-fold replication (so, for example, n = 3 describes TMR), plus one stage with micrometer-scale voting circuitry and m-fold replication. When a 0 is used instead of x(n) (1(m), respectively), we mean that the proposed system has only micrometer-scale stages (nanometer-scale stages, respectively). The figures have been obtained by referring to the appropriate formulae of sections 2 and 3.

SED-based chips:

- 9-junction pump, CNMR in the space domain, 3(3)+1(3) stages, $r_s = 81$, $r_t = 1$
- 5-junction pump, BER in the space domain, 4(2)+1(4) stages, $r_s = 64$, $r_t = 1$
- 7-junction pump, BER in the space domain, 3(2)+1(3) stages, $r_s = 24$, $r_t = 1$
- 9-junction pump, BER in the space domain, 3(2)+1(2) stages, $r_s = 16$, $r_t = 1$

QCA-based chips:

- operated at 77 K, CNMR in the space domain, 3(2)+0 stages, $r_s = 8$, $r_t = 1$
- operated at 77 K, CNMR in the space domain, 0+1(5) stages, $r_s = 5$, $r_t = 1$
- operated at 77 K, BER in the space domain, 2(2)+0 stages, $r_s = 4$, $r_t = 1$
- operated at 77 K, CNMR in the time domain, 0+1(5) stages, $r_s = 1$, $r_t = 6$
- operated at 77 K, BER in the time domain, 2(2)+0 stages, $r_s = 1$, $r_t = 9$

5.2 MEMORY CHIPS

We list here the alternative solutions found for memory chips. The figures have been obtained by referring to the appropriate formulae of section 4. For Reed-Muller codes, a storage time of ~ 10^8 clock cycles has been used. Since this is the minimum allowable storage time for 10^{11} gates (see section 4), we have to correct and re-encode a memory word every two clock cycles. Therefore, $N_c = 1$ in Eq. (20). For Hamming codes we assume $N_c = 0$. The results are quoted for a 16-bit information word. As shown in section 4, the information word has to be split into two independent 8-bit blocks.

SED-based chips:

- 9-junction pump, Reed-Muller code, $r_s = 16$, $r_t = 7.5$

QCA-based chips:

- operated at 77K, Hamming code, $r_s = 1.5$, $r_t = 2.5$
- operated at 77K, Reed-Muller code, $r_s = 16$, $r_t = 7.5$

6 CONCLUSIONS

In this work, we addressed the problem of estimating nanochip fault rates. We considered SED and QCA-based logic gates and surveyed the various space, time and information-redundant strategies available. We proposed a new fault-tolerant technique, named cascaded general modular redundancy (CGMR) and we were able to adapt other standard techniques to our requirements. By using the figures for SED and QCA intrinsic fault rates given in Ref. [1], we were able to propose a number of fault-tolerant solutions. In particular, we had to distinguish between logic and memory chips. We found that a mean time between failures of ~ 1 y at ~ 1 GHz can be guaranteed (and in some cases vastly exceeded) for logic chips, with a redundancy level of a few tens at worse, either in space or in time. For memory chips, a mean time between failures of ~ 1 y at ~ 1 GHz can be guaranteed (and in some cases vastly exceeded) with a redundancy level of ~ 10 at worse, both in space and in time. However, a careful tuning of the devices' operating conditions and design solutions is required. For both logic and memory chips, QCA-based devices can work with lower redundancy levels than SED-based chips. However, they cannot be operated at room temperature.

The intrinsic error rates we analyzed are of a transient nature. Impurities and dislocations, however, could cause permanents faults, which have not yet been considered in our analysis. The fact that the typical dimensions of the devices we considered are ~ 10 nm, so that their area is ~ 10^4 times smaller than present-day devices [1] suggests that single crystal errors will have more significant effects than in microcircuitry. On the other hand, the level of redundancy required by the transient gate fault rates might provide protection against permanent faults, as well. The effects of cosmic rays and natural radioactivity have to be assessed, too. Once again, the level of redundancy required by the intrinsic fault rates might provide protection against cosmic rays and natural radioactivity, too.

As pointed out in Ref. [1], despite the somewhat high level of redundancy we have to provide, the linear dimensions of the nanochips considered in this paper should be of ~ 1 cm and power dissipation should be of ~ 1 W, which seems quite realistic. This is also an a fortiori justification for the choice of our space and time redundancy constraints. However, it is not the purpose of our work to give detailed design solutions. Throughout the paper, a number of simplifying assumption have been made, mainly based on the fact that the intrinsic failing probabilities for both SED and QCA-based gates are very small. These assumptions will be checked through numerical simulations.

ACNOWLEDGEMENT

This paper and associated research were funded as part of the DARPA ULTRA program. Grant number N00014-96-1-0850. Principal investigator: T J Fountain.

REFERENCES

[1] S. Spagocci and T. Fountain, in Proceedings of ECS 98-19, Quantum Confinement V: Nanostructures, M. Cahay, D.J. Lockwood, J.P. Leburton and S. Bandyopadhyay, Eds., pp 582-596, The Electrochemical Society, Pennington, NJ (1999).
[2] H. Pothier, P. Lafarge, C. Urbina, D. Esteve and M.H. Devoret, Europhys. Lett., **17**, 249 (1992).
[3] M.G. Ancona, J. Appl. Phys., **79**, 526 (1996).
[4] H.D. Jensen and J.M. Martinis, Phys. Rev. B, **46**, 13407 (1992).
[5] J.M. Martinis, M. Nahum and H.D. Jensen, Phys. Rev. Lett., **72**, 904 (1994).
[6] M.W. Keller, J.M. Martinis, N.M. Zimmermann and A.H. Steinbach, Appl. Phys. Lett., **69**, 1804 (1996).
[7] C.S. Lent, P.D. Tougaw, W. Porod and G.H. Bernstein, Nanotechnology, **4**, 49 (1993).
[8] P.D. Tougaw and C.S. Lent, J. Appl. Phys., **75**, 1818 (1994).
[9] C.S. Lent, P.D. Tougaw and W. Porod, in PhysComp '94: Proceedings of the Workshop on Physics and Computing, pp. 1-9, IEEE Computer Society Press, Los Alamitos, CA (1994).
[10] P.G. Depledge, IEE Proc. A, **128**, n. 4, 257 (1981).
[11] F.P. Matur, IEEE Trans. Comp., **C 20**, 1376 (1971).
[12] F.P Matur and P.T. de Sousa, IEEE Trans. Rel., **R 24**, n. 2, 108 (1975).
[13] F.P Matur and P.T. de Sousa, IEEE Trans. Rel., **R 24**, n. 5, 296 (1975).
[14] J. Losq, IEEE Trans. Comp., **C 25**, n. 6, 569 (1976).
[15] F.P Matur and P.T. de Sousa, IEEE Trans. Comp., **C 27**, n. 7, 624 (1978).
[16] H.Y. Lo, L.P. Ju and C.C. Su, IEE Proc. G, **137**, n. 1, 1 (1990).
[17] F. Distante, M.G. Sami and R. Stefanelli, in: V. Cantoni, L. Lombardi, M. Mosconi, M. Savini and A. Setti, Eds., Proc. CAMP 1995, pp. 340-349, IEEE Computer Society Press, Los Alamitos, CA (1995).
[18] J.F. Wakerly, Proc. IEEE, **64**, n. 6. 889 (1976).
[19] K.W. Philp and N.D. Deans, Microelectron. Reliab., **37**, n. 4, 581 (1996).
[20] M.G. Ancona, Superlattices and Nanostructures, **20**, n.4, 461 (1996).
[21] H. Schepers, in: J. Vytopil, Ed., Formal Techniques in Real-Time and Fault-Tolerant Systems, pp. 3-31, Kluwer Academic Publishers, Boston, MA (1993).
[22] S. Roman, Introduction to Coding and Information Theory, Springer-Verlag, New York, NY (1997).
[23] R.W. Hamming, Bell Syst. Tech. J., **26**, n. 2, 147 (1950).

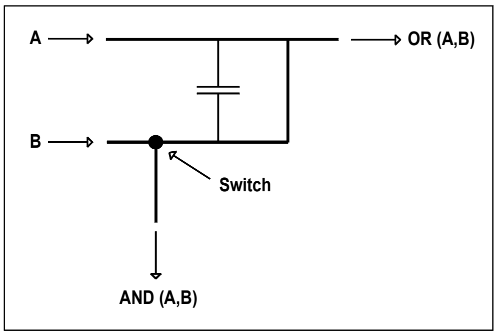

Fig. 1 An AND/OR gate, based on single-electron switching. Any electron appearing at input A is driven to the OR (A,B) output. Any electron appearing at input B is either switched to the AND (A,B) output or to the OR (A,B) output, depending on the presence or absence of the electron coming from input A. Driving happens through the electron pump mechanism. Electron pumps in the circuit are represented as thick lines.

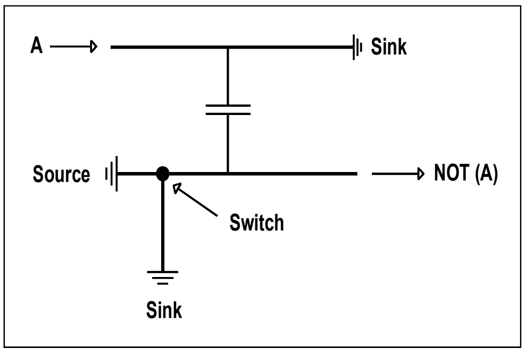

Fig. 2 A NOT gate, based on single electron switching. Any electron appearing at input A is driven to a sink. One electron per clock cycle is taken from a source and either switched to a sink or to the NOT (A) output, depending on the presence or absence of the electron coming from input A. Electron pumps in the circuit are schematized as thick lines.

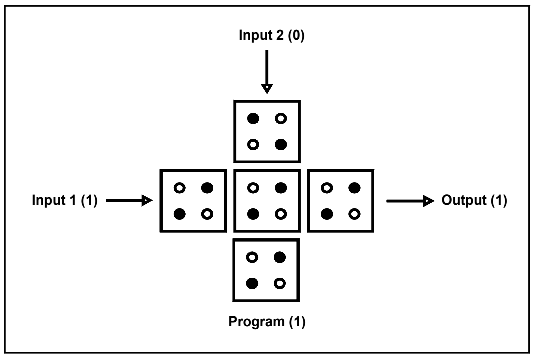

Fig. 3 A programmable AND/OR gate, based on QCAs. The central cell performs majority voting among the two input cells and the control (program) cell. Depending upon whether the control cell is set to 0 or to 1, the device works as an AND gate or as an OR gate. In the example shown the control cell is set to 1, so that we have an OR gate. Quantum dots are represented as circles. Filled circles indicate that a quantum dot is filled with an electron.

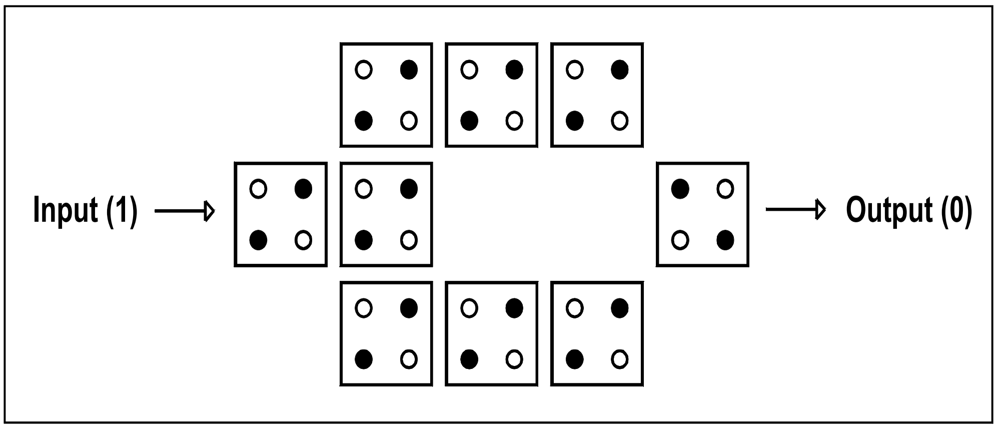

Fig. 4 A NOT gate, based on QCAs. The input line extends one cell beyond the beginning of the two circuit branches. The input signal is propagated unaltered through the branches, due to electrostatic repulsion. The two branches then converge onto the output line. In this case there is diagonal alignment, so that electrostatic repulsion causes the input signal to be inverted. Quantum dots are represented as circles. Filled circles indicate that a quantum dot is filled with an electron.

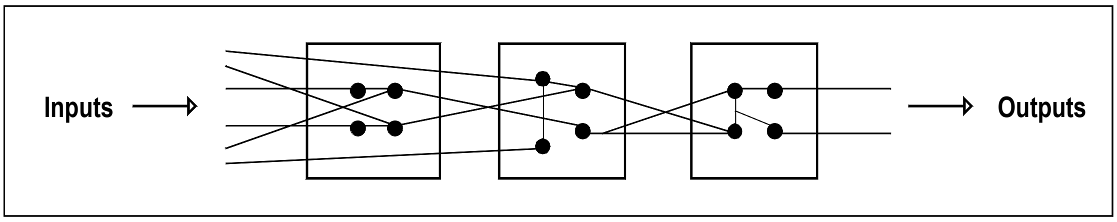

Fig. 5 A model for a logic chip. A logic chip is represented here as a set of interconnected gates. The set is suitably partitioned, so that it can be considered as a linear array of functional units. The functional units have a variable number of inputs and outputs. Gates are represented by filled circles, functional units by squares.

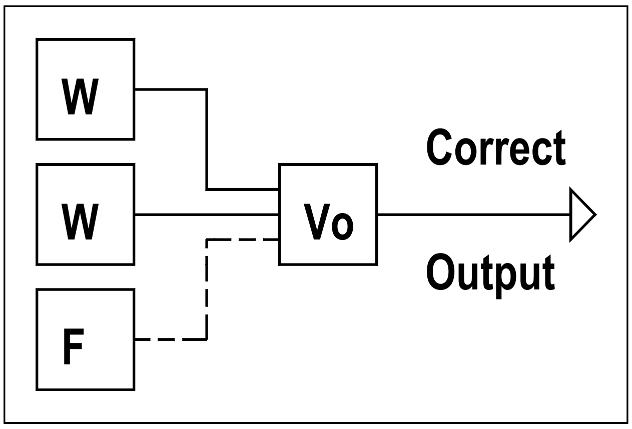

Fig. 6 A TMR unit. Three copies of the potentially faulty devices send their output to a unit, which performs majority voting among them. The answer is taken to be the correct output. W = working device, F = failing device, Vo = voting circuit. Wrong outputs are marked with a dashed line

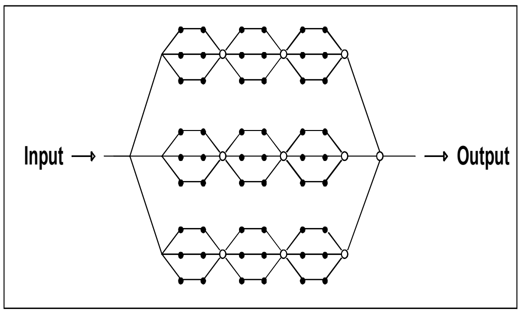

Fig. 7 A 2-stage CTMR arrangement. A linear array of six gates is partitioned into clusters of two gates. In the 1st stage, each cluster is tripled and majority voting is performed on each triplet. In the 2nd stage, the linear chain thus obtained is in turn tripled and majority voting is performed on the triplet. Gates are represented by filled circles, voting units by empty circles. The reasons behind the clustering process are explained in the text.

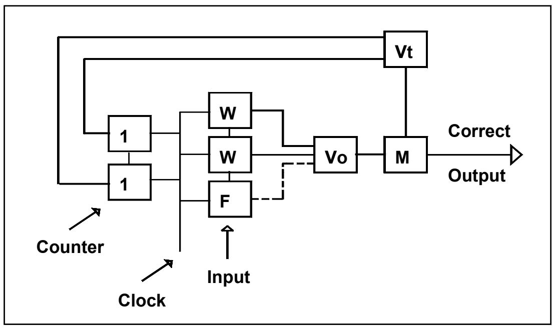

Fig. 8 A voting unit for time domain TMR. The input stream is accumulated in the shift register. The three stages of the shift register send their output to a majority voting unit. The answer of the voting unit is sent to a memory cell. A veto unit (an AND gate) prevents the memory cell from storing the input data coming from the voter unless the counter (as in the picture) gives a "11" output. This happens every four clock cycles. W = data coming from a working device, F = data coming from a failing device, Vo = voting circuit, Vt = veto unit, M = memory cell. Wrong outputs are marked with a dashed line.

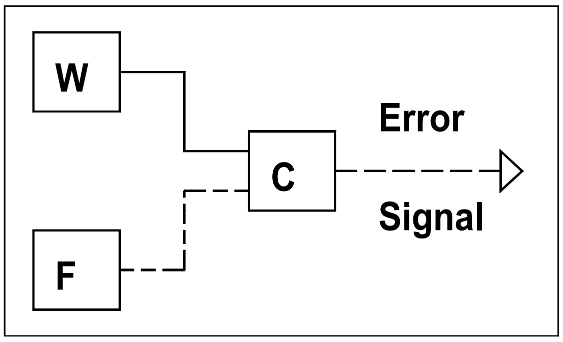

Fig. 9 A BER unit with duplication. Two copies of the potentially faulty devices send their output to a comparator (an XOR gate). The comparator detects any disagreement between the two outputs and emits an error signal. If the error signal in any of the units in the system is 1, the system steps back to its previous state. W = working device, F = failing device, C = comparator. Wrong outputs are marked with a dashed line.

Appendix 3 Parallel Computing

A3.1 Introduction

One of the most promising applications of nanoelectronics is the single-chip implementation of massively parallel computers [1]. Although the present state of nanoelectronic research, and time constraints, prevented us from delving into detailed architectural issues, it is interesting to present, at least as an appendix, a brief overview of parallel computing. A more complete review is given in [1,2]. We notice that neural networks might be considered as parallel computers, as well. The reader is referred to e.g. Ref. [3] for details.

A3.2 Parallel Computer Classification

Flynn's classification of parallel systems [4] is based on the nature of their input streams and devices. In particular, a distinction is drawn between single instruction (SI) and multiple instruction (MI) streams and single device (SD) and multiple device (MD) systems. In particular:

- SISD systems are the ordinary, sequential computers.
- SIMD systems are the data parallel computers we describe later.
- MISD systems are shown by Flynn not to exist [4].
- MIMD systems are the function parallel computers we describe later.

Flynn's classification has the advantage of being concise and clear. However, it hides important details of architecture and functionality. A more complete classification, presented in the following, has also been proposed [1,2]. For each category of parallel computers, Ref. [1] chooses a system and describes it in a short contribution, written by the relevant designers. We refer the reader to such contributions for details.

A3.2.1 Function Parallel Computers

In function parallel computers [2], processors perform different functions, so that parallelism comes from distributing the computer's workload among the different processors, which operate simultaneously. The paradigms of function parallel computing are: pipeline, superscalar and VLIW, graph reduction and MIMD.

A3.2.1.1 Pipelines and Systolic Arrays

In order to explain the basic principles of pipelining, we refer to a concrete example [1]. Let us then consider the function $f(x,y)=\sqrt{[2\cdot(x+y)]}$. Fig. (A3.1) shows how different processing elements might be arranged to perform this calculation. If we had a stream of input pairs and wanted to treat them sequentially, each pair would take five steps. However, we can arrange for each unit to start treating the i th pair as soon as it has finished the (i-1) th one. After a transient, all the units are busy at any time and the array produces a result each clock cycle. See Fig. (A3.6) for systolic arrays.

An example of a configurable pipeline for image processing is the Datacube [5] (up to 16 bus-connected elements). Pipelines appeared at the end of the 1960s as an effective number-crunching technique in the first supercomputers [2]. At the beginning of the 1970s, they were used in the first vector processors [2]. In the 1970s, pipelining gained momentum as an instruction processing technique in mainframes [2]. From the beginning of the 1980s it has been used in microprocessors and now it is the standard instruction processing technique, used in the functional units of processors. A well-known example is the Intel Pentium [2].

A3.2.1.2 Superscalar and VLIW Architectures

In superscalar and very long instruction word (VLIW) architectures [1], shown in Fig. (A3.2), the burden of executing instructions is shared among different execution units working in parallel. In superscalar architectures, each unit is supplied with an instruction stream having an ordinary length. In VLIW architectures, a single and very long instruction stream (up to 512 bits) is issued for all the execution units.

VLIW architectures lead to a less complex design than superscalar architectures. However, they have drawbacks, because compilers targeted to a specific VLIW architecture are not suitable for other architectures. There is, in fact, correlation between the structure of an instruction and the architecture it acts on.

The idea of superscalar architectures originated in the 1970s [6] and was better reformulated in the 1980s [7]. Prototype systems include those developed in the 1980s by IBM [8] and DEC [9]. An example of a commercial system is the Intel PentiumPro [2]. VLIW architectures appeared as early as 1975, although they received their present name in 1983 [10]. The main VLIW prototype systems, produced in the 1980s, are described in Refs. [10,11]. An example of the commercial developments which followed is Trace200 (256 bit word), by Multiflow Computers [2].

A3.2.1.3 Graph Reduction Architectures

The graph reduction paradigm can again be described by a specific example [1]. Let us then consider the function $f(x,y,z)=\sqrt{[(x+y)^3-(x+z)^2]/(xy)}$. Once the triplet (x,y,z) is fixed, f(x,y,z) can be calculated by propagating the input values inside the graph, as in Fig. (A3.3). One can arrange for each processing element (a graph node) to start processing its inputs as soon as they become available. The system is somewhat similar to a pipeline, except that in pipelines information propagates linearly, which is not necessarily the case here.

Examples of graph reduction systems include the GRIP machine [12], a bus-connected array of 128 processors, the Meiko system [13] (implemented on a MIMD mesh), the ALICE graph reduction system [14] and the Manchester Dataflow machine [15], implemented on a switching network. See below for details on the above-mentioned connection arrangements.

A3.2.1.4 MIMD Architectures

In the MIMD (Multiple Instruction Multiple Data) paradigm [1], an array of processors simultaneously performs different functions on different data. The structure of a typical MIMD system is shown in Fig. (A3.4). The workload is distributed among the different processors and, if the tasks are properly allocated, a considerable reduction in completion time is achieved. Typically, the processors in an MIMD system (like e.g. the well known transputer [2]) are conventional CMOS microprocessors, optimized for inter-processor communication.

MIMD systems may have distributed or shared memory. In distributed memory systems, each processor has its own memory. In shared memory systems, memory is shared. Distributed memory systems pose less severe synchronization problems and are more scalable. However, workload distribution is more critical.

In the early 1980s, MIMD systems were incorporating tens of processors. Systems with several hundred processors were common in the mid-1980s. Nowadays, such systems include 10000 or more processor. Distributed memory MIMD systems include the nCUBE [16], the MIT J-machine [17] the Intel Paragon [2] and IBM SP2 [2]. Shared memory MIMD systems include the Cray T3D [18], the Wisconsin multicube [19], the Stanford DASH [20] and FLASH [21].

A3.2.2 Data Parallel Computers

In data parallel computers [2], each processor performs the same function, so that the parallelism comes from simultaneously treating different data. The paradigms of data parallel computing can be classified as associative and SIMD. Vectorization, a data parallel approach, has quite peculiar features [2] and will be treated separately.

A3.2.2.1 Associative Architectures

In associative paradigm, data are processed in parallel according to their content. A specific example [2] is useful here. Let us then consider a parallel spell checker, as in Fig. (A3.5). A linear array of w processors (where w is the maximum length of a word in English) compares the given word to each word in the system dictionary. Each processor takes care of a letter. An AND between the processor outputs signals a match with a dictionary entry. After a transient, the gain in processing time is given by w.

Only a few associative systems have been built up to now. One example is ASP (Associative String Processor) [22], researched at Brunel University and developed at Aspex Microsystems. ASP processors were the bases of the Trax machines [22]. Their wafer-scale version was WASP [22], a linear array of up to 256k processors. The main application was track analysis in particle physics experiments.

A3.2.2.2 SIMD Architectures

In SIMD (Single Instruction Multiple Data) arrays [2], a number of processors, incorporating local memory, perform the same instruction at the same time on different input data. An example might be a low level vision operation like image averaging, where each processor takes care of a pixel. SIMD arrays are usually arranged according to a square mesh and the processors are quite simple (typically 1 bit elements), while their number is maximized. Image processing is not the only application for SIMD computers. Scientific computation offers many problems suitable for the SIMD approach (e.g. finite element or quantum mechanical simulations) [2].

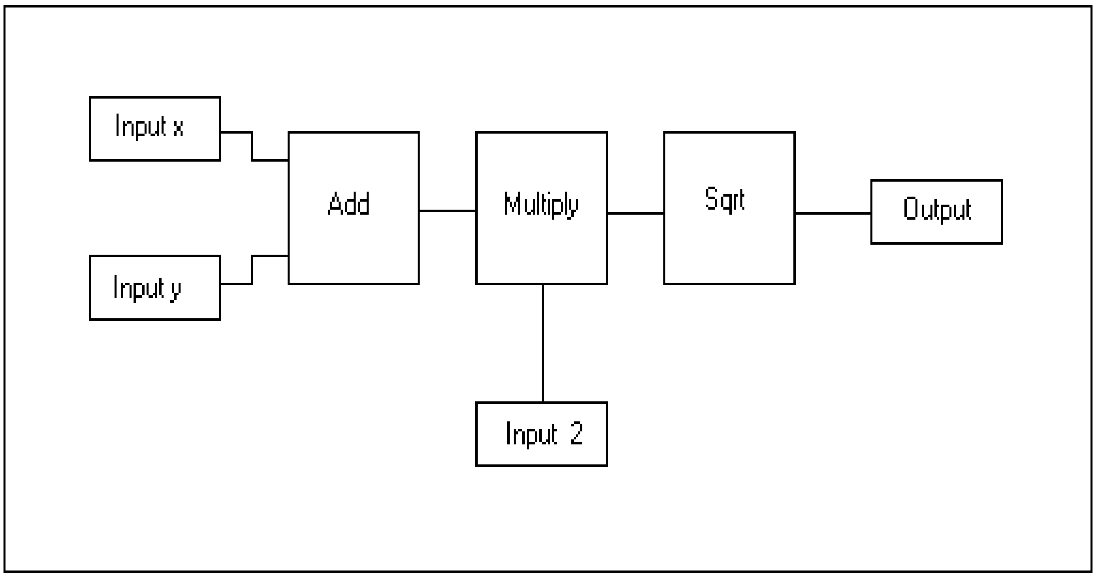

Fig. A3.1 A pipeline. The pipeline shown implements the function $f(x,y)=\sqrt{[2\cdot(x+y)]}$. The input pairs are processed sequentially. However, as soon as a unit has processed the data from the previous unit and deriving from the ith input, it processes the data from the previous unit and deriving from the (i+1)th input. Redrawn from [1].

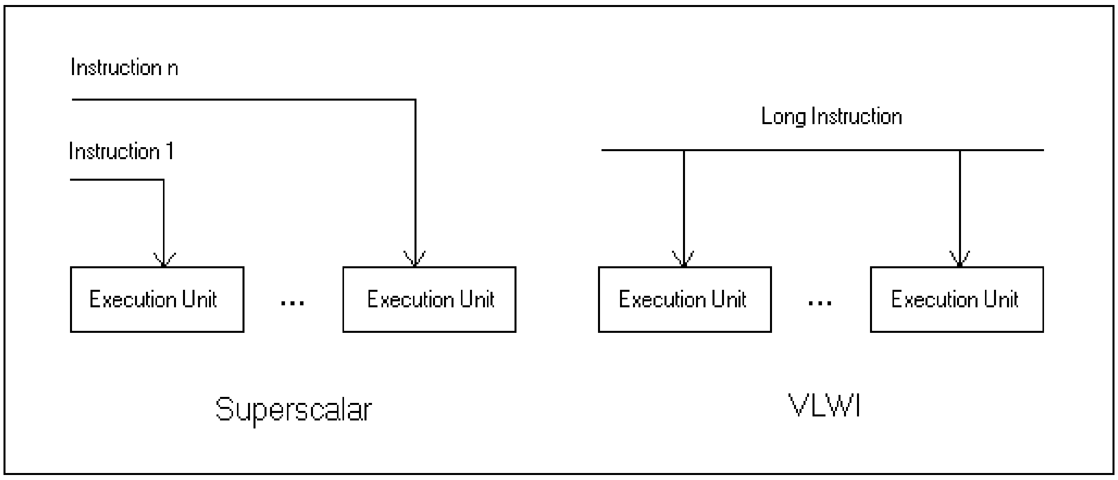

Fig. A3.2 Superscalar and VLIW architectures. The instructions are executed by a number of units working in parallel. However, in superscalar architectures the units receive different instructions. In VLIW architectures, a single and very long instruction is issued for all the units. Redrawn from [2].

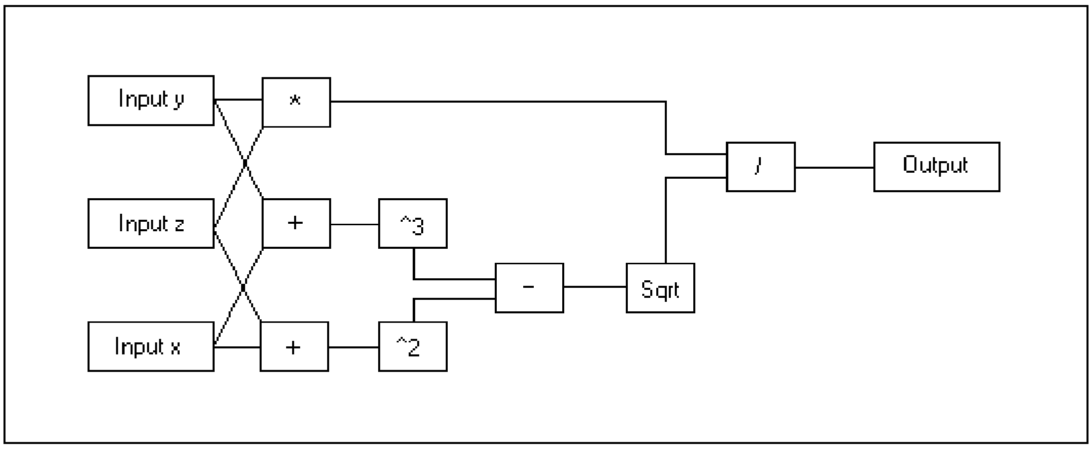

Fig. A3.3 A computational graph. The computational graph shown implements the function f(x,y,z)=√ [(x+y)3-(x+z)2]/(xy). The layout is similar to that of a pipeline, except that each graph node starts processing its inputs as soon as they become available. Redrawn from [1].

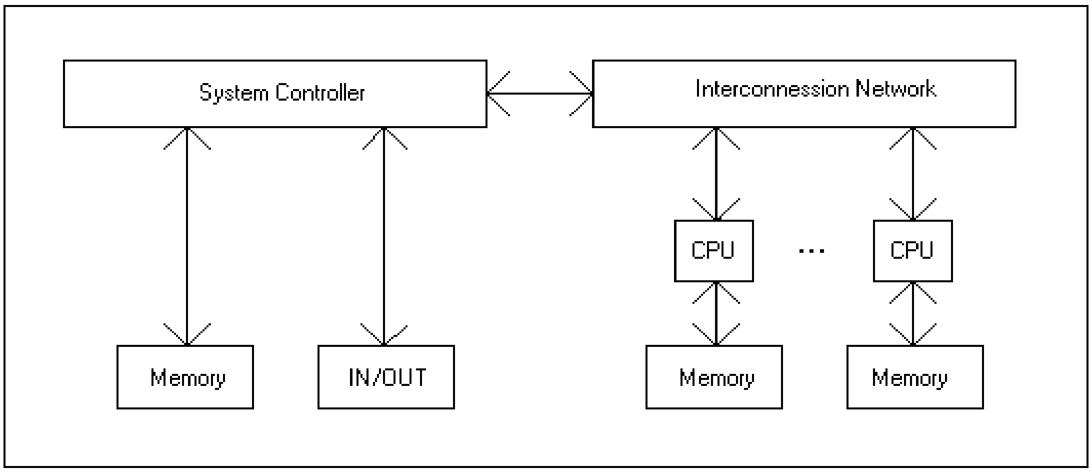

Fig. A3.4 An MIMD array. A number of interconnected processors perform different tasks on different input data at the same time, hence the acronym Multiple Instruction Multiple Data. Redrawn from [1].

The earliest SIMD systems (1960s) were developed in the Iliac program [23,24] (Illinois University) which led to a 32x32 array [24]. The CLIP (Cellular Logic Image Processing) program at UCL led to CLIP4 [25], a 96x96 array. Another important system was DAP (Distributed Array Processor) [26], a 32x32 array developed at Imperial College early in the 1970s. Perhaps the most famous SIMD system is the Connection Machine [27], a hypercube array mainly devoted to scientific computation. The first SIMD system to enter the commercial field was MasPar [28], an array of up to 16k elements. A number of systems by IBM, Cray and DEC then followed [2].

A3.3 Vectorization and Supercomputers

Old fashioned supercomputers, like the famous Crays, operated on vectors [2]. One can understand the vectorization paradigm [2] by observing that the most relevant time burden in computation is often given by the addressing operations. By vectorization, n scalar addressings can be turned into one vector addressing, with a gain of a factor n. Of course, the computing time does not vary, since a vector operation is the juxtaposition of n scalar operations. However, pipelining can be applied to vector operations, so that there can be a gain of a factor n in computing time, as well.

Apart from vectorization, old fashioned supercomputers made use of two other computing paradigms. Vector operations were in fact pipelined, as mentioned above. Also, a number of CPUs (e.g. 16 in the Cray C90 [2]) worked in parallel. This is an application of the SIMD paradigm.

Processing speeds of the order of 10^{10} operations per second were achieved in old fashioned, Cray-like supercomputers [2]. Modern supercomputers, however, make use of semi-standard commercial processors in very large numbers – more than 10,000 in the largest machines. Processing speeds of the order of 10^{12} operations per second are thus achieved [29].

A3.4 Parallel Programming

Parallel languages [2] can either be imperative (how to do) or declarative (what to do). They can involve different levels of abstraction, from microcode (operating on single processor control lines), to assembly, to high level languages.

A classification of parallel languages must also deal with process synchronization and consider whether parallelism is hidden or explicit and which parallel computing paradigm is employed.

A3.4.1 Process Synchronization

One of the main problems in parallel programming is process synchronization. There are three approaches to this problem [2]. Processes can, of course, be opened and closed in an unsynchronized way. It is up to the programmer to insure that no conflict arises. At a higher sophistication level, processes can be opened in an unsynchronized way and closed in a synchronized way [30]. Finally, processes can be opened and closed in a synchronized fashion [30].

A3.4.2 Hidden Parallelism

Let us first consider the case of an SIMD array aimed, say, at image processing operations [1]. The array could be programmed for neighborhood averaging. As long as our language can support a binary image data type and an image averaging function, the procedure can simply be declared as S=Ave (R), where R is the raw image, S is the smoothed image and Ave is the averaging function. A language suitable for SIMD arrays is Fortran-Plus [1].

Pipelines and systolic arrays, see Fig (A3.6), also hide parallelism from the user [2,31,32]. The nature of these paradigms is such that problems are formulated in a sequential way, the parallelism being obtained from a straightforward task allocation between the different processors, which is automatically performed by the system. No special purpose languages are therefore required.

A3.4.3 Explicit Parallelism

Examples of explicitly parallel programming languages are Occam, Parlog and Dactl [1]. Occam is an imperative language which was used for programming transputers. Parlog is a declarative language and, in fact, a parallel version of Prolog. Dactl is based on the idea of letting the programmer write his code without worrying about the details of the parallel machine. The Dactl compiler then turns it into versions suitable for each specific machine.

Explicit parallelism [1] is required by most function parallel systems (pipelines are an exception). Communication in these systems is asynchronous. When a processor needs to send data to another processor, it sends a request. As soon as the receiving processor is ready, communication can take place. The process is called handshaking and requires a high level of processor autonomy.

A3.5 Parallel Architectures

The main parallel architectures are bus, crossbar switch, mesh, graph, pyramid, and hypercube [1]. All these architectures, with the notable exception of the crossbar switch, are non-reconfigurable. Reconfigurable architectures, due to their potential relevance for nanoelectronics [33], are treated in some detail in the next section.

Buses are only able to support a limited number of processors, typically of the order of ten, before saturating [1]. Therefore, they are used to connect a small number of MIMD processors communicating asynchronously, hence the need of a handshaking and an address bus, besides the data bus.

Fig. (A3.8) shows a crossbar switch network [1]. By suitably driving the switches, each pair of processors can be connected and a variety of architectures can be simulated. Crossbar switch arrays are usually associated with a moderate number of processors, communicating in a synchronous fashion.

Among mesh-connected arrays, the 4-connected square mesh, sometimes with four next-neigbour diagonal connections added, is usually preferred [1]. Mesh-connected arrays have poor long-range connectivity. However, they are suitable for local operations like those of image processing and are usually associated with the SIMD paradigm, see Fig. (A3.7).

The graph reduction paradigm naturally maps onto a graph-connected array. In fact, a computational graph like that of Fig. (A3.3) can be immediately translated into a processor arrangement. Artificial Intelligence (AI) applications are often suitable for this connection scheme, since symbolic AI problems can often be reduced to graph-searching problems [2].

Fig. (A3.9) shows a pyramid array. Pyramid arrays can be employed as improved mesh-connected arrays. By passing data to the peak of the pyramid and then back down, it is possible to add long-range connectivity to a mesh array, although serious bottlenecks may occur [34]. Pyramid arrays are also useful when doing image processing at different resolutions or machine vision at different levels [35]. Each vision or image processing level corresponds to a pyramid level.

Fig. (A3.10) shows the iterative construction of hypercubes of increasing orders. Of course, we are talking about hypercubes in the topological sense. What really counts is the level of connectivity provided. Hypercubes are a combination of short-range and long-range connectivity. This makes them attractive for connecting a large number of MIMD processors [1].

A3.6 Reconfigurable Architectures

The aim of reconfiguring links in a parallel array is two-fold [33]. Faulty processors can in fact be bypassed, so that the system can work even if some of its processors are faulty. This is especially relevant for nanoelectronic devices [36]. Link reconfiguration can also be used to implement different machines on the same array.

Details on reconfigurable parallel architectures can be found in Ref. [33]. Here we only refer to the Teramac machine [36], a massively parallel experimental computer, built in the late 1990s in the Hewlett-Packard (HP) laboratories to investigate a wide range of defect-tolerant architectures. Teramac contains a large number of identical chips, many of which had previously been discarded. This was a deliberate choice, aimed at testing the defect-tolerant properties of the architecture.

The chips, field programmable gate arrays (FPGAs), contain a large number of simple computing elements. The computing elements are just memory cells, so that the various logical operations are performed through a lookup table (LUT). The LUTs are connected by crossbar switches so that, by turning the various switches on and off, various parallel architectures can be implemented and faulty elements, as well as links, can be bypassed.

Teramac contains 864 identical FPGAs. About 30% of them are devoted to logic operations. The bulk of the FPGAs are used for communication and signal routing. LUTs comprise less than 10% of the total silicon area. The system has 65,536 LUTs, arranged hierarchically and operating at a clock frequency of 1 MHz.

Teramac is configured by a very long instruction word (300 megabits). Before configuration starts, bit strings are injected into the system, following different paths, to locate faulty computing elements and links. This is done by an external workstation but, in principle, the machine could test itself. The machine is then configured to implement the desired structure while avoiding faulty elements. This can only be done because of path and processor redundancy.

In the prototype built at HP laboratories, 75% of the FPGAs were supplied free of charge by the manufacturer, since they had proved to be faulty. Tests established that about 10% of the computing elements and 10% of the links were faulty. On the whole, about 3% of the computer resources proved to be faulty. Despite this, the computer could be configured into a number of different machines, including a self-testing machine and an image processor for magnetic resonance applications.

Processing speeds of the order of 10^{12} operations per second were achieved. Teramac represents an example of what modern supercomputers [44] are like and, in our opinion, neatly exemplifies how the approach to parallel computing has quite recently changed, due to the renewed interest for fault tolerance issues, with the realization that successful computer operation can be achieved despite some of the processors and/or links being faulty.

A3.7 Conclusions

Non-reconfigurable parallel computers are both complex and specialized devices. In fact, their physical structure and programming model have to be carefully tailored to the specific problem they have to tackle, or their performance is impaired. The future of non-reconfigurable parallel computers, then, probably lies in market niches, such as image processing arrays and pipelining embedded in serial computers.

Should one-chip massively parallel nanocomputers be built, the story might be different. This thesis, we hope, has shown how important fault tolerance is for nanoelectronics. On the other hand, reconfigurable architectures still have much to say as, in our opinion, the Teramac has shown. The future of parallel computing, as far as we can see it, seems then to lie in fault tolerance.

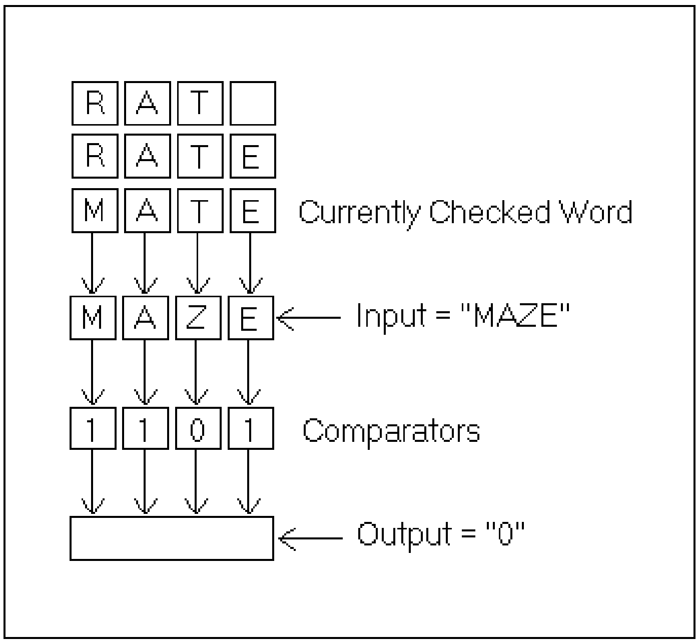

Fig. A3.5 An associative array. In this example, an associative spell checker, the words in the system dictionary are streamed past an array of comparators, each comparator taking care of a letter. An AND among the comparator outputs is used to detect global matches. Redrawn from [1].

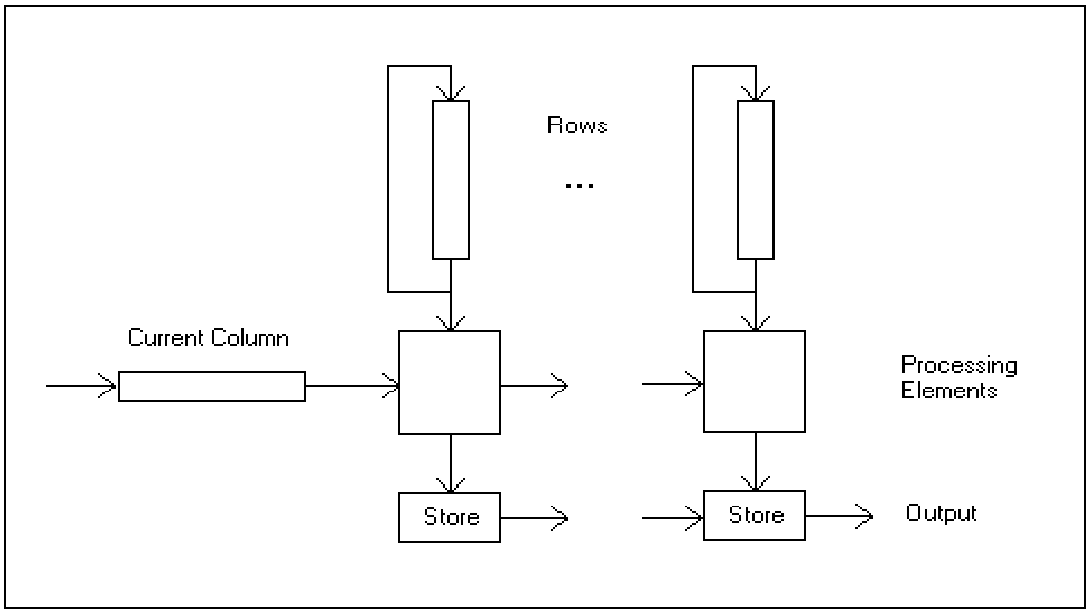

Fig. A3.6 A systolic array. In this example, a systolic matrix multiplier, the current column of, say, matrix B is pumped down the array. Each processing element calculates the scalar product between A and one of the columns of, say, matrix A. The results are stored and represent one of the rows of matrix A·B. Redrawn from [1].

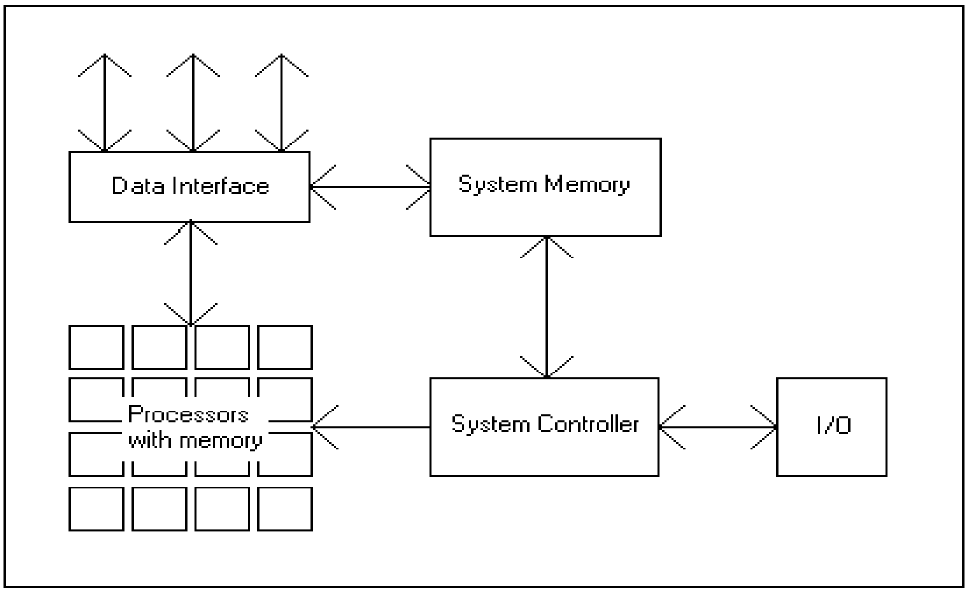

Fig. A3.7 An SIMD array. A number of interconnected processors perform the same task (say, an image processing task) on different input data at the same time, hence the acronym Single Instruction Multiple Data. Redrawn from [1].

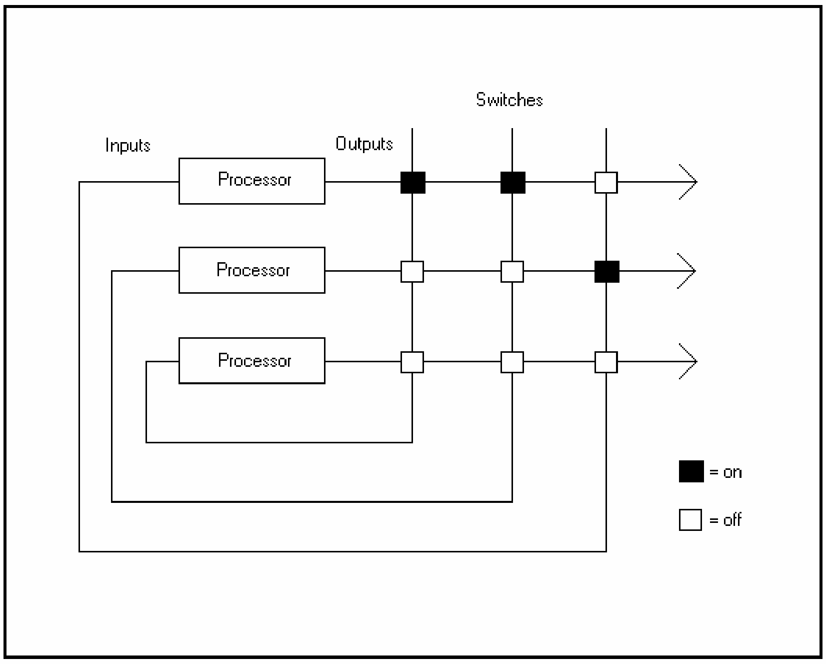

Fig. A3.8 A crossbar switch network. The processors are linked through a network of switches. By suitably switching them on, a variety of connection arrangements can be simulated. Redrawn from [1].

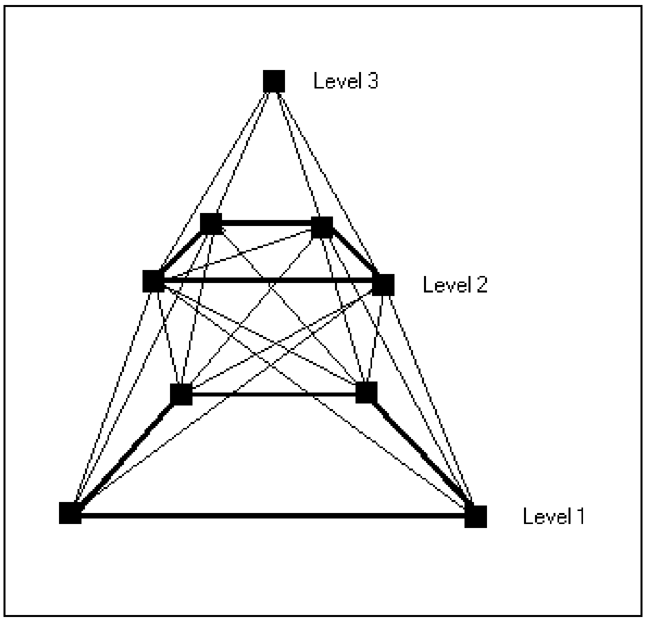

Fig. A3.9 A pyramid array. In this example, there are 3 levels. Level 1 is an MIMD array of processors (a small part of it, and of the other levels, is shown). Information is passed to levels 2 and 3 and back down, to improve connectivity. Redrawn from [1].

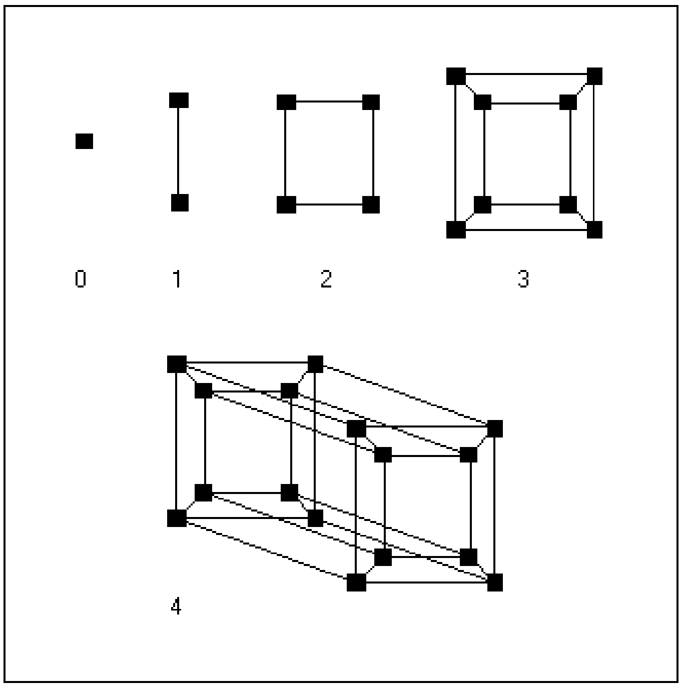

Fig. A3.10 Hypercubes of increasing order. A 0-dimensional hypercube is a point, by definition. A 1-dimensional hypercube is a segment, joining 2 points. A 2-dimensional hypercube is a square, joining the corresponding points of two segments. By iteration, we get a cube and higher dimensional hypercubes. Redrawn from [1].

A3.8 References

[1] T.J. Fountain, *Parallel Computing: Principles and Practice*, Cambridge University Press, Cambridge, UK (1994).

[2] D. Sima, T.J. Fountain and P. Kacsuk, *Advanced Computer Architectures. A Design Space Approach*, Addison-Wesley, Harlow, Essex, UK (1997).

[3] J. Hertz, A. Krogh and R.G. Palmer, *Introduction to the Theory of Neural Computation*, Addison Wesley, Harlow, UK (1994).

[4] M.J. Flynn, Very high speed computing systems, *Proc. IEEE* **54**, 1901-1909 (1966).

[5] D. Simmons, Datacube: Using DSP techniques to do real-time image processing, in: T.J.Fountain, Ref. [1].

[6] G.S. Tjaden and M.J. Flinn, Detection and parallel execution of independent instructions, *IEEE Trans. Comp.* **C-19**, n.10, 889-895 (1970).

[7] R.D. Acosta, J. Kjelstrup and H.G. Thorng, An instruction issuing approach to enhancing performance in multiple functional unit processors, *IEEE Trans. Comp.* **C-35**, n.9, 815-828 (1986).

[8] G.F. Grohoski, Machine organization of the IBM RISC System/6000 processor, *IBM J. Res. Develop.* **34**, n.1, 37-58 (1990).

[9] N.P. Jouppi and D.W. Wall, Available instruction-level parallelism for superscalar and superpipelined machines, in: *Proc. ASPLOS III*, 272-282 (1989).

[10] J.A. Fisher, Very long instruction word architectures and the ELI-512, in: *Proc. 10th AISCA*, 140-150 (1983).

[11] B.R. Rau, D.W.L. Yen, W. Yen and R. Towele, The Cydra 5 departmental supercomputer, *Computer* **22**, 112-125 (1989).

[12] C. Clack, GRIP: The GRIP multiprocessor, in: T.J. Fountain, Ref. [1].

[13] D.M. Watson, Supernode: The Parsys SN1000, in: T.J. Fountain, Ref. [1].

[14] M.J. Reeve and S. Wright, The experimental ALICE machine, in: T.J. Fountain, M.J. Shute, Eds., *Multiprocessor Computer Architectures*, North Holland, Amsterdam, 39-56 (1990).

[15] C. Kirkham, The Manchester Dataflow machine, in: T.J. Fountain, M.J. Shute, Eds., *Multiprocessor Computer Architectures*, North Holland, Amsterdam, 141-154 (1990).

[16] R.S. Wilson, nCUBE: The nCUBE 2 supercomputer, in: T.J.Fountain, Ref. [1].

[17] W.J. Dally, J.A.S. Fiske, J.S. Keen, R.A. Lethin, M.D. Noaks, P.R. Nuth, R.E. Davison and G.A. Fyler, The message-driven processor: A multicomputer processing node with efficient mechanisms, *IEEE Micro* **12**, n.2, 23-39 (1992).

[18] J.G. Fleming, Cray: Cray parallel supercomputers, in: T.J.Fountain, Ref. [1].

[19] J.R. Goodman and P.J. Woest, The Wisconsin Multicube: A new large-scale cache-coherent multiprocessor, in: *Proc. 15th Annual Intl. Symp. on Computer Architecture*, 422-431 (1988).

[20] D. Lenoski, The Stanford DASH multiprocessor, *IEEE Computer* **25**, n.3, 63-79 (1992).

[21] J. Kuskin, The Stanford FLASH multiprocessor, in: *Proc. 21st Annual Intl. Symp. on Computer Architecture*, 302-313 (1994).

[22] I. Jaloweicki, WASP: The associative string processor, in: T.J.Fountain, Ref. [1].

[23] B.H. McCormick, The Illinois pattern recognition computer – ILIAC III, *IEEE Trans. Comp.* **EC-12**, 791-813 (1963).

[24] G.H. Barnes, The ILIAC IV computer, *IEEE Trans Comp.* **C-17**, 746-757 (1968).

[25] M.J.B. Duff, A cellular logic array for image processing, *Pattern Recognition* **5,**

229-234 (1973).

[26] P.M. Flanders, D.J. Hunt, S.F. Reddaway and D. Parkinson, Efficient high speed computing with the distributed array processor, in: *High Speed Computer and Algorithm Organization*, Academic Press, Burlington, MA, USA (1977).

[27] W.D. Hillis, *The Connection Machine*, MIT Press, Boston (1985).

[28] J.R. Nicholls, MasPar MP1: The design of MasPar MP1, a cost-effective massively parallel computer, in: T.J. Fountain, Ref. [1].

[29] www.top500.com.

[30] S.A. Williams, *Programming Models for Parallel Computers*, John Wiley & Sons, Hoboken, NJ, USA (1990).

[31] J.V. McCanny and J.G. McWirther, On the implementation of signal processing functions using one-bit systolic arrays, *Electron Lett.* **18**, 241-243 (1982).

[32] H.T. Kung and C.E. Leiserson, Systolic arrays for VLSI, in: *Proc. Sparse Matrix 1978*, 256-282 (1978).

[33] F. Distante, M. G. Sami and R. Stefanelli, Reconfiguration techniques in the presence of faulty interconnections, in: *Proc. 1st Intl. Conf. on Wafer Scale Integration*, 379-388 (1989).

[34] M.J.B. Duff, Pyramids – Expected performance, in: *Pyramidal Systems for Computer Vision*, Springer-Verlag, Berlin, 59-73 (1986).

[35] G.R. Nudd, T.J. Atherton, N.D. Francis, R.M. Howarth, D.J. Kerbison, R.A. Packwood and G.J. Vaudin, A hierarchical MSIMD architecture for image analysis, *Proc. 10th ICPR*, 642-647 (1990).

[36] J.R. Heath, P.J. Kuekes, G.S. Snider and R.S. Williams, A defect-tolerant computer architecture: Opportunities for nanotechnology, *Science* **280**, 1716-1721 (1998).

[37] J. Rose, A. El Gamal and A. Vincentelli, Architecture of field-programmable gate arrays, *Proc. IEEE* **81**, n.7, 1013-1029 (1993).

Gate	Temperature	Error Rate per Clock Cycle
SED - 5 islands	300 K	10^{-6}
SED – 7 islands	300 K	10^{-8}
SED – 9 islands	300 K	10^{-10}
Koroktov – 5 islands	300 K	10^{-6}
Koroktov – 7 islands	300 K	10^{-8}
Koroktov – 9 islands	300 K	10^{-10}
QCA	77 K	10^{-14}
QCA	300 K	10^{-3}
Parametron	77 K	10^{-19}
Parametron	300 K	10^{-4}

Table 1 Error rates per clock cycle for nanogates of various kinds.

Logic Gate	Error-Correcting Code	Space Redundancy	Time Redundancy
SED – 300 K 9 junctions	Reed-Muller	16	2
SED – 300 K 9 junctions	Reed-Muller	16	2
QCA – 77 K	Hamming	1.5	2
QCA – 77 K	Reed-Muller	16	2
Parametron – 77 K	Hamming	1.5	2
Parametron – 77 K	Reed-Muller	16	2

Table 6 Fault tolerant solutions found for memory nanochips of various kinds.

Logic Gate	**Approach**	**Redundancy Domain**	**Replicas (Nanometre Scale)**	**Levels (Nanometre Scale)**	**Replicas (Micrometre Scale)**	**Levels (Micrometre Scale)**	**Space Redundancy**	**Time Redundancy**
SED – 300 K 9 junctions	CNMR	space	3	3	3	1	81	1
SED – 300 K 5 junctions	BER	space	2	4	4	1	64	1
SED – 300 K 7 junctions	BER	space	2	3	3	1	24	1
SED – 300 K 9 junctions	BER	space	2	3	2	1	16	1

Table 2 Fault tolerant solutions found for single electron logic nanogates.

Logic Gate	**Approach**	**Redundancy Domain**	**Replicas (Nanometre Scale)**	**Levels (Nanometre Scale)**	**Replicas (Micrometre Scale)**	**Levels (Micrometre Scale)**	**Space Redundancy**	**Time Redundancy**
Koroktov 300 K 9 junctions	***CNMR***	space	3	3	3	1	81	1
Koroktov 300 K 5 junctions	BER	space	2	4	4	1	64	1
Koroktov 300 K 7 junctions	BER	space	2	3	3	1	24	1
Koroktov 300 K 9 junctions	BER	space	2	3	2	1	16	1

Table 3 Fault tolerant solutions found for Koroktov's logic nanogates.

Logic Gate	**Approach**	**Redundancy Domain**	**Replicas (Nanometre Scale)**	**Levels (Nanometre Scale)**	**Replicas (Micrometre Scale)**	**Levels (Micrometre Scale)**	**Space Redundancy**	**Time Redundancy**
QCA 77 K	CNMR	space	2	3	0	0	8	1
QCA 77 K	CNMR	space	0	0	5	1	5	1
QCA 77 K	BER	space	2	2	0	0	4	1
QCA 77 K	CNMR	time	0	0	5	1	1	6
QCA 77 K	BER	time	2	2	0	0	1	9

Table 4 Fault tolerant solutions found for QCA-based logic nanogates.

Logic Gate	**Approach**	**Redundancy Domain**	**Replicas (Nanometre Scale)**	**Levels (Nanometre Scale)**	**Replicas (Micrometre Scale)**	**Levels (Micrometre Scale)**	**Space Redundancy**	**Time Redundancy**
Parametron 77 K	CNMR	space	2	3	0	0	8	1
Parametron 77 K	CNMR	space	0	0	5	1	5	1
Parametron 77 K	BER	space	2	2	0	0	4	1
Parametron 77 K	CNMR	time	0	0	5	1	1	6
Parametron 77 K	BER	time	2	2	0	0	1	9

Table 5 Fault tolerant solutions found for parametron-based logic nanogates.