Fault-tolerant Architectures for Nanoelectronic and Quantum Devices

Jie Han

Fault-tolerant Architectures for Nanoelectronic and Quantum Devices

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus Prof.dr.ir. J.T. Fokkema,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op dinsdag 30 november 2004 om 10:30 uur

door

Jie HAN

Bachelor of Science
Tsinghua University
Geboren te Pingshan, Hebei, China

Dit proefschrift is goedgekeurd door de promotor:
Prof.dr. I.T. Young

Toegevoegd promotor:
Dr.ir. P.P. Jonker

Samenstelling promotiecommissie:

Rector Magnificus	Technische Universiteit Delft, voorzitter
Prof.dr. I.T. Young	Technische Universiteit Delft, promotor
Dr.ir. P.P. Jonker	Technische Universiteit Delft, toegevoegd promotor
Prof.dr.ir. P. Dewilde	Technische Universiteit Delft
Dr. P. Hadley	Technische Universiteit Delft
Prof.dr. H. Corporaal	Technische Universiteit Eindhoven
Prof.dr. Yun He	Tsinghua University, Beijing, China
Dr. M. Forshaw	University College London, UK
Prof.dr. L.J. van Vliet	Technische Universiteit Delft, reservelid

The work presented in this thesis was supported by TU Delft in its DIRC (Delft Inter-faculty Research Center) program NanoComp.

Advanced School for Computing and Imaging

This work was carried out in the ASCI graduate school.
ASCI dissertation series number **110**.

Cover: The cover picture was taken in the Veluwe national park of The Netherlands. The small one shows a stone lion guarding the old gate of Tsinghua Yuan (University) in Beijing, China.

A Ph.D. dissertation of the Delft University of Technology
ISBN: 90-9018888-6

Copyright © 2004 by Jie Han
All rights reserved.
Printed by Universal Press, Veenendaal, The Netherlands.

To my teachers,

colleagues,

and friends.

献给

我的祖父母

我的父亲

母亲

岳父母

和妻子

容错性量子及纳米计算机体系结构研究

摘要

韩杰

今天CMOS技术的发展已经进入了亚微米（纳米）领域，预见在未来的15年内将接近于极限。基于纳米范围量子效应的各种新颖的信息处理电子器件已经得到了广泛的研究，并且有一些已经成功地在简单电路上得到了应用。纳米器件及电路研究方面的进展激励了对量子及纳米计算机的体系结构的进一步研究。例如：由于组成元器件的不可靠性和不稳定性，这些结构必须保证在器件和导线操作失败时能够可靠地工作；为了避免散热问题，元器件可能不得不在量子效应领域内操作；由于潜在的器件间的互连问题，纳米电路只能进行本地或局部的连接。

本学术论文针对设计纳米计算机时出现的体系结构问题，寻求一种可靠的解决方案，探索建立基于纳米及量子器件的可行性和计算机系统的可靠性。具体的，对容错的和本地连接的并行计算机结构进行了研究。

论文的结构如下：

第一章简要介绍微电子(*microelectronics*)和纳米电子(*nanoelectronics*)的发展及存在的问题，并讨论这些问题对纳米计算机结构的作用和影响。

第二章介绍纳米电子发展的当前状态和在纳米计算机结构领域的最新研究进展。

第三章叙述对容错体系结构的研究。回顾冯•诺伊曼 *(von Neumann)* 的与非门多路转接技术 *(NAND multiplexing technique)*，并将他在高度冗余领域的研究扩展到了相当程度的低度冗余。论文研究一个多级多路转接系统的马尔科夫 *(Markovian)* 随机本质并揭示其特征。提出一个基于与非们多路转接技术的系统结构，并证明这个结构可以有效地解决单电子隧道效应 *(Single Electron Tunneling: SET)* 器件的随机背景噪声 *(random background charges)* 问题。研究证明，虽然需要相当多的冗余组件，但对于主要受暂时性错误 *(transient errors)* 影响的纳米元器件来说，一个基于与非门多路转接技术的系统结构，可以是容错系统的一个解决方案。

第四章提出一个可同时容纳永久性 *(permanent)* 和暂时性错误的系统结构。在该结构中，多路转接技术被应用到了基本电路，分层次的重新可配置性 *(hierarchical reconfigurability)* 被应用到了整个系统。论文验证了通过在系统概念上增加重新可配置性，原本必需的大量的冗余组件倍数降低到了一个适度水平 —— 不大于 10^2。这个系统结构可以有效地同时容纳制造缺陷 *(manufacturing defects)* 和瞬时错误，并可以在门错误率高达 10^{-2} 的情况下仍然可靠工作。而该错误率对当前的任何微电子系统都是不可接受的。

根据冯•诺伊曼的多路转接技术，论文提出三重交织冗余技术 *(Triplicated Interwoven Redundancy: TIR)*，并将它作为三重模块冗余 *(Triple Modular Redundancy: TMR)* 的一般化推广，但是器件间互连是随机的。一个经典处理器模型和一个基于仿真的可靠性模型被用来评估容错性能。通过比较，模拟了使用 *TIR* 和所谓的四重逻辑 *(quadded logic)* 实现的处理器结构。一般来说，*TIR* 电路的可靠性和等效的 *TMR*

电路相当；但是对于某些特殊连结模式，由于 TIR 结构的内在交织互连本质，TIR 的可靠性较差。被延伸到高阶的 TIR，称之为多重交织冗余技术 (N-tuple Interwoven Redundancy: NIR)。应用五重交织冗余技术可以进一步将所需的系统冗余系数降低到不大于 10。对于 TIR/NIR 和四重逻辑结构来说，用来提高可靠性能的恢复电路（或投票器）的设计和实施是很重要的。只有在选用简单的投票器电路的情况下，才有可能通过使用高阶 NIR 技术获得更好的系统可靠性。TIR 或 NIR 技术特别适用于分子纳米计算机的随机化学或生物合成的制造过程。

第五章，基于约瑟夫森连接 (Josephson junctions) 的超导电路是研究的重点。约瑟夫森电路可能被应用在本地连接的处理器结构中。论文提出可以用约瑟夫森超导电路来实现传统 SIMD 计算机结构和基于矩阵的量子计算机结构。理想状况是，在约瑟夫森电路矩阵的中心进行的量子计算中，传统计算结构可以用作预先、中间和后继处理器。这样可以建立一个异质传统/量子计算机，用来实现同时包含传统和量子计算步骤的算法，比如绍尔的分解因数算法 (Shor's factoring algorithm)。一个可逆性计算结构原则上也可以用约瑟夫森超导电路来实现，但这方面理论还有待研究。

论文还提出一个可用约瑟夫森超导电路实现的新颖计算结构：量子单元非线性网络 (Cellular Nonlinear Networks: CNNs)。由于传统计算结构 (SIMD 矩阵)，量子计算结构，和不完全量子计算结构 (量子 CNNs) 可同时作为同一个器件的应用来研究，不论最终哪一种电子器件被用来实现量子或纳米计算机，约瑟夫森超导电路是目前研究量子和纳米计算机结构的一个很好的媒介。

本章节将结束这份学术论文，然而对纳米和量子计算机体系结构的研究才刚刚开始。

本论文之外，形成这份论文基础的作者发表的科技论文已经激励了许多在容错结构方面的新研究，譬如，为了更好地了解在由马尔科夫链组成的纳米系统中的容错性能，有些理论运用蒙特卡洛模拟 (Monte Carlo simulations)，有些运用叉路理论 (bifurcation theory) 和应用组合变量的确切分析方法。一个基于概率论并应用马尔科夫随机领域 (Markov Random Field: MRF) 的方法被提出用来设计纳米计算机，计算机辅助设计 (CAD) 工具也已经被用来评估容错计算机结构的可靠性和冗余度之间的平衡。起源于冯•诺伊曼的冗余技术实际上属于纠错码 (Error-Correcting Codes: ECC) 的范畴。多路转接技术实际上归结于对重复编码 (repetition codes) 的应用。重复编码是对各个信息变量重复多次以创造冗余。纠错编码及其它容错技术在纳米计算上的应用需要进一步的研究。

新型的计算机系统可能在未来涌现。现在可以设想的有基于分子电路的适应性系统，受生物学启发的自我学习和自我演变系统，非线性动态系统和量子计算机系统。新的计算系统将导致新颖的算法和结构。而算法和结构的选择必须以纳米技术的应用为目的。计算机结构将在根本上影响电子器件和电路的设计，反之亦然：在纳米电子器件中发现的机会和问题将强烈地影响结构的选择。因此，在对纳米计算机的研究中，各学科间的融会贯通至关重要，成功将最终依赖于在多种学科，比如化学、物理、电子工程、计算机科学交互融合和许多其它领域内的共同努力。

Contents

1 Introduction 1
  1.1 From microelectronics to nanoelectronics 1
  1.2 From nanoelectronics to nanoelectronic computers 2
  1.3 Contributions of this dissertation 3

2 Computing Architectures for Nanoelectronic and Quantum Devices 5
  2.1 The current status of nanoelectronics 5
    2.1.1 Resonant tunneling devices (RTDs) 5
    2.1.2 Single electron tunneling devices (SETs) 6
    2.1.3 Quantum cellular automata (QCA) 6
    2.1.4 Rapid single flux quantum (RSFQ) and superconducting circuits of Josephson junctions 7
    2.1.5 Carbon nanotubes (CNTs) and semiconductor nanowires (NWs) . . . 8
    2.1.6 Molecular nanoelectronics 9
  2.2 Computing architectures for nanoelectronic and quantum devices 9
    2.2.1 Defect- and fault-tolerant architectures 9
    2.2.2 Locally-connected (coupled) computing architectures 11
    2.2.3 Quantum computers 12

3 Fault-Tolerance in Nanocomputers: The Multiplexing Approach 15
  3.1 Introduction 15
  3.2 von Neumann's theory on NAND multiplexing 17
    3.2.1 A NAND multiplexing unit 17
    3.2.2 The restorative unit 21
  3.3 Error distributions in a multiplexing unit — an alternative method 25

3.3.1 Theoretical analysis . . . 25
3.3.2 Numerical evaluation . . . 27
3.4 Error distributions in multi-stage systems . . . 34
3.4.1 For modest N . . . 34
3.4.2 For large N . . . 37
3.5 Discussion . . . 39
3.6 Application . . . 39
3.7 Summary . . . 42

4 A Defect- and Fault-Tolerant Architecture and Its Implementation for Nanocomputers 45

4.1 Introduction . . . 45
4.2 The developments of fault-tolerant techniques . . . 46
4.3 The NAND multiplexing technique for correlated errors . . . 48
4.3.1 Error distributions in a multiplexing unit . . . 48
4.3.2 Error distributions in a multi-stage system . . . 50
4.4 Reliability analysis of reconfigurable architectures . . . 51
4.5 A hypothetical architecture for defect- and fault-tolerance . . . 53
4.5.1 Basic circuits implemented with NAND multiplexing . . . 53
4.5.2 Hierarchical reconfigurability at processor, cluster and chip levels . . . 55
4.5.3 Summary and issues . . . 55
4.6 Triple modular redundancy (TMR), quadded logic and triplicated interwoven redundancy (TIR) . . . 59
4.6.1 Triple modular redundancy (TMR) . . . 59
4.6.2 The interwoven redundant logic and quadded logic . . . 61
4.6.3 Triplicated interwoven redundancy (TIR) . . . 63
4.7 Experimental studies on fault-tolerant processor architectures . . . 67
4.7.1 A processor prototype for array architectures and its fault-tolerant implementations . . . 67
4.7.2 A fault injection simulation . . . 70
4.7.3 The effects of critical gates (CGs) in voters . . . 71
4.7.4 A simulation-based reliability model and results . . . 74
4.8 N-tuple interwoven redundancy (NIR) . . . 78

4.9 Discussion 79
4.10 Summary 81

5 Computing with Locally-Coupled Josephson Circuits 83
5.1 Introduction 83
5.2 The superconducting circuit of Josephson junctions 85
5.3 Classical computing with Josephson circuits 87
5.3.1 Circuit topology and simulations 87
5.3.2 Elementary logic gates 89
5.3.3 A processor element (PE) design 92
5.4 Quantum computing with Josephson circuits 96
5.4.1 Introduction to quantum computing 96
5.4.2 Elementary quantum gates 98
5.4.3 Quantum sum and carry 99
5.4.4 Quantum Fourier transform 100
5.4.5 A draft structure of a quantum computer 102
5.5 Quantum cellular nonlinear networks (CNNs) using Josephson circuits 104
5.5.1 Cellular nonlinear networks (CNNs) 104
5.5.2 Formulating Josephson quantum dynamics as CNN state dynamics . 104
5.5.3 Simulations 106
5.5.4 Summary and issues 108
5.6 Implementation issues 109
5.7 Summary 111

Summary 113

Samenvatting 117

Bibliography 121

Acknowledgement 133

Curriculum Vitae 135

Chapter 1

Introduction

1.1 From microelectronics to nanoelectronics

The rapid growth of microelectronics has been based on the continuous miniaturization of electronic components over decades. Since the invention of the transistor, electronic circuits have evolved at an amazing pace from the early integrated circuits (ICs), with tens of components, to nowadays very-large-scale-integrated (VLSI) systems with hundreds of millions of components. This evolution is commonly referred to as being governed by Moore's law, which states that the number of electronic components per chip doubles every 18 months. Today's VLSI circuits are based on the complementary metal-oxide-semiconductor (CMOS) field-effect-transistors (FETs), and the state-of-the-art fabrication process of CMOS has reached a node dimension of 90 nm. However, as CMOS technology enters the nanoelectronic realm (tens of nanometers and below), where quantum mechanical effects start to prevail, conventional CMOS devices are meeting many technological challenges for further scaling. A variety of non-classical CMOS structures have been invented and investigated worldwide. It is generally believed that these novel structures will extend the CMOS technology to 45 nm nodes by the year 2009. If this scaling continues beyond 2009, however, CMOS technology is anticipated to hit a brick wall and cease to decrease in size around 2019 [1]. This will be due to many reasons such as the physical limitations imposed by thermal fluctuations, power dissipations and quantum effects, and the technological limitations in manufacturing methods (e.g. lithography), etc.

Besides the endeavor devoted to the continuous scaling of CMOS by developing advanced device structure, various novel information processing devices based on new physical phenomena have been proposed and some have been successfully demonstrated at the logic circuit level. These devices include resonant tunneling devices (RTDs), single electron tunneling (SET) devices, quantum cellular automata (QCA), rapid single flux quantum (RSFQ) and superconducting circuits of Josephson junctions, carbon nanotubes (CNTs) and silicon nanowire (SiNWs), molecular devices, spin-based devices, etc. [2], [3] They share one or more characteristics such as extremely small dimensions, high switching speed, low power consumption, ease of fabrication and good scaling potential. Many of these devices fall into the scope of nanoelectronics, such as those based on coulomb blockade tunneling and molecules. Some devices, mainly employing superconducting quantum effects, are, however, in the microscopic regime, such as RSFQ and superconducting circuits of Josephson junctions.

In the near term, one or more of these devices are expected to be integrated on a CMOS platform, possibly serving as complementary components to CMOS. In the long term, the research in nanoelectronics may provide opportunities for alternative technologies to the electronics beyond CMOS [4].

A brief survey of the nanoelectronic and quantum effect devices is presented in Chapter 2.

1.2 From nanoelectronics to nanoelectronic computers

The advances at device and circuit levels have raised design issues for computer architectures based on nanoelectronic and quantum devices [5], [6]. The developments of nanoelectronics could eventually lead to extremely large scales of integration, of an order of a trillion (10^{12}) devices in a square centimeter. The architectures of the integrated circuits and systems must be suitable for implementations in nanoelectronic devices. In other words, architectures must optimally make use of the properties and at the same time deal with the drawbacks of the devices. There are many features in nanoscale devices that impose limitations on nanoelectronic architectures, while the most prominent ones have been recognized as: the devices' poor reliabilities, the difficulties in realizing interconnects and the problem of power dissipation [7], [8].

The unreliability of nanoelectronic devices comes from two sources. One is the bottom-up manufacturing process of self-assembly, which will be used at dimensions below those for which conventional top-down fabrication techniques can be used. Since imprecision and randomness are inherent in this self-assembly process, it is almost inevitable that a large number of defective devices will appear due to this fabrication process. The other source of errors is the environment in which the devices will be operating. Due to a reduced noise tolerance of low thresholds of state variables, malfunctions of devices may be induced by external influences such as electromagnetic interference, thermal perturbations, cosmic radiation, etc. Hence, permanent faults or defects may emerge during the manufacturing process, while transient errors may spontaneously occur during operation. The issue of defect- and fault-tolerance is therefore critical for any large integration of unreliable nanoelectronic devices. Several techniques, such as NAND multiplexing, N-tuple modular redundancy (NMR) (e.g. triple modular redundancy (TMR)) and reconfiguration, have been investigated for fault-tolerant implementations in nanocomputer architectures.

The problem of interconnects is partly due to the imperfect manufacturing process, which makes it difficult to produce precise alignments between wires. Another challenge lies on how transformations of interconnects can be made from nanoscale dimensions to the macroscopic world of realizable systems. In addition, long-distance communication seems a problem for nanoelectronic systems, because of the properties of many devices such as low drive capabilities and easy local interactions. For these reasons, the parallel architectures that are highly regular and locally connected have been proposed for nanocomputer implementations. Among those, the single instruction and multiple data (SIMD) computers, quantum cellular automata (QCA) and cellular nonlinear networks (CNNs) have been the subjects of intense research activities.

Thermal power dissipation comes from the device switching energy and the energy needed to drive signals through circuits. The minimum energy needed to switch a bit and the switching frequency are limited by the uncertainty principle. In other words, the power-delay product (minimum power dissipated × switching time) cannot be less than Planck's constant, in the quantum limit [9]. This indicates that a trade-off of clock speed versus device density has to be made, i.e. clock speeds will need to be decreased for very high densities and densities will need to be decreased for very high clock speeds. This implies that a nanocomputer will rely on massive parallel processing rather than on fast operation speed. The problem of power dissipation sets in general a limit to any electron transport device. The strategies to overcome this are to employ novel devices that use alternative variables for logic states, such as spin-based devices, and to search for computing architectures based on novel physical principles, such as quantum mechanical computers.

A brief review of these nanoelectronic and quantum computer architectures is presented in Chapter 2.

1.3 Contributions of this dissertation

- In research on fault-tolerant architectures, the NAND multiplexing technique, as initiated by von Neumann, has been comprehensively studied. In particular, the NAND multiplexing technique is extended from a high degree of redundancy to a fairly low degree of redundancy; the stochastic Markovian characteristics in a multi-stage multiplexing system are discovered and investigated. It has been shown that the Markov chain model presents a general framework in the study of systems based on multiplexing techniques. (Chapter 3)
- A defect- and fault-tolerant architecture, with the multiplexing technique implemented into the fundamental circuits and a hierarchical reconfigurability mapped to the overall system, is proposed. It has been shown that the required redundancy could be brought back to a moderate level by reconfigurability. This architecture is efficiently robust against both manufacturing defects and transient faults, tolerating a gate error rate of up to 10^{-2}, which is in general unacceptable for any current VLSI system. (Chapter 4)
- A novel fault-tolerant technique, the triplicated interwoven redundancy (TIR), is proposed as a general class of triple modular redundancy (TMR), but implemented with random interconnections. The TIR is extended to higher orders, namely, the N-tuple interwoven redundancy (NIR), to achieve higher system reliabilities. The TIR/NIR is in particular suitable for implementation in molecular nanocomputers, which are likely to be fabricated by a manufacturing process of stochastically chemical assembly. Our study suggests that the randomness inherent in the process of molecular self-assembly might not be an obstacle that prevents one from implementing fault-tolerant measures into a molecular architecture, and that a low overhead fault-tolerant architecture might be possible for a future nanosystem. (Chapter 4)
- A classical SIMD computer architecture and an array-based quantum computer structure have been studied as possible applications of superconducting circuits of Josephson junctions. The classical computer may serve as a pre- and post-processor for the quantum computing performed in the heart of the Josephson circuit array, establishing a

heterogeneous quantum/classical computer for, e.g. an implementation of Shor's factoring algorithm. A quantum CNN architecture using the Josephson circuits has also been proposed, presenting a novel computing paradigm for Josephson circuits. Since classical computing architectures (SIMD arrays), quantum computing architectures and semi-quantum computing architectures (quantum CNNs) can be simultaneously studied on the same device, it has been shown that the Josephson circuit is a good vehicle for investigating the architectural issues of quantum and nanoelectronic computer systems, independently from the question which device will be the ultimate implementation vehicle. (Chapter 5)

Chapter 2

Computing Architectures for Nanoelectronic and Quantum Devices

2.1 The current status of nanoelectronics

2.1.1 Resonant tunneling devices (RTDs)

Resonant tunneling devices (RTDs) are usually two terminal devices of vertical semiconductor heterostructures with two insulating layers separating the conducting regions. A negative differential resistance (NDR) is produced by the double barrier structure, which has a resonance peak enabling the resonant tunneling of electrons through the barriers. Due to the fast tunneling process, the RTDs have inherently a very high switching speed (up to 700 GHz), which makes them potentially attractive for high speed switching applications, such as very high frequency oscillators, amplifiers and ADCs [1].

Three terminal devices have been demonstrated by integrating RTDs with conventional FETs (RTD-FETs) [2]. Various designs, including digital logic, threshold logic and memory, were proposed based on the heterostructures of RTD-FETs [10], [11]. However, the combination of RTDs and transistors introduces delays to the intrinsically fast switching speed of RTDs. The operating speed of the hybrid devices can be an order of magnitude slower than the switching speed of RTDs. Furthermore, the complexity of the integrated structure imposes a limit on the scaling properties of the devices, compared with CMOS. Resonant tunneling transistors (RTTs) have been obtained by adding a control terminal to the RTD [12] and RTT-based logic circuits have been demonstrated [13].

A major problem with RTDs is the extreme sensitivity of the device characteristics to the layer thickness, as the tunneling current depends exponentially on the thickness of the tunnel barrier. Difficulties in manufacturing, to produce large-scale RTD circuits with uniform thickness of tunnel barriers, remain. This and other challenges in fabrication may limit the usefulness of RTDs in certain niche applications of high speed switching, digital signal processing, ADC, DAC, etc.

2.1.2 Single electron tunneling devices (SETs)

Single electron tunneling devices (SETs) are three terminal devices where electron movement is controlled with a precision of an integer number of electrons. An electron can tunnel from and to an island (or quantum dot) through a tunneling barrier, which is controlled by a separate gate based on Coulomb blockade. This electron island can accommodate only an integer number of electrons. This number may be up to a few thousand. A single electron transistor is composed of a quantum dot connected to an electron source and to a separate electron drain through tunnel junctions, with the electron injection controlled by a gate electrode. Single electron transistors can be implemented in logic circuits by operating on one or more electrons as a bit of information [14].

SET circuits usually operate at very low temperatures. It is estimated that the maximum operation temperature for $2nm$ SETs is 20 K, with an integration density of approximately 10^{11} cm^{-2} and a speed of the order of 1 GHz [15]. Various logic applications of SETs, including inverters [16], [17], OR and a 2-bit adder [18], have been demonstrated. However, due to the high impedance required for Coulomb blockade, a SET gate would not be able to drive more than one other gate. This has two implications. First, SET logic would have to be based on local architectures, such as cellular arrays and cellular nonlinear networks (CNNs). Second, although SETs may not be suitable for implementations in logic circuits, they could be used for memories. SET-based memory structures have been proposed and experimentally demonstrated [19]-[21].

Background charge fluctuations remain a major issue for the successful operation of a SET-based circuit [14]. Due to electrostatic interactions, correct device functions can be destroyed by impurities and trapped electrons in the substrate. In order to tackle this problem, besides the endeavor to develop novel computing schemes, such as the multi-value SET logic, fault-tolerant architectures, implemented at higher levels of circuits and systems, might be a direction for investigation [46].

2.1.3 Quantum cellular automata (QCA)

Cellular automata (CA) are computing architectures inspired by complex natural and physical systems [22]. CA systems are usually based on regular arrays of simple cells. Each cell in an array interacts with its neighbors and evolves from an initial state into a final state. The evolution of a cell is determined by the cell's initial state and the interactions with its neighbors. A computation can be mapped to such a dynamic process in a CA system.

The concept of quantum cellular automata (QCA) was first proposed as a cell structure of quantum dots coupled via quantum mechanical tunneling [23]. In a typical 4-dot cell, the quantum dots are in the corners of a square cell. Due to electrostatic repulsion, free charges will occupy the dots in diagonally opposite corners of the cell and form two bistable states representing binary bits. Logic states are thus encoded in the spatial distribution of electric charges in a cell and a computation can be performed by the mutual interactions of cells in an array. Basic circuits of logic [24], a latch [25] and shift registers [26] have been experimentally demonstrated for electronic QCA implementations.

The potential advantages of QCA are high switching speed, low power consumption and good scaling capability. It is estimated that the inter-dot distance in a solid state QCA cell would be approximately 20 nm and the inter-cell distance would be 60 nm [27]. In a recently proposed scheme for a molecular QCA cell [28], the inter-dot distance is expected to be about 2 nm, and the inter-cell distance about 6 nm. An optimistic evaluation shows that the intrinsic switching speed of an individual QCA cell can be in the THz range [27].

However, it was shown that by a comparative study of QCA and CMOS circuit performance a practical circuit of solid-state QCA will only have the maximum operating speed of a few MHz [29]. This frequency might be a few GHz for the circuits based on molecular QCA. It was also shown that the maximum operating temperature for a standard solid state QCA cell is about 7 K, indicating that room temperature operation is not possible for solid state QCA systems [27]. Molecular QCA systems might be the only possibility for room temperature operation. Another serious drawback of QCA devices is that they suffer from the problem of background charge fluctuation, because QCA are single electron devices.

Besides the widely studied electronic QCA, the concept of magnetic QCA based on small ferromagnetic structures has been proposed for room temperature operation [30]. For magnetic QCA, logical states are represented by the directions of the cell magnetization and cells are coupled through magnetostatic interactions. The minimum size of magnetic QCA cells is estimated to be about 100 nm, and the maximum switching speed is about 200 MHz. Logic devices including a shift register have been demonstrated for the use of nanoscale ferromagnetic devices [31].

2.1.4 Rapid single flux quantum (RSFQ) and superconducting circuits of Josephson junctions

RSFQ devices are based on the effect of flux quantization in superconducting circuits of Josephson junctions [32]. The Josephson junctions serve as switching elements and binary bits are represented by the presence or absence of flux quanta in the superconducting circuits. A voltage pulse is generated when a magnetic flux quantum is transferred from one circuit to another by switching the Josephson junctions. Complex circuit functions are realized by the propagation and interaction of the voltage pulses in RSFQ circuits. Current RSFQ devices are mainly built on low temperature superconductors ($\sim$ 5 K), while high temperature superconductor ($\sim$ 50 K) technology may eventually be possible for implementations of RSFQ circuits.

The main advantage of the RSFQ circuit is the very high operating speed of up to approximately 770 GHz, which has been achieved in flip-flop circuits [33]. More complex circuits, such as random access memories, adders and multipliers, have been demonstrated [34]. As the superconducting quantum effect occurs at a microscopic scale, the typical dimension of RSFQ devices is a few microns. It has been shown that it might be able to scale the RSFQ circuits down to 0.3 μm and a frequency of 250 GHz [35]. However, further scaling of RSFQ into nanoscale will be a challenge, due to many limiting factors associated with this technology.

The main drawback of the RSFQ technology is the need for cryogenic cooling [36]. A broad scale of applications will strongly depend on the availability of low cost, highly reliable and compact cooling systems. Before great technical progress is made for cryogenic coolers, the RSFQ technology is likely to be limited to niche applications where speed is the dominant requirement.

Superconducting circuits of Josephson junctions can also be used for quantum information processing. A superconducting loop of three Josephson junctions has been proposed and demonstrated as a quantum bit or qubit [37]-[40]. A coherent superposition of two persistent-current states can be obtained when the two classical states are coupled via quantum tunneling through an energy barrier. The classical states of persistent currents can also be used as two binary bits [41]. Logic functions can be realized by coupling two or more bits, i.e. the circuit loops [42]. The interaction between loops is via magnetic interference of the superconductors. A cellular array architecture based on the Josephson circuits is discussed in Chapter 5.

2.1.5 Carbon nanotubes (CNTs) and semiconductor nanowires (NWs)

Carbon nanotubes and semiconductor nanowires are often considered as molecular devices, while they are referred to as one-dimensional (1D) devices in [1]. The potential advantages of 1D structures include enhanced mobility and phase-coherent transport of the electron wavefunctions. These properties may lead to faster transistors and novel wave interference devices. Carbon nanotubes and semiconductor nanowires are important subsets of 1D structures.

A carbon nanotube is a molecular cylinder formed by rolling up an atomic sheet of carbon atoms [52]. Carbon nanotubes typically have diameters of less than 20 nm and lengths of up to several microns. A CNT can be a semiconductor or a metal, which is determined by the tube diameter and the way it is rolled up. The tubes can be doped to make p-n junctions. Transistors have been obtained from CNTs [53]-[55], and logic circuits, such as NOT, NOR, a flip-flop and ring oscillators, have been demonstrated [56], [57]. However, it is still not possible to precisely control whether CNTs are semiconducting or metallic, which makes the fabrication of CNTs a random process.

Semiconductor nanowires could also function as building blocks for nanoscale electronics, and can be fabricated through a directed assembly process [58], [59]. A nanowire, usually with a diameter of $10 - 20$ nm, can be doped as a p- or n-type device. NW FETs have been obtained by making structures of crossed p- and n-type nanowires separated by a thin dielectric [60]. Various logic gates with gains have been demonstrated [61]. More complicated circuits such as address decoders have recently been reported [62]. These results present a step toward the realization of integrated nanosystems based on semiconductor NWs.

The problems associated with 1D structures (CNTs and NWs) include their low drive capability of individual devices, their contact resistance limited by quantum effects, their interconnect problems and yield of fabrication.

2.1.6 Molecular nanoelectronics

Molecular electronic devices are assumed to be based on electron transport properties through a single molecule [63]. The exact mechanism of charge transport in molecules is not yet well understood. Logic circuits based on two-terminal devices [64] and programmable molecular switches [65] have been experimentally realized. A three-terminal FET structure based on a C-60 molecule has been demonstrated, but with a very high contact resistance [66]. The most complicated molecular circuit to date is a 64-bit random access memory, which has been experimentally realized on a 2-dimensional (2D) crossbar circuit [67].

Large-scale molecular circuits can in principle be fabricated through self-assembly, a stochastically chemical or biological process of low cost. The progress of molecular electronics may eventually lead to large-scale integrated circuits, possibly with a density of $10^{12} bits/cm^2$ [68]. However, there are many technological challenges in building large-scale molecular circuits [69]. For examples, there are no or very low gains in molecular circuits, and most molecular devices have low "on-off" current ratios, which make molecular devices fragile to perturbations and noise. The problems of yield in fabrication and reliability in operation due to the stochastically self-assembly process indicate that molecular computer systems would require defect- and fault-tolerant architectures for reliable operations.

2.2 Computing architectures for nanoelectronic and quantum devices

2.2.1 Defect- and fault-tolerant architectures

The very small sizes of molecular and nanoelectronic devices make it possible to build a trillion (10^{12}) devices in a square centimeter. However, for such a densely integrated circuit to perform a useful computation, it has to deal with the inaccuracies and instabilities introduced by fabrication processes and external influences. Permanent faults may emerge during the manufacturing process, while transient ones may spontaneously occur during the computer's lifetime. It is therefore likely that the emerging nanoelectronic devices will eventually suffer from more errors than classical CMOS devices in large-scale integrated circuits. In order to make future systems based on nanoscale devices reliable, the design of fault-tolerant architectures will be necessary.

Fault-tolerant approaches have been of interest since the first generation of electronic computers when computers were constructed from such unreliable components as vacuum tubes. In the 1950s von Neumann initiated the study of using redundant components to obtain reliable synthesis from unreliable components, namely, the multiplexing technique [70]. It has been shown that the multiplexing structure, based on a massive duplication of imperfect devices and randomized imperfect interconnects, can be reliable with a high probability, provided that the failure probability of a component is sufficiently small. Since this study of von Neumann, various fault-tolerant techniques have been developed and successfully implemented in modern computer systems. These includes N-tuple modular redundancy (NMR) (e.g. triple modular redundancy (TMR)), reconfiguration and error correcting codes [71].

NMR and TMR designs, as implied in the multiplexing technique, have been implemented in VLSI systems for high reliability applications, and have been used as benchmarks for evaluating fault-tolerant approaches. In TMR, the most general form of NMR, three identical circuit modules perform the same operation, and a voter accepts outputs from all three modules, producing a majority vote at its output. A reconfigurable architecture is a computer architecture which can be configured or programmed after fabrication to implement a desired computation. Faulty components are detected during testing and excluded during reconfiguration.

Recently, these fault-tolerant techniques have been studied for potential use in nanoelectronic systems [72], [73]. The main results were that the multiplexing technique and NMR generally require a large amount of redundant components and an extremely low error rate of nanoelectronic devices, and that the reconfiguration may be efficient for protection against manufacturing defects if defective devices can be located. In [46], von Neumann's NAND multiplexing technique was extended from high degrees of redundancy to fairly low degrees of redundancy, and the characteristics of a Markov chain is discovered and investigated in a multi-stage multiplexing system, as presented in Chapter 3. It was shown that this multiplexing might be an effective fault-tolerant technique for protection against the increasing transient faults in nanoelectronic systems. Further, a CAD method based on probabilistic model checking has been proposed to evaluate the reliability of fault-tolerant architectures and, in particular, the multiplexing systems [74]; Monte Carlo simulations have been performed to study the error behavior in a multiplexing nanosystem [75]; and a better understanding of the error behavior in the Markov chains of multiplexing systems is obtained through a study using bifurcation theory [76]. For reconfiguration, the Teramac computer [77], though built with conventional CMOS technology, is a successful proof-of-principle model for nanocomputers. The basic components in Teramac are programmable switches (memory) and redundant interconnections. High communication bandwidth is critical for both parallel computation and defect tolerance. Array-based reconfigurable architectures have also been proposed for the applications of two-terminal molecular devices [78] and carbon nanotubes (CNTs) and silicon nanowires (SiNWs) FETs [79].

A hierarchically reconfigurable architecture with multiplexing technique implemented into the fundamental circuits has been studied as a system that is robust against both manufacturing defects and transient faults [47]. In this architecture, the required redundancy could be brought back to a moderate level — no larger than 10^2 — by reconfigurability. A new form of interwoven redundant logic, the triplicated interwoven redundancy (TIR), has been proposed as a general class of triple modular redundancy (TMR), but implemented with random interconnections [50]. The TIR is extended to higher orders, namely, the N-tuple interwoven redundancy (NIR), to achieve higher system reliabilities. The NIR/TIR is in particular suitable for implementation through the manufacturing process of stochastically molecular assembly. This study suggests that a low overhead fault-tolerant architecture may be possible for an implementation of future nanosystems. These are presented in Chapter 4.

The redundancy technique, originating from von Neumann, is basically an error correcting code [80]. Error correcting codes provide a way to cope with the corruption of bits by encoding messages as code words that contain redundant information. The multiplexing construction boils down to the use of a so-called repetition code, in which each symbol of a message is repeated many times to create redundancy. The use of error correcting codes in fault-tolerant nanosystems has also been explored [81].

2.2.2 Locally-connected (coupled) computing architectures

The advances in nanoelectronics have also raised design issues for novel computation structures for nanoelectronic and quantum effect devices. The study of computer architectures started before the first electronic computer. Some fundamental issues were then thought about computation, such as what can, in principle, be computed or effectively computed and how to realize it on a computer, and extensively studied. A remarkable achievement in computation theory was made in 1936 when Turing developed in detail a mathematical model for computation now known as the Turing machine. Turing showed that there is a Universal Turing Machine that can do anything that any specific Turing machine can do. Furthermore, he asserted that, if a computation can be effectively performed on any computer hardware, it can then be effectively done by a Universal Turing Machine. This assertion established a connection between computer hardware that carries out computations and the equivalent theoretical model of a Universal Turing Machine.

Later in the 1940s an architecture model was developed by von Neumann for the practical realization of a computer functional as a Universal Turing Machine. The von Neumann architecture is commonly defined as a computer architecture that sequentially executes a single stream of instructions stored with data in an addressable memory. Early computers were mostly sequential computers based on von Neumann architecture. Sequential computers are however slow due to sequential execution of instructions in programs. Functional parallelism was therefore explored and, with the advancement of VLSI circuits, massively parallel computers have been built and used in various areas of data processing, in particular in the field of high performance image processing (see, for examples, [82]-[85]).

This evolution of computer architectures has been, and will continue to be, driven by the development of underlying technologies of computer hardware. For computers based on nanoelectronic and quantum devices, due to the characteristics of these devices such as low power consumption, low drive capability and easy local interactions, the parallel architectures that are highly regular and locally connected, such as the single instruction and multiple data (SIMD) computers [86], quantum cellular automata (QCA) [87] and cellular nonlinear networks (CNNs) [88], have preferences to be the prototype architectures.

Although they have been studied separately, SIMD computers, the QCA architecture and CNNs all belong to the category of cellular array architectures. SIMD computers consist of assemblies of identical, simple processor elements (PEs), usually associated with local memories and connected to its nearest neighbors in a linear or square array. SIMD processor arrays have been successfully used in various areas of high-performance image and data processing [89]. Cellular automata (CA) represent an alternative computing paradigm to the conventional von Neumann architecture, albeit that the study of CA was also initiated by von Neumann [90]. Typically the QCA architecture has been studied as an implementation of arrays of electrostatically coupled quantum dots [87]. The computing issues of a magnetic QCA based structure has also been investigated [91]. Recent study has shown that the QCA paradigm may also have applications in molecular structures [28]. For a regular and uniform network of QCA, various computation algorithms can be implemented by using the theory of cellular automata. An adiabatic clock scheme can be employed in the operation of a nonuniform layout of QCA to carry out general logic functions [92]. The architectural issues of a cellular array have been discussed in [93] for the implementations of quantum cellular automata (QCA) and resonant tunneling diodes (RTDs).

Cellular nonlinear networks (CNNs) represent a circuit architecture that is capable of high-speed parallel signal processing [88]. A cellular nonlinear network (CNN) is usually an array of identical dynamical systems, or cells, and has mainly local interactions within a finite radius and analog signals as state variables. As a real-time signal processing architecture, CNNs have important applications in image processing and pattern recognition. If local memories are attached, a CNN can be used to build a universal CNN machine, which is as universal as a Turing machine [94]. Because of the local connectivity, which is independent of the number of cells, the CNN architecture is in principle scalable and reliable. The potential applications of CNNs using resonant tunneling diodes (RTDs) [95], single electron transistors (SETs) [96] and tunneling phase logic [97] have been investigated. A quantum CNN has been proposed for the use of quantum dots by exploring their local quantum dynamics and global interactions [98]. In Chapter 5, we present a classical cellular (SIMD) array [42] and a quantum CNN architecture [44] based on superconducting circuits of Josephson junctions. In the quantum CNN architecture, the quantum dynamics of the Josephson circuit is formulated as the state dynamics of a CNN cell and the quantum states of neighboring cells interact with each other only via classical couplings, which distinguishes a quantum CNN architecture from a quantum computer.

2.2.3 Quantum computers

Classical computing models derived from the Turing Machine operate in two distinguishable states — False or True, or simply 0 or 1, and produce a deterministic output. Quantum mechanics however tells us that if a bit can be in one or the other of two distinguishable states, then it can also exist in coherent superpositions of these states [99]. Inspired by the laws of physics that are ultimately quantum mechanical, Deutsch proposed a computing model working upon the principles of quantum mechanics in 1985 [100]. There came the concept of quantum computer. Because of the quantum mechanical superpositions, which suggest a massive parallelism in computation, a quantum computer may be more powerful than any classical computer [101].

In 1994 Shor discovered a quantum algorithm for factorization that is exponentially faster than any known classical algorithm [102]. This algorithm would have immediate applications in cryptography, e.g. in the quick determination of keys to codes such as RSA. There are also other algorithms, such as fast searching [103] and equation solving [104], which suggest that quantum computers could perform certain tasks that are intractable for classical computers. Various physical systems have been proposed to realize a quantum computer, including those using nuclear magnetic resonance (NMR), optical photons, optical cavities, ion traps and solid-state quantum systems [105]. A 5-bit quantum computer for the factoring of 15 has been experimentally realized using NMR [106].

Decoherence is a major issue for quantum computing [107]. Quantum bits or qubits are extremely sensitive to the perturbations from their external environment, and thus may lose their quantum properties before any operation is performed. Among various proposed devices, mesoscopic superconducting circuits of Josephson junctions, produced by modern lithography, appear promising for integration in electronic circuits and for large-scale applications [37], [38]. Recently, the coherent superposition of two macroscopic persistent-current states on a superconducting Josephson circuit has been observed [39], and the coherent quantum dynamics of this Josephson flux qubit has been demonstrated [40]. A sufficiently high quality factor of quantum coherence has been obtained in a superconducting tunnel junction circuit [108]. This may imply that decoherence need not be an obstacle in building quantum computers with macroscopic Josephson circuits [109]. The superconducting circuits of Josephson junctions may be well suited for the realization of an array-based quantum computer architecture [110]. The issues of quantum computing with superconducting circuits of Josephson junctions are briefly presented in Chapter 5 [43].

Chapter 3

Fault-Tolerance in Nanocomputers: The Multiplexing Approach

3.1 Introduction

[1]This chapter presents an evaluation of the NAND multiplexing technique as originally introduced by von Neumann [70]. Our evaluation leads to the possibility at calculating optimal redundancies for nanoelectronic system designs, using statistical analysis of chains of stages, each of which contains many NAND circuits in parallel. Basically, a single NAND (or NOR) gate design is sufficient for the implementation of a complex digital computer. Currently, logic gates are made of reasonably reliable Field Effect Transistor (FET) circuits, future logic circuits may however be built up from less reliable devices, among which the Single Electron Tunnelling (SET) technology is one of the most likely circuit candidates. In order to make future systems based on nanometer-scale devices reliable, the design of fault-tolerant architectures will be necessary.

In the 1950s von Neumann initiated the study of using redundant components to obtain reliable synthesis from unreliable components [70]. He first addressed the question that, given a malfunction probability of ε for unreliable basic gates, can a network be constructed from these gates to compute a Boolean function that deviates with a probability of at most δ while $\delta < 1/2$? The main features of von Neumann's study are that the construction is only possible when the failure probability per gate has a limit strictly smaller than 1/2, that the minimum must not be less than ε, i.e. $\delta \geq \varepsilon$ for all possible ε, and that the network of unreliable gates may have greater depth (a measure of the layers of gates in a network) than a network of reliable gates computing the same function. It has later been shown by others that ε is bounded by 1/2 and that computations with failures due to noise proceed more slowly than in the absence of failures, since a fraction of the layers has to be devoted to correction [111], [112].

In order to improve these results, von Neumann went on assessing the reliability of a network of unreliable components by expanding the size of the network, namely, the multiplexing technique [70]. In this construction, von Neumann considered two sets of basic logic circuits, the Majority Voting and NAND logic. Each logic gate was duplicated N times,

---

[1] The content of this chapter has been published in [45] and [46].

and each input was replaced by a bundle of N lines, thus producing a bundle of N outputs. For NAND logic, the inputs from the first bundle are randomly paired with those from the second bundle to form the input pairs of the duplicated NANDs. Instead of requiring all or none of the lines of the output bundle producing correct answers, a certain critical (or threshold) level Δ is set: $0 < \Delta < 1/2$. A number of larger than $(1 - \Delta)N$ lines carrying the correct signal is interpreted as a positive state of the bundle and a number of less than ΔN lines carrying the correct signal is considered as a negative state. By using a massive duplication of unreliable components, von Neumann concluded that the construction can be reliable with a high probability if the failure probability of the gates is sufficiently small. This construction however requires a large amount of redundancy (N is no less than 10^3), which makes the theory of little use in practice.

As to computational complexity, von Neumann came to the conclusion that a function computed by a network of n reliable gates could be computed by a network of $O(n \log n)$ unreliable gates. In 1977 Dobrushin and Ortyukov provided a rigorous proof to improve von Neumann's heuristic result, showing that logarithmic redundancy is actually sufficient for any Boolean function [113] and, at least for certain Boolean functions, necessary [114]. This argument was later strengthened by Pippenger, Stamoulis and Tsitsiklis [115]. In the 1980s, Pippenger proved that a variety of Boolean functions may be computed reliably by noisy networks requiring only constant multiplicative redundancy [116]. It has also been shown that the complexity measures could be affected by at most constant multiplicative factors when the sets of Boolean functions or the error bounds are changed [117]. For a good literature review on this respect, please refer to [118].

Since nanometer-scale devices will be much smaller than current CMOS devices, the device failure rate increases due to the limit of manufacturing and less amiable operating environments. The unreliability of devices is crucial in that in some cases it prevents promising nanometer-scale devices from being used in any large-scale applications, such as the Single Electron Tunnelling (SET) technology influenced by random background charges [14]. We seek architecture solutions for the integration of unreliable nanoelectronic devices. In this chapter von Neumann's NAND multiplexing is reviewed and extended to a low degree of redundancy; the stochastic Markov nature in the heart of the system is discovered and studied, leading to a comprehensive fault-tolerant theory. The problem of the random background charges in SET circuits is addressed to study a system based on NAND multiplexing as a fault-tolerant architecture for the integration of unreliable nanometer-scale devices.

The structure of the chapter is as follows. In section 2, von Neumann's NAND multiplexing theory is briefly reviewed and, in section 3, it is extended to a low degree of redundancy. We then study the stochastic Markov characteristics of multi-stage multiplexing systems in section 4. In section 5 we present a discussion. In section 6 the application of NAND multiplexing in a SET based nanoelectronic computer architecture is presented. Section 7 summarizes this chapter. This chapter is based on [45] and [46].

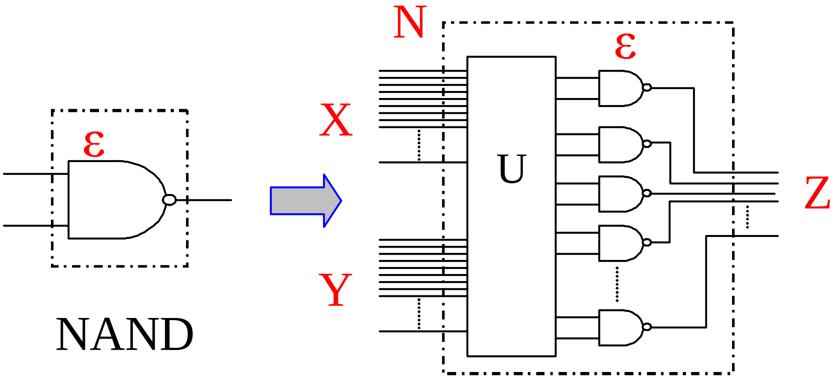


Figure 3.1: The scheme of NAND multiplexing technique.

3.2 von Neumann's theory on NAND multiplexing

3.2.1 A NAND multiplexing unit

The structure and the question

Consider a NAND gate. Replace each input of the NAND gate as well as its output by a bundle of N lines, and duplicate the NAND N times, as shown in Figure 3.1. The rectangle U is supposed to perform a "random permutation" of the input signals in the sense that each signal from the first input bundle is randomly paired with a signal from the second input bundle to form the input pair of one of the duplicated NANDs.

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

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

The theory without errors

Assume first that $\varepsilon = 0$. Let Z^c be the complementary set of Z. Let $(\bar{p}, \bar{q}, \bar{r})$ be the numbers of elements of (X, Y, Z^c) respectively, so that $\bar{p} = \bar{x} \cdot N$, $\bar{q} = \bar{y} \cdot N$ and $\bar{r} = (1 - \bar{z}) \cdot N$. The problem is then to determine the distribution of the stochastic variable $\bar{r}$ in terms of the given $\bar{p}$ and $\bar{q}$, i.e., to determine the probability of a given $\bar{r}$ in combination with given $\bar{p}$ and $\bar{q}$.

Considering that $Z^c = X \cdot Y$, then Z^c, $X - Z^c$, $Y - Z^c$ and $N - X - Y + Z^c$ form the

X	Y	Z	
....	....	....	....
0	0	1	
0	0	1	$N - X - Y + Z^c$
0	0	1	$N - \bar{p} - \bar{q} + \bar{r}$
....	....	....	....
0	1	1	
0	1	1	$Y - Z^c$
0	1	1	$\bar{q} - \bar{r}$
0	1	1	
....	....	....	....
1	0	1	$X - Z^c$
1	0	1	$\bar{p} - \bar{r}$
....	....	....	....
1	1	0	
1	1	0	
1	1	0	Z^c
1	1	0	$\bar{r}$
1	1	0	
....	....	....	....
$\bar{p} = 7$	$\bar{q} = 9$	$\bar{r} = 5$	

Table 3.1: A possible realisation with N=14.

four disjoint sub-sets of the entire output bundle, with $\bar{r}$, $\bar{p}-\bar{r}$, $\bar{q}-\bar{r}$ and $N-\bar{p}-\bar{q}+\bar{r}$ elements, see Table 3.1.

At the input side, there are

$$C_X = \binom{N}{\bar{p}} = \frac{N!}{\bar{p}! \cdot (N-\bar{p})!} \tag{3.1}$$

possible permutations of the set X with $\bar{p}$ elements and

$$C_Y = \binom{N}{\bar{q}} = \frac{N!}{\bar{q}! \cdot (N-\bar{q})!} \tag{3.2}$$

possible permutations of a the set Y with $\bar{q}$ elements. These sets offer at the output side the following joint permutations

$$\begin{aligned} C_O &= \binom{N}{\bar{r}} \cdot \binom{N-\bar{r}}{\bar{p}-\bar{r}} \cdot \binom{N-\bar{r}-\bar{p}+\bar{r}}{\bar{q}-\bar{r}} \\ &= \frac{N!}{\bar{r}! \cdot (\bar{p}-\bar{r})! \cdot (\bar{q}-\bar{r})! \cdot (N-\bar{p}-\bar{q}+\bar{r})!}. \end{aligned} \tag{3.3}$$

C_O is given by Multinomial coefficients. The probability P of Z^c having $\bar{r}$ elements is then:

$$P = \frac{C_O}{C_X \cdot C_Y} = \frac{p!(N-p)!q!(N-q)!}{r!(p-r)!(q-r)!(N-p-q+r)!N!} \tag{3.4}$$

Substituting the $\bar{x}, \bar{y}, \bar{z}$ expressions for $\bar{p}, \bar{q}, \bar{r}$ and using Stirling's formula give

$$P \sim \frac{1}{\sqrt{2\pi N}} \sqrt{\bar{a}} e^{-\bar{\theta} N} \tag{3.5}$$

with

$$\bar{a} = \frac{\bar{x}(1-\bar{x})\bar{y}(1-\bar{y})}{(\bar{z}+\bar{x}-1)(\bar{x}+\bar{y}-1)(1-\bar{z})(2-\bar{x}-\bar{y}-\bar{z})} \tag{3.6}$$

$$\begin{aligned} \bar{\theta} = {} & (\bar{z}+\bar{x}-1)\ln(\bar{z}+\bar{x}-1) + (\bar{z}+\bar{y}-1)\ln(\bar{z}+\bar{y}-1) \\ & +(1-\bar{z})\ln(1-\bar{z}) + (2-\bar{x}-\bar{y}-\bar{z})\ln(2-\bar{x}-\bar{y}-\bar{z}) \\ & -\bar{x}\ln\bar{x} - (1-\bar{x})\ln(1-\bar{x}) - \bar{y}\ln\bar{y} - (1-\bar{y})\ln(1-\bar{y}). \end{aligned} \tag{3.7}$$

From this we have

$$\frac{\partial\bar{\theta}}{\partial\bar{z}} = \ln\frac{(\bar{z}+\bar{x}-1)(\bar{z}+\bar{y}-1)}{(1-\bar{z})(2-\bar{x}-\bar{y}-\bar{z})} \tag{3.8}$$

$$\frac{\partial^2\bar{\theta}}{\partial^2\bar{z}} = \frac{1}{\bar{z}+\bar{x}-1} + \frac{1}{\bar{z}+\bar{y}-1} + \frac{1}{1-\bar{z}} + \frac{1}{2-\bar{x}-\bar{y}-\bar{z}} \tag{3.9}$$

and hence $\bar{\theta} = 0$ and$\frac{\partial\bar{\theta}}{\partial\bar{z}} = 0$ for

$$\bar{z} = 1 - \bar{x}\bar{y} \text{ [70].} \tag{3.10}$$

Consequently $\bar\theta > 0$ for all valid $\bar z$ in the problem but $\bar z = 1 - \bar x\bar y$, as $\frac{\partial^2\bar\theta}{\partial^2\bar z} > 0$. This implies that for all $\bar z$ that significantly deviate from $1 - \bar x\bar y$, i.e. $\bar z \neq 1 - \bar x\bar y$, P tends to go to 0 very rapidly when N grows large. It is therefore sufficient to evaluate for $\bar z \sim 1 - \bar x\bar y$.

If $\bar z \sim 1 - \bar x\bar y$, then

$$\bar a \sim \frac{1}{\bar x(1-\bar x)\bar y(1-\bar y)}, \quad \bar\theta \sim \frac{(\bar z - (1-\bar x\bar y))^2}{2\bar x(1-\bar x)\bar y(1-\bar y)} \tag{3.11}$$

and hence

$$P \sim \frac{1}{\sqrt{2\pi\bar x(1-\bar x)\bar y(1-\bar y)N}} e^{-\frac{(\bar z-(1-\bar x\bar y))^2}{2\bar x(1-\bar x)\bar y(1-\bar y)}N} \tag{3.12}$$

As N is assumed to be very large, the set Z with $\bar z \cdot N$ elements is so dense that a continuous domain can be assumed. The distribution of $\bar z$ can then be described by a probability density $\bar\sigma$, with $P = \bar\sigma d\bar z$. Since the minimum variance of $\bar z$ is $1/N$, i.e. $d\bar z = 1/N$, we have $\bar\sigma = PN$.

Therefore:

$$\bar\sigma \sim \frac{1}{\sqrt{2\pi}\sqrt{\bar x(1-\bar x)\bar y(1-\bar y)/N}} e^{-\frac{1}{2}\left(\frac{\bar z-(1-\bar x\bar y)}{\sqrt{\bar x(1-\bar x)\bar y(1-\bar y)/N}}\right)^2} \tag{3.13}$$

This means that $\bar z$ is approximately normally distributed with mean $1 - \bar x\bar y$ and a dispersion (standard deviation) $\sqrt{\bar x(1-\bar x)\bar y(1-\bar y)/N}$. The normal distribution decreases rapidly when $\bar z$ is near to $1 - \bar x\bar y$.

As

$$\bar z = (1 - \bar x\bar y) + \bar\delta\sqrt{\bar x(1-\bar x)\bar y(1-\bar y)/N}, \tag{3.14}$$

with $\bar\delta$ a stochastic variable, normally distributed with mean 0 and standard deviation 1, it can be seen that $\bar z$ is approximately given by $1 - \bar x\bar y$, i.e. $\bar z = 1 - \bar x\bar y$ with a high probability, when N is large.

The theory with errors

Next, consider the error rate of a NAND $\varepsilon \neq 0$. The number of errors committed by the N logic units is then a random variable that is approximately normally distributed with mean εN and standard deviation $\sqrt{\varepsilon\,(1-\varepsilon)\,N}$.

Assume that the number of actual stimulated output lines now is $\bar r'$. For the $\bar r$ correctly stimulated outputs, hence, each faulty NAND effectively reduces $\bar r'$ by one line in the output bundle. Thus also the number of errors in the output bundle is approximately normally distributed, with mean $\varepsilon\bar r$ and standard deviation $\sqrt{\varepsilon\,(1-\varepsilon)\,\bar r}$. For the $N - \bar r$ not stimulated outputs, each error increases $\bar r'$ by one. The number of these errors is also approximately normally distributed with mean $\varepsilon(N-\bar{r})$ and standard deviation $\sqrt{\varepsilon\left(1-\varepsilon\right)\left(N-\bar{r}\right)}$. Thus $\bar{r}'-\bar{r}$ is also approximately normally distributed with mean

$$\varepsilon\left(N-\bar{r}\right)-\varepsilon\bar{r}=\varepsilon(N-2\bar{r}) \tag{3.15}$$

and standard deviation

$$\sqrt{\left(\sqrt{\varepsilon(1-\varepsilon)\bar{r}}\right)^2+\left(\sqrt{\varepsilon(1-\varepsilon)(N-\bar{r})}\right)^2}=\sqrt{\varepsilon(1-\varepsilon)N}. \tag{3.16}$$

Consequently,

$$\bar{r}'=\bar{r}+2\varepsilon(\frac{N}{2}-\bar{r})+\bar{\delta}\sqrt{\varepsilon(1-\varepsilon)N}, \tag{3.17}$$

where $\bar{\delta}$ is normally distributed with mean 0 and standard deviation 1.

From above (actually $\bar{z}=\bar{r}/N$ here and let $\bar{z}'=\bar{r}'/N$), we have

$$\bar{z}'=\bar{z}+2\varepsilon(\frac{1}{2}-\bar{z})+\bar{\delta}\sqrt{\varepsilon\left(1-\varepsilon\right)/N}. \tag{3.18}$$

Finally, taking (3.14), we have

$$\bar{z}'=(1-\bar{x}\bar{y})+2\varepsilon(\bar{x}\bar{y}-\frac{1}{2})+\bar{\delta}\sqrt{((1-2\varepsilon)^2\bar{x}(1-\bar{x})\bar{y}(1-\bar{y})+\varepsilon(1-\varepsilon))/N}, \tag{3.19}$$

with $\bar{\delta}$ a stochastic variable, normally distributed with mean 0 and standard deviation 1.

For large N, von Neumann thus concluded that $\bar{z}$ is a stochastic variable, approximately normally distributed. He also gave an upper bound for the failure probability per gate that can be tolerated, $\varepsilon_0=0.0107$, when $\Delta=0.07$. In other words, if $\varepsilon\geq\varepsilon_0$, the failure probability of the NAND multiplexing network (with the threshold $\Delta=0.07$) will be larger than a fixed, positive lower bound, no matter how large a bundle size N is used.

3.2.2 The restorative unit

If we assume that the two input bundles have almost the same stimulated or non-stimulated levels (which is likely in circuits), i.e. $\bar{x}==\bar{y}$, it is then intuitively known that

- if almost all lines of one input bundle are stimulated and almost all lines of the other bundle are non-stimulated, then the error probability of the output bundle (NAND; hence the probability of the number of lines that are non-stimulated) will approximately be the same as the error probability in either one of the input bundles;
- if almost all lines of both input bundles are non-stimulated, then the error probability of the output bundle (NAND; hence the probability of the number of lines that are non-stimulated) will be smaller than the error probability in either one of the input bundles;
- if almost all lines of both input bundles are stimulated, then the error probability of the output bundle (NAND; hence the probability of the number of lines that are stimulated) will be larger than the error probability in either one of the input bundles.

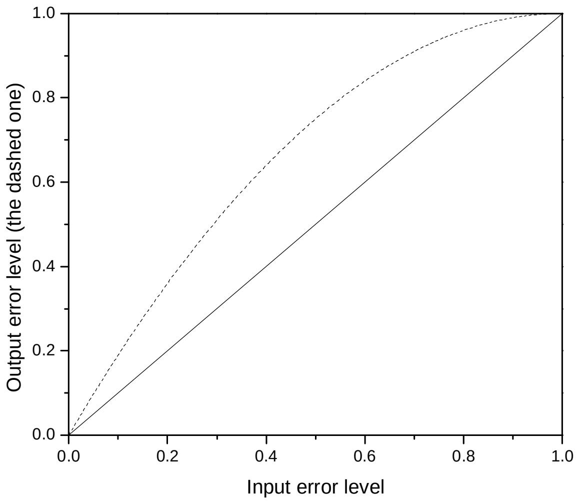


Figure 3.2: The function of a NAND multiplexing unit with non-stimulated inputs as errors.

For this last case, we need a unit that restores the original stimulation level without destroying the NAND function. This can be seen as follows.

If $\bar{x}N$ of the N incoming lines are stimulated, then the probability of a NAND being stimulated (by at least one non-stimulated input) is approximately (assuming ε is small or $\varepsilon = 0$)

$$\bar{z}' = 1 - \bar{x}^2. \tag{3.20}$$

This indicates that, at a high probability, approximately $\bar{z}'N$ outputs will be stimulated, provided N is large.

Let $\bar{x}'$ be the non-stimulated (error) level of the inputs, i.e. $\bar{x}' = 1 - \bar{x}$. Replacing $\bar{x}$ with $\bar{x}'$ in (3.20):

$$\bar{z}' = 2\bar{x}' - \bar{x}'^2 \tag{3.21}$$

The function (3.21) is plotted in Figure 3.2. It shows that, when the error level $\bar{x}'$ varies from 0 to 1/2, $\bar{z}'$ is monotonically increasing and $\bar{z}' \geq \bar{x}'$. This means that the non-stimulated inputs give rise to more stimulated outputs, i.e. the error level is amplified. If, for example, the original error probability was 0.2, the output error probability is 0.36. Consequently, we need a unit that restores the original stimulation level.

The restorative unit can be made by using the same NAND multiplexing technique while duplicating the outputs of the executive unit as the inputs. This is shown in Figure 3.3.

If $\bar{x}N$ of the N incoming lines are stimulated and ε is very small, the probability of the output of the restorative unit being stimulated $\bar{z}'$ is approximately given by (3.20).

We now plot $\bar{z}'$ against $\bar{x}$ as in Figure 3.4. It shows that instead of restoring the excitation level, the restorative unit inverts the output of the executive unit, i.e. it transforms the most stimulated bundles to most non-stimulated and vice versa. In addition it produces for a value

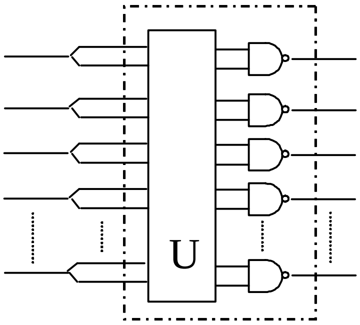


Figure 3.3: A restorative unit.

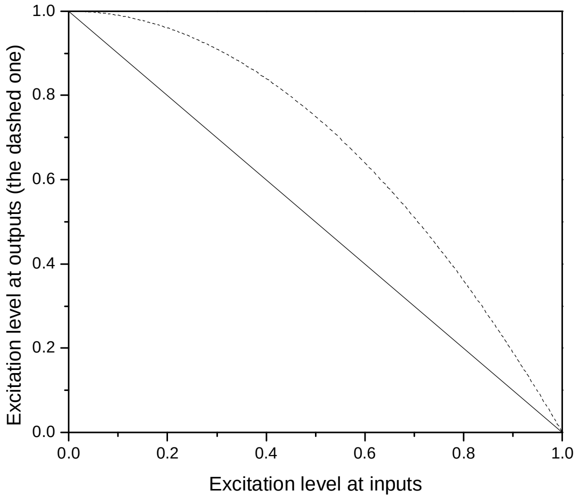


Figure 3.4: The function of a single restorative unit.

of $\bar{x}$ close to 1 a value of $\bar{z}'$ less close to 0, and for $\bar{x}$ close to 0 a $\bar{z}'$ much closer to 1. This suggests that the operation needs to be iterated to construct a proper restoration.

Now let the restoring unit consist of two of the restoration units in series, as shown in Figure 3.5. This unit transforms an input excitation level $\bar{x}N$ into an output excitation level of (approximately):

$$\bar{z}' = 1 - (1 - \bar{x}^2)^2 = 2\bar{x}^2 - \bar{x}^4 \tag{3.22}$$

The $\bar{z}'$ is plotted against $\bar{x}$ as shown in Figure 3.6, with $0 \le \bar{x} \le 1$. The curve intersects the diagonal $\bar{z}' = \bar{x}$ three times at: $\bar{x} = 0$, $\bar{x}_0 = 0.618$ and $\bar{x} = 1$. If $0 < \bar{x} < \bar{x}_0$, then $0 < \bar{z}' < \bar{x}$; while $\bar{x}_0 < \bar{x} < 1$ implies that $\bar{x} < \bar{z}' < 1$. This indicates that the restorative unit brings every $\bar{x}$ nearer to either 0, when $\bar{x}$ is not larger than 0.618, or 1, when $\bar{x}$ is not smaller than 0.618. This process has the required restoring effect and hence the unit shown in Figure 3.5 gives an effective restoration mechanism.

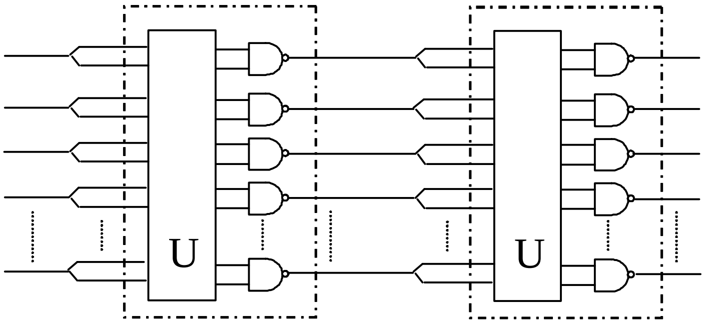


Figure 3.5: A 2-stage restorative unit.

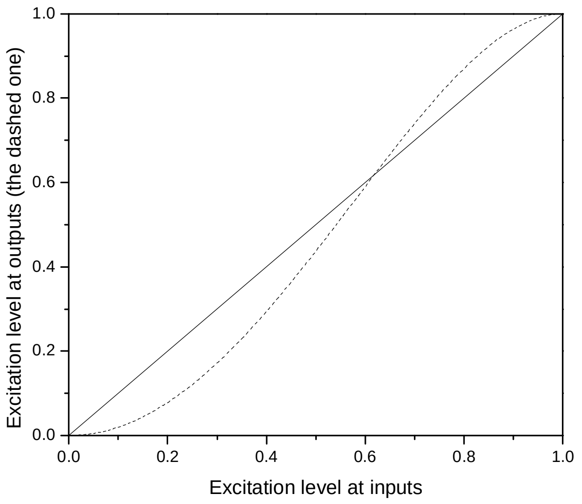


Figure 3.6: The function of a 2-stage restorative unit.

In summary, von Neumann had built a multiplexing system with two types of units, the first being the executive unit, which performs the NAND function and the second a restorative unit which annuls the degradation caused by the executive unit. The restorative unit was made by using the same NAND multiplexing technique by duplicating the outputs of the executive unit as the inputs. To keep the NAND function, the multiplexing unit was iterated to give the effective restoring mechanism, see Figure 3.7.

Here is the end of presentation of von Neumann's work; new results from our investigations start in the next section.

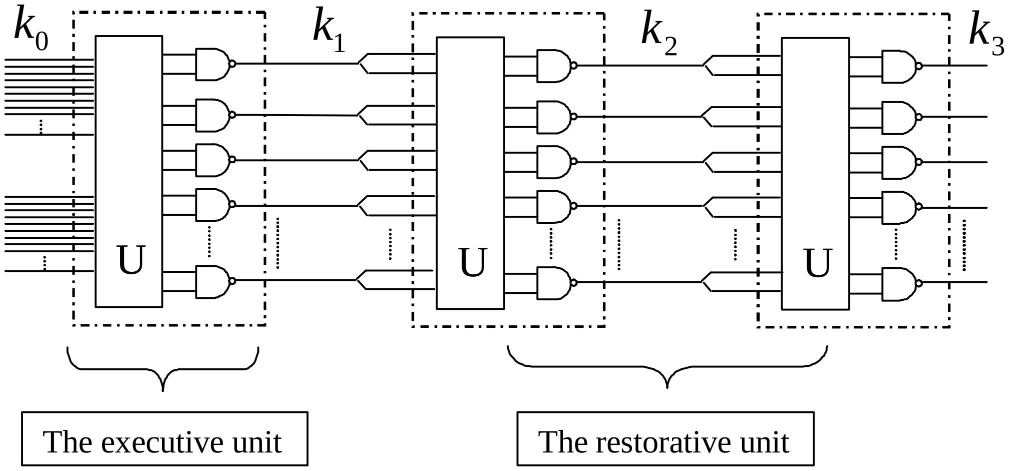


Figure 3.7: A NAND multiplexing system with the executive and restorative units.

3.3 Error distributions in a multiplexing unit — an alternative method

3.3.1 Theoretical analysis

The NAND multiplexing unit was constructed as in Figure 3.1. In this section an alternative method is given to extend the study of the NAND multiplexing technique from a high degree to a fairly low degree of redundancy.

Let us consider a single NAND gate in the NAND multiplexing scheme. If we still assume that there are $\bar{x}N$ and $\bar{y}N$ input lines stimulated, then the probability of the output of the NAND gate that is found stimulated (by at least one non-stimulated input) is approximately $\bar{z}' = 1-\bar{x}\bar{y}$ (assuming that the NAND gate is fault-free). If each NAND gate has a probability ε of making an error, the probability of its output being stimulated is given by:

$$\begin{aligned} P &= P(stimulated|NAND\ defective)P(NAND\ defective) \\ &\quad +P(stimulated|NAND\ not\ defective)P(NAND\ not\ defective). \end{aligned}$$

For gate errors of von Neumann type, this probability is:

$$\bar{z}_v = (1-\bar{x}\bar{y})(1-\varepsilon) + \bar{x}\bar{y}\varepsilon = (1-\varepsilon) - (1-2\varepsilon)\bar{x}\bar{y}. \tag{3.23}$$

For more common fault models such as Stuck-at-0 and Stuck-at-1, the probabilities become respectively

$$\bar{z}_0 = (1-\bar{x}\bar{y})(1-\varepsilon) \tag{3.24}$$

and

$$\bar{z}_1 = 1-(1-\varepsilon)\bar{x}\bar{y}. \tag{3.25}$$

For each NAND gate, thus, the probability of the output to be stimulated (event 1) is $\bar{z}$, $\bar{z} \in \{\bar{z}_v, \bar{z}_0, \bar{z}_1\}$, and the probability to be non-stimulated (event 0) is $1-\bar{z}$. For given numbers of stimulated inputs (i.e. $\bar{x}N$ and $\bar{y}N$), the probability that an output is stimulated or not is actually not independent, but rather relevant to others. When N is relatively large, however, this relevance has little significant effect such that it can be ignored. If the N NAND gates function independently, therefore, the occasion whether an output is stimulated or not in the NAND multiplexing unit can be modeled by a Bernoulli sequence. Hence the probabilities of stimulated outputs are given by the binomial distribution. The probability of k out of N outputs being stimulated is then:

$$P(k) = \binom{N}{k}\bar{z}^k(1-\bar{z})^{N-k}. \tag{3.26}$$

When N is large and $\bar{z}$ is small, the Poisson Theorem gives:

$$P(k) \approx \lim_{N\to\infty}\binom{N}{k}\bar{z}^k(1-\bar{z})^{N-k} = \frac{\bar{\lambda}^k e^{-\bar{\lambda}}}{k!}, \tag{3.27}$$

where

$$\bar{\lambda} = N\bar{z}. \tag{3.28}$$

Given N very large and $\bar{z}$ very small, therefore, the distribution of probability of k outputs from the N output lines of the NAND multiplexing unit being stimulated is approximately a Poisson distribution.

If both inputs of the NAND gates are expected to be in stimulated states, the stimulated outputs are then considered to be faulty. To evaluate the effect of faults, the probability of possible errors below an acceptable threshold level, i.e. $P(k \leq n)$, needs to be computed. Since the number of the stimulated outputs is a stochastic variable, which is described by the binomial distribution, the De Moivre-Laplace Theorem [120], when N is large and $0 < \bar{z} < 1$, applies:

$$\lim_{N\to\infty} P\{\frac{k-N\bar{z}}{\sqrt{N\bar{z}(1-\bar{z})}} \leq m\} = \int_{-\infty}^{m}\frac{1}{\sqrt{2\pi}}e^{\frac{-t^2}{2}}dt, \tag{3.29}$$

replacing

$$m = \frac{n-N\bar{z}}{\sqrt{N\bar{z}(1-\bar{z})}}, \tag{3.30}$$

then

$$P(k \leq n) \approx \int_{-\infty}^{n}\frac{1}{\sqrt{2\pi}\sqrt{N\bar{z}(1-\bar{z})}}e^{-\frac{1}{2}(\frac{t-N\bar{z}}{\sqrt{N\bar{z}(1-\bar{z})}})^2}dt. \tag{3.31}$$

Since N is very large, the set of k outputs is so dense that a continuous domain can be assumed. Let $k = \bar{u}\cdot N$ and $f(\bar{u})$ be the probability density, then $d\bar{u} = 1/N$ and $P = f(\bar{u})d\bar{u}$. The probability density of $\bar{u}$ can now be obtained as:

$$f(\bar{u}) = \frac{1}{\sqrt{2\pi}\sqrt{\bar{z}(1-\bar{z})/N}}e^{-\frac{1}{2}(\frac{\bar{u}-\bar{z}}{\sqrt{\bar{z}(1-\bar{z})/N}})^2}. \tag{3.32}$$

This shows that the probability of the number of stimulated outputs of the NAND multiplexing unit could be approximated by a normal distribution with mean $N\bar{z}$ and standard deviation $\sqrt{N\bar{z}(1-\bar{z})}$, when N is large and $0 < \bar{z} < 1$.

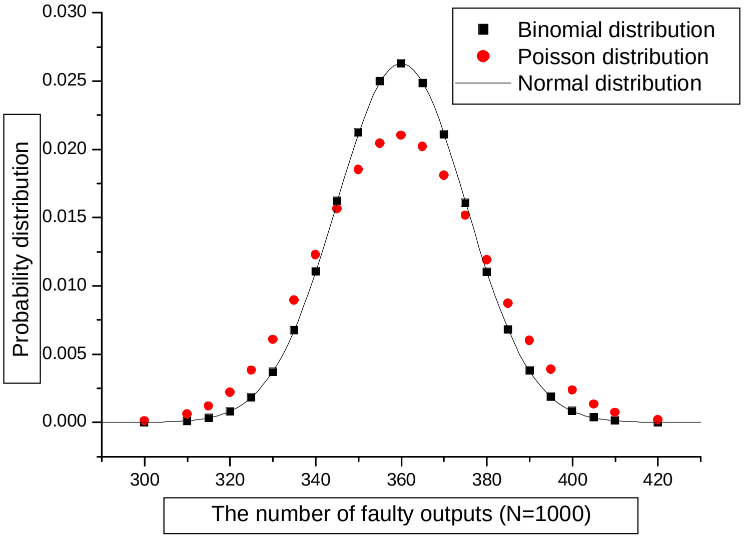


Figure 3.8: The error distributions (von Neumann: $\varepsilon = 10^{-4}$).

3.3.2 Numerical evaluation

Consider next the fault distribution of the NAND multiplexing unit for different N and ε within certain range. We assume that the largest possible error rate ε for a future nanoelectronic system is 0.1, meaning that one of ten devices is faulty on average. Consequently, the ε under investigation will be in the range of $[0, 0.1]$. We further assume that the input excitation rates are identical to each other, i.e., $\bar{x} == \bar{y}$. This is often true for circuits using similar devices.

For von Neumann faults, hence, the error probability of one output of the NAND multiplexing unit, i.e. the probability of an output line being stimulated, becomes:

$$\bar{z} = (1 - \varepsilon) - (1 - 2\varepsilon)\bar{x}^2. \tag{3.33}$$

For simplicity, we assume $\bar{x}' = 1 - \bar{x}$. Replacing $\bar{x}$ with $\bar{x}'$ in (3.33):

$$\bar{z} = (2\varepsilon - 1)\bar{x}'^2 + 2(1 - 2\varepsilon)\bar{x}' + \varepsilon. \tag{3.34}$$

For $\varepsilon \in [0, 0.1]$, the formula (3.34) is monotonically-increasing as $\bar{x}'$ varies from 0 to 0.5. For a typical $\bar{x}'$, say, 0.1, $\bar{z} \in [0.19, 0.25]$. This condition does not favor a conclusion in the direction of a Poisson distribution.

We proceed with a study on the approximation of the Poisson and the normal distribution to the binomial distribution for different sizes of the NAND multiplexing unit, i.e. for different N. We first take $N = 1000$. Specifying $\bar{x} = 0.8$ and $\varepsilon = 10^{-4}$, for different types of gate errors (von Neumann, Stuck-at-0 and Stuck-at-1), the probability (density) of the binomial, Poisson and normal distribution against the number of faulty outputs are plotted in Figures 3.8, 3.9 and 3.10.

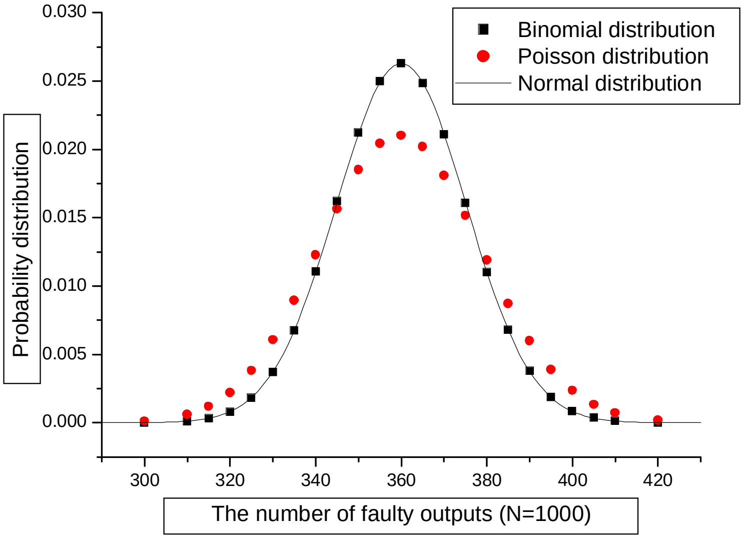


Figure 3.9: The error distributions (Stuck-at-0: $\varepsilon = 10^{-4}$).

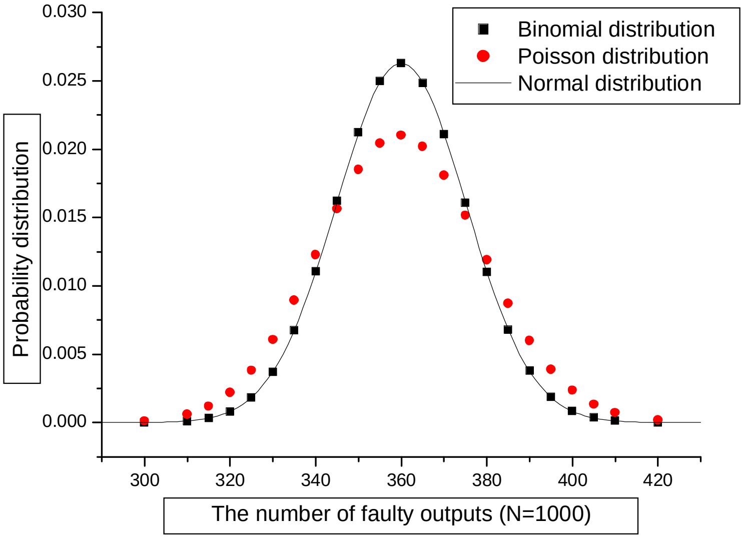


Figure 3.10: The error distributions (Stuck-at-1: $\varepsilon = 10^{-4}$).

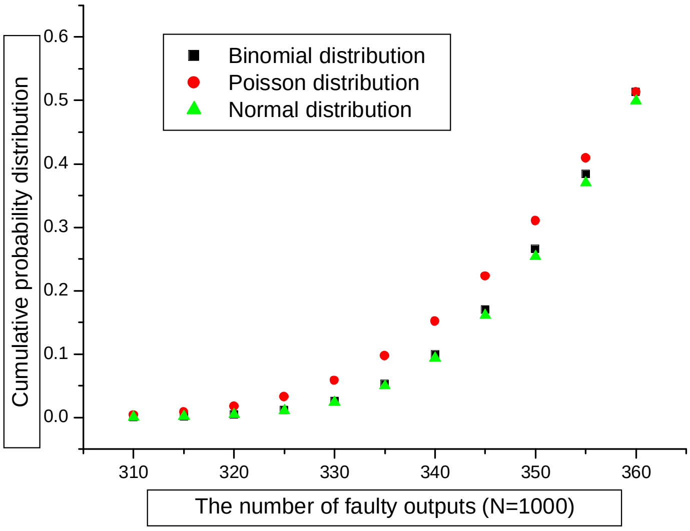


Figure 3.11: The cumulative error distributions (von Neumann: $\varepsilon = 10^{-4}$).

As the probability of possible errors below an acceptable threshold level $P(k \leq n)$ is an important feature to evaluate the approximation, the cumulative probability distribution $P(k \leq n)$ for the binomial, Poisson and normal distribution are plotted as well, in Figures 3.11, 3.12 and 3.13 for different types of gate errors.

As can be seen in both sets of the figures, the normal distribution is in good accordance with the binomial distribution, while the Poisson distribution is not. Further study shows that the approximation for the normal distribution is very well kept when $\bar{x}$ varies in the range $[0.7, 0.9]$ and ε varies in the range $[0, 0.1]$. Since the gate error rate ($\varepsilon = 10^{-4}$), compared to the input error level ($\bar{x} = 0.8$), is relatively small, the output error distribution is largely determined by the input error level. As revealed in the figures, the variance of either the probability distributions or the cumulative distributions for these three types of gate errors (von Neumann, Stuck-at-0 and Stuck-at-1) is hardly discernable.

Now consider the case that $N = 100$. We still let $\bar{x} = 0.8$ and $\varepsilon = 10^{-4}$. The fault probability distributions are shown in Figures 3.14, 3.15 and 3.16, and the cumulative distributions are in Figures 3.17, 3.18 and 3.19.

As can be seen, though the samples of probability density of the normal distribution fits in quite well with the binomial distribution, the discrete binomial distribution is no longer appropriately described by the normal distribution in terms of the cumulative distribution, due to the declined bundle size N. This indicates that neither normal nor Poisson gives good approximation to the binomial for cumulative probability distribution.

Hence, in terms of probability (density) and cumulative probability distribution, the faulty probability of the NAND multiplexing unit can be given by the normal distribution when N is large (typically $N > 1000$). For modest N, the error distribution can be described with the binomial distribution. Obviously, the larger N is, the better the approximation.

If N is very small, however, neither a normal nor a binomial distribution is appropriate for

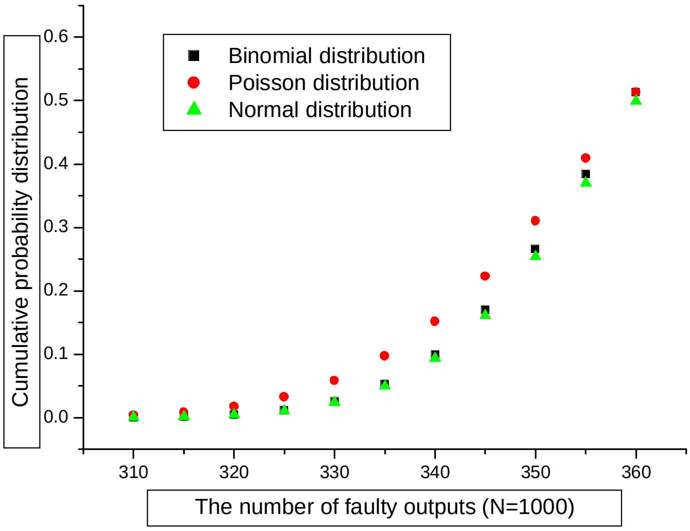


Figure 3.12: The cumulative error distributions (Stuck-at-0: $\varepsilon = 10^{-4}$).

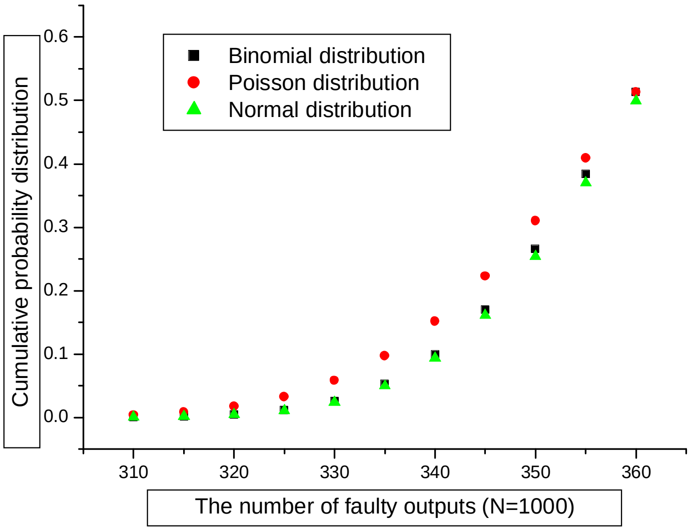


Figure 3.13: The cumulative error distributions (Stuck-at-1: $\varepsilon = 10^{-4}$).

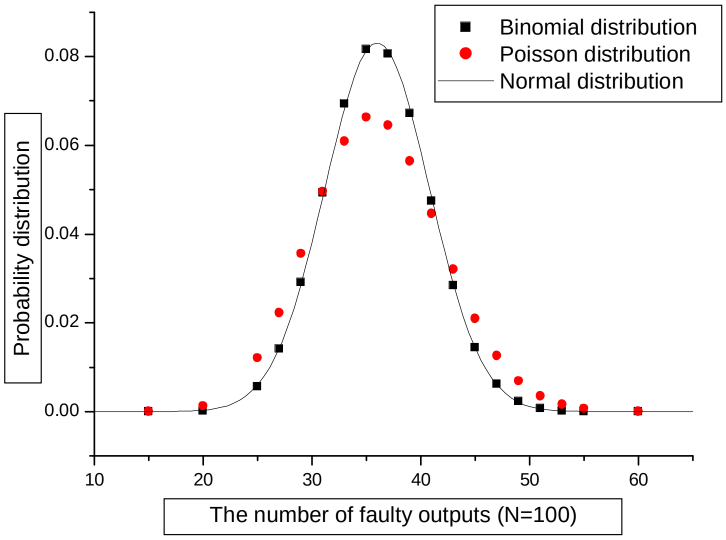


Figure 3.14: The error distributions (von Neumann: $\varepsilon = 10^{-4}$).

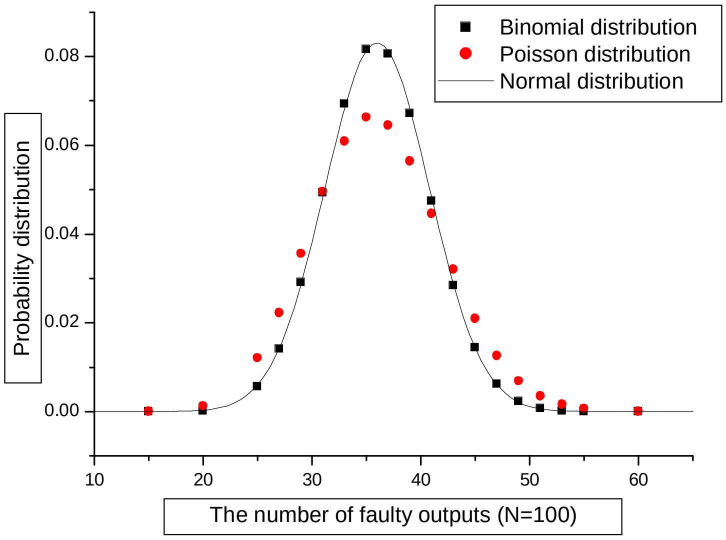


Figure 3.15: The error distributions (Stuck-at-0: $\varepsilon = 10^{-4}$).

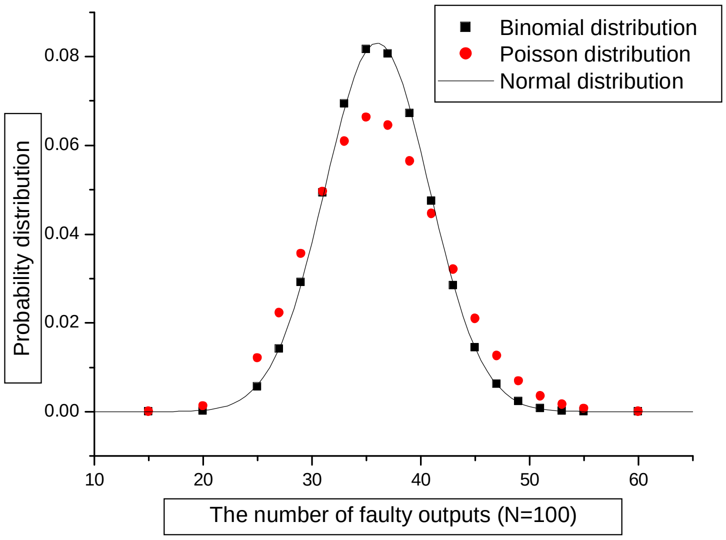


Figure 3.16: The error distributions (Stuck-at-1: $\varepsilon = 10^{-4}$).

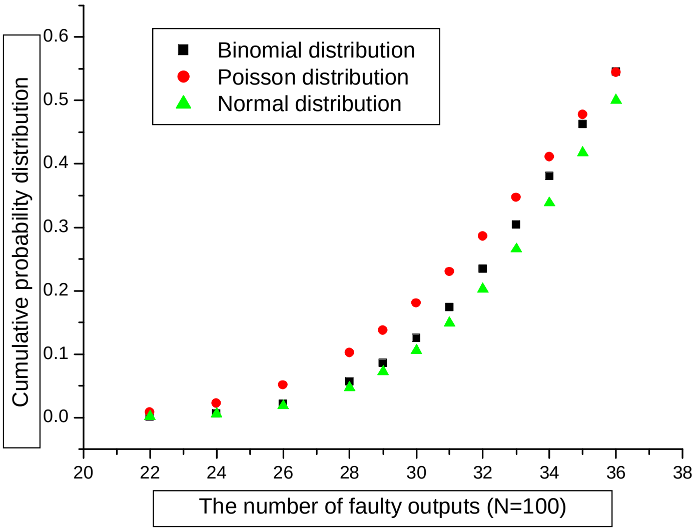


Figure 3.17: The cumulative error distributions (von Neumann: $\varepsilon = 10^{-4}$).

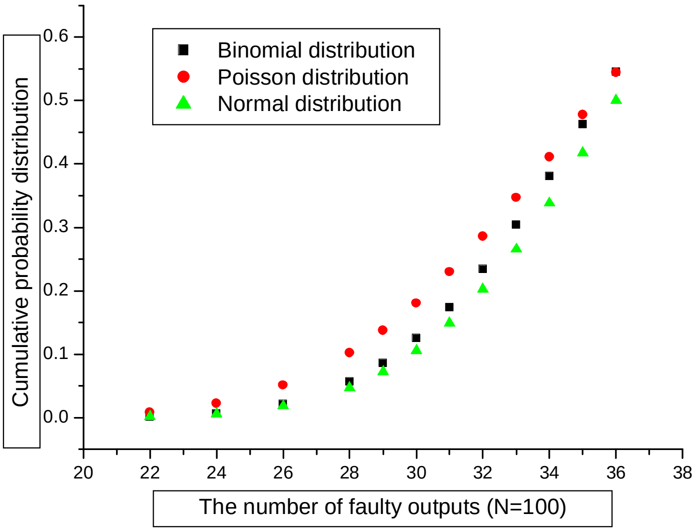


Figure 3.18: The cumulative error distributions (Stuck-at-0: $\varepsilon = 10^{-4}$).

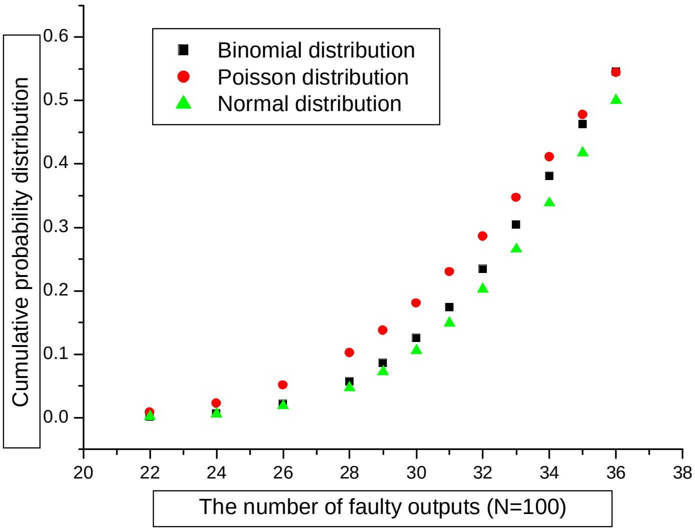


Figure 3.19: The cumulative error distributions (Stuck-at-1: $\varepsilon = 10^{-4}$).

the modelling of the output errors in the NAND multiplexing unit, because of the increased significance of the input errors' dependency on the reduced bundle size. In this case the fault analysis results in a significant increase in complexity and we propose a simulation based method to investigate the fault-tolerance. This is presented in Chapter 4.

3.4 Error distributions in multi-stage systems

3.4.1 For modest N

The 3-stage multiplexing system

We have discussed the set-up of a NAND multiplexing system with an executive unit and a restorative unit, as depicted in Figure 3.7. If there are k_0 of the N incoming lines stimulated for both inputs of the executive unit in the NAND multiplexing system, and each NAND gate has a definite probability ε of making an error (the von Neumann type; without the loss of generality), according to equations (3.23) and (3.26) the probabilities of the stimulated outputs k_1, k_2 and k_3 of the three multiplexing units in cases of the corresponding stimulated inputs k_0, k_1 and k_2 are given by:

$$P_1(k_1|k_0) = \binom{N}{k_1} \bar{z}_1^{k_1}(k_0)(1 - \bar{z}_1(k_0))^{N-k_1}, \tag{3.35}$$

$$P_2(k_2|k_1) = \binom{N}{k_2} \bar{z}_2^{k_2}(k_1)(1 - \bar{z}_2(k_1))^{N-k_2}, \tag{3.36}$$

$$P_3(k_3|k_2) = \binom{N}{k_3} \bar{z}_3^{k_3}(k_2)(1 - \bar{z}_3(k_2))^{N-k_3}, \tag{3.37}$$

where

$$\bar{z}_1(k_0) = (1 - \varepsilon) - (1 - 2\varepsilon)(\frac{k_0}{N})^2, \tag{3.38}$$

$$\bar{z}_2(k_1) = (1 - \varepsilon) - (1 - 2\varepsilon)(\frac{k_1}{N})^2, \tag{3.39}$$

$$\bar{z}_3(k_2) = (1 - \varepsilon) - (1 - 2\varepsilon)(\frac{k_2}{N})^2. \tag{3.40}$$

Noting the stochastic nature of k_1, k_2 and k_3, the probabilities of them being stimulated in all cases are then obtained by:

$$P_1(k_1) = \sum_{k_0=0}^{N} P_1(k_1|k_0)P_1(k_0), \tag{3.41}$$

$$P_2(k_2) = \sum_{k_1=0}^{N} P_2(k_2|k_1)P_1(k_1), \tag{3.42}$$

$$P_3(k_3) = \sum_{k_2=0}^{N} P_3(k_3|k_2)P_2(k_2). \tag{3.43}$$

In equations (3.41), (3.42) and (3.43) the most significant parts are the conditional probabilities, $P_3(k_3|k_2)$, $P_2(k_2|k_1)$ and $P_1(k_1|k_0)$ (which is $P_1(k_1)$ with fixed k_0). For any identical set of inputs and outputs, all the three conditional probabilities are the binomial distribution with identical parameters, i.e.,

$$P(k_l|k_{l-1}) = \binom{N}{k_l} \bar{z}^{k_l}(k_{l-1})(1-\bar{z}(k_{l-1}))^{N-k_l}, \tag{3.44}$$

where

$$\bar{z}(k_{l-1}) = (1-\varepsilon) - (1-2\varepsilon)(\frac{k_{l-1}}{N})^2. \tag{3.45}$$

Therefore a $(N+1) \times (N+1)$ matrix $\mathbf{\Psi}$, whose elements are $P(k_l|k_{l-1})$, $k_l, k_{l-1} \in [0, 1, 2, ...N]$, can be made as shown in (3.46), so that all conditional probabilities for any set of (k_l, k_{l-1}) are included.

$$\mathbf{\Psi} = \begin{bmatrix} P(0|0) & P(1|0) & P(2|0) & .... & P(N|0) \\ P(0|1) & P(1|1) & P(2|1) & .... & P(N|1) \\ P(0|2) & P(1|2) & P(2|2) & .... & P(N|2) \\ .... & .... & .... & .... & .... \\ P(0|N) & P(1|N) & P(2|N) & .... & P(N|N) \end{bmatrix} \tag{3.46}$$

Accordingly, given a fixed input distribution:

$$\mathbf{P}_0 = [p_0,\ p_1,\ p_2\ ...\ p_N], \tag{3.47}$$

where p_i is the probability of i inputs being stimulated, the stimulated output distributions of (3.41), (3.42) and (3.43) are given by:

$$\mathbf{P}_1 = [P_1(0), P_1(1), ...P_1(N)] = \mathbf{P}_0\mathbf{\Psi}, \tag{3.48}$$
$$\mathbf{P}_2 = [P_2(0), P_2(1), ...P_2(N)] = \mathbf{P}_0\mathbf{\Psi}^2, \tag{3.49}$$
$$\mathbf{P}_3 = [P_3(0), P_3(1), ...P_3(N)] = \mathbf{P}_0\mathbf{\Psi}^3. \tag{3.50}$$

If $N = 100$ and $\varepsilon = 10^{-2}$, for example, the output error distributions of the 3 stages are depicted as Figure 3.20, given 90% of input lines being stimulated.

As can be seen in Figure 3.20, the error level is amplified after the first stage of the multiplexing, so that the error rates are distributed at the scale of approximately 10% ~ 30%. The error distributions are then shifted to the other side of the diagram by the second stage, and, after the third stage, the correct output distribution is elevated to a new level. This is in agreement with previous discussion.

A stochastic Markov chain

The number of stimulated outputs of each NAND multiplexing stage is actually a stochastic variable and its state space is $A = [0, 1, 2, ...N-1, N]$. If we name this variable $\bar{\xi}_n$, where n

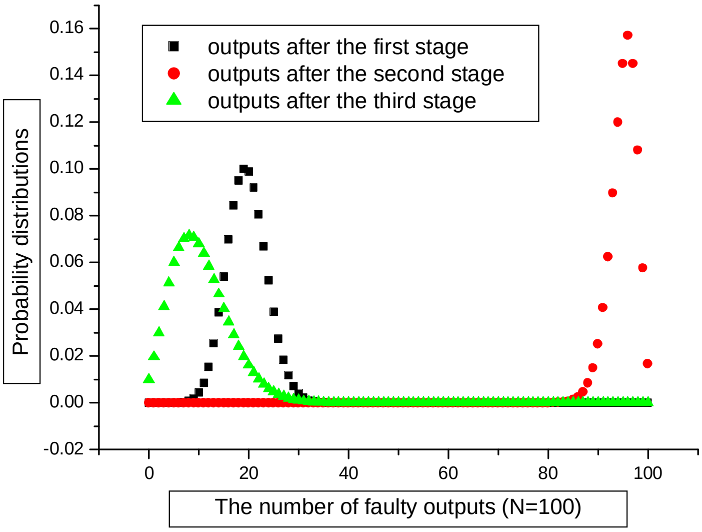


Figure 3.20: The output error distributions in a 3-stage multiplexing system ($\varepsilon = 10^{-2}$).

is the index of the multiplexing unit, the evolution of $\bar{\xi}_n$ in the NAND multiplexing system is a stochastic process. With fixed N and ε, the distribution of $\bar{\xi}_n$ for every n is totally determined by the number of stimulated inputs of the nth multiplexing unit. This can be mathematically described by:

$$\begin{aligned} P(\bar{\xi}_n &\in A|\bar{\xi}_1 = k_1, \bar{\xi}_2 = k_2, ...\bar{\xi}_{n-1} = k_{n-1}) \\ &= P(\bar{\xi}_n \in A|\bar{\xi}_{n-1} = k_{n-1}). \end{aligned} \tag{3.51}$$

Equation (3.51) shows a Markovian behavior, the condition for a stochastic process to be a Markov process. The evolution of $\bar{\xi}_n$ in the NAND multiplexing system, therefore, is a Markov process, or a Markov chain for discrete states and parameters.

In a stochastic Markov chain, the transition probability, which indicates the conditional probability from one specified state to another, is the most significant factor. Since the transition probability matrix $\boldsymbol{\Psi}$ for each $\bar{\xi}_n$ is identical and irrelevant with regard to n, $\bar{\xi}_n$ evolves as a homogeneous Markov chain. Therefore an initial probability distribution and a transition probability matrix as (3.46) are sufficient to get all output distributions. If a NAND multiplexing system has n individual stages in series and its transition probability matrix is given by (3.46), the output distribution of it is then:

$$\mathbf{P}_n = \mathbf{P}_0 \boldsymbol{\Psi}^n. \tag{3.52}$$

The NAND multiplexing system with one executive and two restorative stages can be described as three stochastic variables $\bar{\xi}_1, \bar{\xi}_2$ and $\bar{\xi}_3$. In principle a system with arbitrary number of NAND multiplexing stages, say, $n = 5, 7, 9, ...$, can be built (note that an odd number is necessary to keep the NAND function). When n gets large, $\boldsymbol{\Psi}^n$ approaches a constant matrix $\boldsymbol{\pi}$, i.e.,

$$\lim_{n \to \infty} \boldsymbol{\Psi}^n = \boldsymbol{\pi}. \tag{3.53}$$

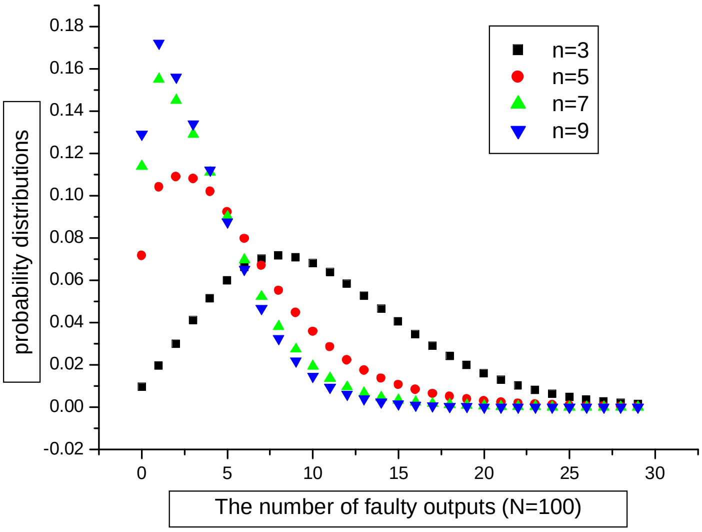


Figure 3.21: The output error distributions in an n-stage multiplexing system ($\varepsilon = 10^{-2}$).

Each row of $\boldsymbol{\pi}$ is identical. This indicates that, as n becomes extremely large, not only the transition probabilities in a NAND multiplexing system will get stable, but also the output distribution will become stable and independent of the number of multiplexing stages.

Let us take an example, a NAND multiplexing system with $N = 100$ and $\varepsilon = 10^{-2}$ (for a von Neumann error). We study the output distribution of this system with different restorative stages, i.e. $n = 3, 5, 7, 9, \ldots$. With the transition probability matrix computed (we do not show the 101×101 matrix) and given that the 90% of the inputs are stimulated, the stimulated output distributions of systems with various stages n, are plotted in Figure 3.21.

As can be seen, the output error distributions move to the lower end as the number of multiplexing stages increases, indicating a reliability improvement resulting from the use of more multiplexing units. Actually, it appears that there is a high probability that the system gives only a few faulty outputs as the number of stages goes up. Further evaluation shows that, when the NAND multiplexing system consists of 3 stages, the probability that less than 10% of the outputs is faulty (stimulated) is approximately 0.49; when the number of multiplexing stages in a system rises to 7, this probability increases to 0.93. A more thorough study of this is given in Section 5.

3.4.2 For large N

If N is rather large (typically $N > 1000$), the output error of each NAND multiplexing stage is approximately normally distributed. If for the lth multiplexing stage there are k_{l-1} stimulated inputs and accordingly k_l stimulated outputs, according to equation (3.32) the probability density of u_l $(u_l = k_l/N)$ is given by:

$$f(u_l|u_{l-1}) = \frac{1}{\sqrt{2\pi}s(u_{l-1})}e^{-\frac{1}{2}(\frac{u_l - z(u_{l-1})}{s(u_{l-1})})^2}, \tag{3.54}$$

where

$$s(u_{l-1}) = \sqrt{z(u_{l-1})(1 - z(u_{l-1}))/N}, \tag{3.55}$$

$$z(u_{l-1}) = (1-\varepsilon) - (1-2\varepsilon){u_{l-1}}^2. \tag{3.56}$$

Then the probability of the multiplexing stage having k_l stimulated outputs under the condition of k_{l-1} inputs is approximately:

$$P(k_l|k_{l-1}) = f(u_l|u_{l-1})\triangle u_l, ...\triangle u_l \sim 1/N. \tag{3.57}$$

The probability of k_l outputs being stimulated in all cases for $0 \leq k_{l-1} \leq N$ is then:

$$P(k_l) = \sum_{k_{l-1}=0}^{N} P(k_l|k_{l-1})P(k_{l-1}). \tag{3.58}$$

Replacing

$$P(k_l) = f(u_l)\triangle u_l, \tag{3.59}$$

and

$$P(k_{l-1}) = f(u_{l-1})\triangle u_{l-1}, \tag{3.60}$$

we have in all cases that the probability density of k_l outputs being stimulated is:

$$f(u_l) = \sum_{u_{l-1}=0}^{1} f(u_l|u_{l-1})f(u_{l-1})\triangle u_{l-1}. \tag{3.61}$$

In the limit we obtain:

$$f(u_l) = \int_0^1 f(u_l|u_{l-1})f(u_{l-1})du_{l-1}. \tag{3.62}$$

Equation (3.62) is an inductive expression, from which conclusions on the outputs of any NAND multiplexing system can be derived from its initial inputs. As the number of NAND multiplexing stages increase, however, it becomes extremely hard to be computed. A practical way is to use the mean of the previous outputs as the fixed inputs of the successive stage. We show this as follows.

Consider the 3-stage multiplexing system in Figure 3.7, for instance. If N is large, the probability density of the outputs u_3 in term of the fixed inputs u_0 is given by:

$$\begin{aligned} f(u_3) &= \int_0^1 f(u_3|u_2)(\int_0^1 f(u_2|u_1)f(u_1)du_1)du_2 \\ &= \int_0^1\int_0^1 f(u_3|u_2)f(u_2|u_1)f(u_1|u_0)du_2du_1 \end{aligned} \tag{3.63}$$

where the initial condition is given by:

$$f(u_1) = \int_0^1 f(u_1|u_0)f(u_0)du_0 = f(u_1|u_0) \tag{3.64}$$

While the equation (3.63) is hard to be computed, it is practical to use the mean of the previous outputs as the fixed inputs of the successive stage. The output distribution of the system in Figure 3.7 is then:

$$f(u_3) = \frac{1}{\sqrt{2\pi}\sqrt{z(u_2)(1-z(u_2))/N}} e^{-\frac{1}{2}\left(\frac{u_3 - z(u_2)}{\sqrt{z(u_2)(1-z(u_2))/N}}\right)^2}, \tag{3.65}$$

with

$$z(u_2) = (1-\varepsilon) - (1-2\varepsilon)z(u_1)^2 \tag{3.66}$$
$$z(u_1) = (1-\varepsilon) - (1-2\varepsilon)z_0^2 \tag{3.67}$$

If, for example, there is a 3-stage NAND multiplexing system with $N = 1000$ and $\varepsilon = 10^{-5}$, given that 90% of the initial inputs are stimulated, i.e., $z_0 = 0.9$, the relative excited level of outputs is approximately normally distributed, with a mean of 0.071 and a standard deviation of 0.008. Using equation (3.65), a probability of less than 10% of the outputs being faulty (stimulated) can be easily evaluated as being approximately 0.9998. If the NAND multiplexing system consists of more stages of restorative units, i.e., $n \geq 5$, this probability approaches 1.

3.5 Discussion

We now study the fault-tolerance of a NAND multiplexing system while we vary the I/O bundle sizes. It might be interesting to evaluate the performance of a NAND multiplexing system with $\varepsilon = 10^{-5}$ and 90% of its inputs stimulated, and the probability that no more than 10% of its outputs is stimulated. Systems with various restorative stages have been investigated. The probability distributions versus the number of multiplexing stages are shown in Figure 3.22 for different bundle sizes: $N = 10$, $N = 100$ and $N = 1000$. Let us take as example $N = 100$. The probability that less than 10% of the outputs is faulty (stimulated) is approximately 0.70 in a 3-stage system while this is 0.99 in a 7-stage system. As the number of multiplexing stages increases, it shows that the reliability of the signals greatly improves, but, on the other hand, the rate of the improvement is getting smaller.

If we pick the number of multiplexing stages to be $n = 7$, then the system has a good performance while the required redundancy ($7N$) is not too high. The fault tolerance of the system for a varying number of error rates ε of the NAND circuits can be studied in this specific case. In Figure 3.23 the probability distribution of errors less than 10% are drawn against the error rate of an individual NAND gate, with $n = 7$. It is obvious that the NAND multiplexing system has a better fault-tolerance when the bundle size N grows. The trade-off, however, has to be made between performance and redundancy. Another conclusion is that the NAND multiplexing technique hardly works when the error rate of basic logic devices approaches 0.1.

3.6 Application

To give an example of how the suggested fault-tolerant architecture is applicable to nano-electronic system, we address the problem of random background charges in SET circuits.

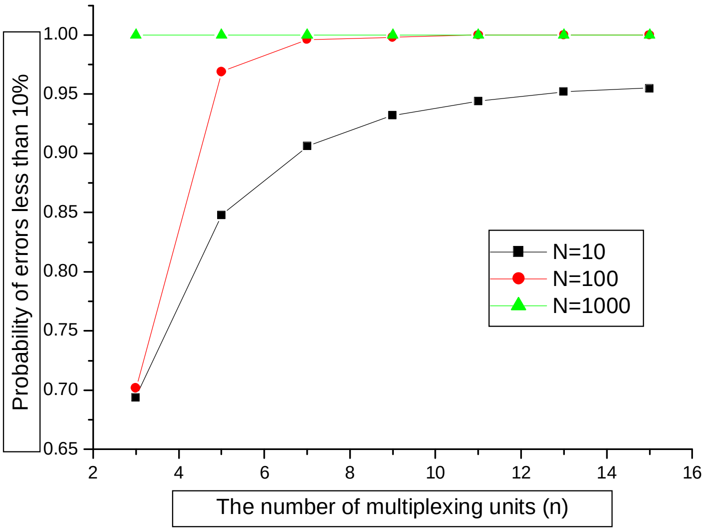


Figure 3.22: Error distribution versus Number of stages.

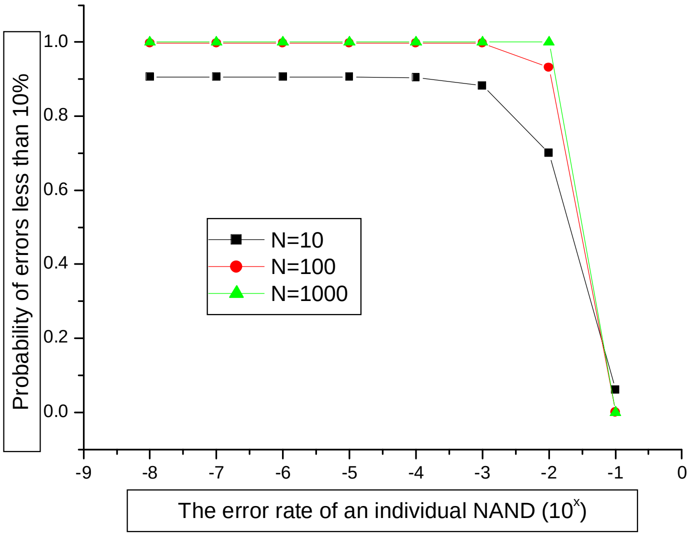


Figure 3.23: Error distribution versus Error rate of a NAND gate.

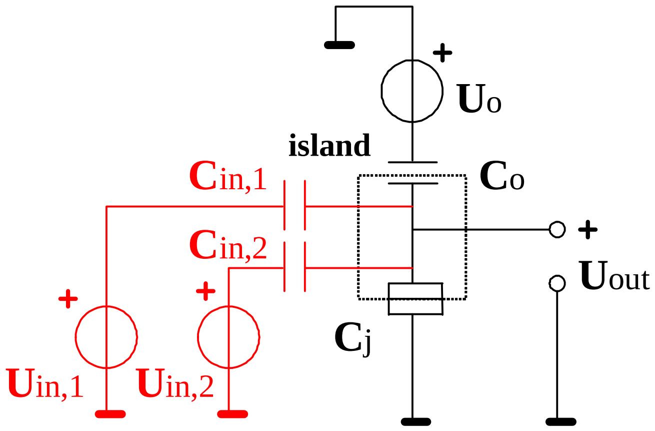


Figure 3.24: An unreliable NAND implemented into SET circuits.

SET devices and circuits have been extensively studied as one of those prospective substitutions to CMOS in digital logic and memory [14]. With appropriate configuration a simple SET circuit can function as NAND logic, as shown in Figure 3.24 [121].

The SET NAND gate consists of a single tunnel junction C_j and one capacitor C_0 as well as two input capacitors. When properly functioning, the output voltage is either low when both the inputs are high, or high in other cases. A so called island is created, so that the single electron can tunnel from and to it through the junction. The island can be made as small as a few nanometers, thus an ultra dense system could be integrated. Unfortunately, the SET circuit suffers from random background charges. Impurities and trapped electrons in the substrate induce image charges Q_0 on the surface of the island. If Q_0 is comparable to e (a single electron charge), the correct device function is destroyed. Optimistically, with a minimum device density of $10^{10}/cm^2$, about one in 1000 devices will have a considerable background charge fluctuation ($|Q_0| > 0.1e$) [14], i.e. a gate error rate of $\varepsilon = 10^{-3}$. This is generally unacceptable for any VLSI system.

However, if in future SET chips with 10^{12} devices are eventually realizable, we could use the NAND multiplexing to achieve fault-tolerance. Although it is difficult to speculate on the architecture of future nanochips, it seems plausible to make it a massively parallel computer consisting of a large number of rather simple processors with associated memories [72], [73]. Within a digital computer, the bulk of the logic gates is spent on memory and caches. The processor itself is made from a number of functional units, each of which can be separated into function blocks. Let us assume that the function block on the most refined level evaluates its inputs and produces a stable output within one clock cycle. Within this function block, many logic circuits may be cascaded, however to avoid timing problems (hazard) usually the number of circuits cascaded and hence the possible paths from inputs to outputs through the various logic circuits is kept within bounds, and hence their path lengths are similar. Such function blocks are found everywhere in the processor and in memory. In this section we make an abstraction of such a function block and assume at first, to be able to make a statistical analysis, that it is made entirely out of n stages of N parallel NAND gates. In a design with unreliable logic, the upper bound is that we must replace each logic gate with $n \cdot N$ unreliable gates. However, we will show in Chapter 4 that, due to the logic design of the function block, we may end up with less redundancy.

To evaluate the reliability, we assume that each processor has a 10-bit output and for each bit 40 logic devices are required. If we implement the multiplexing with $N = 250$ in such processors, then in each processor there are 10^5 devices. We further assume that a processor has a logical depth of 10, which is sufficient for general computation tasks, resulting that the NAND multiplexing will be repeated 10 times. In this implementation, which has 10 stages of multiplexing units, the restorative mechanism is achieved by the successive multiplexing units, therefore the special restorative units would not be necessarily present and, hence, the redundancy level reduces to N from $n \cdot N$ in a n-stage system. For circuits with a few stages of logic, additional restorative stages could be needed to reach the required error bounds.

In such a processor, if no more than 10% of the outputs being faulty is regarded as reliable and perfect inputs are given, then the unreliability of the 1-bit NAND multiplexing output after 10 stages is approximately 10^{-8}. Since each processor only works reliably if none of the output bits fails, the reliability of the processor is then given by:

$$R_p = (1 - U_r)^l, \tag{3.68}$$

where U_r is the unreliability of 1-bit NAND multiplexing output and the processor has an l-bit output. If on the chip there are m processors, the reliability of the whole chip is then given by:

$$R_c = R_p^m. \tag{3.69}$$

We assume that 10% of the total 10^{12} devices are allocated to processors (others for memories, communications, etc.), therefore the number of processors on the nanochip is about 10^6, i.e. $m = 10^6$. Thus the ultimate reliability of the conceived nanochip can be calculated to approximately be 0.9, at the expense of hundreds of redundant components. This indicates that future nanochips with 10^{12} devices, implemented using the NAND multiplexing technique, might be working at an acceptable reliability level, virtually having $10^9 \sim 10^{10}$ effective devices. This could be competitive in future nanoelectronics.

3.7 Summary

A fault-tolerant technique, based on a massive duplication of imperfect devices and randomized imperfect interconnects, had been comprehensively studied. With a given number of identical NAND gates N, input error rate $\bar{x}$, and the error rate of the NAND logic ε, the probability of the number of faulty outputs within a NAND multiplexing unit is proposed to be modeled by a binomial distribution for modest N and by a normal distribution for large N. When N is small, neither of these distributions gives a good approximation due to the dependence of input errors. In this case we propose to study the error behavior through the conducting of simulations.

The error distributions in a multi-stage multiplexing system evolve as a stochastic homogeneous Markov process (chain). The NAND multiplexing system can have more stages to improve the capacity of fault-tolerance. However, the rate of improvement decreases as the number of multiplexing stages increase. When the number of stages becomes large, the output error distribution will become stable and independent of the number of multiplexing stages.

A system architecture based on NAND multiplexing is investigated by studying the problem of random background charges in SET circuits. Although the conceived fault-tolerant architecture requires a rather large amount of redundant components, which makes it inefficient for the protection against permanent faults — normally compensated by reconfiguration techniques, it might be a system solution for ultra large integration of highly unreliable nanometer-scale devices affected by dominant transient errors.

Nevertheless the reliability evaluation of the SET-based architecture is sketchy. This is partially due to the adoption of a highly abstract system model. In the next chapter, we propose a defect- and fault-tolerant architecture based on the multiplexing and reconfiguration technique. A prototype processor structure is presented and used in fault-tolerant implementations. A fault injection simulation is proposed to investigate the error distribution in the structure and the evaluation is carried out through a simulation based reliability model.

Chapter 4

A Defect- and Fault-Tolerant Architecture and Its Implementation for Nanocomputers

4.1 Introduction

[1]Recent progress in molecular electronics has motivated much effort in the research of architectures that are suitable for the implementation of a nanoelectronic computer. One of the factors that make a molecular architecture different from today's CMOS architecture lies on the way how molecular circuits will be fabricated. As opposed to current top-down fabrication approaches, it appears likely that molecular circuits will be assembled through a bottom-up manufacturing process. This bottom-up assembly provides an approach to make large-scale circuits composed of extremely small devices at a low cost, while it imposes many limitations on nanoelectronic architectures. The most prominent of these is the imprecision and randomness in the self-assembly process. This means that it will be difficult to create devices at precise locations and to make precise alignment between wires. The randomness due to the stochastic nature of chemical self-assembly will inevitably raise the densities of defects occurring in molecular devices and interconnects. Defect tolerance is thus a major issue in nanoarchitecture design [69].

There are basically two ways of handling this randomness in molecular assembly. One relies on reconfigurability. The idea is to build spares into a fairly regularly manufactured array architecture and to design the architecture, incorporated with defects, to be reconfigurable. After fabrication, the architecture is tested to locate defective components. The detected defects are then avoided during reconfiguration and good resources are configured to perform computation. Teramac, though built with conventional CMOS technology, is such a successful proof-of-principle model [77]. Array-based reconfigurable architectures have also been proposed for the applications of two-terminal molecular devices [78] and carbon nanotubes (CNTs) and silicon nanowires (SiNWs) FETs [79]. The testability of the structure and fault diagnosis remain a major challenge for such architectures.

---

[1] The content of this chapter has been published in [47], [48], [49] and [50]. To keep its integrity, some results obtained from previous chapters are summarized and rewritten in this chapter.

The other way of handling is to train the stochastically assembled, generally disordered structure to useful logical functions. These architectures place little precision requirements on manufacturing while substantial training and configuration time is required postfabrication. The Nanocell, based on the random assembly of molecular switches, is such an architecture [122]. In a Nanocell once the internal topology is formed by random assembly, it remains static, yet disordered. The Nanocell is then trained by changing the states, on or off, of the molecular switches. Some logical functions such as inverters, NAND gates, half-adders and full adders have been obtained from simulated Nanocells. Another example of this is the two-dimensional crossbar based demultiplexor proposed by HP [123]. The address lines of the demultiplexor are accessed by a set of nanowires by randomly forming contacts between the wires and the address lines. After fabrication the random connections are determined through measurements and, by using more address lines, the circuit can be used as a demultiplexer with a good probability. Due to the general presence of redundant components or connections, these architectures of random assembly are inherently defect-tolerant.

In this chapter, we first present a defect- and fault-tolerant architecture combining the multiplexing and reconfiguration technique, through which we show that the required redundancy could be brought back to a moderate level — no larger than 10^2 — by reconfigurability. We further propose a fault-tolerant technique suitable for implementations by the manufacturing process of stochastically molecular assembly, the triplicated interwoven redundancy (TIR), in which logic gates are triplicated and randomly interconnected. A processor structure is presented for the implementation of the TIR as well as the quadded logic, both with the use of NAND logic. A fault injection simulation is used to investigate the reliability of the TIR as well as that of the equivalent quadded circuit. The TIR is extended to higher orders of redundancy, namely, the N-tuple interwoven redundancy (NIR). The system performance of these architectures are evaluated by studying the reliability, defined as the probability of system survival, through a simulation based reliability model. Our evaluation shows that the suggested architecture is efficiently robust against both permanent and transient faults for the integration of highly unreliable nanometre-scale devices.

This chapter is organized as follows. In section 2 we review the developments of fault-tolerant techniques. In section 3 the NAND multiplexing is briefly presented and developed to account for correlated gate errors. Section 4 gives the reliability analysis of reconfigurable architectures. In section 5 we present the implementation of a defect- and fault- tolerant architecture based on NAND multiplexing and reconfigurable architectures. In section 6 we present the ideas of TMR, quadded and TIR techniques. The experimental studies on the fault-tolerant implementations of a 1-bit processor structure are presented in section 7. In section 8 the TIR is extended to higher orders of NIR. Section 9 presents a discussion. Finally, section 10 summarizes the chapter. This chapter is based on [47], [48], [49] and [50].

4.2 The developments of fault-tolerant techniques

In von Neumann's multiplexing structure, each logic gate was duplicated N times, and each input was replaced by a bundle of N lines, thus producing a bundle of N outputs. A restoring organ was then placed after each logic operation to provide more reliable information on the output bundle by using the redundant information available. In the multiplexing structure a restoring organ was a duplication of the same circuits that performed the logic, whereas it can be any collection of decision elements or switching circuits. It has been shown that the structure can be reliable with a high probability by using a massive duplication of unreliable components, provided that the failure probability of a gate is sufficiently small [70].

As implied in von Neumann's theory, N-tuple modular redundancy (NMR) designs (e.g. triple modular redundancy (TMR)) have been used as benchmarks for evaluating fault-tolerant approaches and were implemented in VLSI for high reliability applications [71]. NMR techniques, generally implemented at modular level instead of gate level, use redundant components to mask the effect of faults. In TMR, the most general form of NMR, three identical modules perform the same operation, and a voter accepts outputs from all three modules, producing a majority vote at its output. This majority voter functions as a restoring organ, bringing the outputs to a more reliable level.

In the 1960s a different redundant technique, quadded logic, was introduced by Tryon for use with AND, OR and NOT logic [124], and later by Jensen for use with NOR logic [125]. Quadded logic requires four times as many circuits and corrects errors without using restoring organs. A quadded logic circuit thus corrects errors and performs the desired logic function at the same time. In quadded logic, a gate is replaced by a stage of four gates and each has twice as many inputs as the nonredundant one. The four outputs of each stage are divided into two sets of two outputs each, connected in a systematic way to the gates in a succeeding stage. These redundancies in logic gates and interconnections give a quadded logic circuit the capability of error correction.

In 1965, these ideas on fault-tolerant logic were generalized by Pierce to a theory termed interwoven redundant logic [126]. Although a procedure for obtaining upper bounds for an interwoven logic network's failure probability was given, the numerical evaluation of a network's reliability seemed to be complicated. The reliability analysis of a quadded network was proposed by using a minimum cut method in [125]. In the 1970s a combinatorial procedure was developed to calculate the reliability of an interwoven logic network [127], and an algorithm was given for the accurate reliability evaluation of a TMR network [128]. Nevertheless, the analytical methods are either extremely complex or present inaccurate predictions for practical designs. In 1989, a systematic approach was proposed to the general design of a fault-tolerant system using redundancy [129], and, in 1994, a fault injection simulation method was proposed to investigate the effect of transient faults in VLSI circuits [130].

More recently, TMR has been examined for potential use in nanoelectronic systems [131], [132]. Von Neumann's NAND multiplexing has also been studied as a possible fault-tolerant technique for the integration of nanoelectronic devices [73], [46], as extensively presented in Chapter 3. The main results are that von Neumann's NAND multiplexing technique is extended to a fairly low degree of redundancy; that the stochastic Markov characteristics of a multi-stage multiplexing system is studied; and that in a large multiplexing network the restoring organs might not be needed. An architecture based on NAND multiplexing is briefly discussed for the use of SET circuits, showing that multiplexing might be an effective fault-tolerant technique for protection against the increasing transient faults in nanoelectronic architectures, while it appears to be less efficient for protection against manufacturing defects or permanent faults, which are normally compensated for by reconfiguration techniques.

A reconfigurable architecture is a computer architecture which can be configured or programmed after fabrication to implement a desired computation. Faulty components are detected during testing and excluded during reconfiguration. Reconfigurable architectures have been investigated as well for the solution of integration of highly unreliable nanometer-scale devices, in particular as defect-tolerant architectures against manufacturing errors.

The Teramac computer [77], built at HP laboratories, is such a defect-tolerant reconfigurable machine. The basic components in Teramac are programmable switches (memory) and redundant interconnections. High communication bandwidth is critical for both parallel computation and defect tolerance. With about 10% of logic cells and 3% of total resources defective, Teramac could still operate 100 times faster than a high-end single-processor workstation for some of its configurations.

The Embryonics architecture [133] is inspired by the biological growth and operation of all living beings. It is based on four hierarchical levels: a molecule (a multiplexer-based element of a programmable circuit), a cell (a small processor with an associated memory), an organism (an application-specific multiprocessor system), and the population of identical organisms. Each cell contains complete sets of instructions, the genomes, which make each cell universal and potentially apt for self-repair and self-replication. The objective of developing highly robust integrated circuits capable of self -repair and self-replication makes the Embryonics architecture a potential paradigm for future nanometer-scale computation systems.

4.3 The NAND multiplexing technique for correlated errors

4.3.1 Error distributions in a multiplexing unit

In von Neumann's construction of NAND multiplexing, a NAND gate, as well as its inputs and output, is duplicated by N times, as depicted in Figure 4.1. A "random permutation" of the input signals is performed in the sense that each signal from the first input bundle is randomly paired with a signal from the second input bundle to form the input pair of one of the duplicated NANDs.

If X, Y and Z, with $(\bar{x} \cdot N,\ \bar{y} \cdot N,\ \bar{z} \cdot N)$ elements, are the sets of lines being stimulated in the input and output bundles, then $(\bar{x},\ \bar{y},\ \bar{z})$ are relative levels of excitation of the two input bundles and the output bundle. Let $\bar{r} = 1 - \bar{z}$. Clearly $\bar{r}$ is the relative level of the non-stimulated outputs. Assuming that the NAND gate is fault-free, the probability of the output of a NAND gate that is found non-stimulated (by both stimulated inputs) is approximately $\bar{r}' = \bar{x}\bar{y}$. If each NAND gate has a probability ε of making a von Neumann error, the probability of its output being non-stimulated is:

$$\bar{r}_v = \bar{x}\bar{y} + \varepsilon(1 - 2\bar{x}\bar{y}); \tag{4.1}$$

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

$$\bar{r}_0 = \varepsilon + (1 - \varepsilon)\bar{x}\bar{y} \tag{4.2}$$

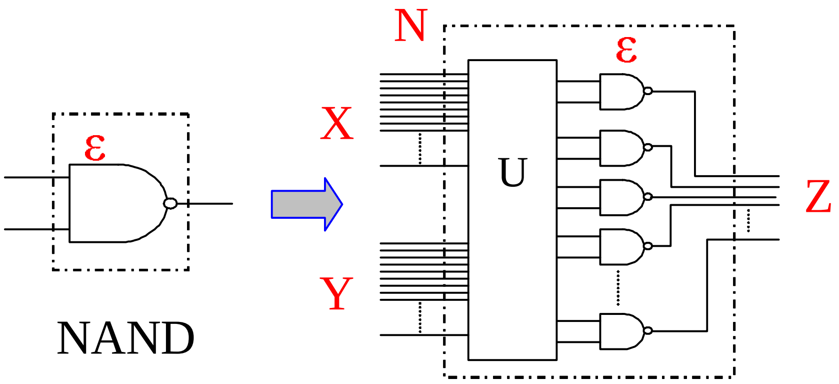


Figure 4.1: A NAND multiplexing unit.

and

$$\bar{r}_1 = (1 - \varepsilon)\bar{x}\bar{y}. \tag{4.3}$$

For modest N, as has been seen in Chapter 3, where similar results were obtained for the relative stimulated level $\bar{z}$, the probability distribution of the outputs being non-stimulated can approximately be given by the binomial distribution, if the N NAND gates function independently. Thus the probability of k out of N outputs being non-stimulated is:

$$R(k) = \binom{N}{k} \bar{r}^k (1 - \bar{r})^{N-k}. \tag{4.4}$$

with $\bar{r} \in [\bar{r}_v, \bar{r}_0, \bar{r}_1]$.

If both inputs of the NAND gates are expected to be in stimulated states, the non-stimulated outputs are then considered as reliable ones. If the faulty devices in the multiplexing circuits are independent and uniformly distributed, formula (4.4) could be used to evaluate the output reliability. This may be reasonable when the dominant faults are transient ones. For manufacturing defects or permanent faults, however, the binomial distribution model is not sufficient to describe the actual manufacturing imperfections. The device components are then not statistically independent but rather correlated since defects tend to cluster on a chip [134]. Formula (4.4) is therefore not appropriate for reliability calculations. (Although it is not yet clear what the future nanocomputers will be based on and how they will be built, it might be helpful to learn from present manufacturing processes.)

Variability of the manufacturing defects can be modeled with a continuous probability distribution function $f(r)$ of an estimated component reliability r. Compounding the formula (4.4) with respect to this distribution function results in

$$R(k) = \int_0^1 \binom{N}{k} \bar{r}^k (1 - \bar{r})^{N-k} f(r) dr. \tag{4.5}$$

The success of the approach depends on finding appropriate parameters for the formula. Here we follow Stapper's beta distribution model [135], which gives:

$$R(k) = \binom{N}{k} \bar{r}^k \left(\prod_{i=0}^{k-1} \frac{\mu + i}{\mu + i\bar{r}} \right) \times (1 - \bar{r})^{N-k} \left(\prod_{j=0}^{N-k-1} \frac{\mu + j\bar{r}/(1 - \bar{r})}{\mu + k\bar{r} + j\bar{r}} \right), \tag{4.6}$$

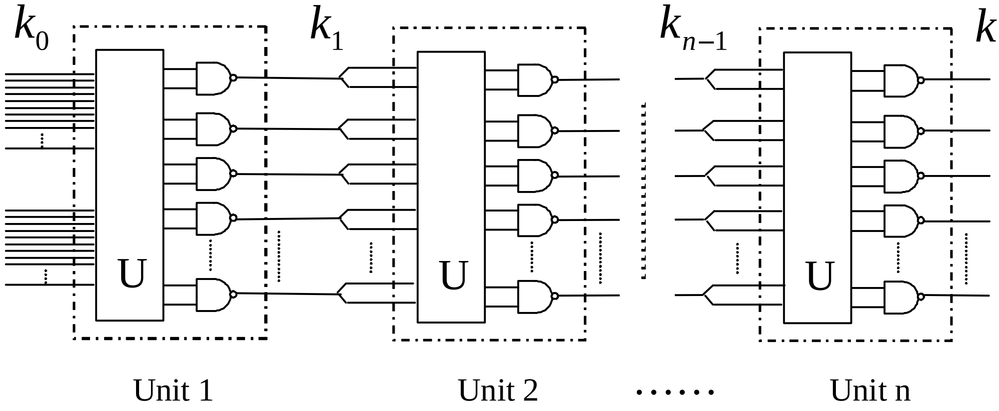


Figure 4.2: A multi-stage NAND multiplexing system.

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

4.3.2 Error distributions in a multi-stage system

If the outputs of a NAND multiplexing unit are duplicated as the inputs of the succeeding one, a multi-stage system can be built as depicted in Figure 4.2. In such a system the number of stimulated (or non-stimulated) outputs of each NAND multiplexing stage is actually a stochastic variable; it evolves as a Markov chain since the outputs of one stage are totally determined by the inputs and device error distribution of the same stage.

If there are k_{l-1} of the N incoming lines stimulated for both inputs of the lth unit and each NAND gate has a fixed probability ε of making an error, according to the formula (4.6), the probability of having k_l' non-stimulated outputs in case of the corresponding k_{l-1} stimulated inputs is given by:

$$\begin{aligned} R(k_l'|k_{l-1}) &= \binom{N}{k_l'} \bar{r}^{k_l'}(k_{l-1}) \left(\prod_{i=0}^{k_l'-1} \frac{\mu+i}{\mu+i\bar{r}(k_{l-1})} \right) \times \\ & (1-\bar{r}(k_{l-1}))^{N-k_l'} \left(\prod_{j=0}^{N-k_l'-1} \frac{\mu+j\bar{r}(k_{l-1})/(1-\bar{r}(k_{l-1}))}{\mu+k_l'\bar{r}(k_{l-1})+j\bar{r}(k_{l-1})} \right) \end{aligned} \tag{4.7}$$

where $\bar{r}(k_{l-1})$ is a variation of the equation (4.1), (4.2) or (4.3) with $\bar{x} = \bar{y} = \frac{k_{l-1}}{N}$.

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

$$P(k_l|k_{l-1}) = R((N-k_l)|k_{l-1}) \tag{4.8}$$

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

$$P(k_l) = \sum_{k_{l-1}=0}^{N} P(k_l|k_{l-1})P(k_{l-1}) \tag{4.9}$$

The formula (4.9) is inductive in the sense that, given an initial probability distribution and conditional probabilities, the output probability at any stage can be obtained. In this Markov chain an $(N+1) \times (N+1)$ transition probability matrix $\boldsymbol{\Psi}$, whose elements are $P(k_l|k_{l-1})$, $k_l, k_{l-1} \in [0, 1, 2, ...N]$, can be obtained as (4.10), so that all conditional probabilities for any set of (k_l, k_{l-1}) are included.

$$\boldsymbol{\Psi} = \begin{bmatrix} P(0|0) & P(1|0) & P(2|0) & .... & P(N|0) \\ P(0|1) & P(1|1) & P(2|1) & .... & P(N|1) \\ P(0|2) & P(1|2) & P(2|2) & .... & P(N|2) \\ .... & .... & .... & .... & .... \\ P(0|N) & P(1|N) & P(2|N) & .... & P(N|N) \end{bmatrix} \tag{4.10}$$

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

$$\mathbf{P}_0 = [p_0, p_1, p_2 ... p_N] \tag{4.11}$$

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

$$\mathbf{P}_n = \mathbf{P}_0 \boldsymbol{\Psi}^n \tag{4.12}$$

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

$$\lim_{n \to \infty} \boldsymbol{\Psi}^n = \boldsymbol{\pi}. \tag{4.13}$$

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

4.4 Reliability analysis of reconfigurable architectures

The idea behind reconfigurable architectures is that the defects due to manufacture can be detected, located and then avoided. The reconfigurable computer concept is greatly assisted by the use of field programmable gate arrays (FPGAs) [136]. Fundamentally a FPGA contains a regular array of logic units, which are called configurable logic blocks (CLBs). Each CLB can communicate with its neighbors, and the CLBs are further grouped in blocks, then clusters of blocks. The CLBs can be individually reprogrammed so that a wide variety of logic or memory structures can be mapped onto the array of CLBs. When a part (or all) of a CLB is not working, the defective components are easy to locate and exclude from the working components. The Teramac machine [77], as a successful example of the reconfiguration concept, uses 864 identical FPGA chips, among which 75% (647) are partially defective. The first task of Teramac after it was built was to run self-diagnostic software, by which the defects were detected and located, and a defect database was generated. By reading the database, user applications are mapped onto good resources. Teramac "has been successfully configured into a number of parallel architectures and used for extremely demanding computations".

In processor arrays, the basic logic circuit blocks are referred to as processing elements (PEs), which are usually associated with local memories. In very large chips, the reliability can be improved by adding spare PEs to the design. Clearly, the more spares added, the higher the resulting reliability will be. Instead of trying to achieve complete fault tolerance, defined as survival to a number of faults equal to the number of spares, most research aims at optimizing probability of survival, defined as the percentage of fault configurations that can be successfully overcome by the reconfiguration approach [137]. Reconfiguration approaches may be local or global. In local approaches, arrays are divided into subarrays. Spare elements are added to each individual subarray and reconfiguration is performed internally to each subarray. In global approaches, a set of spare elements is added to the whole array (usually as spare rows and columns along the edges of the array). Global approaches usually involve far more complex reconfiguration algorithms than local solutions [137].

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

$$R_n = \sum_{m=n-r}^{n} R_{mn} \ . \tag{4.14}$$

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

$$R_{mn} = \binom{n}{m} R_0^m (1 - R_0)^{n-m} \ . \tag{4.15}$$

Once again the defective modules in an array are not uniformly distributed but rather correlated, therefore the binomial distribution formula (4.15) is not sufficient for reliability evaluation. Stapper's model can be used to improve the reliability calculation of correlated modules [135]:

$$\bar{R}_{mn} = \binom{n}{m} \bar{R}_0^m \left(\prod_{i=0}^{m-1} \frac{\mu + i}{\mu + i\bar{R}_0} \right) \times \left(1 - \bar{R}_0\right)^{n-m} \left(\prod_{j=0}^{n-m-1} \frac{\mu + j\bar{R}_0/(1-\bar{R}_0)}{\mu + m\bar{R}_0 + j\bar{R}_0} \right) , \tag{4.16}$$

where μ is a variable parameter indicating the amounts of fault clustering and $\bar{R}_0$ is the average or expected single module reliability. Typical values of μ are dependent upon the other parameters in formula (4.16).

Formula (4.16) calculates the probability that exactly m out of n identical modules operate correctly. It can be applied to the reliability analysis of parallel processors with redundancy and fault-tolerant VLSI systems.

4.5 A hypothetical architecture for defect- and fault-tolerance

4.5.1 Basic circuits implemented with NAND multiplexing

The fact that a NAND gate is a universal logic device makes it possible to use the NAND multiplexing technique on any logic operation, even though the multiplexing theory can be easily extended to application of other specific logic. In processors, function blocks or modules, many logic circuits may be cascaded. The function blocks or processors can be composed of Arithmetic and Logic Units (ALUs), Look-Up Tables (memories), or simply multiplexers. In this section we make an abstraction of such a function block and assume, as we did in Chapter 3, that it is made entirely out of stages of parallel NAND gates.

If a processor is implemented using NAND multiplexing, then the obtained structure will be a NAND multiplexing system with a lower bound of redundancy of N, as all the components are duplicated N times. The performance of a multiplexing system can be evaluated by investigating the probability that the number of faulty outputs is or is not within a critical threshold level Δ. In other words, those outputs with errors less than this threshold will be regarded as reliable and their complementaries will be unreliable. The threshold level, together with N, may have impact on the maximum tolerable value of the device failure rate ε. To make a reasonable analysis, we will take a bundle size of 50 and a threshold of 10%, i.e. $N = 50$ and $\Delta = 10\%$.

If each processor has a logical depth of 11, which is sufficient for general computation tasks, then the reliability of the 1-bit NAND multiplexing output after 11 stages can be studied with various device error rates ε, using the NAND multiplexing theory. With perfectly fault-free inputs, the probability distributions of output errors (unreliability distribution) are evaluated against the error rate of an individual NAND with $\mu = 5, 10, 15, 25, 50, 100$ and $infinity$. The results are shown in Tables 4.1, 4.2 and 4.3 for the fault models of von Neumann, Stuck-at-0 and Stuck-at-1.

As revealed in the tables, the output reliability varies due to μ, i.e. the amount of fault clustering, indicating the influence of fault distributions on system reliability. It can also be seen that von Neumann fault model brings the largest system performance degradation (system unreliability). Since we are interested in the maximum device error rate that can be tolerated in general, we will take $\mu = 50$ and $\varepsilon = 10^{-2}$ for the von Neumann fault; then the reliability of the 1-bit output can be obtained from Table 4.1 as $R_0 = 0.868$. In the following section we will take $R_0 = 0.868$ as the average reliability of a 1-bit multiplexing circuit.

	$\mu = 10$	$\mu = 25$	$\mu = 50$	$\mu = 100$	$\mu = \infty$
$\varepsilon = 10^{-6}$	$1.516 \cdot 10^{-5}$	$1.042 \cdot 10^{-5}$	$7.264 \cdot 10^{-6}$	$5.271 \cdot 10^{-6}$	$2.984 \cdot 10^{-6}$
$\varepsilon = 10^{-5}$	$1.518 \cdot 10^{-4}$	$1.043 \cdot 10^{-4}$	$7.293 \cdot 10^{-5}$	$5.302 \cdot 10^{-5}$	$2.991 \cdot 10^{-5}$
$\varepsilon = 10^{-4}$	$1.519 \cdot 10^{-3}$	$1.047 \cdot 10^{-3}$	$7.349 \cdot 10^{-4}$	$5.360 \cdot 10^{-4}$	$3.055 \cdot 10^{-4}$
$\varepsilon = 10^{-3}$	$1.539 \cdot 10^{-2}$	$1.093 \cdot 10^{-2}$	$7.929 \cdot 10^{-3}$	$5.987 \cdot 10^{-3}$	$3.703 \cdot 10^{-3}$
$\varepsilon = 10^{-2}$	0.1686	0.1479	0.1315	0.1194	0.1036

Table 4.1: Output unreliabilities of a 1-bit NAND multiplexing circuit (von Neumann fault).

	$\mu = 10$	$\mu = 25$	$\mu = 50$	$\mu = 100$	$\mu = \infty$
$\varepsilon = 10^{-6}$	$1.081 \cdot 10^{-5}$	$8.382 \cdot 10^{-6}$	$6.497 \cdot 10^{-6}$	$5.026 \cdot 10^{-6}$	$2.923 \cdot 10^{-6}$
$\varepsilon = 10^{-5}$	$1.082 \cdot 10^{-4}$	$8.386 \cdot 10^{-5}$	$6.501 \cdot 10^{-5}$	$5.029 \cdot 10^{-5}$	$2.927 \cdot 10^{-5}$
$\varepsilon = 10^{-4}$	$1.082 \cdot 10^{-3}$	$8.401 \cdot 10^{-4}$	$6.525 \cdot 10^{-4}$	$5.059 \cdot 10^{-4}$	$2.965 \cdot 10^{-4}$
$\varepsilon = 10^{-3}$	$1.089 \cdot 10^{-2}$	$8.580 \cdot 10^{-3}$	$6.777 \cdot 10^{-3}$	$5.367 \cdot 10^{-3}$	$3.347 \cdot 10^{-3}$
$\varepsilon = 10^{-2}$	0.1142	0.1009	$8.964 \cdot 10^{-2}$	$8.062 \cdot 10^{-2}$	$6.744 \cdot 10^{-2}$

Table 4.2: Output unreliabilities of a 1-bit NAND multiplexing circuit (Stuck-at-0 fault).

	$\mu = 10$	$\mu = 25$	$\mu = 50$	$\mu = 100$	$\mu = \infty$
$\varepsilon = 10^{-6}$	$4.346 \cdot 10^{-6}$	$2.041 \cdot 10^{-6}$	$7.668 \cdot 10^{-7}$	$2.448 \cdot 10^{-7}$	$6.140 \cdot 10^{-8}$
$\varepsilon = 10^{-5}$	$4.360 \cdot 10^{-5}$	$2.038 \cdot 10^{-5}$	$7.892 \cdot 10^{-6}$	$2.694 \cdot 10^{-6}$	$6.135 \cdot 10^{-7}$
$\varepsilon = 10^{-4}$	$4.361 \cdot 10^{-4}$	$2.044 \cdot 10^{-4}$	$7.931 \cdot 10^{-5}$	$2.693 \cdot 10^{-5}$	$6.217 \cdot 10^{-6}$
$\varepsilon = 10^{-3}$	$4.395 \cdot 10^{-3}$	$2.104 \cdot 10^{-3}$	$8.450 \cdot 10^{-4}$	$3.000 \cdot 10^{-4}$	$7.065 \cdot 10^{-5}$
$\varepsilon = 10^{-2}$	$4.726 \cdot 10^{-2}$	$2.720 \cdot 10^{-2}$	$1.441 \cdot 10^{-2}$	$7.239 \cdot 10^{-3}$	$2.142 \cdot 10^{-3}$

Table 4.3: Output unreliabilities of a 1-bit NAND multiplexing circuit (Stuck-at-1 fault).

4.5.2 Hierarchical reconfigurability at processor, cluster and chip levels

We further assume that each processor has a 32-bit processing capacity. For a 32-bit processor, if no redundant circuits are applied, it is only reliable as all of the output bits are reliable. Instead of exactly making a 32-bit circuit we build in the processor some redundant processing circuits, so that the spares can be configured to replace defective ones (Figure 4.3 (c)).

If each 1-bit circuit has a similar structure, the reliability of the 32-bit processor with redundant circuits can be evaluated against the number of spare bit circuits, using the formulae (4.14) and (4.16), as plotted in Figure 4.4. The left-most data in the figure indicate the reliability of a processor with no redundancy. The effect of the variability parameter μ here is rather significant. The improvement of reliability by using redundancy is explicit; in particular, when μ is large, i.e. when faults are barely correlated. Assuming that errors are not strongly correlated in processor and upper levels (due to the relatively large circuit area), we take $\mu = 20$ for further evaluations. Thus a 32-bit processor with 16 redundant bit circuits will have a reliability of approximately 0.981.

The development of nanotechnology makes it eventually possible to realize extremely large-scale integration, of the order of 10^{12} devices per chip. If on such a chip each processor has about 10^6 devices (logic, memory, communications, etc.), the number of processors on the chip will be about 10^6 ($2^{10} \times 2^{10}$). Instead of being connected globally, the processors can be assembled into 1024 (2^{10}) processing clusters, each containing 1024 (2^{10}) processors and executing tasks independently. The clusters further compose the chip. Both clusters and processors can be connected in a 2-dimensional (32×32) array, in which some columns are redundant, as depicted in Figure 4.3 (a and b). The reconfigurable strategy therefore can be implemented on both chip and cluster level.

Similarly, the performance of a cluster (chip) with redundant processors (clusters) can be evaluated using the formulae (4.14) and (4.16). The reliability of a cluster against the number of spare columns is plotted as Figure 4.5, with $\mu = 20$. The left-most points indicate the reliability of a cluster with no spares. By using 4 columns of processors as redundant, the reliability of the cluster is elevated from 0.215 to 0.985, i.e., a cluster having 128 (4×32) redundant processors has a reliability of approximately 0.985.

The reliability of a chip with 1024 clusters is plotted against the number of spare columns of clusters with $\mu = 20$ as shown in Figure 4.6. It can be seen that the reliability of a chip will be greatly improved by using redundant components. If 128 of the 1024 total clusters (4 out of 32 columns) are used as spare ones, then the reliability of the chip will reach a level of approximately 99%, provided that faulty components can be effectively substituted by spare ones.

4.5.3 Summary and issues

In summary, we have discussed the setup of a massively parallel fault-tolerant architecture. The NAND multiplexing technique is implemented in the fundamental circuits and reconfigurable structures are mapped to the processor, cluster and chip level. Containing up to

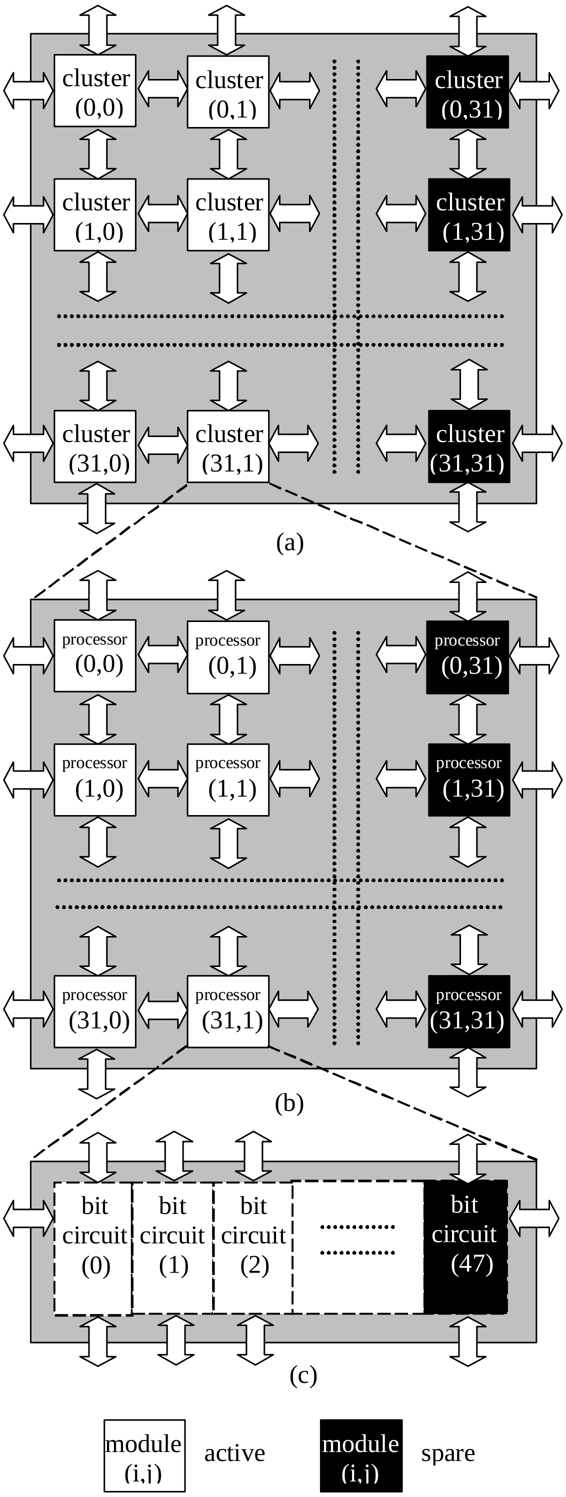


Figure 4.3: The architecture: (a) chip (b) cluster (c) processor; all with columns of spares.

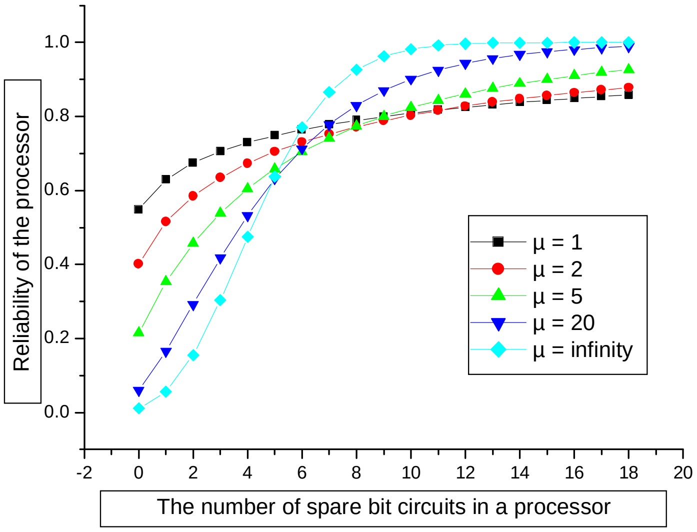


Figure 4.4: The reliability of a processor with spare bits.

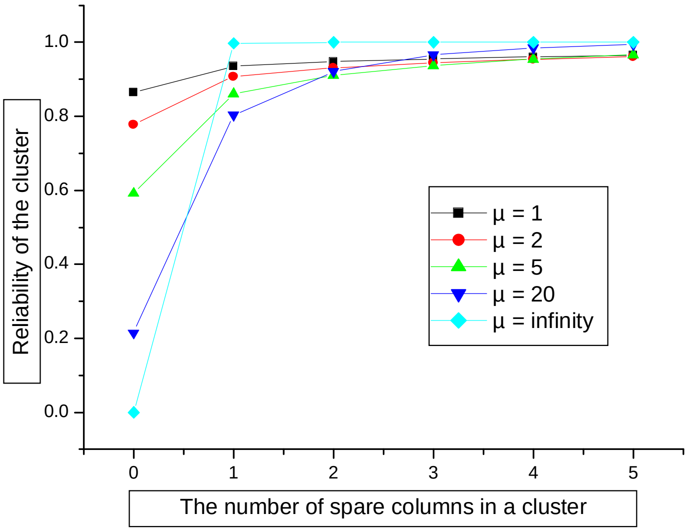


Figure 4.5: The reliability of a cluster with spare processors.

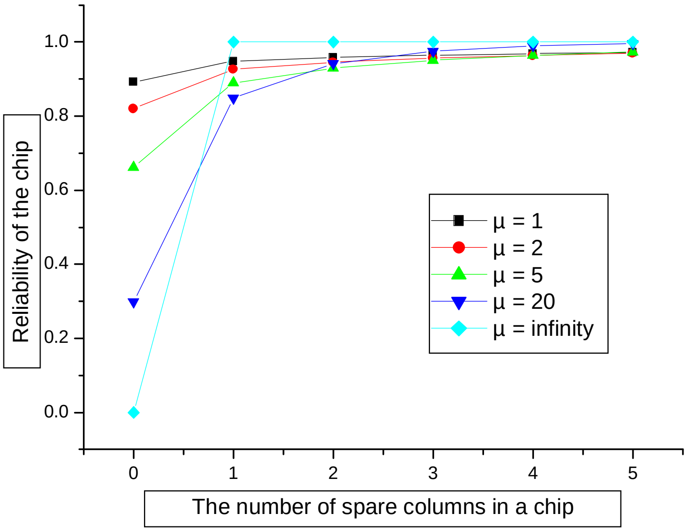


Figure 4.6: The reliability of a chip with spare clusters.

10^{12} devices the conceived chip can have about 10^6 medium-sized processors and tolerate a device error rate of up to 10^{-2}, which is generally unacceptable for any current VLSI system. Redundant components are used at various levels and, as our evaluation shows, they are critical for the survival of the architecture.

In contrast with the architecture presented in Chapter 3 where plain NAND multiplexing was used to recover from transient errors, resulting in a massive redundancy, we now accept a higher error rate on the lowest level with considerably less redundancy, but compensate for this using a hierarchical reconfigurability. This leads to an acceptable failure rate of permanent defects for the entire system, and simultaneously forms a protection against transient errors (online error detection might be needed). The error detection problem remains open for further research.

The system is expected to have a total redundancy factor of $(50 \times \frac{3}{2} \times \frac{8}{7} \times \frac{8}{7} \times (\textit{the fraction of other necessary spare components})) \approx 100$. This indicates that the future nanochips with 10^{12} devices might be working at an acceptable reliability level, virtually having about 10^{10} effective devices.

In this architecture, however, there were two issues. First, the numerical analysis employing the multiplexing theory gives an approximate evaluation of the output reliability of the circuit that consists of chains of multiplexing stages. A larger bundle size N presents a better approximation. When N becomes small, however, the deviation resulting from this analysis could be large. This limitation means that the analytical approach using the multiplexing theory cannot be applied to the study of circuits in which fewer redundancies are involved (e.g. $N < 10$). To solve this problem, a CAD method based on probabilistic model checking has been proposed to evaluate the reliability of fault-tolerant architectures and, in particular, the multiplexing systems [74]; Monte Carlo simulations have also been performed to study the error behaviors in a multiplexing nanosystem [75]. Second, in order to make a statistical analysis, a highly abstract circuit model consisting of chains of identical

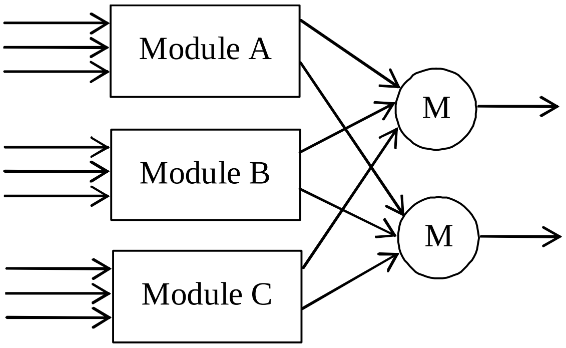


Figure 4.7: A typical configuration of TMR.

multiplexing stages has been adopted. This is obviously not true in a practical implementation. To improve these, we propose a new form of interwoven redundant logic, the N-tuple interwoven redundancy (NIR), and present an experimental study through the simulation of a realistic processor model. Before turning to the study of simulations, we present in the next section the ideas of triple modular redundancy (TMR), the interwoven redundant logic and the triplicated interwoven redundancy (TIR).

4.6 Triple modular redundancy (TMR), quadded logic and triplicated interwoven redundancy (TIR)

4.6.1 Triple modular redundancy (TMR)

In a TMR system, a nonredundant circuit is divided into smaller modules, each module is "triplicated" and majority gates are used to collect outputs from all of the three modules, producing majority votes at their outputs. A typical TMR structure with a simple configuration is shown in Figure 4.7. In a serial circuit the majority gate for each output is also triplicated and serves as an intermediate restoring device. The TMR circuits of triplicated modules followed by triplicated voters has been widely studied as an important redundancy technique (see e.g. [138], [128] and [71]). It has also been used as a benchmark against which other redundancy schemes are evaluated.

For TMR systems, various reliability evaluation algorithms have been proposed, but most involve expensive computation. For a simple network, as the one in Figure 4.7, a classical model has been widely used for reliability evaluation. This reliability modeling assumes that a TMR network continues to operate as long as at least two of the circuit modules are fault-free. If all three of the modules work independently, the reliability is then given by

$$R_{TMR} = \binom{3}{3} R_0^3 + \binom{3}{2} R_0^2 (1 - R_0) = 3R_0^2 - 2R_0^3, \tag{4.17}$$

where R_0 is the reliability of a circuit module. If a module has N components, each of which has a failure rate of p_f, and if it works as long as all of its components work correctly, the reliability of the module is then given by

$$R_0 = (1 - p_f)^N. \tag{4.18}$$

When the reliability of a module falls in a certain range (with a limit of lower bounds strictly larger than 1/2), as implied in equation (4.17), a TMR system behaves as one of its constituent modules, but with an improved reliability, but also at the expense of three times the number of components plus the majority gates. As implied in equation (4.18), however, the reliability of a module imposes a demanding requirement on a module's size - the modules involved in TMR should be modest in size in relation to the error rate of an individual component in the circuits, or, in other words, a module with a large number of components will present a serious limit on the upper bound of the device error rate that could be tolerated by the use of TMR [132]. In principle, using the modules with the lowest complexity gives the best system reliability of TMR, provided with perfect majority gates [138].

A TMR circuit can be further triplicated. The obtained circuit thus has nine copies of the original module and it requires two layers of majority gates to collect information at outputs. This process can be repeated if necessary, resulting in a technique called cascaded triple modular redundancy (CTMR). A study on the use of CTMR in nanoelectronic circuits was presented in [131]. It was shown that the use of CTMR in a nanochip with a large number (e.g. 10^{11} or 10^{12}) of nanoscale devices would require an extremely small error rate of nanoelectronic devices. However, the method may be effective for use in modest or small circuit modules. Another disadvantage of the CTMR scheme is that it introduces an exponential growth in redundancy as the cascaded layers increase.

In CMOS circuitry the assumption that the majority gates are ideally fault-free can be realized by using larger and more reliable components. However, it will be highly unlikely or inefficient to implement the majority gates with other technologies in future circuits made up of nanoelectronic or molecular devices. Instead it is straightforward to envision that all the circuit elements are made from similar technologies. Thus the majority gates are also prone to errors, with a device error rate as the same as that of the working modules. Assuming a majority gate is composed of M components and it works only when it is fault-free, its reliability is given by

$$R_M = (1 - p_f)^M. \tag{4.19}$$

If a TMR only gives reliable outputs when both of the majority gates and modules are fault-free, the reliability of a TMR circuit becomes

$$R_{TMR} = R_M^{N_{op}}(3R_0^2 - 2R_0^3), \tag{4.20}$$

where N_{op} is the number of a module's outputs, i.e. the number of majority voter circuits.

If the majority gates are not perfectly reliable, then the reliability of a majority gate should be larger than that of a module in a TMR circuit. Otherwise, if R_M is comparable with or less than R_0, we have $R_{TMR} \leq R_0^{N_{op}}(3R_0^2 - 2R_0^3)$. Since $R_0 < 1$ and $(3R_0^2 - 2R_0^3) < 1$, R_{TMR}, the reliability of the TMR, can never be better than R_0, the reliability of the original circuit. This indicates that, when circuit modules and majority gates are made of similar devices, a module must be larger in scale than a majority circuit in a TMR to get a reliability improvement.

4.6.2 The interwoven redundant logic and quadded logic

The concept of the interwoven redundant logic derived from the reliability theories as developed by von Neumann and his contemporaries [126]. The faults considered in the interwoven redundant logic are interpreted as the $0->1$ and $1->0$ faults, without a distinction between the von Neumann and stuck-at types. The error correction mechanism in the interwoven redundant logic depends upon asymmetries in the effects of these two different types of binary errors. The effect of a fault is determined by the value of the fault and the type of the gate. Consider a NAND gate, for instance. If one of its inputs is in the binary 0 state while it should nominally be in the binary 1 state, possibly caused by a faulty gate or faulty interconnection, then the NAND's output will be static at the 1 state regardless of the values of other inputs. If an input is in the 1 state while it should be in the 0 state, the output will not be stuck but dependent on other inputs. Thus, there are two types of faults for a logic gate. One is critical in the sense that its occurrence on one of the inputs leads to a stuck output; the other is subcritical in the sense that its occurrence alone does not cause an output error. Hence, a $1->0$ fault is critical for a NAND gate, while it is subcritical for a NOR gate. A $0->1$ fault is critical for an OR logic, while it is subcritical for an AND gate.

If it is not possible to ensure that all errors are subcritical errors, one way to improve the reliability of an interwoven logic circuit is to arrange the logic so that a critical error occurring in one layer of logic becomes a subcritical error in the second layer. Then, at the output of the third layer, the effect of the subcritical error from the second layer may be reduced; it is therefore likely that there are no or fewer errors in the end of the circuit. For instance, the output error caused by a critical fault of an AND gate is the subcritical input error for an OR gate, and the output error caused by a critical fault of an OR gate is the subcritical input error for an AND gate. Alternating layers of AND and OR gates therefore corrects errors by switching them from critical faults to subcritical ones. Similarly, the output error caused by a critical fault of a NAND (or NOR) gate becomes the subcritical error in the next layer of NAND (or NOR) gates.

Quadded logic is an *ad hoc* configuration of the interwoven redundant logic. It requires four times as many circuits, interconnected in a systematic way. It corrects errors and performs the desired computation at the same time. Quadded logic has been studied respectively for use with AND, OR and NOT logic [124], and for use with NOR logic [125]. To illustrate it, we show the schematics of a complementary half adder (computing the complements of carry and sum, denoted as cc and cs) in Figure 4.8 and its quadded form in Figure 4.9, both implemented with NAND gates (including inverters, which can be seen as a special form of NANDs).

In the quadded implementation in Figure 4.9, each NAND gate in Figure 4.8 is replaced by a group of four NAND gates, each of which has twice as many inputs as the one replaced. The four outputs of each group are divided into two sets of two outputs, each providing inputs to two gates in a succeeding stage. The interconnections in a quadded circuit are hence eight times as many as those used in the nonredundant form.

In this pattern of interconnection, any single error introduced in the network can be corrected by the network itself, provided that the network is large enough. To show this in Figure 4.9, assume that $B1$ in stage B is wrongly in the 0 state when it should be in the

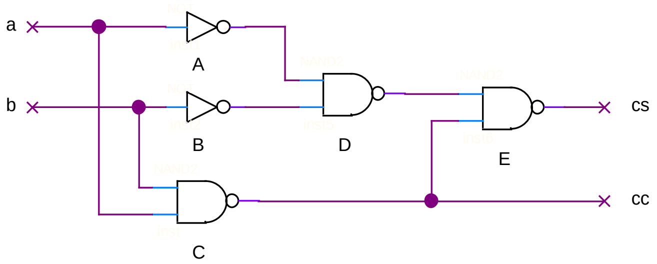


Figure 4.8: A nonredundant complementary half adder implemented with NAND logic.

1 state (a critical $1 - > 0$ error for NAND). Due to this error, the outputs of $D1$ and $D3$ of stage D must be 1; this can be erroneous, but it would be a subcritical $0 - > 1$ error. Since the outputs $D2$ and $D4$ of stage D are not in error (thus in the correct 0 state), the subcritical errors at outputs $D1$ and $D3$ are masked at stage E, producing the expected (correct) 1 state at all the outputs of stage E. As has been seen, a subcritical $0 - > 1$ error is even more promptly corrected in the NAND network. In general, a single critical error in a quadded circuit will be eliminated after passing through two stages and a single subcritical error will be corrected in the next stage after its occurrence.

The interconnection patterns in a quadded network are important to the network's capability of error correction, yet the rules are simple. The outputs of four gates, numbered from 1 to 4 as in Figure 4.9, are divided into two sets. Each set forms a pair of inputs and each pair feeds the two gates with the same numbers as the set in succeeding stages. If the four outputs are divided into two sets of $(1, 3)$ and $(2, 4)$, for instance, the set $(1, 3)$ will provides inputs to gates 1 and 3 in the next stage and the set $(2, 4)$ will provides inputs to gates 2 and 4. There are three possible ways to break four inputs into two sets to form an interconnection pattern: $(1, 2)$ and $(3, 4)$, $(1, 3)$ and $(2, 4)$, $(1, 4)$ and $(2, 3)$. The rule to arrange these patterns is that the interconnection pattern at the outputs of a stage must be different from the interconnection patterns of any of its input variables.

The principle of quadded logic is also applicable to the circuits of storing devices. In Figure 4.10 a clocked D-type flip-flop is drawn and the quadded implementation is shown in Figure 4.11. For a quadded circuit containing loops, as the flip-flop in Figure 4.11, a critical error is possibly corrected in one layer. For example, any critical error in stage B will not appear in stage E. For interconnections, however, special attention is needed so that the rules are followed everywhere in the circuit loops.

The error correction property of a quadded NAND network is in fact due to its logical characteristics. Let us take a look at the outputs of stage B: $B1$, $B2$, $B3$ and $B4$ in Figure 4.9. After passing through two NAND stages, the outputs of stage B can be represented at stage E by the Boolean function:

$$B1B3 + B2B4.$$

All $B's$ in this function should be the same in the absence of errors, but any single error in the $B's$ will not affect the correct value of the function.

In a quadded circuit, a single error can be corrected in at most two logic layers. For the errors occurring on the edge of a circuit, however, they may not be eliminated at outputs

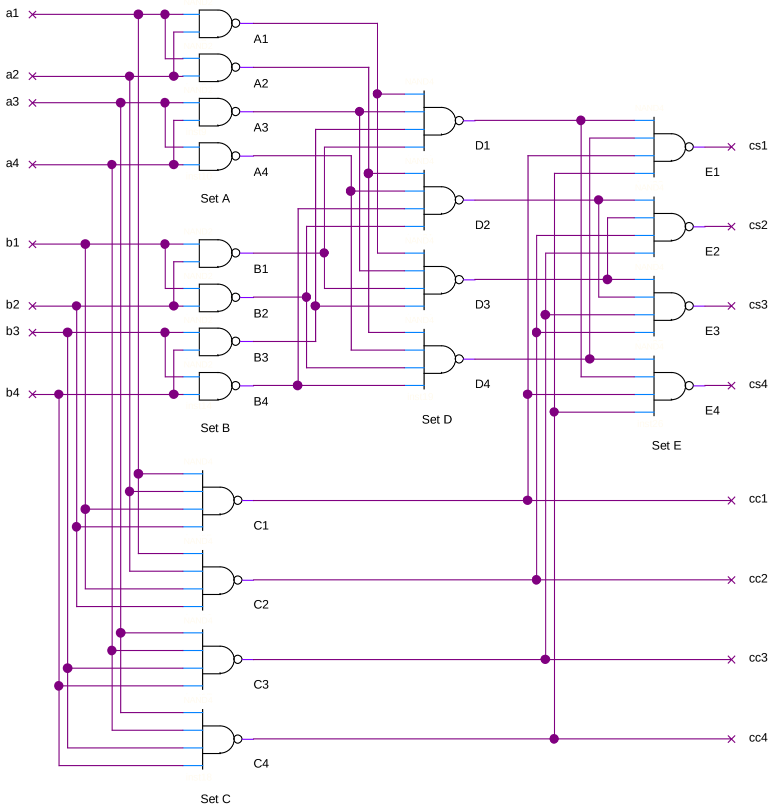


Figure 4.9: A quadded implementation of the complementary half adder.

(more specifically, a critical error within the last two layers or a subcritical error in the last layer will not be corrected at outputs). Because a single error is corrected within a rather short logical path, many multiple errors do not interact. Hence multiple errors can also be corrected in many cases. This is a particular merit of the quadded logic.

4.6.3 Triplicated interwoven redundancy (TIR)

The idea of triplicated interwoven redundancy (TIR) originates from von Neumann's multiplexing technique and the general concept of the interwoven redundant logic. We illustrate it via the TIR implementations of the complementary half adder in Figure 4.8 and the clocked D-type flip-flop model in Figure 4.10, shown in Figure 4.12 and 4.13 respectively. In the TIR complementary half adder and the TIR flip-flop each NAND gate in the nonredundant forms is replaced by a triplication and all the interconnections are accordingly triplicated as well. A TIR circuit thus has three times as many gates and interconnections as the corresponding nonredundant circuit.

Unlike the quadded logic, where systematic interconnections are required, the interconnections in a TIR circuit are in principle arranged in a random way. In practical implementation the random interconnections can be substituted by arbitrarily selected static ones that have specific routes. In a TIR circuit made up of 2-input NAND gates, for instance, there are six possible pair connections $\{(1,1),(2,2),(3,3)\}$, $\{(1,1),(2,3),(3,2)\}$, $\{(1,2),(2,3),(3,1)\}$,

$\{(1,2),(2,1),(3,3)\}$, $\{(1,3),(2,1),(3,2)\}$ and $\{(1,3),(2,2),(3,1)\}$ (by (i,j) we mean here the output of gate i in a triplication is paired with the output of gate j in another triplication to form the inputs of a gate in the next stage). The total interconnect patterns become 36 (6×6) if a distinction is made among the gate orders of a triplication in the next stage. One method to arrange interconnections is to randomly adopt one of the 36 connection patterns for all connecting pairs in adjacent layers. As shown in Figure 4.12, the interconnect patterns used in the three layers from inputs to outputs of the circuit are $\{(1,1),(2,2),(3,3)\}$, $\{(1,2),(2,3),(3,1)\}$ and $\{(1,3),(2,1),(3,2)\}$, while the circuit can be with any other interconnect patterns. We shall show in the next section through experimental studies that, in most cases, there is little difference in error correcting capacity by using one specific interconnect pattern or another.

It is interesting to notice that, if the pattern $\{(1,1),(2,2),(3,3)\}$ is used in all layers for all interconnections, the three modules in Figure 4.12 will independently perform computation, actually working as a TMR circuit, as depicted in Figure 4.14. The TIR is hence a general class of TMR implemented with random interconnections and the TMR is a particular configuration of TIR with regular interconnections. The randomness in interconnects of TIR is particularly interesting to the integration of molecular electronics, for which stochastic chemical assembly is most likely to be used as the manufacturing method.

Similarly as in TMR, a decision element is needed in a general TIR circuit as a restoring device. In Figure 4.15, a design of a 2-out-of-3 voter with NAND logic is shown. This voter can be connected to each set of the output triplication and produces a majority signal as the final output.

In a TMR circuit, due to the use of a majority voting mechanism, the failure of only one output signal would not cause the circuit as a whole to fail. Hence the effect of any single error in the working circuit can be eliminated by the voters, provided that the majority gates are fault-free. If the majority gates are subject to errors too, the errors in the circuit may not be effectively corrected by the voters. Furthermore, the errors in the voter circuits can be fatal, even when the operating circuits are fault-free.

In a general TIR circuit (i.e. a TIR with random interconnections), all components are interwoven and therefore an error is not confined to affect only one or one set of outputs. For instance, the effect of an error can be amplified via a fanout or branch circuit. Hence, a single error in a TIR circuit could cause the circuit to fail, in spite of the use of majority voters. This may intuitively imply that a general TIR circuit might be less reliable than an

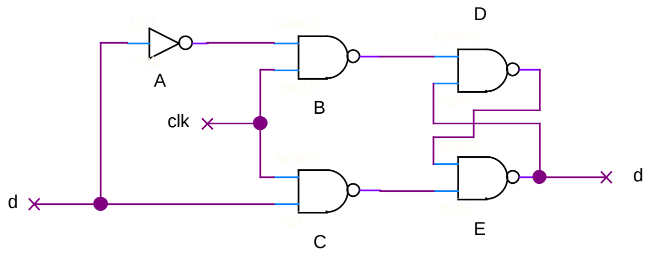


Figure 4.10: A clocked D-type flip-flop implemented with NAND logic.

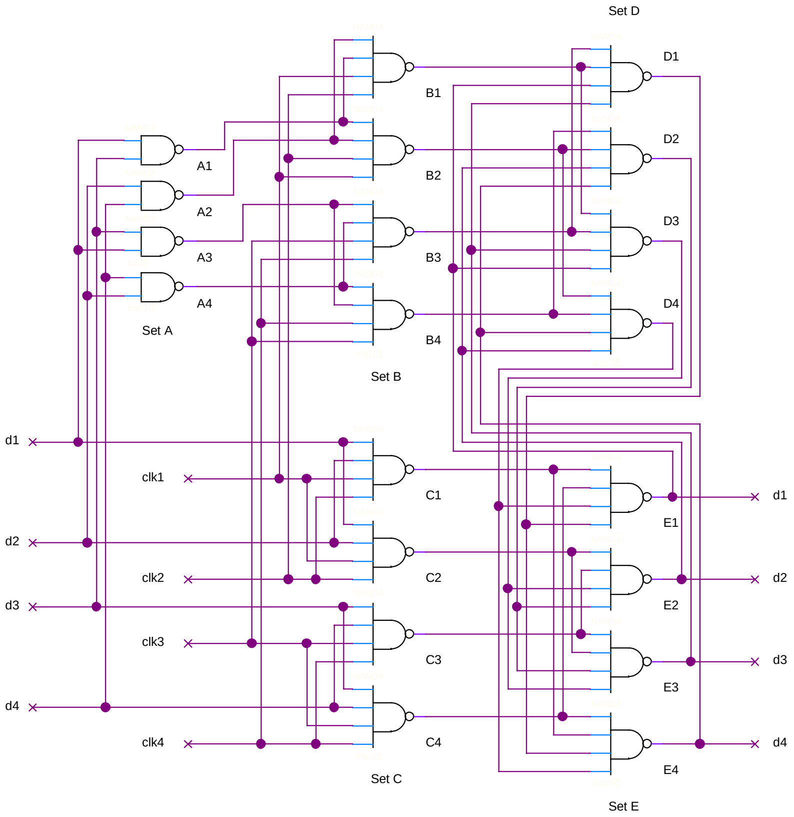


Figure 4.11: A quadded implementation of the clocked flip-flop.

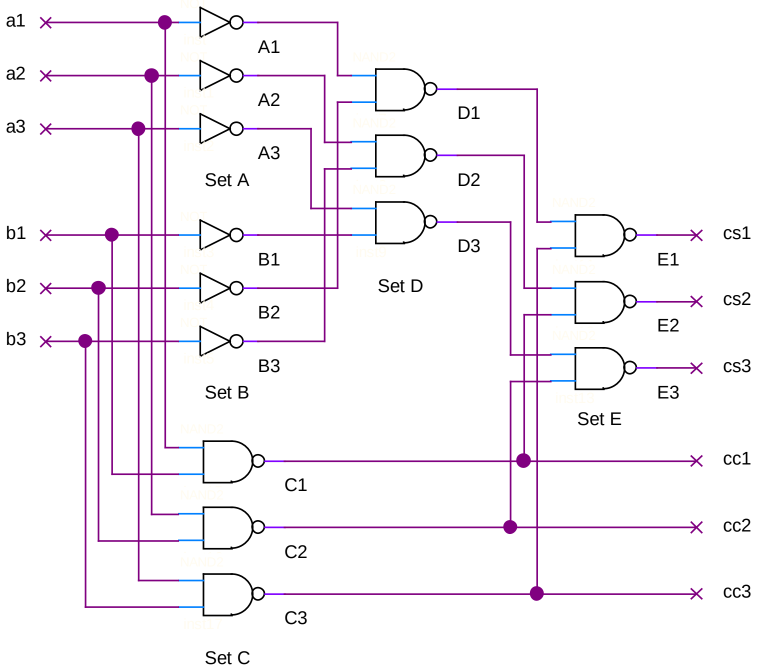


Figure 4.12: A TIR implementation of the complementary half adder.

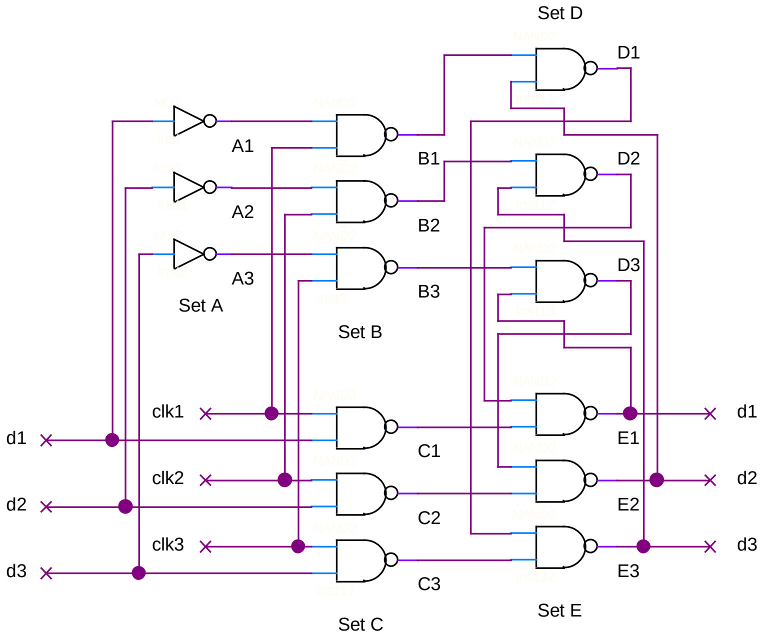


Figure 4.13: A TIR implementation of the clocked flip-flop.

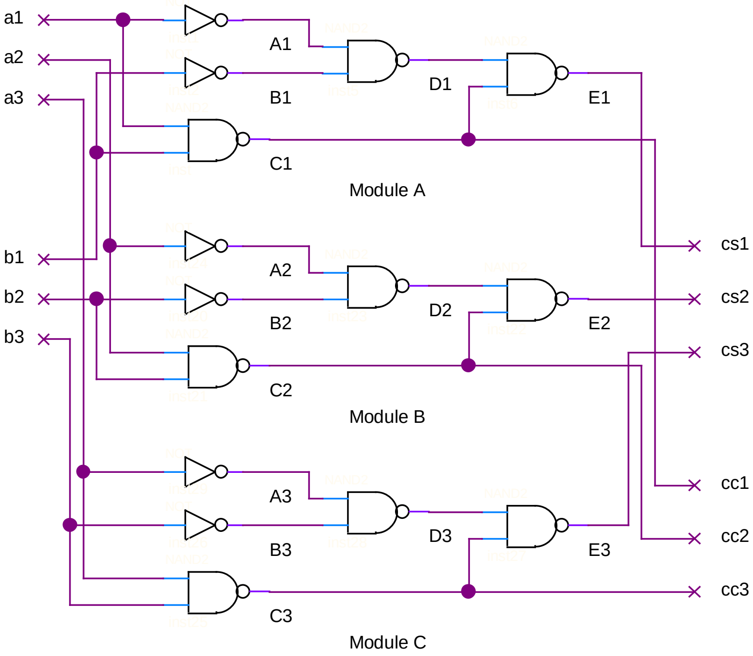


Figure 4.14: The TMR configuration of the TIR complementary half adder.

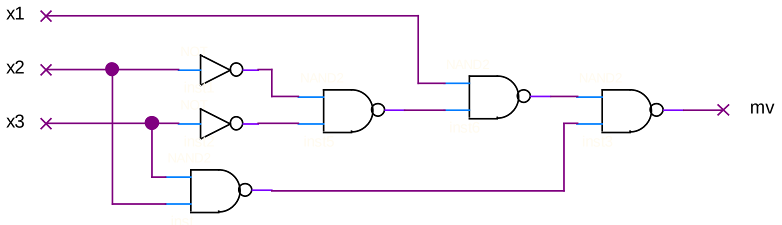


Figure 4.15: A 2-out-of-3 majority gate implemented with NAND logic.

equivalent TMR circuit. However, we shall show in the next section that this variation in reliability is in most cases negligible.

Furthermore, the occurrence of multiple errors in a TIR circuit does not necessarily cause the failure of the circuit. The multiple errors in the operating circuit may contribute to the same faulty signal of an output or they may each produce a faulty signal for different outputs such that the majority of any output still gives a correct signal. Also, the errors in the voter circuits may compensate for the effects of errors in the operating circuit such that an increased number of faults can be tolerated in a TIR circuit. In the next section, the error behaviors of a TIR circuit are studied by using a simulation based reliability model.

4.7 Experimental studies on fault-tolerant processor architectures

4.7.1 A processor prototype for array architectures and its fault-tolerant implementations

A typical processor structure for a processor array consists of an arithmetic and logic unit (ALU), registers and a multiplexer, as shown in Figure 4.16. Upon the arrival of clock signals, the input signals are latched in the registers and a computation procedure is triggered. At the end of the computation, results are sent to memory and neighbors. This processor prototype has been widely used to model processor arrays, and many variations based upon it have been successfully implemented in parallel processors for image processing and pattern recognition (see, for example, [82]-[85]).

In this processor, local memories are usually attached and connected to the inputs and outputs for data processing. The quadded and TIR structures can in principle be employed in memories to deal with errors while commonly used fault-tolerant measures for memory are reconfiguration and error correcting codes. In this study the fault-tolerance issues for memories are not specifically addressed.

In a 1-bit processor, the ALU is basically composed of a full adder, which can be constructed from the circuits of the semi-half adder in Figure 4.8 plus a few auxiliary gates. The registers, used to store the inputs from memories and neighboring processors, can be realized by the clocked flip-flops shown in Figure 4.10. The fault-tolerant implementations

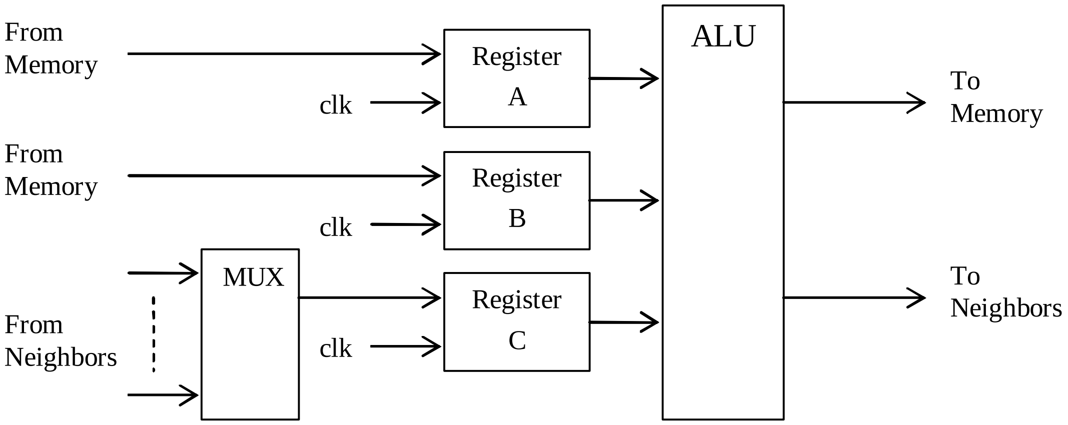


Figure 4.16: A prototype structure of a PE for a processor array.

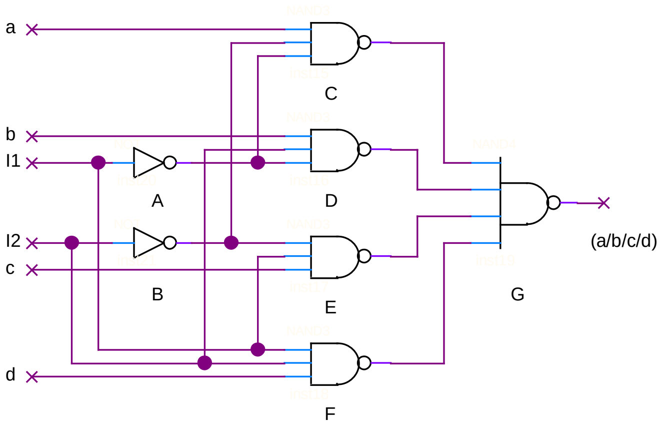


Figure 4.17: A 4-to-1 multiplexer.

of the ALU and registers can thus be obtained from quadded and TIR circuits of the complementary half adder and flip-flop discussed in the previous section.

The design of a 4-to-1 multiplexer is shown in Figure 4.17. Which one of the four inputs is selected at the output is dependent upon the combination of the two instruction bits $I1$ and $I2$. For a linear processor array, however, a 2-to-1 multiplexer requiring only one instruction bit I is sufficient for the design, as shown in Figure 4.18. For simplicity, we only show the schematics of the quadded and TIR implementations of the 2-to-1 multiplexer, in Figure 4.19 and 4.20 respectively.

Hence, all the elements composing the processor structure in Figure 4.16 are available in their fault-tolerant implementations of quadded and TIR circuits. In the next section a fault injection simulation is proposed to study the reliability of these fault-tolerant processor structures.

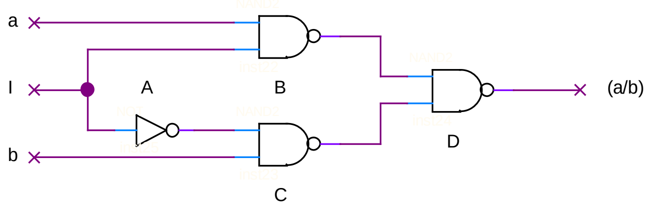


Figure 4.18: A 2-to-1 multiplexer.

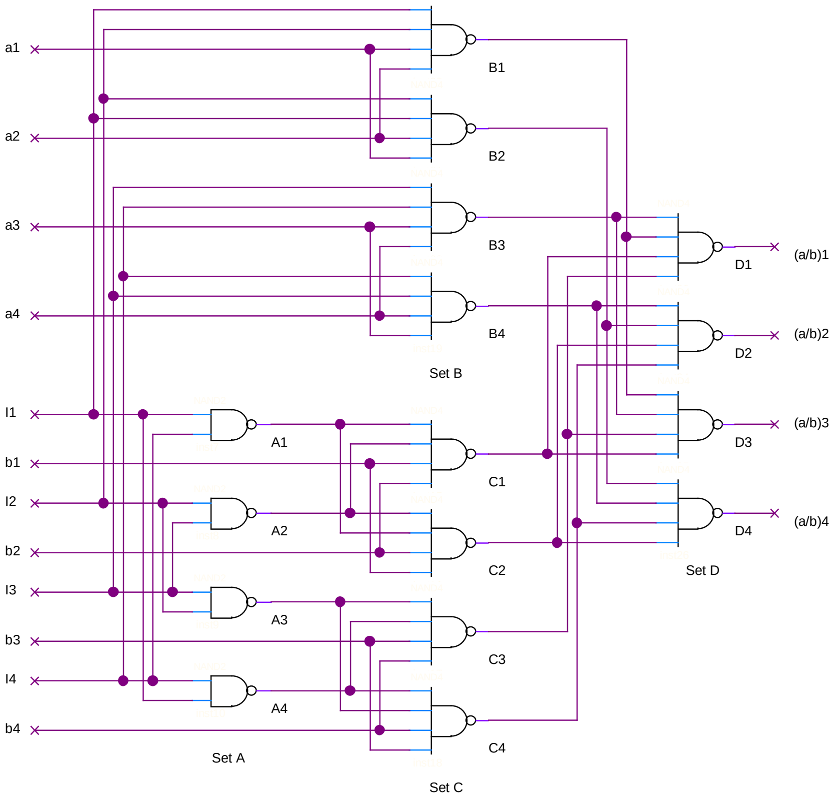


Figure 4.19: A quadded implementation of the 2-to-1 multiplexer.

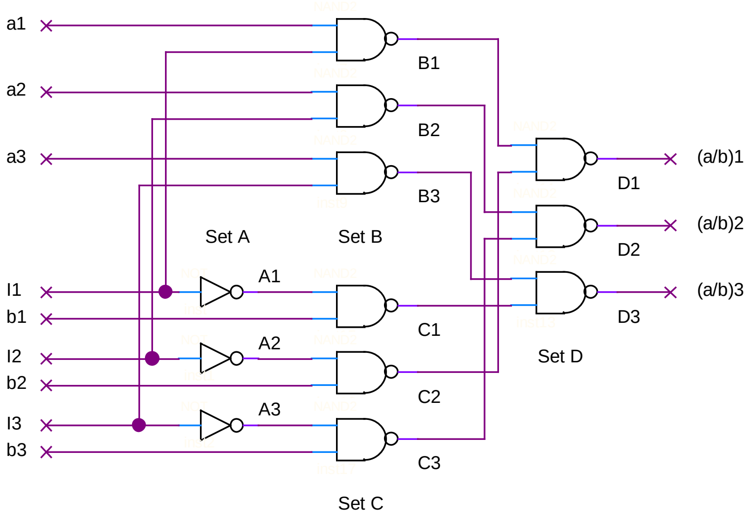


Figure 4.20: A TIR implementation of the 2-to-1 multiplexer.

4.7.2 A fault injection simulation

The reliability of an interwoven logic network has been evaluated by using the minimal cut [125] and a combinatorial method [127]; they were however both extremely cumbersome, especially for large circuits. In this study, a fault injection simulation procedure has been performed to investigate the effects of multiple component failures in quadded and TIR structures. In the simulation, faults were considered to appear in logic gates, thus producing possibly faulty signals at the outputs of the erroneous gates. In manufacturing, errors can also occur on interconnections. Interconnection faults, though not considered in this study, can readily be modeled in the simulation by taking into account possible errors on both input and output lines. This simulation procedure is in principle applicable to any fault-tolerant circuit. It goes as follows:

1. Start at all good states. A number of faulty gates (the number, m, starting from 0 and up to the maximum number of faulty gates in the circuit) is randomly selected from all of the gates in the fault-tolerant circuit and a randomly selected stuck-at-0 or stuck-at-1 fault is emulated at the output of each faulty gate.

2. A set of input pattern is applied to the fault-tolerant circuit, in which faults have been injected. The outputs produced in the circuit are then compared with the correct ones. If the fault-tolerant circuit provides the correct outputs, repeat Step 2 until the complete set of input patterns have been tested. Otherwise, increase the number of failed simulations, k_m, by 1.

3. Increase the number of simulations performed thus far, n, by 1. If n is less than the total number of simulations to be performed, N, go to Step 1. Otherwise, go to Step 4.

4. Compute the failure rate of the simulated fault-tolerant circuit by the number of injected faults, $F_m = k_m/N$. Increase the number of faults injected into the fault-

tolerant circuit, m, by 1. If m is no larger than the maximum number of faulty gates in the fault-tolerant circuit, go to Step 1. Otherwise, end the simulation.

In this procedure, a simulation is carried out with all possible input patterns of the simulated circuit. In other words, a circuit is only reliable when it succeeds in producing correct outputs for all input patterns. Faults are injected at once before the circuit is put into operation. This is likely to be the case for manufacturing defects. This fault injection simulation can be modified to account for clustered or correlated defects for reliability modeling [134], [135]; random faults are however assumed in this investigation as has been the case in most of the reliability models [71]. A simulation procedure was proposed in [130] for the modeling of transient faults that spontaneously appear during system operation.

This simulation procedure has been applied to the quadded and TIR as well as the nonredundant implementations of the processor structure in Figure 4.16. At the end of each simulation, a failure rate by the number of faults injected into the fault-tolerant circuit, F_m, is obtained as the number of times the fault-tolerant circuit fails, k_m, divided by the total number of fault injection simulations performed, N ($N = 10000$ in all simulations performed here).

4.7.3 The effects of critical gates (CGs) in voters

The failure rates obtained for the nonredundant, quadded and TIR circuits are shown in the first two rows of data in Tables 4.4 and 4.5 for up to 8 faults. As can be seen from the tables, the failure rates obtained for the quadded and TIR circuits are significantly lower than that of the nonredundant circuit. It can also be seen, however, that the quadded circuit has a failure rate of nearly 20% and that the TIR circuit also has a non-zero failure rate in the presence of a single fault. This might not be surprising because in both of the quadded and TIR circuits there are certain critical gates (CGs) in the sense that the failure of any CG will with a high probability cause the failure of the whole circuit.

In general, the number of CGs a fault-tolerant circuit has, is approximately the sum of the number of CGs related to each output. If any CG is only connected to one output and each output has the same number of CGs, the number of CGs in a fault-tolerant circuit is given by

$$N_{cg} = N_{op} N_c, \tag{4.21}$$

where N_{op} is the number of outputs and N_c is the number of CGs related to each output.

In a quadded circuit, for instance, N_c is the number of gates within the last two layers of an output (considering the case of critical errors). For the quadded processor circuit in this study, the number of CGs is obtained by $2 \cdot (4 \cdot 3)$ (2 *outputs*$\cdot$4 *times of the* 3 *nonredundant gates including the output gate and the two gates providing inputs to it*). The failure rate of a quadded circuit in the presence of a random single error is basically dependent on the fraction of the CGs to the total gates of that circuit. One way to improve the reliability of a quadded circuit is therefore to lower the fraction of the CGs in it.

A restorative device, made up of two layers of 2-input NAND gates as shown in Figure 4.21, can be attached to each output of a quadded circuit. In any circuit containing these restoring devices, the CGs become the gates in the restoring layers and the number of CGs is

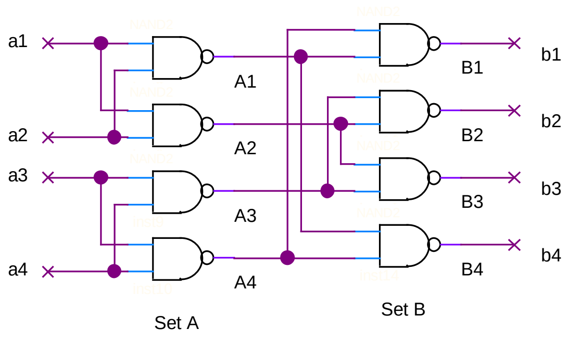


Figure 4.21: A restorative device for quadded circuits.

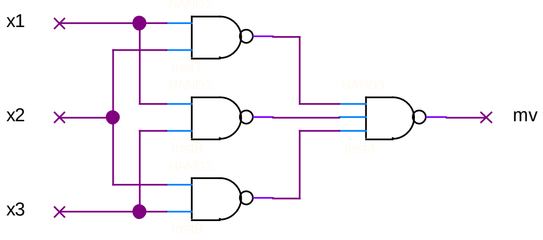


Figure 4.22: A 2-level majority voter.

thus reduced to the constant minimum $N_{op}\cdot(4\cdot 2)$. For the processor model in this study, the number of CGs (and thus approximately the fraction of CGs to the total gates) is reduced by 1/3 and the failure rates are lowered through the use of the restorative devices, as the data show in Table 4.4. This structure of restoring devices can be applied to any quadded circuit.

In a TIR circuit, the majority voters function as restoring devices. Since the reliability of any voter is critical to the survival of the whole circuit (though in some cases the failures of gates in a voter may compensate for the failures of other gates and thus an imperfect voter may still produce a correct output), the number of CGs in a TIR circuit is obtained through equation (4.21), in which the parameter N_c indicates the number of gates in a voter circuit. Thus the reliability of a TIR circuit could be improved by reducing the complexity of restorative circuits. A simpler design of a 2-level majority voter, consisting of four NAND gates, is shown in Figure 4.22. As our simulations reveal in Table 4.5, however, the use of this voter does not produce a better performance. Instead, it increases the failure rates, compared to that obtained by using the voter in Figure 4.15, which consists of four NAND gates and two inverters (or six NAND gates). Hence, a degradation caused by a slightly increased complexity of a voter design can simultaneously be compensated for by a better fault-tolerance this specific structure affords. This exhibits a compensating effect of faults in the voters of a TIR circuit.

The designs in Figures 4.15 and 4.22 are favorable for applications based on "conventional" transistors. In quantum and nanoelectronic regime, however, the implementation of majority logic could be greatly simplified. A simple structure of a single majority gate,

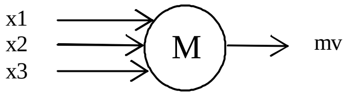


Figure 4.23: A single majority gate.

Table 4.4: The failure probabilities of the nonredundant processor structure and the quadded implementations without restorative devices, with unreliable restorative (UR) devices and perfect restorative (PR) devices. (m: the number of random faults in a circuit)

	$m = 1$	$m = 2$	$m = 3$	$m = 4$	$m = 5$	$m = 6$	$m = 7$	$m = 8$
$Nonredundant$	0.775	0.949	0.988	0.998	0.999	1.000	1.000	1.000
$Quadded$	0.191	0.392	0.564	0.700	0.808	0.880	0.931	0.960
$Quadded\ (UR)$	0.115	0.260	0.418	0.568	0.690	0.793	0.868	0.915
$Quadded\ (PR)$	0.000	0.061	0.165	0.301	0.432	0.565	0.678	0.771

as schematically shown in Figure 4.23, has been proposed for possible implementations using quantum-dot cellular automata [87], single electron tunneling (SET) devices [139] and superconducting circuits of Josephson junctions [42]. Apparently the best way to improve reliability is to use highly or "perfectly" reliable components in restorative circuits. The quadded and TIR circuits with these various designs of restoring devices have been studied using the fault injection simulation, and the results are shown in Tables 4.4 and 4.5.

As can be seen, the TIR circuit using single majority voters presents a better reliability than those using more complex voter designs. Both of the quadded and TIR structures provide best performance with the use of perfect restoring circuits (by which the numbers of CGs are reduced to 0). The failure rates of the quadded circuits are lowered with the use of restoring devices and the gate reliability in restoring circuits is in particular significant for a quadded implementation. In general, the number of CGs (as well as the fraction of

Table 4.5: The failure probabilities of the nonredundant processor structure and the TIR implementations with the unreliable voters (UVs) in Figure 4.15, the 2-level voters (2Vs) in Figure 4.22, the single gate voters (SVs) in Figure 4.23 and perfect voters (PVs). (m: the number of random faults in a circuit)

	$m = 1$	$m = 2$	$m = 3$	$m = 4$	$m = 5$	$m = 6$	$m = 7$	$m = 8$
$Nonredundant$	0.775	0.949	0.988	0.998	0.999	1.000	1.000	1.000
$TIR(UVs)$	0.057	0.498	0.788	0.924	0.974	0.992	0.997	0.999
$TIR\ (2Vs)$	0.079	0.523	0.799	0.925	0.974	0.991	0.998	0.999
$TIR\ (SVs)$	0.022	0.453	0.757	0.906	0.966	0.987	0.996	0.998
$TIR\ (PVs)$	0.000	0.428	0.738	0.900	0.963	0.986	0.994	0.998

CGs to the total gates) in a fault-tolerant circuit is reduced with the use of a simpler design for restorative devices and thus the failure rate of the circuit is expected to be lowered. This thought of minimizing the fraction of CGs in design is in principle applicable to the restorative construction of any fault-tolerant circuit. In the next subsection we present a simulation based reliability model, which is used to further investigate these fault-tolerant constructions.

4.7.4 A simulation-based reliability model and results

If in a fault-tolerant circuit each logic gate has the same error rate, and errors are randomly and independently distributed, the probability of a number of faulty gates follows the binomial distribution, i.e.

$$P(m) = \binom{N}{m} p_f^m (1 - p_f)^{N-m}, \tag{4.22}$$

where p_f is the error rate of a logic gate in the circuit, m and N are respectively the number of faulty gates and the total number of gates in the circuit.

By considering the number of faults, m, to be a random variable, the failure rate F_m can be used as a failure distribution in m, and $F_m = k_m/N$. A reliability distribution R_m, which gives the probability that the fault-tolerant circuit survives as a function of the number of faults in the circuit, m, is obtained from the failure distribution F_m, as follows:

$$R_m = 1 - F_m = 1 - k_m/N. \tag{4.23}$$

The reliability distribution R_m gives the probability that a fault-tolerant circuit will continue to properly operate in the presence of m faults. The reliability of the fault-tolerant circuit is therefore obtained by summing over all conditional reliabilities with the presence of faults, i.e.

$$\begin{aligned} R_{FT} &= \sum_{m=0}^{N} R_m P(m) && (4.24) \\ &= \sum_{m=0}^{N} \binom{N}{m} (1 - k_m/N) p_f^m (1 - p_f)^{N-m}. && (4.25) \end{aligned}$$

Hence, the reliability of a fault-tolerant circuit can be obtained from the simulation based formula (4.25).

In the simulation the random interconnections in TIR circuits are substituted by randomly selected static ones, of which TMR is a specific configuration with regular interconnects. We have simulated the TIR processor structures with various interconnect patterns (including the one of TMR). Each pattern is obtained by arbitrarily selecting the interconnects among gates, except for the one particularly specified for TMR. The reliabilities obtained from the simulation based formula (4.25) are plotted against the error rate of a NAND gate in Figure 4.24 for six sets of different interconnect patterns.

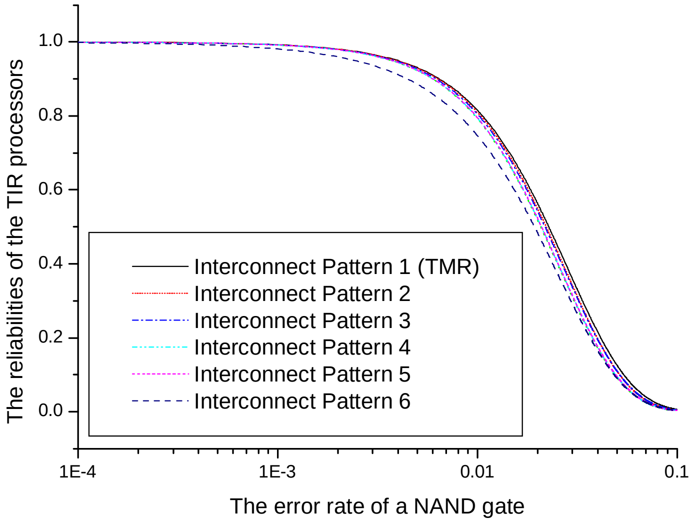


Figure 4.24: The reliabilities of TIR circuits with random interconnects (including the TMR with regular interconnects).

As revealed in the figure, the TIR structures with randomly selected interconnections present slightly lower probabilities of system survival than the TMR structure. In most cases, however, these differences in reliability are hardly discernable. For a small fraction of interconnect patterns (e.g. interconnect pattern 6 in Figure 4.24), this variation becomes noticeable. Our further investigations show that this is due to a fanout effect of erroneous signals and a malicious interconnect combination of the output gates in the flip-flop (gates D and E in Figure 4.13) of Register C (in Figure 4.16). As a result, any single error in the multiplexer (MUX in Figure 4.16), if propagated into the flip-flop through the fanout circuit, causes failures of two outputs of the flip-flop. These failures present a majority and thus cannot be corrected by the voters attached to the circuit.

Fortunately, there is just one of 36 possible interconnect combinations of the output gates that would cause the failure of the flip-flop (and thus the processor) due to a single error in the multiplexer. Generally, the variations in reliability obtained from various configurations of TIR (including TMR) are small so that they are negligible. These TIR circuits are virtually equivalent in terms of reliability without a distinction in interconnect patterns. This randomness in interconnects is a prominent feature of TIR, albeit in some cases it may introduce a decrease in reliability improvements. For simplicity, we take a configuration with fixed interconnects (specifically the one with interconnect pattern 2 in Figure 4.24) as a typical TIR structure for further simulations.

Next we made a comparison of the TIR model by simulations and the classical TMR model by theory. The TIR reliabilities obtained from formula (4.25) and the TMR reliabilities obtained from formulae (4.17) and (4.20) are plotted against the error rate of a single gate in Figure 4.25 with the gate error rate p_f varying in $(10^{-4}, 10^{-1})$. As can be seen, the classical TMR model gives a rather pessimistic estimation of the system reliability, since it does not take into consideration the compensating effects of critical and subcritical faults.

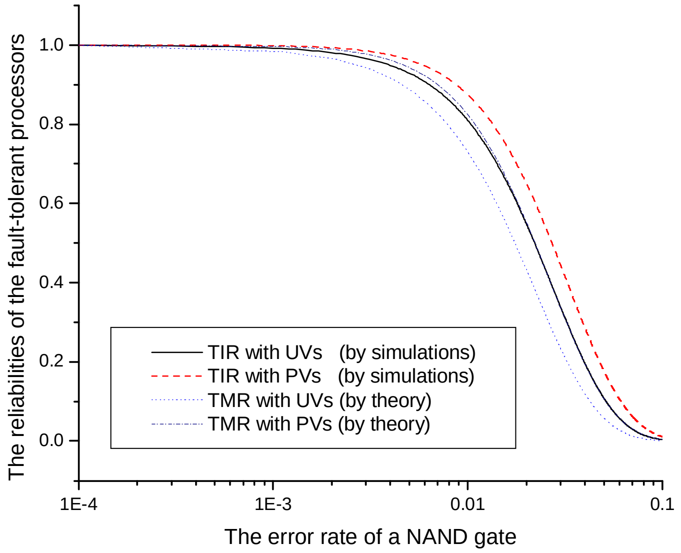


Figure 4.25: The reliabilities of TIR circuits by simulations and TMR circuits by theory, with unreliable voters (UVs) and perfect voters (PVs).

So far the simulations of TIR circuits were carried out with the voter circuit in Figure 4.15. We further present the results obtained by using different voter designs. The reliabilities of the TIR circuits, as well as those obtained from the nonredundant form, are plotted against the error rate of a single NAND gate in Figure 4.26 with p_f varying in $(10^{-4}, 10^{-1})$. It can be seen that the TIR circuit with perfect voters performs the best, while the circuit with single majority gates for voters presents a better performance than those using more complex designs. In general, better reliabilities are obtained for TIR circuits by using simpler designs in restorative devices. However, a performance degradation due to a complex voter structure could be compensated by an error correction capacity this structure may provide. The voter structure in Figure 4.15, implemented with NAND gates, is such an example. It presents a nearly equivalent fault-tolerance as the simpler voter structure in Figure 4.22. Though the TIR structures are more reliable than the nonredundant one, this is only true when the gate error rate is strictly not larger than a threshold. For the TIR processor model in our study, this value is approximately 10^{-2} for the voters in Figures 4.15 and 4.22, while it increases with the use of a more compact voter design.

The reliabilities of the quadded circuits, as well as those obtained from the nonredundant form, are plotted against the error rate of a single NAND gate in Figure 4.27. Somewhat surprisingly, the performance of the quadded form without any restoring devices is inferior to that of the nonredundant circuit. This is largely due to the relatively large number of CGs in the original quadded circuit, as we discussed in the previous section. With the use of restorative devices, the quadded circuits present better performance and, in particular, a fairly large boost in reliability is obtained with the use of “perfectly” reliable devices in restoring circuits.

To compare their performance, the reliabilities obtained from the quadded and TIR circuits are shown in Figure 4.28. The voter in Figure 4.15 is used for the simulations of

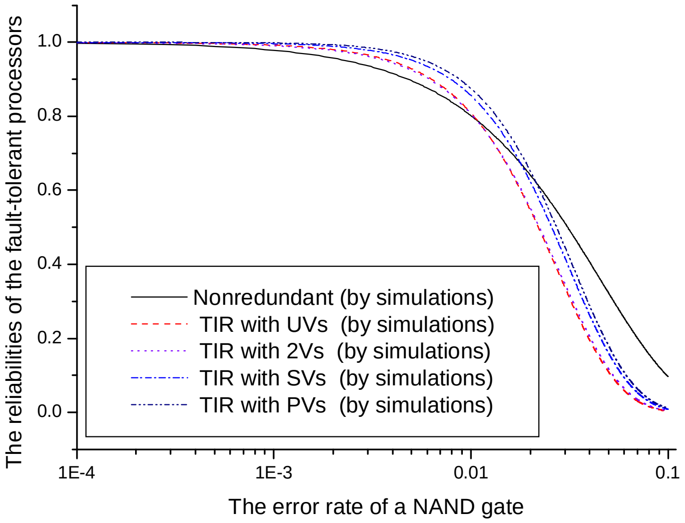


Figure 4.26: The reliabilities obtained by simulations of the nonredundant and TIR circuits with the unreliable voters(UVs) in Figure 4.15, the 2-level voters (2Vs) in Figure 4.22, the single gate voters (SVs) in Figure 4.23 and perfect voters (PVs).

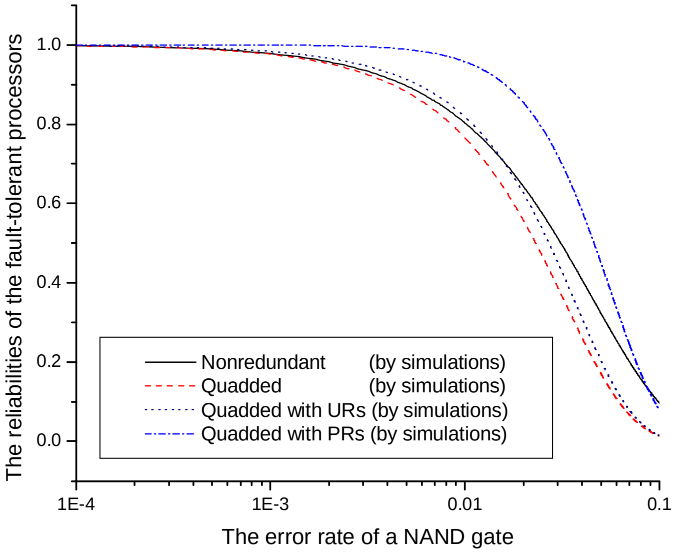


Figure 4.27: The reliabilities of nonredundant and quadded circuits by simulations, without restorings, with unreliable restorings (URs) and perfect restorings (PRs).

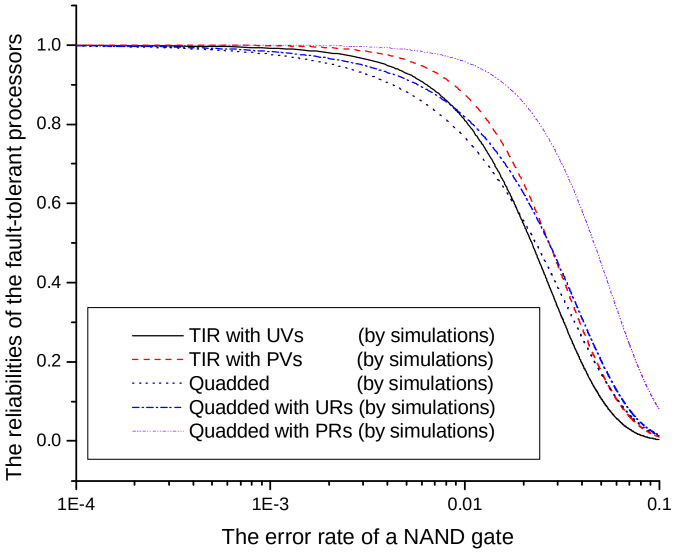


Figure 4.28: The reliabilities of TIR and quadded circuits by simulations, with unreliable voters or restorings (UVs or URs) and perfect voters or restorings (PVs or PRs).

the TIR structures. In the region that is of practical interest (roughly where the module reliability is no less than 0.5), as revealed in Figure 4.28, the TIR circuit with unreliable voters is more robust in the protection against errors than the quadded circuit without any restorative devices, and its reliability is comparable with that of the quadded circuit with unreliable restorative devices. For those with perfect voters or restoring devices, however, the quadded implementation is superior in reliability to the TIR implementation.

4.8 N-tuple interwoven redundancy (NIR)

Implementing critical components with larger reliable devices is in principle possible in present CMOS technology. In nanoelectronic or molecular implementation, however, it would be highly unlikely or inefficient to use CMOS devices in restorative circuits. This is for two reasons: first, CMOS transistors are large and would decrease the density of the molecular or nanoelectronic circuit [132], and second, the interconnects between CMOS and molecular electronics would be a technical challenge [123]. However, the reliability of a TIR circuit can still be further improved without employing perfect critical components. One approach is to use higher-order redundancies. The TIR, as a general class of TMR, can readily be extended to, say, N-tuple interwoven redundancy (NIR), similarly as TMR to NMR (in contrast, the quadded logic is hardly scalable to higher orders). Thus, the NIR is a general class of NMR, but with random interconnections.

The degree of redundancy R used to construct an NMR system is determined from the desired number of faulty circuit modules to be masked, E, by [129]

$$2E + 1 \leq R \leq (E + 1)^2. \tag{4.26}$$

A grouping parameter, K, is used to design the voter circuit of an NMR. For a 2-level voter circuit, K indicates the number of inputs of a gate in the first level of the voter [129], and

$$E + 1 \leq K \leq R - E. \tag{4.27}$$

For TMR and the 2-level voter circuit in Figure 4.22, for instance, we have $E = 1$, $R = 3$ and $K = 2$. For a general NMR system, the number of gates in the first level of a voter can be obtained by selecting K-out-of-R elements, and is given by

$$c = \binom{R}{K} = \frac{R!}{K!(R-K)!}. \tag{4.28}$$

In practice, applications of NMR systems are mainly restricted to odd numbers of replications, i.e. $R = 3, 5, 7, 9, \ldots.$

To investigate the fault-tolerance of NIR systems, we have studied the NIR implementations of the processor structure in Figure 4.16, with $R = 3$ (i.e. TIR)$, 5, 7$ and 9. Similarly to TIR, an NIR voter circuit can follow a design of the 2-level structure in Figure 4.22 or a design of the single majority gate structure in Figure 4.23. For the 2-level voter design, the numbers of NAND gates in NIR voters for $R = 3, 5, 7$ and 9 ($E = 1, 2, 3$ and 4) are $4, 11, 36$ and 127, according to equation (4.28). We can see that the size of a voter grows faster than the increase of the redundancy in an NIR circuit. This implies that the performance gain by a higher degree of redundancy may be degraded by an increased complexity of an NIR voter. For a single majority gate design, however, the complexity of an NIR voter is kept the same or slightly increased due to an increase in gate interconnections in a higher order of NIR.

The reliabilities obtained by simulations are plotted against the gate error rate in Figure 4.29, for the NIR circuits using these two types of voters. As can be seen, indeed, the use of a higher order of redundancy ($R = 5, 7$ or 9) does not offer a better fault-tolerance if a 2-level voter design is used in an NIR system. In fact, the system degenerates to a level that is less reliable than the nonredundant one. The higher the degree of redundancy is, the worse the reliability. If a single majority gate is used as a voter, however, a higher reliability results in an NIR system with a higher degree of redundancy, i.e., an improved system reliability is obtained by using a higher order of NIR. These indicate the significance of voter implementations in an NIR system.

With the use of single majority voters, a reliability improvement can only be obtained when the gate error rate in an NIR system is lower than a threshold. As revealed in Figure 4.29, this threshold has a higher value in a higher-order NIR system. For instance, a TIR system ($R = 3$) does not afford an advantage over the nonredundant structure until the error rate of a gate reaches approximately 0.02, while an equivalent NIR structure with $R = 9$ provides a better performance as long as the gate error rate is not larger than approximately 0.05.

4.9 Discussion

In section 4 an abstract processor model of chains of multiplexing layers is given and it is analytically shown that, with a bundle size of 50, the multiplexing processor has a probability

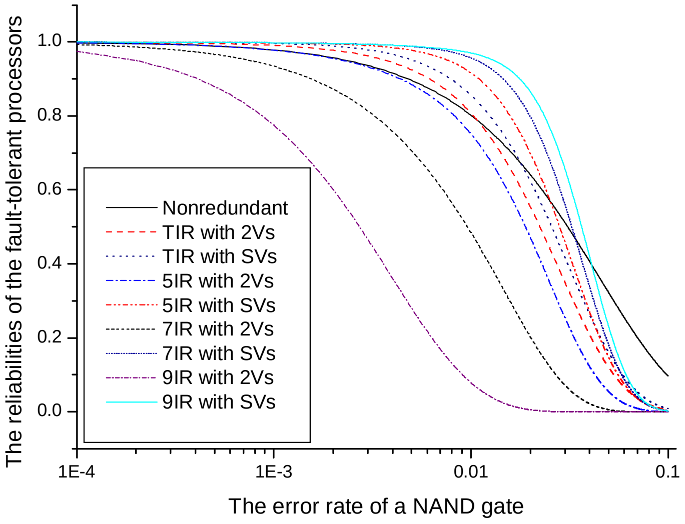


Figure 4.29: The reliabilities obtained by simulations of the nonredundant and NIR circuits with 2-level voters (2Vs), following the design in Figure 4.22, and the single gate voters (SVs), following the design in Figure 4.23, for R=3 (i.e. TIR), 5, 7 and 9.

of survival of 0.868 for a gate error rate of $\varepsilon = 10^{-2}$. Then it is further demonstrated that a hierarchically reconfigurable architecture of these processors will have an overall reliability of approximately 99%. In section 7, we present a realistic processor design and the reliabilities of its quadded and NIR (TIR) implementations are investigated through a simulation based approach. The proposed simulation procedure has been performed on various circuit configurations.

With a gate error rate of 10^{-2}, for example, the nonredundant structure of the processor has a reliability of 0.800 while the TIR structure has a reliability of 0.810 with the use of the unreliable voters shown in Figure 4.15 and a reliability of 0.875 with the use of perfect voters. If these TIR processors are used as the building blocks of the reconfigurable architecture as discussed in section 4 and the majority circuits are made from the same unreliable devices as the operating circuits, the overall reliability of the architecture would be significantly worse than that obtained in section 4. The outcome would be that only a smaller error rate could be tolerated in such an architecture. With the use of TIR with unreliable voter circuits the largest boost occurs at the gate error rate of 0.005 where the reliability is raised from 0.895 to 0.928. The system reliability of the architecture cannot be improved by employing higher orders of NIR with the use of unreliable 2-level voters.

If the critical voter circuits are made of single majority gates or “perfectly” reliable devices, however, the reliability level of the architecture would be achieved with a TIR implementation and the required redundancy would be substantially reduced. This system reliability can further be improved by using a higher order of NIR with single majority gates. If we apply 5-tuple interwoven redundancy (5IR) with single gate voters, for example, a processor reliability of 0.916 will be obtained for a gate error rate of 10^{-2}. Thus, a better system reliability results than that obtained in section 4. This indicates that the redundancy required in the reconfigurable architecture can be reduced by a factor of 10, i.e. from 10^2 down to approximately 10, through the implementation of 5-tuple interwoven redundancy in the basic circuits.

4.10 Summary

We have presented a hypothetical defect- and fault- tolerant architecture, in which von Neumann's NAND multiplexing is implemented in basic circuits and reconfigurable architectures are hierarchically mapped to the overall system. The system is expected to be working at an acceptable reliability level at the expense of having about 10^2 times redundant components. The triplicated interwoven redundancy (TIR) has been proposed as a general class of triple modular redundancy (TMR), but implemented with random interconnections. The TIR implementations of a 1-bit processor element, as well as the quadded implementations, are presented, and a fault injection simulation is performed to investigate the fault-tolerance, aiming at a low redundancy design in fault-tolerant architectures. It is shown that the simulated TIR circuits present better reliability evaluations than theoretical TMR models, due to the compensating effects of multiple errors. The TIR is extended to higher orders, namely, the N-tuple interwoven redundancy (NIR), in order to achieve higher system reliabilities. The use of 5-tuple interwoven redundancy leads to an economical redundancy factor of less than 10 for the system architecture proposed in section 4.

The critical components are very important to the reliability of a quadded processor structure. The quadded processor structure affords little or no advantage without the use of any restorative devices or with unreliable restorative devices, but presents a great advantage with the use of perfect restoring devices. In general, the reliability of a TIR circuit is comparable with that of an equivalent TMR circuit while, for certain interconnect patterns, the TIR structure may present an inferior performance to TMR due to its interwoven nature in gate interconnections. The percentage of these configurations that present relatively low reliabilities depends on the specific structure of a TIR circuit, though in our study this occurs only at a small probability. In this regard, further measures might be needed to raise the reliability of a TIR circuit.

The design and implementation of restorative devices (voters) are important for the NIR (TIR) structure. In general, a better reliability results in an NIR circuit by using a simpler design of restorative devices, while a performance degradation due to a complex voter structure could be compensated by the specific structure itself. With the use of conventional 2-level voters, as shown in our study, a higher order of NIR presents a degraded system reliability, because of a significantly increased size of the voters. With the use of single majority voters, which are favorable for implementations in nanotechnology, the NIR structure offers an advantage over the nonredundant form when the device error rate is not larger than a threshold value, while the optimal gain in reliability is dependent on the circuit size and component error rate. With the single gate voter design, a better system reliability is obtained by using a higher order of NIR.

The NIR (TIR), derived from von Neumann's multiplexing technique, is particularly interesting for the physical implementation of molecular nanocomputers. First, an NIR circuit, unlike NMR, does not require a systematic interconnect pattern. Instead any randomly formed static interconnect pattern is workable. This is highly favorable to the stochastic process of chemical assembly in which randomness is inherent. Second, any logical gate or even higher level of functioning blocks, e.g. half adders, full adders, or latches, can in principle be the basic elements of an NIR structure. (The consideration to arrange critical and subcritical errors in an alternate order, as we do in the study using NAND gates, is likely to give the structure a better fault-tolerance.) Third, the components of an NIR circuit do not have to be defect-free, because the structure itself, as has been seen, is defect-tolerant. Hence the NIR structure presents minimum precision requirements on the manufacturing of devices and interconnects. Moreover, since it might be easy, i.e. with a high probability, to train a randomly assembled molecular circuit to an NIR or NIR-like circuit, only a modest configuration time is required after fabrication. Finally, if an NIR circuit module is used as the building block of a reconfigurable architecture, the reconfigurability can then be limited to the module or higher levels, instead of gate or device level. This would greatly lower the difficulty in system testing and fault diagnosis. Since the NIR simultaneously provides the architecture protection from transient errors in system operation, an NIR based reconfigurable architecture is robust against both manufacturing defects and transient faults for systems based on molecular or nanoelectronic devices.

Chapter 5

Computing with Locally-Coupled Josephson Circuits

5.1 Introduction

[1] Historically, parallel processing offered considerable performance advantages in many areas of computing. Several approaches have been explored and prototype systems were constructed. Due to the characteristics of many nanoelectronic devices, such as low power consumption, low drive capability and easy local interactions, parallel architectures that are highly regular and locally connected (typically the single instruction and multiple data (SIMD) computers [86] and cellular nonlinear networks (CNNs) [88]) have been studied as candidate architectures for nanocomputers.

The SIMD computers consist of assemblies of identical, simple processor elements (PEs), usually each connected to its nearest neighbors in a linear or square array. Instructions are fed in a parallel stream to every PE and each instruction is executed simultaneously by PEs. Memories are distributed uniformly across the array such that each PE can access its local storage directly. SIMD systems have been successfully used in various areas of data processing (see, for examples, [86], [140] and [89]). The field of high performance image processing in particular has brought forward archetypical systems, such as the SPA (square processor array) CLIP4 [82], MPP [83], and currently the successful systems of LPA (linear processor array) IMAP-CE [84] and XETAL (SPA) [85]. The architectural issues of a SIMD array have been discussed in [93] for the implementations of quantum cellular automata (QCA) and resonant tunneling diodes (RTDs).

Cellular nonlinear networks (CNNs) represent a circuit architecture that is capable of high-speed parallel signal processing [88]. A CNN is usually a two or three dimensional regular array of identical cells with analog signals as state variables. The cells are locally interconnected and directly communicate with each other through their nearest neighbors. As a real-time signal processing architecture, CNNs have important applications in image processing and pattern recognition, such as nonlinear digital filters, noise removal, feature extraction, etc. [88] Because of the local connectivity, which is independent of the number of cells, the CNN architecture is in principle scalable and reliable. The potential applications

---

[1] The content of this chapter has been published in [41], [42], [43] and [44].

of CNNs using resonant tunneling diodes (RTDs) [95] and single electron transistors (SETs) [96] have been investigated, and a quantum CNN has been proposed using quantum dots by exploring their local quantum dynamics and global interactions [98].

Quantum computing has been extensively studied as a computing paradigm employing quantum mechanics [105]. A quantum computer performs a massively parallel processing on quantum mechanical superpositions of quantum bits or qubits. To perform a quantum computation, one must be able to prepare qubits in a desired initial state, to coherently preserve and manipulate superpositions of qubits' states, to couple qubits together, to measure their states, and to keep them relatively free from external interactions that induce noise and decoherence [141]. Essentially, any two-state quantum system that can be addressed, controlled, measured, coupled to its neighbors and decoupled from its environment, is potentially useful for quantum computing. Various physical systems have been proposed to realize a quantum computer [105]. Among these, mesoscopic superconducting circuits of Josephson junctions, produced by modern lithography, appear promising for integration in electronic circuits and for large-scale applications [37], [38]. Recently, the coherent superposition of two macroscopic persistent-current states on a superconducting Josephson circuit has been observed [39], and the coherent quantum dynamics of this Josephson flux qubit has been demonstrated [40]. The Josephson circuit has thus come up as a promising candidate for realizing a quantum computer. These superconducting circuits of Josephson junctions can also be designed to work in the classical domain. Classical processing circuits can be obtained by the same manufacturing process and can thus be used as supplementary or control circuitry affiliated with the circuits devoted to quantum processing. Boolean logic can further be obtained from the circuits, establishing a classical parallel computing architecture.

In this chapter, we first present a close look at the Josephson circuit in section 2, focusing on the properties we are interested in. In section 3, we present a classical array architecture based on the Josephson circuits. We start the design with elementary logic gates and end up with a memory and PE array. In section 4, we discuss issues of quantum computing with the Josephson qubits and describe a feasibility study on how an architecture of a heterogeneous system based on the Josephson circuits could be set up to implement Shor's quantum factorization algorithm. Although the necessary classical computing could as well be done with conventional CMOS technology, the study on the Josephson circuits might bring insight that cannot easily be achieved with other, e.g. spin based, devices for quantum computers, because of the problems unsolved in combining them with conventional technologies. In section 5, we propose a quantum CNN architecture using the Josephson circuits. In this architecture, the quantum dynamics of the Josephson circuit are formulated as the state dynamics of a CNN cell. Each cell is thus a quantum dynamical system. The quantum states of neighboring cells interact with each other only via classical couplings, which distinguishes a quantum CNN architecture from a quantum computer. In section 6 some implementation issues are discussed. Section 7 concludes the chapter. This chapter is based on [41], [42], [43] and [44].

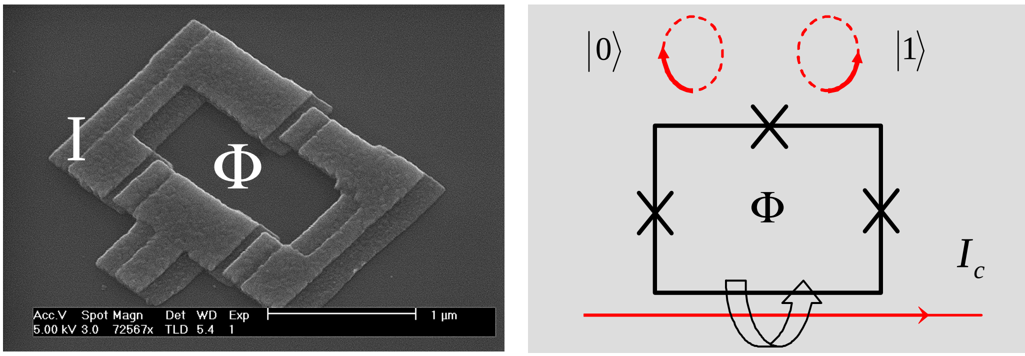


Figure 5.1: The superconducting circuit of Josephson junctions: a scanning electron microscope (SEM) picture (courtesy of J. E. Mooij) and a schematic representation [37].

5.2 The superconducting circuit of Josephson junctions

The Josephson circuit in principle consists of a loop with three Josephson junctions in series that encloses a magnetic flux that is provided by an external magnet (Figure 5.1 [37]). In particular when the enclosed magnetic flux is close to half a superconducting flux quantum Φ_0 ($=h/2e$, where h is Planck's constant), the system behaves as a particle in a double-well potential, where the classical states in each well correspond to persistent currents of opposite sign.

In the circuit, superconductors are coupled with Josephson junctions. The current through a Josephson junction is

$$I_J = I_0 \sin\gamma \tag{5.1}$$

with

$$I_0 = (4\pi e/h)E_J; \tag{5.2}$$

I_0 is the critical current of the Josephson junction, E_J is the Josephson energy and γ is the gauge-invariant phase difference. Two of the junctions in the loop have equal Josephson energy E_J and the third has αE_J. If the applied magnetic flux is $f\Phi_0$ with $f \in [1/2 - f_c, 1/2 + f_c]$, where $f_c \leq 1/4$, this circuit has two stable persistent currents. Energy levels and persistent currents of the circuit as a function of applied flux Φ_{ext} are plotted in Figure 5.2 [38]. The amplitude of the persistent currents I_p is

$$I_p \approx 2\pi\alpha E_J/\Phi_0. \tag{5.3}$$

With $\alpha = 0.8$, I_c is about 0.5μA. The self-generated flux due to persistent currents is about $10^{-3}\Phi_0$ [38]. The states of the persistent currents can be changed by tuning the magnetic flux and magnetic interaction can be made by inductive coupling. The persistent currents can be used as classical binary bits when the applied flux is far away from the degeneracy point at $\Phi_0/2$. The issues of classical computing with the circuits are discussed in section 3.

When the enclosed flux is close to $\Phi_0/2$, the two classical states of the Josephson circuit are coupled via quantum tunneling through the barrier between the wells. The circuit is thus a macroscopic quantum system with two base states $|0\rangle$ and $|1\rangle$ with opposite circulating

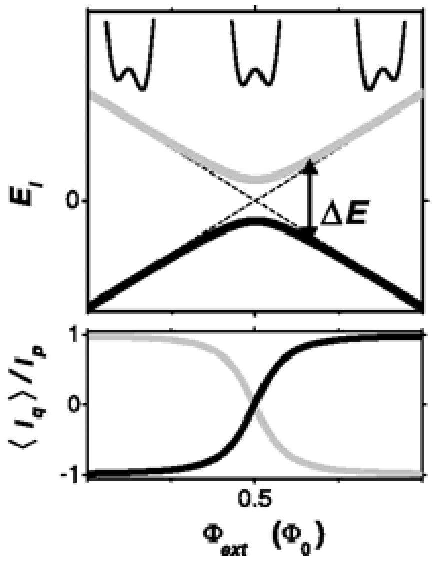


Figure 5.2: Energy levels and persistent currents of the Josephson circuit [39].

persistent currents, corresponding to the two lowest energy levels of the circuit, the ground state and the first excited state. The separation of the energy levels is controlled by varying the flux bias (Figure 5.2).

The quantum state of the circuit is given by:

$$|\Psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,, \tag{5.4}$$

with

$$|\alpha|^2 + |\beta|^2 = 1, \tag{5.5}$$

where α, β are complex numbers, and $|\alpha|$, $|\beta|$ are probability amplitudes. With regard to any measurement, this quantum system behaves as $|0\rangle$ or $|1\rangle$ with a probability of $|\alpha|^2$ or $|\beta|^2$.

The quantum dynamics of the circuit is described by the time-dependent Schrödinger equation:

$$i\hbar\frac{d\,|\Psi\rangle}{dt} = \mathbf{H}\,|\Psi\rangle\,, \tag{5.6}$$

with the Hamiltonian:

$$\mathbf{H} = I_p\Phi_0(f - \frac{1}{2})\boldsymbol{\sigma}_z - \frac{\lambda}{2}\boldsymbol{\sigma}_x, \tag{5.7}$$

where I_p is the classical magnitude of the persistent currents, f the magnetic flux in the loop in units of the flux quantum Φ_0, λ the energy level repulsion, and $\boldsymbol{\sigma}_{z,x}$ the Pauli matrices [38].

The quantum states of the circuit can be operated by resonant microwave modulation of the enclosed magnetic flux by a superconducting control line (as I_c in Figure 5.1). Measurement can be made with superconducting quantum interference devices (SQUIDs) [38]. Two or more Josephson circuits can be coupled through mutual inductance by means of the flux

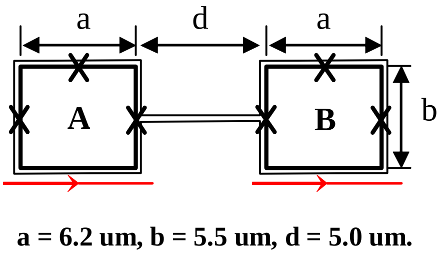


Figure 5.3: An inverter (NOT) circuit.

that the persistent currents generate. A wide variety of potential designs for the couplings are available. For instance, a coupling can be switched on or off via an external transporter attached with a SQUID loop, as proposed in [37]. The Josephson circuit is suitable for integrations in electronic circuits and scaling to large arrays. The issues of quantum computing with these circuits are discussed in section 4.

5.3 Classical computing with Josephson circuits

5.3.1 Circuit topology and simulations

In the classical regime, two or more Josephson circuits can be coupled through interactions of the magnetic field generated by external control currents or the persistent currents. Basically, two schemes are possible. One is to make a coupling directly through magnetic interference. The other is to make a coupling assisted with a superconductive flux transporter. The transporter is placed on top of the circuit loops and insulated by a thin layer. With the parameters suggested in [38], it has been shown that the coupling between circuit loops is stronger with the facility of a transporter, by evaluating the coefficients that describe the effective Hamiltonian and determine the interactions of the Josephson circuits [142]. To make a robust design, however, the layout and parameters of a circuit need to be carefully considered.

In this study, we propose a circuit topology with the use of a flux transporter for the coupling of two Josephson circuits, as shown in Figure 5.3. The inductive influences from control lines on top of the transporter are also considered. In order to independently manipulate a circuit, a control line should be strongly coupled with one loop while weakly coupled with the other. A circuit with more loops is proposed in Figure 5.4. This circuit is based on the same topology as the one in Figure 5.3, and the couplings between neighbor loops are similar to the coupling of the circuit in Figure 5.3. We show in the next section the circuits in Figures 5.3 and 5.4 actually work as an inverter (NOT) and a not-majority voter (NMV).

The simulations of these circuits were carried out with a software package named FastHenry [143], which computes the frequency dependent self and mutual inductances and resistances between conductors of complex shape. The inductances obtained from the circuit in Figure 5.3 are shown in Table 5.1. As can be seen, the mutual inductance between

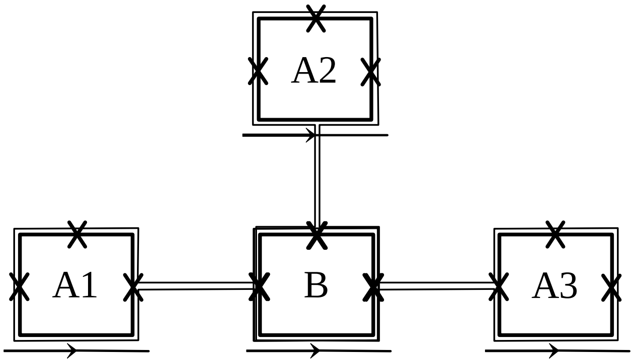


Figure 5.4: A not-majority voter (NMV) circuit.

Table 5.1: The self and mutual inductances of the inverter circuit (pH).

	Transporter	Loop 1	Loop 2	Control line 1	Control line 2
Transporter	L_t=30	M_{t1}=9.7	M_{t2}=9.7	M_{tc1}=3.1	M_{tc2}=3.1
Loop 1	M_{t1}=9.7	L_1=11	M_{21}=-0.069	M_{l1c1}=2.3	M_{l1c2}=0.028
Loop 2	M_{t2}=9.7	M_{12}=-0.070	L_2=11	M_{l2c1}=0.028	M_{l2c2}=2.3
Control line 1	M_{tc1}=3.1	M_{l1c1}=2.3	M_{l2c1}=0.026	L_{c1}=4.3	M_{c1c2}=0.28
Control line 2	M_{tc2}=3.1	M_{l1c2}=0.026	M_{l2c2}=2.3	M_{c1c2}=0.28	L_{c2}=4.3

Table 5.2: The self and mutual inductances of the NMV circuit (pH).

	Tran 1	Tran 2	Tran 3	Loop 1	Loop 2	Loop 3	Loop Out
Tran 1	L_{t1}=30	M_{t1t2}=8.9	M_{t1t3}=11	8.3	-0.099	-0.071	M_{ot1}=8.3
Tran 2	M_{t1t2}=8.9	L_{t2}=30	M_{t2t3}=11	-0.077	6.4	-0.079	M_{ot2}=6.5
Tran 3	M_{t1t3}=11	M_{t2t3}=11	L_{t3}=30	-0.070	-0.099	7.3	M_{ot3}=7.3
Loop 1	8.3	-0.089	-0.074	L_1=11	-0.023	-0.001	-0.065
Loop 2	-0.10	6.5	-0.10	-0.019	L_2=11	-0.019	-0.072
Loop 3	-0.073	-0.089	7.3	-0.001	-0.023	L_3=11	-0.064
Loop Out	M_{ot1}=8.3	M_{ot2}=6.5	M_{ot3}=7.3	-0.062	-0.066	-0.063	L_{out}=11

a circuit loop and its control line is 2.3 pH[2], about two orders larger than the mutual inductance between the control line and the other loop, approximately 0.026 - 0.028 pH. The mutual inductance between the transporter and the loops is 9.7 pH, while the mutual inductance between the two loops is about 0.069 pH. The unnecessary couplings are thus effectively suppressed. The simulation result of the circuit in Figure 5.4 is shown in Table 5.2. For simplicity, the inductances of control lines are omitted. In this circuit, the mutual inductances of transporters are approximately 9 - 11 pH and the mutual inductances between the output loop and the transporters are approximately 6.5 - 8.5 pH. These variations are induced by the physical asymmetries due to the implementation and topology of the circuit. In a large-scale circuit, this will imply precision requirements on circuit design and fabrication process. Given the inductances, the magnetic interactions of the Josephson circuits can numerically be analyzed. This is presented in the next subsection.

5.3.2 Elementary logic gates

An inverter and a not-majority voter

Consider first the circuit in Figure 5.3. Prepare an initial magnetic flux threading the two loops at $1/2\Phi_0$. The magnetic flux in the superconducting transporter is then Φ_0. If a flux change $\Delta\Phi_x$ due to external sources (e.g. the control currents) is introduced into a loop as an input, then a clockwise (or anti-clockwise, depending on the orientation of the flux change) persistent current is generated in the circuit. Since the transporter tends to keep the total flux unchanged (i.e. to keep it at multiples of a flux quantum) [144], there will be a current I_t generated on the transporter and it satisfies:

$$\Delta\Phi_x + I_t L_t + I_{c1} M_{t1} + I_{c2} M_{t2} = 0, \tag{5.8}$$

where L_t is the self inductance of the transporter, I_{c1} and I_{c2} are respectively the persistent currents on the two loops, M_{t1} and M_{t2} are the mutual inductances between the two loops and the transporter. The magnetic flux change in the second loop is then

$$\Delta\Phi_2 = I_{c1} M_{12} + I_t M_{t2} + I_{c2} L_2, \tag{5.9}$$

where M_{12} is the mutual inductance between the two loops and L_2 is the self inductance of loop 2. Because the persistent currents on the loops have similar amplitudes and opposite directions, we have $I_{c2} \approx -I_{c1}$, and, as can be seen in Table 5.1, $M_{t1} = M_{t2}$ and $M_{12} \approx 0$. Equation (5.8) becomes:

$$\Delta\Phi_x + I_t L_t \approx 0. \tag{5.10}$$

The magnetic flux change in the second loop is then:

$$\Delta\Phi_2 \approx I_t M_{t2} + I_{c2} L_2 \approx -(M_{t2}/L_t)\Delta\Phi_x + \Delta\Phi_p, \tag{5.11}$$

where $\Delta\Phi_p = I_{c2} L_2$ is the flux change induced by the persistent current on loop 2.

If $\Delta\Phi_x$ is relatively large compared to $\Delta\Phi_p$ (about $10^{-3}\Phi_0$), say, $\Delta\Phi_x = 0.1\Phi_0$ (corresponding to a control current of approximately 900μA), $\Delta\Phi_p$ is then negligible. This is

---

[2] Here, pH = pico-Henry, a measure of inductance, not acidity.

usually true for operating in the classical realm. Taking the data in Table 5.1, we have then $\Delta\Phi_2 \approx -(1/3)\Delta\Phi_x$. This indicates that a fraction of the flux change in loop 1 due to external sources is transported to loop 2, but with a reversed orientation, the characteristic of an inverter.

Consider next the circuit in Figure 5.4. If a flux change $\Delta\Phi_{xi}$ $(i = 1, 2, 3)$ is introduced in each of the input loops and the magnetic flux generated by the persistent current on each loop is small compared to $\Delta\Phi_{xi}$, we have:

$$\Delta\Phi_{x1} + I_{t1}L_{t1} + I_{t2}M_{t1t2} + I_{t3}M_{t1t3} \approx 0, \tag{5.12}$$

$$\Delta\Phi_{x2} + I_{t2}L_{t2} + I_{t1}M_{t1t2} + I_{t3}M_{t2t3} \approx 0, \tag{5.13}$$

$$\Delta\Phi_{x3} + I_{t3}L_{t3} + I_{t1}M_{t1t3} + I_{t2}M_{t2t3} \approx 0, \tag{5.14}$$

where I_{t1}, I_{t2}, I_{t3} are the currents on transporters, L_{t1}, L_{t2}, L_{t3} are the self inductances of transporters, and M_{t1t2}, M_{t1t3}, M_{t2t3} are the mutual inductances between transporters. The flux change in the output loop is then given by:

$$\Delta\Phi_{out} \approx M_{ot1}I_{t1} + M_{ot2}I_{t2} + M_{ot3}I_{t3}, \tag{5.15}$$

where M_{ot1}, M_{ot2}, M_{ot3} are the mutual inductances between the output loop and transporters. Taking the data in Table 5.2, the flux change in the output loop is approximately $\Delta\Phi_{out} \approx -(1/8 \text{ - } 1/6)(\Delta\Phi_1 + \Delta\Phi_2 + \Delta\Phi_3)$. This indicates that a fraction of the total flux changes of input loops is transported to the output loop, but with an opposite direction.

If the initial magnetic flux in a Josephson circuit is lower or higher than $1/2\Phi_0$, a flux change can be introduced by using microwave pulses to activate or deactivate the circuit's state. A flux change is then induced in the loop by the reverse of a persistent current, i.e. $\Delta\Phi_x = 2I_{c1}M_{t1}$, and a flux change results in the second loop. This change can further be detected by applying microwave pulses, similar to the method used in manipulating the quantum processing circuits. This method may be applicable when classical and quantum computing are simultaneously implemented in the Josephson circuits.

Hence, the Josephson circuit has the states $+$ (clockwise current), $-$ (anticlockwise current), 0 (no current, i.e. suppressed by external control lines), digitally this can be associated with $+1$, -1 and 0, in Boolean Logic with TRUE, FALSE and Don't Care. Two or more circuits can be coupled through the interaction of magnetic field. For the circuit in Figure 5.3, as has been seen, any flux variation in a loop always causes a reverse change of the magnetic flux in the other loop, and so does the variation of a persistent current. Logically, this circuit functions as an inverter (or NOT gate), i.e. $B = NOT(A)$. For the circuit in Figure 5.4, the output loop obtains an "addition" of the flux changes of those it is coupled to, but with an opposite orientation. This circuit thus works on the principle of a not-majority voter (NMV), i.e. $B = NMV(A1, A2, A3)$. As revealed in the analysis, however, a major issue of the Josephson logic is the lack of a signal gain, which might suggest that only a limited number of logical gates is possible in a Josephson circuit module without signal restoration. A potential solution to this is to use an external magnetic field (via the control lines, for instance) to restore a signal, while this issue will not be adressed any further in our study.

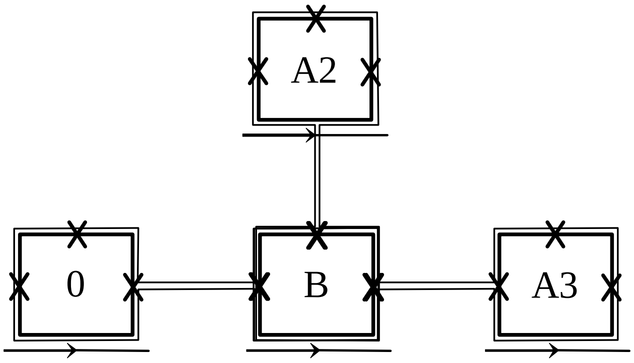


Figure 5.5: A NAND gate.

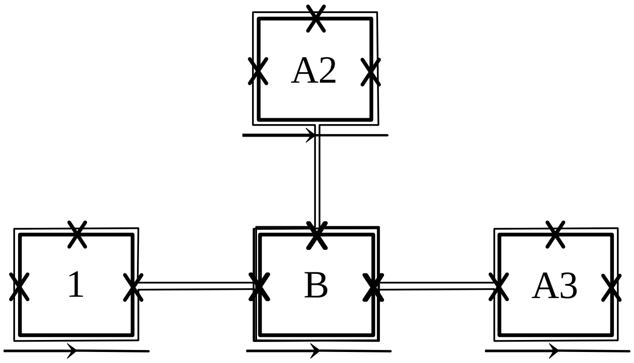


Figure 5.6: A NOR gate.

A universal gate

Now we have an inverter and a NMV. By setting one of its inputs as an instruction bit, a NMV can be configured to a NAND or NOR gate. In Figure 5.5 we have $B = NAND(A2, A3)$ and in Figure 5.6 we have $B = NOR(A2, A3)$, as indicated in the truth table (Table 5.3). A NMV is hence a universal logic circuit, with which any logical function can be constructed. In addition, the topology of the NMV circuit can be configured to a fan-in or fan-out circuit (in a reversed manner though) by setting the instruction bits and control currents. In fact, the NMV circuit in Figure 5.4 can serve as a data processing unit, which can be used as a fundamental element in the construction of a large data processing network.

So far the circuits we discussed are coupled to their nearest neighbors, and long-range communication seems to be a hard task. With the use of a transporter, however, it may be possible to realize fast data propagating. Assume that a big transporter is put on the top of a chain, which consists of a number of Josephson loops (Figure 5.7). If each loop in the chain interacts equally with others, then all the loops switch simultaneously as soon as a flux variation is introduced into the input loop. In this way, a signal is propagated to all the cells in a chain at once. Similarly, the status of an array can be obtained by measurements with SQUIDs (Figure 5.8).

$I/A1$	$A2$	$A3$	B	$Function$
0	0	0	1	$NAND$
0	0	1	1	
0	1	0	1	
0	1	1	0	
1	0	0	1	NOR
1	0	1	0	
1	1	0	0	
1	1	1	0	

Table 5.3: True table of NAND and NOR from NMV.

5.3.3 A processor element (PE) design

A typical design of a processor element (PE) is usually an arithmetic and logic unit (ALU), registers and local memories. The memory ranges from a few bits up to megabytes. PEs are connected to a central controller that broadcasts instructions and has data buffers for inputs and outputs. In this section, we present the designs of a full adder (as a simple ALU), a memory array and a PE array structure, all based on the superconducting circuits of Josephson junctions.

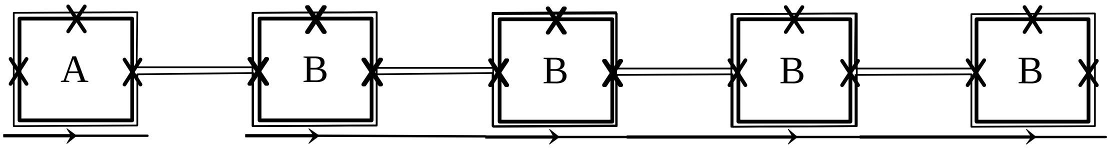


Figure 5.7: Fast data propagating in the Josephson circuits.

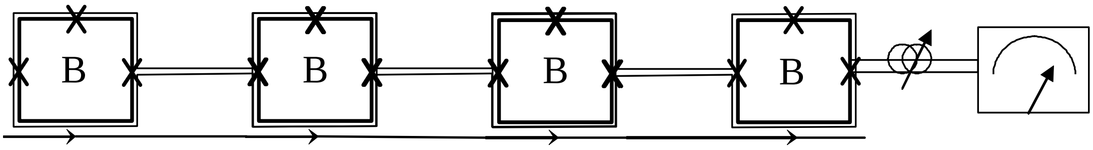


Figure 5.8: Gathering array status with SQUIDs.

An ALU design

An ALU can be as simple as a full adder, a schematic drawing of which is shown in Figure 5.9. This adder consists of three NMV gates and two inverters, clocked by control currents. With

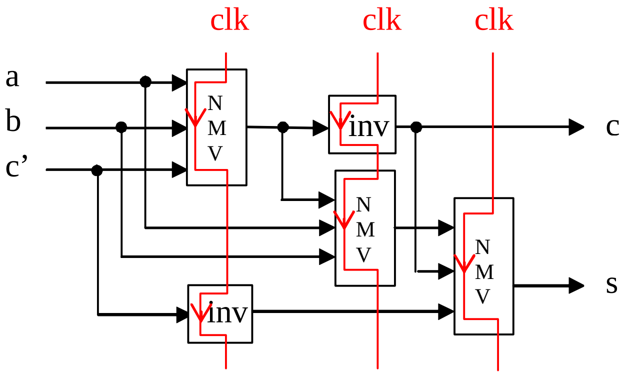


Figure 5.9: A schematic drawing of a full adder.

appropriate configurations, it performs not only an arithmetic addition, but also many logical functions. To implement the adder, we map its functional blocks onto a Josephson circuit array and execute a desired function by configuring instruction bits and injecting input data into the array. A possible implementation is shown in Figure 5.10. Since instructions and input data can swiftly be switched into the array through fast propagating paths (as data buses), a Josephson array can readily be configured to perform desired computations.

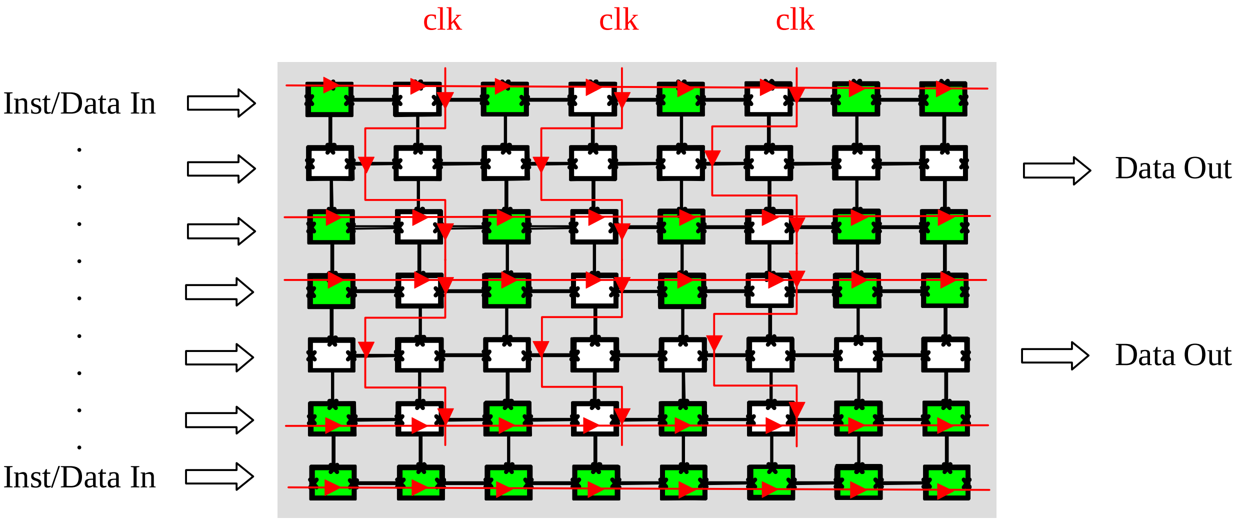


Figure 5.10: An implementation of the full adder in a Josephson circuit array.

A memory array

It is arguable that the most important component of modern computing systems is not the processor, but the memory. Fortunately, the Josephson circuit possesses natural properties to be a good memory cell, where a bit of information is stored as a persistent current. Writing and reading of information are carried out with the control currents. Figure 5.11 shows a 4×4 memory array implemented in the Josephson circuits with column (ca) and row (ra) addressing. A decoder for a row selection is shown as well.

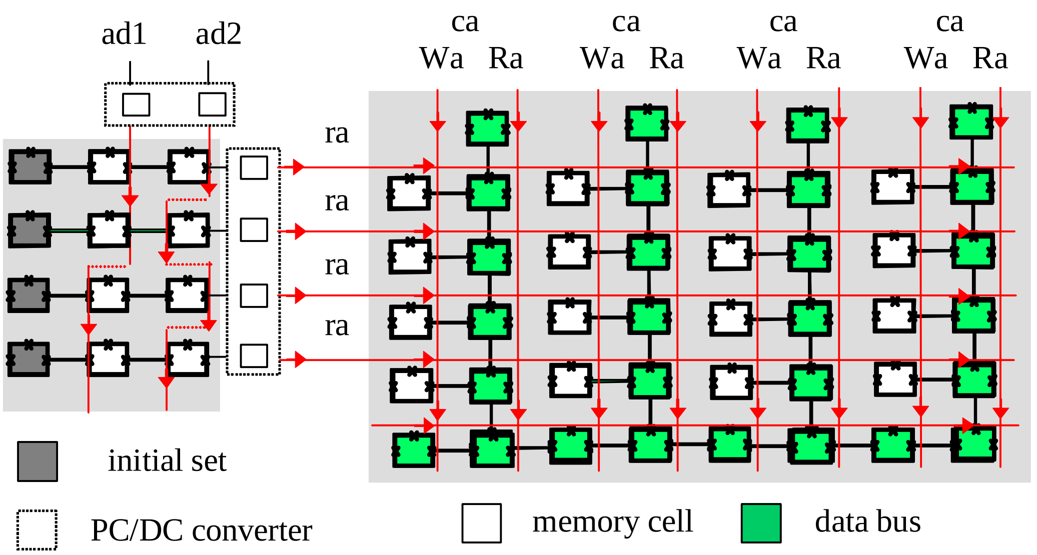


Figure 5.11: A 4×4 memory array with a 2-bit row decoder.

For a WRITE operation, a bit is fed into the array via the data buses and a cell is selected by a row address (ra) and a column WRITE address (Wa). The address lines are designed in such a way that a memory cell is selected only when both of its row address (ra) and column WRITE address (Wa) lines are active. A memory cell cannot independently be refreshed by either a row or a column address line. As soon as a cell is activated, the bit in the data buses is stored in. Other cells are not affected, because none or only one of their address lines is active.

For a READ operation, the cell adjacent to the being-accessed memory cell is selected by both of its row address (ra) and column READ address (Ra). Since other cells in the same column are suppressed during reading, this cell will get the stored information from its adjacent memory cell, without interacting with its neighbors in the same column. As soon as a bit is obtained, it is shifted out via the data bus.

A two-bit address decoder is as well shown in Figure 5.11. This decoder consists of rows of Josephson cells. In each row the left-most cell is fixed in an active state, and each cell is coupled to its nearest neighbors in the row. Address signals (ad1 and ad2) arrive at the PC/DC converters, by which the persistent currents (PCs) are translated into DC signals carried on control current lines. The control lines are inductively coupled to the cells in the decoder, and are designed in such a way that a cell is activated or suppressed by positioning its control line on one or the other side of the cell. The signal in the left-most cell is then passed through an activated cell while blocked by a suppressed one. By systematically arranging the control lines, as shown in Figure 5.11, it is possible that, for any address signal, there is only one row in the decoder, capable of passing a PC signal from one end to the other. The decoded information, i.e. the passed PC signal, is then transmitted into a PC/DC converter, by which it is translated to a DC signal that is transmitted into the memory array as a raw address (ra).

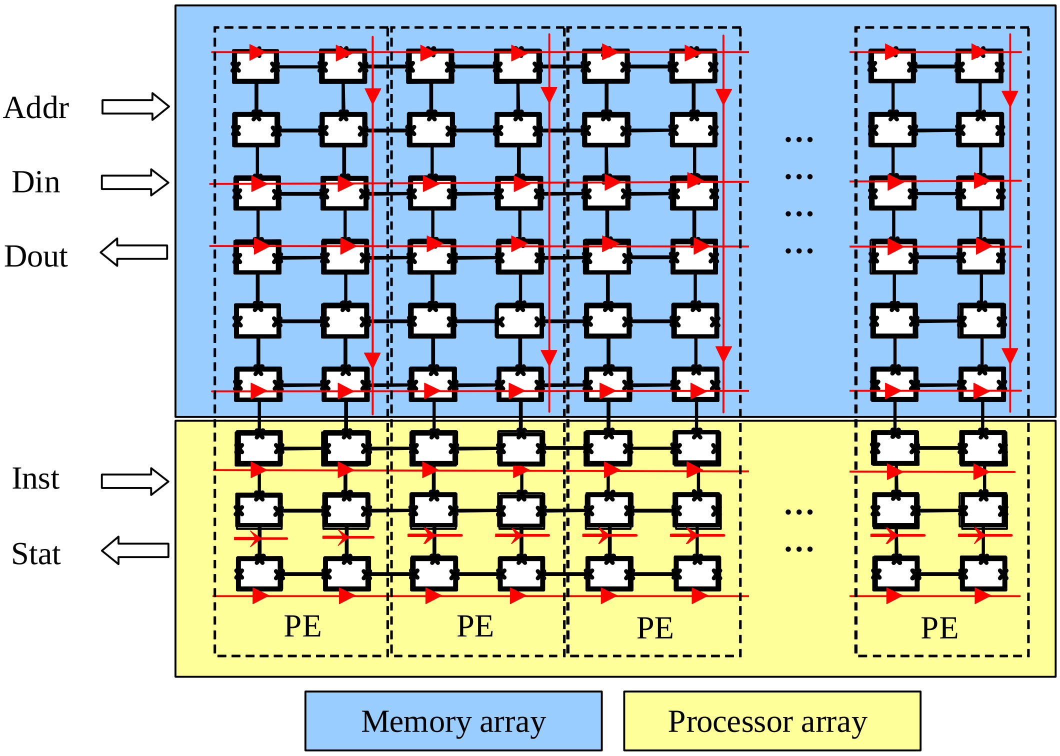


Figure 5.12: An implementation of a PE array on the Josephson circuits.

A PE design

If the couplings among the Josephson circuits are well established, then each of them can be manipulated and addressed by the control currents, and, hence, data processing, storing and communicating can be realized in a Josephson array with the switchings of control currents. We present an implementation of such a Josephson processing network, an array of PEs attached with local memories, as schematically shown in Figure 5.12. Instructions and data are fed into the array while intermediate status and outputs are read out, possibly only via the edges of the array. The advantage of the structure is its extremely simple and regular topology. The manipulation of the array is accomplished by configuring the circuit states as inputs and instructions, and by applying the control currents as addressing and clock signals. The extreme simplicity in structure is thus accompanied by an enormously increased complexity in configuration or programming after fabrication.

Quantum-effect devices perhaps have more variability in their characteristics than conventional ones. We used a simulation tool, FastHenry, to facilitate the designs of elementary

Figure 5.13: A multi-layer Josephson processing array.

	Gate length (μm)	Memory (MBits/cm^2)	On-chip clock (GHz)
CMOS in 2002	0.08	700	2.5
Josephson in 20??	4	25	100

Table 5.4: A comparison of the Josephson superconducting and CMOS technology.

logic gates. In a large-scale network, however, more factors are involved, and an increased complexity in magnetic interference will result. Further fault-tolerant measures will be needed in order to cope with the inaccuracy induced by fabrication and the imprecision of magnetic field or microwave pulses applied during operation. In this regard, we envision a multi-layer Josephson array (Figure 5.13), in which redundancy schemes, e.g. consensus collecting or error correcting codes, can be incorporated.

Nevertheless, difficulties remain in the manipulation of a Josephson array, due to the electromagnetic interference among devices. It is unknown if a controller that provides effective control signals will be available. Furthermore, we compare the properties of the Josephson superconducting technology with those of current CMOS technology, as in Table 5.4. As can be seen, the proposed Josephson technology does not outperform CMOS in terms of device densities of logic and memory. This makes the hope to build a computer with the Josephson circuits rather gloomy. However, the Josephson memory, though bulky, appears to be a promising candidate for an ultra-fast memory structure, because of the properties of high switching speed, low power consumption and relatively simple and regular structure. It may be integrated with circuit components based on other technologies in a heterogeneous architecture or a System-on-Chip (SoC) based architecture.

5.4 Quantum computing with Josephson circuits

5.4.1 Introduction to quantum computing

Classical computation based on Boolean logic acts on classical bits, while a quantum computer composed of quantum logic gates performs operations on quantum bits or qubits. Each qubit represents an elementary unit of information. Corresponding to the classical bits of "0" and "1", a qubit has a basis $\{|0\rangle, |1\rangle\}$ for computation. However, unlike a classical bit, which is one of the two distinguishable states "0" or "1", a qubit exists as a superposition of basis states, represented mathematically as a complex linear combination of the two states $|0\rangle$ and $|1\rangle$, i.e.

$$|\Psi\rangle = a|0\rangle + b|1\rangle . \tag{5.16}$$

With regard to any measurement, the superposition above behaves like $|0\rangle$ with a probability $|a|^2$ and like $|1\rangle$ with a probability $|b|^2$, and we have

$$|a|^2 + |b|^2 = 1. \tag{5.17}$$

More generally, while a string of L classical bits exists in any of the Boolean states $x = 000\ldots0$ through $111\ldots1$, a string of L qubits exists in any state of the form

$$|\Psi\rangle = \sum_{x=00\ldots0}^{11\ldots1} c_x \, |x\rangle \, , \tag{5.18}$$

where c_x are complex numbers such that

$$\textstyle\sum_x |c_x|^2 = 1. \tag{5.19}$$

Consider a register composed of three physical bits, for instance. A classical 3-bit register can store one of eight different numbers, i.e. one of the eight possible binary configurations $000, 001, 010, \ldots, 111$ that represent the numbers 0 to 7. A quantum register composed of three qubits can store up to eight numbers at the same time, in the form of quantum superpositions as exhibited in equation (5.18). Mathematical operations can be executed at the same time on all of the numbers held in the register and the initial superpositions of numbers evolve into different superpositions during a computation. In a quantum computer, therefore, the same mathematical operation can be performed on, say, 2^L input numbers in a single computational step, and the result is a superposition of all of the corresponding outputs. It is this massive parallelism inherent in a quantum computation that suggests that a quantum computer may outperform any classical computer for solving some previously intractable problems.

A quantum computation can be defined as a unitary evolution of a quantum network that takes its initial state into some final states [145]. A quantum network is a quantum computing device consisting of quantum logic gates, and each quantum logic gate executes a unitary operation on one or more qubits. A quantum computer implements a unitary matrix operation on the quantum state of the quantum computer's register. Quantum computations are then always accomplished by building up quantum logic circuits out of quantum logic gates.

In the Josephson circuit, the persistent currents create a magnetic flux. The flux states obey all five functional requirements for a quantum bit: (1) The superconducting circuit is at a sufficiently low temperature that the flux qubits can easily be prepared in their ground states. (2) The flux states can be manipulated precisely with magnetic fields. (3) Two flux qubits can be readily coupled inductively, and the inductive coupling can be turned on and off. (4) The flux of the states can be detected and measured using a SQUID. (5) The flux states can be made insensitive to background charges and effectively decoupled from their environment [38].

All the ingredients for quantum computation are now available. The superconducting persistent current qubits can be initiated, manipulated, coupled to each other, read out and insulated from the environment. A quantum computer is thus in principle possible. As in the case of classical computers, certain sets of quantum logic gates are universal in the sense that any quantum computation can be performed by wiring members of them together. In the following we start with such a set of universal quantum gates, a single qubit rotation and a controlled-NOT gate, or CNOT, then present quantum sum and carry, quantum Fourier transform, and finally propose a draft structure of a quantum computer for an implementation of Shor's factoring algorithm [102].

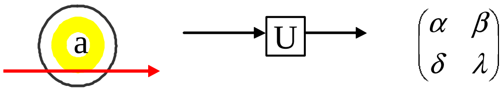

a
U
$\begin{pmatrix} \alpha & \beta \\ \delta & \lambda \end{pmatrix}$

Figure 5.14: Single qubit rotation.

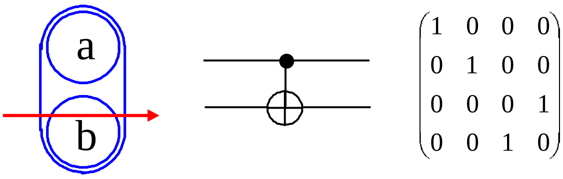

a
b
$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$

Figure 5.15: A controlled-NOT (CNOT) gate.

5.4.2 Elementary quantum gates

An arbitrary single qubit rotation can be written as $e^{-i\sigma t} = \cos t\sigma - i \sin t\sigma$ for some Pauli matrix $\sigma = a\sigma_x + b\sigma_y + c\sigma_z$, where $a^2 + b^2 + c^2 = 1$. The persistent current qubit can be rotated precisely by apply a magnetic pulse for a certain duration (Figure 5.14, the unitary matrix describing the unitary transformation of states is also shown). For example, a pulse would be at 4 GHz and last for about 125 nsec.

A controlled-NOT is a two-qubit quantum logic gate that flips the second qubit if the first qubit is in state 1. That is, it takes $|00\rangle \; -> |00\rangle$, $|01\rangle \; -> |01\rangle$, $|10\rangle \; -> |11\rangle$, $|11\rangle \; -> |10\rangle$ (Figure 5.15). By exploring the magnetic interference of two qubits, a so-called controlled-rotation gate can be constructed, as shown in Figure 5.16. The target bit in a controlled-rotation gate can be precisely rotated according to the state of the control bit. This is characterized by a unitary matrix A_α. Given a π pulse, in particular, a controlled-NOT operation is obtained from a controlled-rotation gate. It has been shown that a controlled-rotation gate is universal for quantum computing and the controlled-NOT gates can be combined with single qubit rotation gates to realize any quantum logic function.

Up to this moment, a single qubit has been measured [39] and the coherent quantum dynamics have been demonstrated [40] in the quantum transport laboratory of applied physics at the TU-Delft. A double-qubit circuit has been fabricated and experiments are being performed [146]. With a controlled operation, it becomes possible to implement a simple Deutsch-Jozsa (D-J) algorithm, which has been demonstrated in an NMR quantum computer [147].

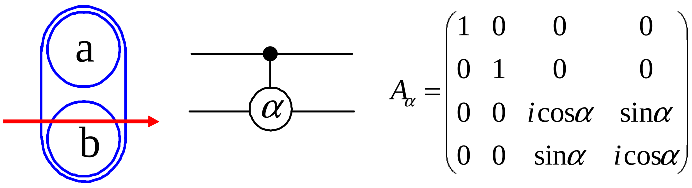

a
b
α
$A_\alpha = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & i\cos\alpha & \sin\alpha \\ 0 & 0 & \sin\alpha & i\cos\alpha \end{pmatrix}$

Figure 5.16: A controlled-rotation gate.

a	b	c	a	b	c
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	**1**	**0**	**1**	**1**	**1**
1	**1**	**1**	**1**	**1**	**0**

Table 5.5: The truth table of Toffoli gate.

The Deutsch-Jozsa (D-J) algorithm [148] determines whether an unknown function is constant or balanced. For a function $f(x)$ that transforms N bits of information to one bit, it is a constant function if output $f(x) = 0$ for all x, or $f(x) = 1$ for all x; it is a balanced function if output $f(x) = 0$ for exactly half of its inputs and $f(x) = 1$ for the remaining inputs. On a classical computer $2^{N-1} + 1$ function calls are needed to determine with certainty whether a function is constant or balanced, while the D-J quantum algorithm determines whether an unknown function is constant or balanced using only one function call. This algorithm illustrates that a quantum computer can perform a computation in less steps than any classical computer. Since the simplest case of the algorithm can basically be carried out on a controlled-NOT gate, it can be the first step for the Josephson qubits to be in practice for a quantum computation.

5.4.3 Quantum sum and carry

Quantum computation performs unitary transformations and any unitary operation is reversible. This is the reason that quantum arithmetic cannot be directly deduced from its classical Boolean counterpart (it is obvious that most of classical logic gates are not reversible). Quantum arithmetic must be built from reversible logical components.

Before we turn to quantum sum and carry, we first present a 3-qubit quantum gate, the Toffoli gate or the controlled-controlled-NOT gate. The truth table of the Toffoli gate is shown in Table 5.5. The target bit c undergoes a NOT operation when the two control bits a and b are both in state 1, and is unaffected when the two control bits are in other states. The Toffoli gate can be constructed from the controlled-rotation and controlled-NOT gates [149], as depicted in Figure 5.17, where time flows from left to right.

The addition of two quantum registers $|a\rangle$ and $|b\rangle$ can be written as $|a, b\rangle\ -> |a, a + b\rangle$. As the input (a, b) can be reconstructed from the output $(a, a + b)$ and there is no loss of information in the process, this computation can be performed reversibly. The quantum sum of three qubits can be implemented with two CNOT gates while the quantum carry is

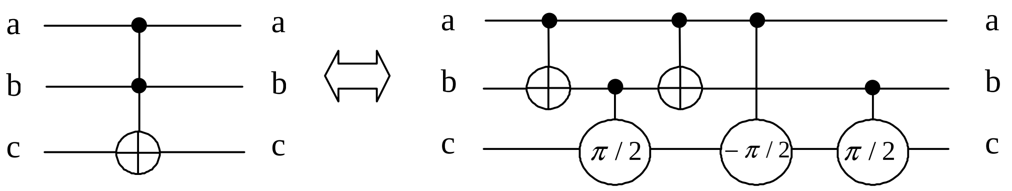


Figure 5.17: The Toffoli gate and its implementation.

obtained by using two Toffoli gates and one CNOT gate, as shown in Figure 5.18, where times goes from left to right. During a computation, a sequence of microwave pulses with different frequencies and durations is applied on each qubit. Those in resonance will switch their states. An implementation of such a process for the quantum carry is schematically shown in Figure 5.19.

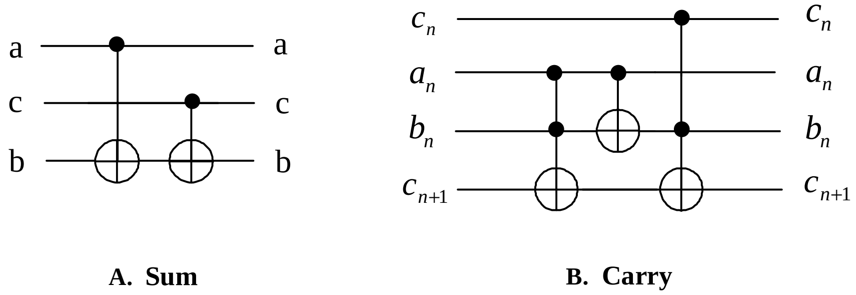
Quantum sum and carry are basic elements for a quantum computation. Many functional quantum networks can be based on these two primitives. A modular exponentiation $U_{x,n}\left|a,0\right\rangle\ ->\left|a,x^a(\bmod n)\right\rangle$, probably the most significant part in Shor's quantum factoring algorithm, can be constructed from the networks of quantum sum and carry [150].

5.4.4 Quantum Fourier transform

The discrete Fourier transform modulo q is a unitary transformation in q dimensions. Consider a number a with $0 \leq a < q$, we perform the transformation that takes the state $|a\rangle$ to the state

$$\frac{1}{q^{1/2}}\sum_{c=0}^{q-1}\exp(2\pi iac/q)\left|c\right\rangle. \tag{5.20}$$

If we take $q = 2^l$, an integer can be represented in binary logic as $|a_{l-1}a_{l-2}...a_0\rangle$. For the quantum Fourier transform, we only need to use two types of quantum gates [141]. One of these gates is a single qubit rotation X_j, which operates on the jth bit of a quantum



A. Sum **B. Carry**

Figure 5.18: Quantum sum and carry.

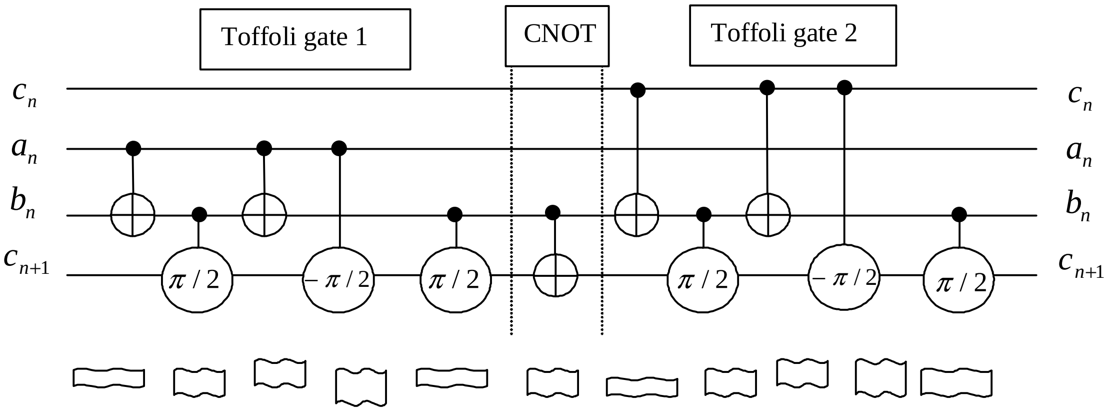


Figure 5.19: Quantum carry implemented by microwave pulses.

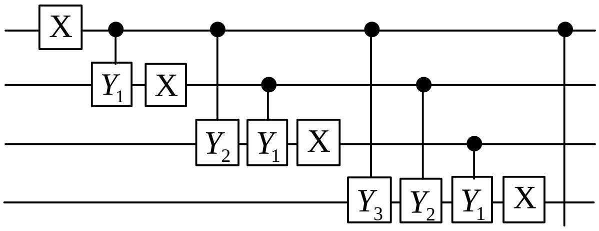


Figure 5.20: An implementation of quantum Fourier Transform.

computer:

$$X_j = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \tag{5.21}$$

Another is a controlled-rotation gate $Y_{j,k}$, which operates on the bits in positions j and k with $j < k$:

$$Y_{j,k} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\theta_{k-j}} \end{bmatrix}, \tag{5.22}$$

where $\theta_{k-j} = \pi/2^{k-j}$. To perform a quantum Fourier transform, we apply the gates X_j in reverse order from X_{l-1} to X_0, and between X_{j+1} and X_j we apply all the gates $Y_{j,k}$ where $k > j$. A part of the transform is implemented as shown in Figure 5.20. By applying this sequence of transformations, we obtain a quantum state

$$\frac{1}{q^{1/2}} \sum_{c=0}^{q-1} \exp(2\pi i a c/q) \left| b \right\rangle, \tag{5.23}$$

where b is the binary number obtained by reading the bits of c from right to left.

A	x^A	$f(A)$
0	1	1
1	4	4
2	16	16
3	64	1
4	256	4
5	1024	16
6	4096	1

A	x^A	$f(A)$
0	1	1
1	8	8
2	64	1
3	512	8
4	4096	1
5	32768	8

Table 5.6: Factoring 21: (a) x=4 and (b) x=8.

Since both the X and Y gates can be implemented by single qubit rotation or the controlled-rotation gates, we estimate that it would eventually be possible to realize the quantum Fourier transform with the Josephson qubits.

5.4.5 A draft structure of a quantum computer

Although there is still a long way to go before a useful quantum computer comes into practice, people believe that it can solve problems that are intractable for present classical computers. One of these problems is the prime factorization of large numbers. For factoring an integer n, the best factoring algorithm available for classical computing takes a run time $\exp(c(\log n)^{1/3}(\log\log n)^{2/3})$ for some constant c. It is an exponential-time algorithm, which is inefficiently computable with the growing of integer n. The quantum algorithm for factoring by Shor takes $O((\log n)^2(\log\log n)(\log\log\log n))$ steps on a quantum computer, along with some polynomial (in $\log n$) processing time on a classical computer [102], [145].

To factor an odd number n, Shor's algorithm first finds the least integer r such that $x^r = 1(\mathrm{mod}\, n)$, i.e. the period of $f(A) = x^A$ for A from 0 to $n-1$, where x is random, with $x < n$ and $gcd(x, n) = 1$. Here, $gcd(a, b)$ is the greatest common divisor of a and b and it can be effectively computed with Euclid's algorithm on a classical computer. Then it finds factors of n by calculating $gcd(x^{r/2} - 1, n)$ and $gcd(x^{r/2} + 1, n)$ if r is even and $x^{r/2} \neq \pm 1 \bmod n$, otherwise, it repeats the algorithm.

For example, factoring $n = 21$. First we choose $x = 4$, and calculate $f(A) = 4^A \bmod 21$ for some A, then we obtain the period $r = 3$, as the data shown in Table 5.6 (a). Here r is not even, so we repeat the algorithm with $x = 8$. Calculating $f(A) = 8^A \bmod 21$, we obtain $r = 2$, as in Table 5.6 (b). Since r is even and $x^{r/2} = 8$, we calculate $gcd(x^{r/2} + 1, n)$ and $gcd(x^{r/2} - 1, n)$, and obtain $gcd(9, 21) = 3$ and $gcd(7, 21) = 7$, the two factors of 21.

To perform Shor's factoring algorithm, we start with two quantum registers, one of which is initiated to be in the superpositions $|a\rangle$ and the other is in the state $|0\rangle$. We compute $x^a(\mathrm{mod}\, n)$ in the second register and keep the first register in the $|a\rangle$ state, i.e. performing the modular exponentiation

$$U_{x,n}\,|a, 0\rangle \quad -> |a, x^a(\mathrm{mod}\, n)\rangle$$

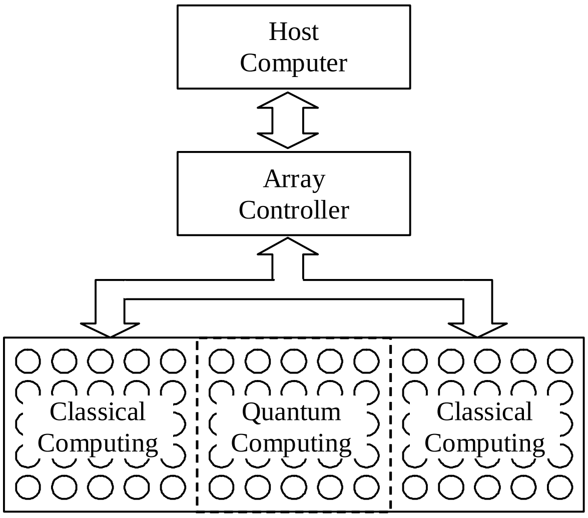


Figure 5.21: A draft structure of an array-based quantum computer.

which can basically be built on a reversible network of quantum sum and carry [150]. Next, we perform the quantum Fourier transform on the first register to get period r. Finally, we calculate the factors of n with Euclid's algorithm by classical computing.

We see that in the realization of Shor's factoring algorithm classical computing is not dispensable; in addition, the experimental realization of a quantum network can be significantly simplified by using classical substitutions. An ultimate architecture for a quantum computer seems to be an integration of both quantum and classical components. As has been seen, the superconducting circuits of Josephson junctions can be designed to independently perform quantum and classical computing. We may eventually be able to build a heterogeneous Josephson array computer, with the quantum computing performed in the heart of the array and accompanied by classical computing components. A possible structure is shown in Figure 5.21.

Shor's factoring algorithm is probabilistic. Preliminary computations are needed for the quantum processing and the (intermediate) outputs of the quantum computing need to be processed classically in order to obtain appropriate results. This process has to be repeated if necessary. For the architecture in Figure 5.21, the classical processing network serves as such a pre- and post-processor for the quantum computing network in the center of the system. The interface between classical and quantum processing will be crucial for a practical implementation. Because classical and quantum computing based on the same device can now be studied simultaneously, which is impossible in certain circumstances, e.g. in the study of ensemble spin-based quantum computers, the array-based architecture of Josephson circuits is a good vehicle for studying the quantum computer paradigm, albeit that the possibility to realize such a quantum computer greatly depends on the progress in experiments on fundamental quantum devices and circuits.

5.5 Quantum cellular nonlinear networks (CNNs) using Josephson circuits

5.5.1 Cellular nonlinear networks (CNNs)

A CNN is an N dimensional array of analog circuit cells with identical components and structures. Each cell is a dynamical system. It interacts directly with its neighbors within a radius and, indirectly, with other cells through a global interaction of continuous-time dynamics. The dynamics of a CNN is determined by a set of state dynamics of each cell in the array [88]. The state dynamics of a cell, indexed by k, is described by the CNN state equation:

$$\frac{dx_k}{dt} = f(x_k) + \sum_{i \in r} g_i(y_i) + \sum_{j \in r} h_j(u_j), \tag{5.24}$$

where x_k is the state variable, and y_i, u_j are inputs and outputs of neighboring cells within the area of effective interactions. The cell interactions, as can be seen, are implied in the linear or nonlinear functions (g_i and h_j in equation (5.24)) of the variables associated with neighbors (y_i and u_j in equation (5.24)). The state variable of a cell is thus completely determined by its inputs and the synaptic effect of neighboring interactions. The output of a cell is given as a function of the state variable:

$$u_k = q(x_k). \tag{5.25}$$

The CNN state equation (5.24) can be readily adapted to applications of image processing and pattern recognition [88], and the CNN architecture, when attached with local memories, can be used to construct a CNN universal machine, which is as universal as a Turing machine [94].

5.5.2 Formulating Josephson quantum dynamics as CNN state dynamics

For a Josephson circuit based CNN (schematically shown in Figure 5.22), the cell dynamics are obtained as the quantum dynamics of the Josephson circuit. The state variable of a cell, indexed by k, is then:

$$|\Psi_k\rangle = \alpha_k |0\rangle + \beta_k |1\rangle \tag{5.26}$$

with

$$|\alpha_k|^2 + |\beta_k|^2 = 1. \tag{5.27}$$

The variable $|\Psi_k\rangle$ can be effectively represented by the unit three-dimensional sphere, often called the Bloch sphere. For the Bloch sphere representation, we have

$$\alpha_k = \cos \frac{\theta_k}{2} \tag{5.28}$$

and

$$\beta_k = e^{i\varphi_k} \sin \frac{\theta_k}{2} \tag{5.29}$$

where θ_k and φ_k are real numbers [105]. Let $A_k = \cos\frac{\theta_k}{2}$ and $B_k = \sin\frac{\theta_k}{2}$, then we have

$$\alpha_k = A_k, \tag{5.30}$$

and

$$\beta_k = B_k e^{i\varphi_k}, \tag{5.31}$$

where A_k, B_k are amplitude variables and φ_k is a phase variable.

A variety of methods is available for the coupling of Josephson circuits (e.g. to use flux transporters [37]). To a first analysis, however, we model the coupling as the interaction of flux generated by the superconducting currents through mutual inductance. Due to the neighboring couplings, the magnetic frustration of a cell k (as the one in the center of Figure 5.22) is changed over the initial frustration f_0 to

$$f_k = f_0 + \sum_{i \in r} M_{ik} I_i / \Phi_0, \tag{5.32}$$

where I_i is the persistent current of a neighbor circuit and M_{ik} is the mutual inductance.

The inductive couplings produce a magnetic flux change, which causes an energy shift in the circuit. If we assume that the coupling between cells is only via classical degrees of freedom, i.e., that there are no quantum entanglements among cells, a Josephson network is then an array of individual quantum systems. The magnetic frustration change of a circuit under interaction is dependent on the states of the circulating currents of neighbors. This change, incorporated in the Hamiltonian, can be described by including a coupling factor $F(A_i)$, a function of A_i, in equation (5.32). The Hamiltonian of a cell k becomes:

$$\mathbf{H}_k = I_p((f_0 - \frac{1}{2})\Phi_0 + \sum_{i \in r} F(A_i) M_{ik} I_i)\boldsymbol{\sigma}_z - \frac{\lambda}{2}\boldsymbol{\sigma}_x. \tag{5.33}$$

The dynamics of the Josephson network is simply described by a set of Schrödinger equations for each cell,

$$i\hbar \frac{d\left|\Psi_k\right\rangle}{dt} = \mathbf{H}_k \left|\Psi_k\right\rangle. \tag{5.34}$$

Taking equations (5.26), (5.27), (5.30), (5.31) and (5.33), the Schrödinger equation (5.34) can be written as a couple of equations for the amplitude variable A_k (or B_k, which is essentially equivalent to A_k) and the phase variable φ_k respectively:

$$\hbar \frac{dA_k}{dt} = -\frac{\lambda}{2}\sqrt{1 - A_k^2}\sin\varphi_k, \tag{5.35}$$

$$\hbar \frac{d\varphi_k}{dt} = I_p((f_0 - \frac{1}{2})\Phi_0 + \sum_{i \in r} F(A_i) M_{ik} I_i) + \frac{\lambda}{2}\frac{A_k}{\sqrt{1 - A_k^2}}\cos\varphi_k. \tag{5.36}$$

As can be seen, equations (5.35) and (5.36) are analogous to the CNN state equation (5.24). The Josephson CNN cell dynamics are characterized by two variables A_k and φ_k. When a measurement is made, the phase information carried on φ_k is discarded, and the amplitude information carried on A_k is obtained as the probability amplitude. The output

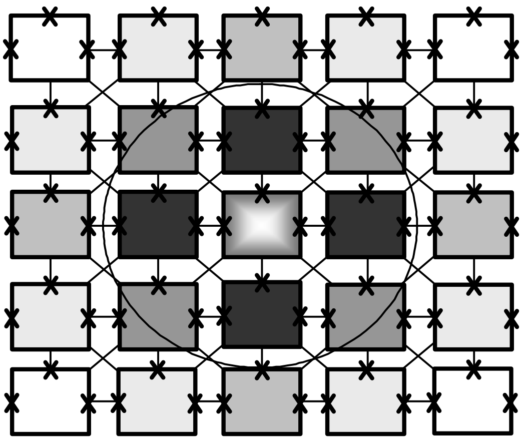


Figure 5.22: A quantum CNN using Josephson circuits.

is therefore the probability that a cell behaves as one specific state of the persistent currents. The output equation is given by:

$$u_k = A_k^2. \tag{5.37}$$

Inputs are prepared in the form of the cells' initial states. Each cell is influenced by its neighbors through mutually inductive couplings. The mutual inductance between cells can simply be a function of their physical distance; the synaptic effect of couplings is therefore a weighted sum of neighboring couplings, as shown in Figure 5.22, where the grey levels of cells imply different coupling weights.

5.5.3 Simulations

To illustrate how a Josephson circuit based CNN works, we present a study on a simple network.

Consider the network in Figure 5.22. Each cell is interacted with neighbors through the flux the persistent currents generate. The varying of the currents induces flux changes in the neighboring circuits, and therefore change the energies of those circuits. For the layout shown in Figure 5.22, the flux change in one cell will result in an opposite effect (i.e. a flux change with an opposite direction) on neighbors. The state of a cell can be manipulated by applying microwave pulses at a frequency equal to the energy splitting of the circuit; the microwave amplitude and pulse length determine the relative probability of the cell being in each base state.

Since every cell in the network has the same state equation as others (without loss of generality, we ignore boundary effects), the global properties of the network can be understood by studying the local properties of a single cell. We further assume that each cell interacts only with its nearest neighbors, i.e., that the interaction with far away cells is ignored. Thus we focus the study on a 3×3 network, as the one encircled in Figure 5.22.

We start the simulation with all cells being initially prepared in the ground state. To study the state dynamics under interaction, we apply microwave pulses to promote the states of a number of cells in the network to the first excited state, inducing the network to evolve due to interactions. Then we observe the changes of the state dynamics of cells. To a first approximation, we focus on the quantum dynamics of the cell $(2,2)$, i.e. the cell in the center of the network, and assume that other cells are static either at the (initial) ground state or at the (promoted) first excited state.

In this simulation, the cells are operating at a magnetic field just below a half quantum flux $(0.5\Phi_0)$, namely, $0.496\Phi_0$ (i.e. $f_0 = 0.496$). Other circuit parameters are adopted from [40], as $a = 0.8$, $E_J = 260\ GHz$ and $\lambda = 3.4\ GHz$. The mutual inductances are obtained by FastHenry to be approximately 0.56 pH for adjacent cells and 0.14 pH for diagonally adjacent cells.

The actual state dynamics of cell $(2,2)$ depends on the synaptic effects of the states of neighbors as well as its initial state. If the initial state of each cell (at $t = 0$) is prepared at the ground state, we have $|\Psi_{ij}(0)\rangle = |0\rangle$, i.e.,

$$A_{ij}(0) = 1, \quad \varphi_{ij}(0) = 0, \ i, j \in [1, 2, 3],$$

If, at the same time, a number of the cells (except the cell $(2,2)$) are excited to the first upper state, i.e., the states are inverted to as $|\Psi_{ij}(0)\rangle = |1\rangle$, the initial conditions for the network become then:

$$\begin{aligned} A_{ij}(0) &= 0, \quad \varphi_{ij}(0) = 0, \quad for\ chosen\ i, j, \\ A_{ij}(0) &= 1, \quad \varphi_{ij}(0) = 0, \quad otherwise. \end{aligned}$$

For example, consider the three distinct sets of initial conditions in Figure 5.23 (a1, a2, a3), shown in the form of $(A_{ij}(0), \varphi_{ij}(0))$. Because of the inverse of the persistent currents, the coupling factor $F(A_{ij}) = -2$ for each $A_{ij}(0) = 0$; and $F(A_{ij}) = 0$ for each $A_{ij}(0) = 1$. In all the cases, the initial states of cell $(2,2)$ are the same, as $A_{22}(0) = 1$, $\varphi_{22}(0) = 0$. The neighbors of cell $(2,2)$ are fixed in the state of $A_{ij}(t) = 0$ or 1, which can be interpreted as the pixel values of binary images [89]. In Figure 5.23, each cell is shaded by encoding its content to a grey-level image pixel: $(1,0) = black$, $(0,0) = white$ and $(0.7,\pi) = grey$.

The amplitude state dynamics of cell $(2,2)$, $A_{22}(t)$, is plotted in Figure 5.24 (a1, a2, a3), for the three different sets of initial conditions. The minimum values of the amplitude variable, as can be seen, are approximately 0.84, 0.87 and 0.89, corresponding to different initial conditions. Although the variations of these values, due to the relatively weak coupling between cells, are rather small, they clearly indicate that the synaptic effects of different sets of neighboring couplings result in the different changes of cell dynamics, given the same initial states. The variation introduced by neighboring interaction can be strengthened by exploring other coupling methods (e.g. to use flux transporters).

Furthermore, the oscillation frequencies of the cell states are distinguishable. This means that a cell may be individually addressed and its state may be manipulated by microwave pulses during data processing. In other words, a CNN network based on Josephson circuits can be potentially programmable.

The dynamics of a cell also depends on its initial state. If cell $(2,2)$, for instance, is initially prepared at a superposition of the states $|0\rangle$ and $|1\rangle$ at $t = 0$, i.e. $|\Psi_{22}(0)\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$, the state variables are then $A_{22}(0) = \frac{1}{\sqrt{2}} \approx 0.7$, $\varphi_{22}(0) = \pi$. If the initial conditions of other cells are assumed to be unchanged, as shown in Figure 5.23 (b1, b2,

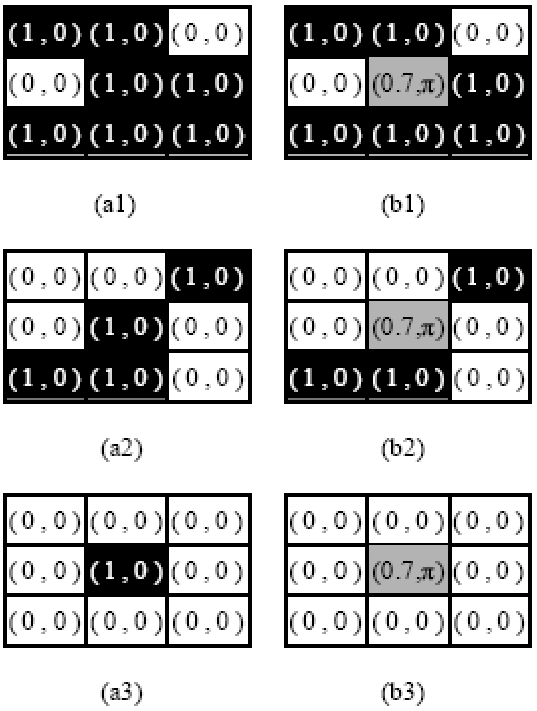


Figure 5.23: The six sets of initial conditions; cells are encoded as grey-level image pixels and shaded in the figure as $(1,0) = \textit{black}$, $(0,0) = \textit{white}$ and $(0.7,\pi) = \textit{grey}$.

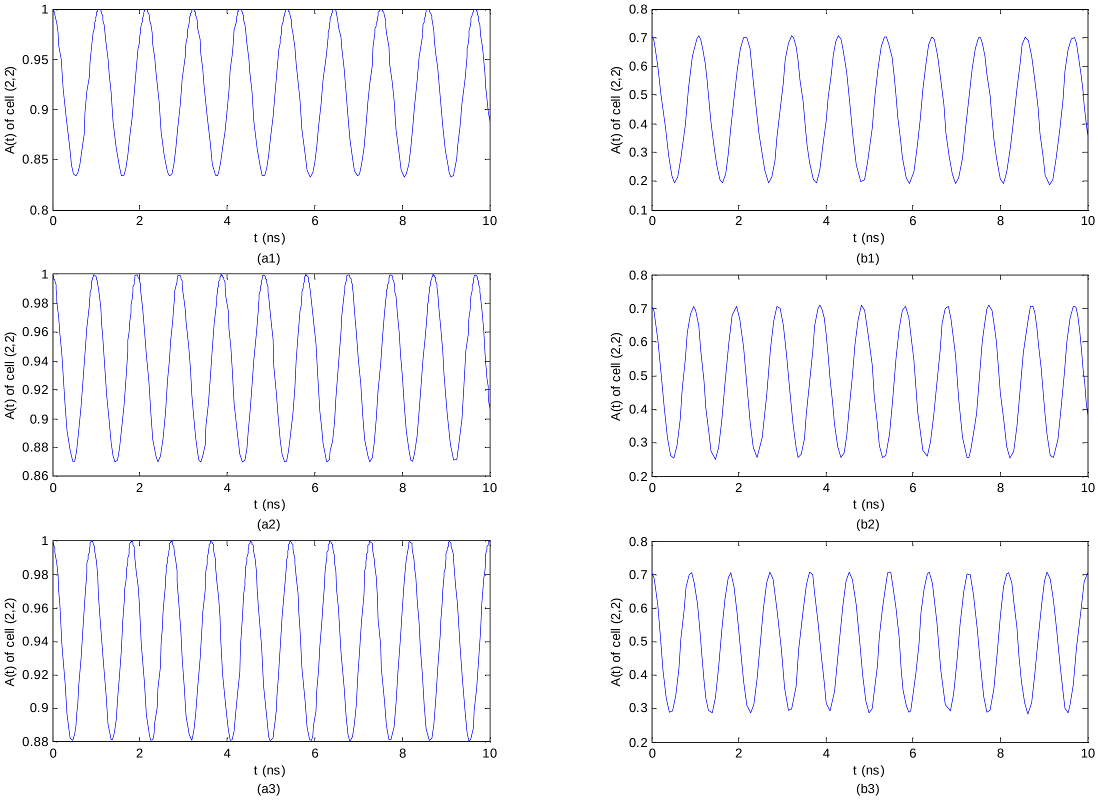
b3), the amplitude state dynamics of cell $(2,2)$, $A_{22}(t)$, is plotted in Figure 5.24 (b1, b2, b3), for the three different sets of initial conditions. The minimum values of the amplitude variable, as can be seen, are approximately 0.20, 0.26 and 0.29, which distinguish these state dynamics, resulting from the different initial conditions in Figure 5.23 (b1, b2, b3).

It is interesting to notice that the oscillation frequencies obtained from Figure 5.24 (a1) and Figure 5.24 (b1), Figure 5.24 (a2) and Figure 5.24 (b2), Figure 5.24 (a3) and Figure 5.24 (b3), are identical. This indicate that, with the same configurations (circuit parameters, interaction conditions, etc.), a CNN network based on Josephson circuits will produce different output states, according to different inputs or initial states.

5.5.4 Summary and issues

Quantum CNNs using superconducting circuits of Josephson junctions have been presented. The local quantum dynamics of the Josephson circuit is used as the state dynamics of CNNs by formulating the Schrödinger equation to a set of equations for an amplitude variable and a phase variable. The states of a cell are controlled by changing the enclosed magnetic flux of the circuit. Cells interact with their neighbors through the mutually inductive couplings. Output states are measured with superconducting magnetometers (SQUIDs). This quantum CNN architecture presents a novel computing paradigm, other than quantum computing and classical computing based on binary logic, for the use of Josephson circuits.

The interactions of variables of cells are assumed to be only via classical degrees of



(a1) (b1) (a2) (b2) (a3) (b3)

Figure 5.24: The state dynamics of cell (2,2) corresponding to the initial conditions of Figure 5.23.

freedom, i.e., no quantum entanglements among cells. This assumption distinguishes a quantum CNN architecture from a quantum computer. It is yet unknown if this assumption is feasible in an actual implementation.

Since any interaction with the external environment introduces decoherence to quantum coherent states, it would be difficult to keep the local coherent states away from decoherence under the circumstances that each Josephson circuit in an array is coupled with its neighbors and the control and measurement circuitry. In general, difficulties remain in the controlling of the states, in inserting inputs and measuring outputs of a large network, due to the decoherence problem. In the next section, a magnetic field gradient scheme is proposed for operations on integrated circuits of Josephson junctions.

5.6 Implementation issues

As noted in previous sections, requirements for a Josephson circuit based system are that the cells can be prepared in well-defined quantum states, that the states can be precisely manipulated according to certain initial conditions (as inputs), and that the states can be measured and read out (as outputs).

The quantum state of a Josephson circuit is manipulated by radiation in resonance of the energy difference of the two base states. This energy difference is typically a few GHz. Microwave pulses can be injected into the circuits for control operations.

Josephson array oscillators consisting of parallel arrays of Josephson junctions have been studied as on-chip oscillators controlling quantum circuits of Josephson junctions [151]. The oscillators have a single frequency, and the frequency, as well as the amplitude, is tunable by independent variables. Oscillators based on RSFQ (rapid single flux quantum) technology have also been proposed for the driving and controlling of Josephson circuits [152], [153]. Measurement can be carried out by weakly coupling one or an ensemble of the Josephson circuits to a SQUID. These schemes aimed at an ultra-fast, high precision and on-chip control of integrated quantum circuits.

For the fabrication of integrated Josephson circuits, there are basically two strategies. One is to incorporate the superconducting control circuitry on a single chip as the Josephson circuits. The other is to develop a flip-chip, i.e., the working Josephson circuits are fabricated on one chip and the measurement and control circuitry are fabricated on another chip; the two chips can then be bonded to be inductively coupled [154].

In both methodologies, however, the mechanism of handles to address each individual cell (either in a quantum CNN or a quantum computation circuit), parameterize it, or input signals to it, have to be developed. In this section we present a magnetic bias scheme that makes it possible to individually address a cell. This scheme is inspired from the MRI (magnetic resonance imaging) technique [155].

Since the energy difference required for resonance in frequency is determined by the enclosed flux of a Josephson circuit, we would be able to control each individual cell if each of them has a unique magnetic flux. A gradient in the magnetic field will help to accomplish this.

A magnetic field gradient is a variation in the magnetic field with respect to position. A one-dimensional linear magnetic field gradient G_x along the x axis, for example, indicates that the magnetic field at position x is given by a linear equation:

$$B_x = B_0 + x \cdot G_x, \tag{5.38}$$

where B_0 is the magnetic field at the initial reference point. A linear magnetic field gradient G_y along the y axis similarly gives the magnetic field at position y.

Applying simultaneously the linear magnetic field gradient G_x and G_y to a two-dimensional array leads to a magnetic field distribution at the (x, y) plane, described by:

$$B_{xy} = B_0 + x \cdot G_x + y \cdot G_y. \tag{5.39}$$

If a cell in this array is regarded as rectangular, and it has values in the coordinates as (x_1, x_2) and (y_1, y_2), its enclosed flux will be:

$$F = \int_{y_1}^{y_2} \int_{x_1}^{x_2} B_{xy} dx dy. \tag{5.40}$$

Simplifying equation (5.40) with (5.39), and assuming the area of the cell $S = (x_2 - x_1)(y_2 - y_1)$ and the center of the cell (x_c, y_c) at $x_c = \frac{1}{2}(x_1 + x_2)$ and $y_c = \frac{1}{2}(y_1 + y_2)$, we obtain the enclosed magnetic flux as:

$$F = S(B_0 + x_c \cdot G_x + y_c \cdot G_y). \tag{5.41}$$

The area of each cell, S, can be seen as the same, if the variation introduced by the fabrication process is ignored. The magnetic flux in each cell is then distinguished by the variables indicating its position, x_c and y_c. The gradient parameters G_x and G_y can be chosen so that the magnetic flux enclosed in each cell is unique. The magnetic field gradient can be realized through a pair of gradient coils with currents of opposite directions.

The use of a magnetic field gradient provides a unique magnetic bias to each cell, so that, when single frequency microwave pulses are applied to the whole array or a chip, only one cell will respond due to the resonance in frequency. It is also possible to have a number of selected cells respond at one frequency by tuning the parameters G_x and G_y. If, for instance, only the gradient G_x or G_y is used (by setting $G_y = 0$ or $G_x = 0$), the cells in a column or a row will be addressed by a single frequency radiation.

The magnetic field gradient scheme presents a practical approach for the manipulation of an integrated circuit of Josephson junctions. However, the use of magnetic field gradient presents challenges. First, each cell has a different flux bias, which implies each cell operates with different energies. This will increase the complexity in setting circuit parameters and implementing algorithms. Second, to individually manipulate a cell, microwave pulses with different frequencies are injected in a sequential order. Other measures are required to have cells working in a synchronized pattern. Finally, the accuracy and stability of a flux bias that can be achieved by using a magnetic gradient need to be investigated.

5.7 Summary

A classical SIMD computer architecture and an array-based quantum computer structure have been presented as possible applications of superconducting circuits of Josephson junctions. The classical computer may serve as a pre- and post-processor for the quantum computing performed in the heart of the Josephson circuit array, establishing a heterogeneous quantum/classical computer for, e.g. an implementation of Shor's factoring algorithm.

Quantum computing represents an important application based upon the reversibility of computation [156], while reversible computing is also of interest in classical digital systems where little or no energy is dissipated [157]. The Josephson circuit, operating at a low temperature and consuming little power, might be favorable for the implementation of a reversible circuit. A reversible network could thus be obtained from the superconducting circuits and perform both quantum and classical computations. These issues on reversible computing are, however, not specifically discussed in our study, and await further investigations.

A quantum CNN architecture using the Josephson circuits, with the quantum dynamics formulated as the CNN state dynamics, has been proposed, presenting a novel computing paradigm for Josephson circuits. Since classical computing architectures (SIMD arrays), quantum computing architectures and semi-quantum computing architectures (quantum CNNs) can be simultaneously studied on the same device, the Josephson circuit is a good vehicle for investigating the architectural issues of quantum and nanoelectronic computer systems, independently from the question of which device will be the ultimate implementation vehicle.

Summary

The progress in CMOS technology has entered the sub-micron realm, and the technology will approach its limits within about 15 years. Already various novel information processing devices, based on quantum mechanical effects at the nanometer scale, have been widely investigated and some have been successfully demonstrated at the circuit level. This advance in nanoelectronic devices has also motivated efforts in the research of nanoelectronic and quantum computer architectures. Due to the components' poor reliabilities, these architectures will have to be robust against device and interconnect failures. In order to avoid power dissipation problems, the components will have to be applied in the quantum mechanical domain, while due to potential problems in interconnects, the components should be locally interconnected only.

This dissertation is devoted to pursuing solutions to architectural issues that come up when designing a nanoelectronic computer. It explores the possibility of building viable and reliable computer systems from novel nanoelectronic and quantum devices. In particular, parallel processor architectures that are fault-tolerant and locally-coupled have been researched.

Chapter 1 presents an introduction to the issues that play a role in nanoelectronics, in contrast with microelectronics, and discusses implications for nanocomputer architectures.

A brief review of the current status in nanoelectronics and recent progress in nanoarchitecture research is presented in Chapter 2.

Chapter 3 describes research on fault-tolerant architectures. We review von Neumann's NAND multiplexing technique and extended his study from a high degree of redundancy to a fairly low degree of redundancy. We show the stochastic Markovian nature of a multi-stage multiplexing system and work out its characteristics. We develop a system architecture based on the NAND multiplexing structure that copes with the problem of random background charges in single electron tunneling (SET) circuits. Our study shows that, although a rather large amount of redundant components is required, an architecture based on the multiplexing technique could be a fault-tolerant system solution for the integration of unreliable nanoelectronic devices affected by dominant transient errors.

In addition, in Chapter 4, a defect- and fault-tolerant architecture is proposed, that uses the multiplexing technique for its fundamental circuits and a hierarchical reconfigurability in the overall system. It is shown that the required redundancy could be brought back to a moderate level — no larger than 10^2 — by adding reconfigurability to the system concept. This architecture is robust in an efficient way against both manufacturing defects and transient faults, and tolerates a gate error rate of up to 10^{-2}, which, for any current microelectronic system, would be unacceptable.

Derived from von Neumann's multiplexing technique, we propose triplicated interwoven redundancy (TIR), as a generalization of triple modular redundancy (TMR), but then with random interconnections. A prototype processor architecture and its simulation-based reliability model have been set-up and are used to evaluate the fault-tolerance. The processor is, by way of comparison, implemented using both TIR as well as so-called quadded logic. In general, the reliability of a TIR circuit is comparable with that of an equivalent TMR circuit while, for certain interconnect patterns, the TIR structure may present an inferior performance to TMR, due to its interwoven nature in gate interconnections. TIR can be extended to higher orders, which we label N-tuple interwoven redundancy (NIR). The use of 5-tuple interwoven redundancy leads to an economical redundancy factor of less than 10 for the reconfigurable system architecture. It has been shown that the design and implementation of restorative devices (voters) are important for TIR/NIR and quadded structures. Only with a simple voter design is it possible to obtain — with a higher order of NIR — a better system reliability than with TIR. TIR or NIR is in particular suitable for implementation in molecular nanocomputers, which are likely to be fabricated by a manufacturing process of stochastic chemical assembly.

In Chapter 5, superconducting circuits of Josephson junctions have been investigated with as aim to possibly use them in locally-connected processor structures. Both a classical SIMD computer architecture and an array-based quantum computer structure are presented that use the same basic circuit, the Josephson junctions. Our ideal is that the classical computer can serve as a pre-, post- and intermediate processor for the quantum computation that is performed in the heart of the Josephson circuit array. As such, it then establishes a heterogeneous quantum/classical computer for implementations of algorithms such as Shor's factoring algorithm which mixes classical computation steps with quantum computation steps in a single algorithm. Although not specifically worked out and discussed in this study in detail, an architecture in the form of an all-reversible computing network based on superconducting circuits of Josephson junctions, could in principle be used for this.

A quantum CNN (cellular nonlinear networks) architecture using the Josephson circuits has also been proposed, presenting a novel computing paradigm for Josephson circuits. Since classical computing architectures (SIMD arrays), quantum computing architectures and semi-quantum computing architectures (quantum CNNs) can be simultaneously studied on the same device, the Josephson circuit is a good vehicle for investigating the architectural issues of quantum and nanoelectronic computer systems, independently from the question of which device will be the ultimate implementation vehicle.

This last chapter concludes this dissertation, which can be placed in the "early days" of research on architectures of nanoelectronic and quantum computers.

And beyond this thesis: The scientific papers that form the foundation of the chapters in this thesis have meanwhile been followed up by many new studies in fault-tolerant techniques such as using Monte Carlo simulations [75], bifurcation theory [76] and an exact analysis using combinatorial arguments [158] to investigate the error behavior in a multiplexed nanosystem of Markov chains. Moreover, a probabilistic-based methodology has been proposed for designing nanocomputer architectures based on Markov Random Fields (MRF) [159], and CAD tools are being developed to automate the evaluation of various fault-tolerant schemes and their reliability/redundancy trade-offs [74]. The redundancy techniques, originating from von Neumann, are basically error-correcting codes (ECC). The multiplexing construction boils down to the use of a repetition code, in which each symbol of a message is repeated many times to create redundancy [81]. The use of error-correcting codes, as well as the issue of fault-tolerance in nanocomputing in general, awaits further investigation.

Novel computing systems, envisioned now as adaptive systems based on molecular electronics [160], biology-inspired self-learning and -evolving systems [161], nonlinear dynamical systems [44] and quantum computers, may in the long term emerge, possibly leading to new types of algorithms and architectures. The choice of algorithms and architectures must aim towards applications in nanotechnology. An architecture will strongly influence the design of devices and circuits, and vice versa: the opportunities and problems found in nanoelectronic devices and circuits will strongly influence the choice of an architecture. In research on nanocomputer architectures, therefore, an interdisciplinary approach must be followed and the success will eventually rely upon a multidisciplinary effort in the fields of chemistry, physics, electrical engineering, computer science, and, perhaps, many others.

Samenvatting

De vooruitgang in CMOS technologie is het submicron rijk ingegaan en de huidige technologie zal zijn grenzen binnen ongeveer 15 jaar bereiken. Nu al zijn er diverse nieuwe processor schakelingen, die gebaseerd zijn op de quantum mechanische effecten behorend bij de nanometerschaal, breed onderzocht en enkele zijn zelfs succesvol gebleken op circuitniveau. Deze vooruitgang in nano-electronica schakelingen was mede de motivatie voor onderzoek op het gebied van nano- en quantumcomputer architecturen. Vanwege de slechte betrouwbaarheid van de basis componenten, zal een dergelijke architectuur bestand moeten zijn tegen fouten zowel in de basis schakelingen als de interconnecties. Om vermogensdissipatie problemen te vermijden, zullen de componenten in het quantummechanische domein moeten worden toegepast, terwijl door te verwachten problemen bij de interconnecties, de componenten slechts plaatselijk onderling dienen te worden verbonden.

Dit proefschrift is gewijd aan het zoeken naar oplossingen voor architectuur kwesties die boven komen bij het ontwerpen van een nano-electronica computer. Het verkent de mogelijkheden om haalbare en betrouwbare computersystemen te bouwen gebaseerd op nieuwe nano-electronische en quantum-fysische circuits. In het bijzonder zijn fout tolerante, lokaal gekoppelde massief parallelle processor architecturen onderzocht.

Hoofdstuk 1 geeft een inleiding in de kwesties die een rol spelen in nano-electronica, in contrast met de micro-elektronica en de implicaties voor nano-computerarchitecturen worden besproken.

Een kort overzicht van de huidige status in de nano-electronica en de recente vooruitgang in nano-architectuur onderzoek wordt gepresenteerd in Hoofdstuk 2.

Hoofdstuk 3 beschrijft onderzoek naar fouttolerante architecturen. Wij beschouwen von Neumann's NAND multiplexing techniek en breiden het onderzoek uit van zijn hoge mate van redundantie naar een vrij lage graad van redundantie. Wij tonen hier de stochastische Markoviaanse aard van een dergelijk multiplexing systeem aan en werken de kenmerken uit. Wij ontwikkelen daarna een systeemarchitectuur die op de NAND multiplexing structuur is gebaseerd en die aan het probleem van "random background charges" in "Single Electron Tunneling" (SET) circuits het hoofd biedt. Onze studie toont aan dat, hoewel er een tamelijk grote hoeveelheid redundante componenten wordt vereist, een architectuur gebaseerd op de multiplexingtechniek een fouttolerante systeemoplossing zou kunnen zijn bij het integreren van onbetrouwbare nano-electronische schakelingen, die gedomineerd worden door tijdelijke (transient) fouten.

Hier aan toegevoegd wordt, in Hoofdstuk 4, een defect- en fouttolerante architectuur voorgesteld, die de multiplexing techniek gebruikt in zijn basiscircuits en hiërarchische herconfigureerbaarheid voor het hele systeem. Aangetoond wordt, dat de vereiste redundantie van de multiplexing in de basiscircuits dan teruggebracht kan worden naar een redelijk niveau, niet groter dan 10^2, door herconfigureerbaarheid aan het systeemconcept toe te voegen. Een dergelijke architectuur is op een efficiënte manier robuust tegen zowel defecten ten gevolge van de productie als wel voorbijgaande (transient) fouten, en tolereert poortfouten tot een ratio van 10^{-2}, wat voor elk huidig micro-elektronisch systeem onaanvaardbaar zou zijn.

Afgeleid uit von Neumann's multiplexing techniek, introduceren we drievoudig verwoven redundantie (Triplicated Interwoven Redundancy: TIR), als een generalisatie van drievoudige modulaire redundantie (Triple Modular Redundancy: TMR), maar dan met willekeurige interconnecties. Een architectuur van een prototype processor en het bijbehorende — op simulatiegebaseerde — betrouwbaarheidsmodel zijn opgesteld en zijn gebruikt om de fouttolerantie te evalueren. De processor is ter vergelijking, naast met TIR, ook met zogenaamde quadded logica uitgevoerd. In het algemeen, is de betrouwbaarheid van een TIR circuit vergelijkbaar met dat van een gelijkwaardige TMR circuit, terwijl voor bepaalde interconnectiepatronen de TIR structuur zelfs inferieure prestaties vergeleken met TMR kan hebben door de verweven aard van de interconnecties. TIR kan tot hogere orden worden uitgebreid, die wij N-tuple verweven redundantie (N-tuple interwoven redundancy: NIR) noemen. Het gebruik van 5-tuple verweven redundantie leidt tot een economische redundantiefactor van minder dan 10 voor de systeemarchitectuur. Er wordt aangetoond dat het ontwerp en de implementatie van herstellende schakelingen (meerderheids-stemmers of "voters") belangrijk voor TIR/NIR en quadded structuren zijn. Slechts met een eenvoudig ontwerp van de voter is het mogelijk om — bij een hogere orde van NIR — a betere systeembetrouwbaarheid te verkrijgen dan met TIR. TIR en NIR zijn in het bijzonder geschikt voor implementatie in moleculaire nano-computers, die zeer waarschijnlijk door een productieproces van stochastische chemische assemblage zullen worden vervaardigd.

In Hoofdstuk 5, zijn supergeleidende circuits van Josephson juncties onderzocht, met als doel hen in lokaal verbonden processorstructuren te kunnen gebruiken. Er wordt zowel een klassieke SIMD computerarchitectuur als een pijplijn-gebaseerde quantumcomputer structuur voorgesteld die hetzelfde basiscircuit, de Josephson junctie gebruiken. Idealiter is dat een dergelijke klassieke computer als pre-, post- en interprocessor voor quantum berekeningen kan dienen die dan in het hart van de Josephson junctie circuit worden uitgevoerd. Als zodanig, vormt het dan een heterogene quantum & klassieke computer geschikt voor implementaties van algoritmen zoals het factoriseringsalgoritme van Shor, dat voortdurend in één enkel algoritme klassieke berekeningsstappen en quantum berekeningsstappen door elkaar heen mengt. Hoewel niet specifiek uitgewerkt en in detail besproken in deze studie, zou in principe een architectuur in de vorm van een geheel omkeerbaar processornetwerk gebaseerd op supergeleidende ringen van Josephson juncties, hiervoor kunnen worden gebruikt.

Een quantum cellulair niet-lineaire netwerk (CNN) architectuur, gebaseerd op Josephson juncties circuits, wordt eveneens geïntroduceerd in dit hoofdstuk, als een mogelijk nieuw gegevensverwerkingparadigma voor Josephson circuits. Omdat zowel de klassieke processor architectuur (SIMD array), de quantumprocessor architectuur en de semi-quantumprocessor architectuur (quantum CNNs) gelijktijdig op het zelfde apparaat kunnen worden bestudeerd, is een Josephson circuit een goed voertuig om de architectuur kwesties van quantum- en nano-electronische computersystemen te onderzoeken, onafhankelijk van de vraag waarvan het apparaat uiteindelijke geïmplementeerd wordt.

Dit laatste hoofdstuk besluit het proefschrift, dat kan worden geplaatst in “de begin dagen” van onderzoek naar nano-electronische en quantum-fysische computer architecturen.

En voortbordurend op het thema van dit proefschrift: De wetenschappelijke artikelen die de basis vormden van de hoofdstukken van dit proefschrift, zijn ondertussen al weer opgevolgd door vele nieuwe studies in fouttolerante technieken om het foutengedrag in een gemultiplext nano-systeem van Markov kettingen te onderzoeken, zoals het gebruiken van de Monte Carlo simulaties [75], vertakkings (bifurcation) theorie [76] en een exacte analyse gebruik makend van combinatorische argumentatie [158]. Voorts is er een probabilistisch-gebaseerde methodologie voorgesteld voor het ontwerpen van nano-computer architecturen die op Markov Random Fields (MRF) wordt gebaseerd [159], en worden er CAD hulpmiddelen ontwikkeld om de evaluatie van diverse fouttolerante schema’s en hun betrouwbaarheid / redundantie uitruil te automatiseren [74]. De redundantietechnieken, voortkomend uit von Neumann’s techniek, zijn in wezen foutverbeterende codes (Error Correcting Codes: ECC). De multiplexing constructie komt neer op het gebruik van een herhalingscode, waarin elk symbool van een bericht vaak wordt herhaald om redundantie te creëren [81]. Het gebruik van het fout-verbeterende codes, evenals de kwestie van fout-tolerantie voor nano-computing in het algemeen, wacht op verder onderzoek.

Nieuwe processorsystemen, die nu worden gezien als zichzelf aanpassende systemen gebaseerd op moleculaire elektronica [160], op door de biologie geïnspireerde zelf lerende en evoluerende systemen [161], op niet-lineaire dynamische systemen [44] en op quantumcomputers, zullen op de lange duur naar voren komen, wellicht leidend tot nieuwe soorten algoritmen en architecturen. Het onderzoek van algoritmen en architecturen zou nu naar toepassingen in nano-technologie moeten streven. Een architectuur zal sterk het ontwerp van apparaten en circuits beïnvloeden, en vice versa: de mogelijkheden en problemen die in nano-electronische apparaten en circuits worden gevonden zullen sterk de keuze van een architectuur beïnvloeden. In het onderzoek naar nano-computer architecturen moet daarom een interdisciplinaire benadering worden gevolgd en het succes zal uiteindelijk gebouwd zijn op een multidisciplinaire inspanning uit de gebieden scheikunde, natuurkunde, elektrotechniek, computerwetenschap, en wellicht vele anderen.

Bibliography

[1] *International Technology Roadmap for Semiconductors*, 2003 ed. http://public.itrs.net/.

[2] R. Compañó (ed.), *Technology Roadmap for Nanoelectronics*, 2nd ed. European Commission IST programme Future and Emerging Technologies, 2000.

[3] K. Nikolić, M. Forshaw and R. Compañó, "The current status of nanoelectronic devices," *Int. J. of Nanosci.* vol. 2, pp. 7-29, 2003.

[4] G. Bourianoff, "The future of nanocomputing," *IEEE Computer*, pp. 44-53, August 2003.

[5] K. L. Wang, "Issues of nanoelectronics: a possible roadmap," *J. of Nanosci. and Nanotech.*, vol. 2, pp. 235-266, 2002.

[6] P. Beckett and A. Jennings, "Towards nanocomputer architecture," in *Proc. Seventh Asia-Pacific Computer Systems Architecture Conference (ACSAC'2002)*, Australian Computer Society, Inc.

[7] J. D. Meindl, Q. Chen and J. A. Davis, "Limits on silicon nanoelectronics for terascale integration," *Science*, vol. 293, pp. 2044-2049, 2001.

[8] M. Forshaw, R. Stadler, D. Crawley and K. Nikolić, "A short review of nanoelectronic architectures," *Nanotechnology*, vol. 15, pp. S220-S223, 2004.

[9] S. Lloyd, "Ultimate physical limits to computation," *Nature*, vol. 406, pp. 1047-1054, 2000.

[10] P. Mazumder, S. Kulkarni, B. Bhattacharya, J. P. Sun and G. I. Haddad, "Digital circuit applications of resonant tunneling devices," *Proc. IEEE*, vol. 86 pp. 664-686, 1998.

[11] 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.*, vol. 19, pp. 7-9, 1998.

[12] M. A. Reed, W. R. Frensley, R. J. Matyi, J. N. Randall and A. C. Seabaugh "Realization of a three-terminal resonant tunneling device: the bipolar quantum resonant tunneling transistor," *Appl. Phys. Lett.*, vol. 54, pp. 1034-1036, 1989.

[13] J. Stock, J. Malindretos, K. M. Indlekofer, M. Pottgens, A. Forster and H. Luth, "A vertical resonant tunneling transistor for application in digital logic," *IEEE Trans. Electron Dev.*, vol. 48, pp. 1028-1032, 2001.

[14] K. K. Likharev, "Single-electron devices and their applications," *Proc. IEEE*, vol. 87, pp. 606-632, 1999.

[15] R. H. Chen, A. N. Korotkov and K. K. Likharev, "Single-electron transistor logic," *Appl. Phys. Lett.*, vol. 68, pp. 1954-1956, 1996.

[16] C. P. Heij, P. Hadley and J. E. Mooij, "Single-electron inverter," *Appl. Phys. Lett.,* vol. 78, pp. 1140-1142, 2001.

[17] Y. Ono, Y. Takahashi, K. Yamazaki, M. Nagase, H. Namatsu, K. Kurihara and K. Murase, "Si complementary single-electron inverter with voltage gain," *Appl. Phys. Lett.,* vol. 76, pp. 3121-3123 MAY 22 2000.

[18] S. Kasai and H. Hasegawa, "GaAs and InGaAs single electron hexagonal nanowire circuits based on binary decision diagram logic architecture," *Physica E,* vol. 13, pp. 925-929, 2002.

[19] N. J. Stone, H. Ahmed and K. Nakazato, "A high-speed silicon single-electron random access memory," *IEEE Electron Device lett.*, vol. 20, pp. 583-585, 1998.

[20] Z. A. K. Durrani, A. C. Irvine and H. Ahmed, "Coulomb blockade memory using integrated single-electron transistor/metal-oxide-semiconductor transistor gain cells," *IEEE Trans. Electron Dev.*, vol. 47, pp. 2334-2339, DEC 2000.

[21] H, Mizuta, H. O. Muller, K. Tsukagoshi, D. Williams, Z. Durrani, A. Irvine, G. Evans, S. Amakawa, K. Nakazato and H. Ahmed, "Nanoscale coulomb blockade memory and logic devices," *Nanotechnology*, vol. 12 (2), pp. 155-159, JUN 2001.

[22] E. F. Codd, *Cellular Automata*, Academic Press, London, UK, 1968.

[23] C. S. Lent and P. D. Tougaw, "A device architecture for computing with quantum dots," *Proc. IEEE*, vol. 85, pp. 541-557, 1997.

[24] A. O. Orlov, I. Amlani, R. 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.,* vol.77, pp. 295-297, 2000.

[25] A. O. Orlov, R. Kummamuru, R. Ramasubramaniam, G. Toth, C. S. Lent, G. H. Bernstein and G. L. Snider, "Experimental demonstration of a latch in clocked quantum-dot cellular automata," *Appl. Phys. Lett.,* vol.78, pp. 1625-1627, 2001.

[26] A. O. Orlov, R. Kummamuru, R. Ramasubramaniam, C. S. Lent, G. H. Bernstein and G. L. Snider, "Clocked quantum-dot cellular automata shift register," *Surface Science*, vol. 532, pp. 1193-1198, 2003.

[27] P. D. Tougaw and C. S. Lent, "Dynamic behavior of quantum cellular automata," *J. of Appl. Phys.*, vol. (80) pp. 4722-4736, 1996.

[28] C. S. Lent, B. Isaksen and M. Lieberman, "Molecular quantum-dot cellular automata," *J. of Amer. Chem. Soci.*, vol. 125, pp. 1056-1063, 2003.

[29] K. Nikolić, D. Berzon and M. Forshaw, "Relative performance of three nanoscale devices — CMOS, RTDs and QCAs — against a standard computing task," *Nanotechnology*, vo. 12 (1), pp. 38-43, JUN 2001.

[30] R. P. Cowburn and M. E. Welland, "Room temperature magnetic quantum cellular automata," *Science*, vol. 287 (5457), pp. 1466-1468, FEB 25 2000.

[31] D. A. Allwood, G. Xiong, M. D. Cooke, C. C. Faulkner, D. Atkinson, N. Vernier and R. P. Cowburn, "Submicrometer ferromagnetic NOT gate and shift register," *Science*, Vol 296, pp. 2003-2006 , 14 June 2002.

[32] P. Bunyk, K. Likharev and D. Zinoviev, "RSFQ technology: physics and devices," Int. J. High Speed Electronics and Systems, vol. 11 (1), pp. 257-305, 2001.

[33] W. Chen, A. V. Rylyakov, V. Patel, J. E. Lukens and K. K. Likharev, "Rapid single flux quantum T-flip flop operating up to 770 GHz," *IEEE Tran. Applied Superconductivity*, vol. 9, pp. 3212-3215, 1999.

[34] D. K. Brock, "RSFQ technology: circuits and systems," Int. J. High Speed Electronics and Systems, vol. 11 (1), pp. 307-362, 2001.

[35] A. M. Kadin, C. A. Mancini, M. J. Feldman and D. K. Brock, "Can RSFQ logic circuits be scaled to deep submicron junctions?" *IEEE Tran. Applied Superconductivity*, vol. 11, pp. 1050-1055, 2001.

[36] D. K. Brock, E. K. Track and J. M. Rowell, "Superconductor ICs: the 100-GHz second generation," *IEEE Spectrum*, vol. 37, pp. 40-46, 2000.

[37] J. E. Mooij, T. P. Orlando, L. Levitov, L. Tian, Caspar H. van der Wal and Seth Lloyd, "Josephson persistent-current qubit," *Science*, vol. 285, pp. 1036-1039, 1999.

[38] T. P. Orlando, J. E. Mooij, L. Tian, Caspar H. van der Wal, L. S. Levitov, Seth Lloyd and J. J. Mazo, "Superconducting persistent-current qubit," *Phys. Rev. B*, vol. 60, pp. 15398-15413, 1999.

[39] 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 macroscopic persistent-current states," *Science*, vol. 290, pp. 773-777, 2000.

[40] I. Chiorescu, Y. Nakamura, C. J. P. M. Harmans and J. E. Mooij, "Coherent quantum dynamics of a superconducting flux qubit," *Science*, vol. 299, pp. 1869-1871; published online 13 Feb. 2003, 10.1126/science. 1081045

[41] P. Jonker and J. Han, "On quantum and classical computing with arrays of superconducting persistent current qubits," in *Proc. CAMP2000, Fifth IEEE International Workshop on Computer Architectures for Machine Perception* (Padova, Italy, Sep.11-13, 2000), IEEE Computer Society, Los Alamitos, California, USA, pp. 69-78, 2000.

[42] J. Han and P. Jonker, "Novel computing architecture on arrays of Josephson persistent current bits," in *Proc. MSM 2002, Fifth International Conference on Modeling and Simulation of Microsystems* (San Juan, Puerto Rico, USA, April 22-25), pp. 636-639, 2002.

[43] J. Han and P. Jonker, "On quantum computing with macroscopic Josephson qubits," in *Proc. IEEE-NANO 2002, 2nd IEEE Conference on Nanotechnology* (Washington D.C., USA, Aug.26-28), pp. 305-308, 2002.

[44] J. Han and P. Jonker, "Quantum cellular nonlinear networks using Josephson circuits," in *Proc. IEEE-NANO 2003, 3rd IEEE Conference on Nanotechnology* (San Francisco, CA, USA, Aug.12-14), pp. 457-460, 2003. An extended version is submitted for publication in IEEE Trans. Nanotechnology.

[45] J. Han and P. Jonker, "A fault-tolerant technique for nanocomputers: NAND multiplexing", in *Proc. ASCI 2002, The Eighth Annual Conference of the Advanced School for Computing and Imaging* (Lochem, NL, June 19-21), ASCI, Delft, pp. 59-66, 2002.

[46] J. Han and P. Jonker, "A system architecture solution for unreliable nanoelectronic devices," *IEEE Trans. Nanotechnology*, vol. 1, no. 4, pp. 201-208, 2002.

[47] J. Han and P. Jonker, "A defect- and fault-tolerant architecture for nanocomputers," *Nanotechnology*, vol. 14, no. 2, pp. 224-230, 2003.

[48] J. Han and P. Jonker, "A study on fault-tolerant circuits using redundancy," in *Proc. VLSI 2003 Multiconference in Computer Science and Engineering* (Las Vegas, NV, USA, June 23-26), pp. 65-69, 2003.

[49] J. Han and P. Jonker, "From massively parallel image processors to fault-tolerant nanocomputers," in *Proc. 17th International Conference on Pattern Recognition* (Cambridge, UK, Aug.23-26), Vol.3, IEEE Computer Society Press, Los Alamitos, pp. 2-7, 2004.

[50] J. Han and P. Jonker, "Fault-tolerance in nanocomputers: random interwoven redundancy," to appear in *IEEE Trans. VLSI.*

[51] M. Forshaw, D. Crawley, P. Jonker, J. Han, and C. Sotomayor Torres, *Nano_Arch_Review: A Review of the Status of Research and Training into Architectures for Nanoelectronic and Nanophotonic Systems in the European Research Area*, EU 6th Framework Programme "Information Society Technologies" FP6/2002/IST/1, Specific Support Action, Contract no. 507519, University College, London, UK, July 2004.

[52] L.C. Venema and C. Dekker, "Carbon nanotubes," in *New Frontiers of Science and Technology*, edited by L. Esaki (Frontiers Science Series No. 31), Universal Acad. Press, pp. 293-300, 2000.

[53] S. J. Tans, A. R. M. Verschueren, and C. Dekker, "Room-temperature transistor based on a single carbon nanotube," *Nature*, vol. 393, pp. 49-52, 1998.

[54] H. W. Ch. Postma, T. F. Teepen, Z. Yao, M. Grifoni and C. Dekker, "Carbon nanotubes single-electron transistors at room temperature," *Science*, vol. 293, pp. 76-79, 2001.

[55] V. Derycke, R. Martel, J. Appenzeller and Ph. Avouris, "Carbon nanotube inter- and intramolecular logic gates," *Nano Letters*, vol. 1, pp. 453-456, 2001.

[56] A. Bachtold, P. Hadley, T. Nakanishi and C. Dekker, "Logic circuits with carbon nanotube transistors," *Science,* vol. 294, pp. 1317-1320, 2001.

[57] A. Javey, Q. Wang, A. Urai, Y. Li and H. Dai, "Carbon nanotube transistor arrays for multi-stage complementary logic and ring oscillators," *Nano Lett.*, vol. 2, pp. 929-932, 2002.

[58] Y. Huang, X. F. Duan, Q. Wei and C. M. Lieber, "Directed assembly of one-dimensional nanostructures into functional networks," *Science*, vol. 291, pp. 630-633, 2001.

[59] N. A. Melosh, A. Boukai, F. Diana, B. Gerardot, A. Badolato, P. M. Petroff and J. R. Heath, "Ultrahigh-density nanowire lattices and circuits," *Science,* vol. 300, pp. 112-115, 2003.

[60] X. F. Duan, Y. Huang, Y. Cui, J. F. Wang and C. M. Lieber, "Indium phosphide nanowires as building blocks for nanoscale electronic and optoelectronic devices," *Nature*, vol. 409 (6816), pp. 66-69, JAN 4 2001.

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

[62] Z. Zhong, D. Wang, Y. Cui, M. W. Bockrath and C. M. Lieber, "Nanowire crossbar arrays as address decoders for integrated nanosystems," *Science,* vol. 302, pp. 1377-1379, 2003.

[63] C. Joachim, J. K. Gimzewski and A. Aviram, "Electronics using hybrid-molecular and mono-molecular devices," *Nature*, vol. 408, pp. 541-548, 2000.

[64] C. P. Collier, E. W. Wong, M. Belohradsky, F. M. Raymo, J. F. Stoddart, P. J. Kuekes, R. S. Williams and J. R. Heath, "Electronically configurable molecular-based logic gates," *Science,* vol. 285, pp. 391-394, 1999.

[65] C. P. Collier, G. Mattersteig, E. W. Wong, Y. Luo, K. Beverly, J. Sampaio, F. M. Raymo, J. F. Stoddart and J. R. Heath, "A [2]catenane-based solid state electronically reconfigurable switch," *Science,* vol. 280, pp. 1172-1175, 2000.

[66] H. Park, J. Park, A. K. L. Lim, E. H. Anderson, A. P. Alivisatos and P. L. McEuen, "Nanomechanical oscillations in a single-C-60 transistor," *Nature*, vol. 407, pp. 57-60, 2000.

[67] Y. Luo, C. P. Collier, J. O. Jeppesen, K. A. Nielsen, E. Delonno, G. Ho, J. Perkins, H. R. Tseng, T. Yamamoto, J. F. Stoddart and J. R. Heath, "Two-dimensional molecular electronics circuits," *ChemPhysChem*, vol 3 (6), pp. 519-525, JUN 17 2002.

[68] G. Y. Tseng and J. C. Ellenbogen, "Toward nanocomputers," *Science,* vol. 294, pp. 1293-1294, 2001.

[69] M. R. Stan, P. D. Franzon, S. C. Goldstein, J. C. Lach and M. M. Ziegler, "Molecular electronics: from devices and interconnect to circuits and architecture," in *Proc. IEEE*, vol. 91, pp. 1940-1957, 2003.

[70] J. von Neumann, “Probabilistic logics and the synthesis of reliable organisms from unreliable components,” *Automata Studies*, Shannon C.E. & McCarthy J., eds., Princeton University Press, Princeton N.J. pp. 43-98, 1956.

[71] D. P. Siewiorek and R. S. Swarz, *Reliable Computer Systems: Design and Evaluation,* 1998. A K Peters, Natick, Massachusetts, USA.

[72] K. Nikolic, A. Sadek, M. Forshaw, “Architectures for reliable computing with unreliable nanodevices,” in *Proc. IEEE-NANO 2001*, pp. 254 -259, 2001.

[73] K. Nikolic, A. Sadek, M. Forshaw, “Fault-tolerant techniques for nanocomputers,” *Nanotechnology, 13*, pp. 357-362, 2002.

[74] G. Norman, D. Parker, M. Kwiatkowska and S. K. Shukla, “Evaluating the reliability of defect-tolerant architectures for nanotechnology with probabilistic model checking,” in *Proc. Int. Conf. on VLSI Design*, pp. 907-912, 2004.

[75] A. S. Sadek, K. Nikolić and M. Forshaw, “Parallel information and computation with restitution for noise-tolerant nanoscale logic networks,” *Nanotechnology, 15*, pp. 192-210, 2004.

[76] Y. Qi, J. Gao and J. A. B. Fortes, “Probabilistic computation: a general framework for fault-tolerant nanoelectronic systems,” Advanced Computing and Information Processing Laboratory, University of Florida, Tech. Rep. TR-ACIS-03-002, Nov. 2003. [online] http://www.acis.ufl.edu/techreports/acis03002.pdf

[77] J. R. Heath, P. J. Kuekes, G. S. Snider and R. S. Williams, “A defect-tolerant computer architecture: opportunities for nanotechnology,” *Science,* vol. 280, pp. 1716-1721, 1998.

[78] S. C. Goldstein and M. Budiu, “Nanofabrics: spatial computing using molecular nanoelectronics,” in *Proc. 28th Int. Symp. Computer Architecture*, pp. 178-189, 2001.

[79] A. DeHon, “Array-based architecture for FET-based nanoscale electronics,” *IEEE Trans. Nano*, vol. 2, pp. 23-32, 2003.

[80] M. Purser, *Introduction to error-correcting codes*, 1995. Artech House, Norwood, MA, U.S.A.

[81] F. Peper, J. Lee, F. Abo, T. Isokawa, S. Adachi, N. Matsui and S. Mashiko, “Fault-tolerance in nanocomputers: a cellular array approach,” *IEEE Trans. Nanotechnology*, vol. 3, no. 1, pp. 187-201, 2004.

[82] M. J. B. Duff, “CLIP4,” *Special Computer Architectures for Pattern Processing*, CRC Press, Boca Raton, Florida, USA, pp.65-86, 1982.

[83] K. E. Batcher, “Design of a massively parallel processor,” *IEEE Trans. Comput.*, vol. 29, pp. 836-840, 1980.

[84] S. Kyo, T. Koga and S. Okazaki, “IMAP-CE: a 51.2 GOPS video rate image processor with 128 VLIW processing elements,” in *Proc. Int. Conf. Image Processing*, pp. 294-297, 2001.

[85] R. Kleihorst, A. Abbo, A. van der Avoird, M. O. de Beeck, L. Sevat, P. Wielafe, R. van Veen, and H. van Herten, “Xetal: a low-power high-performance smart camera processor,” in *proc. IEEE ISCAS 2001*, pp. 215-218, 2001.

[86] T. J. Fountain, *Processor Arrays: Architecture and Applications*, 1987. Academic Press, London, UK.

[87] C. S. Lent, P. D. Tougaw, W. Porod and G. H. Bernstein, “Quantum cellular automata,” *Nanotechnology*, vol. 4, pp. 49-57, 1993.

[88] L. O. Chua and L. Yang, “Cellular Neural Networks: Theory,” *IEEE Tran. Circuits Sys.*, vol. 35, pp. 1257-1272; and “Cellular Neural Networks: Applications,” *ibid.* pp. 1273-1290, 1988.

[89] P. P. Jonker, *Morphological Image Processing: Architecture and VLSI Design.* Dordrecht, The Netherlands: Kluwer Academic Publishers, 1992.

[90] J. von Neumann, *Theory of Selft-Reproducing Automata*, Univ. of Illinois Press, Urbana, IL, U.S. 1966.

[91] M. C. B. Parish and M. Forshaw, “Physical constraints on magnetic quantum cellular automata,” *Appl. Phys. Lett.,* vol. 83, pp. 2046-2048, 2003.

[92] D. Berzon and T. Fountain, “Computer memory structures using QCAs,” Tech. rep. in Image Processing Group at University College London, U.K. 1998, Report No. 98/1.

[93] T. J. Fountain, M. J. B. Duff, D. G. Crawley, C. D. Tomlinson and C. D. Moffat, “The use of nanoelectronic devices in highly parallel computing systems,” *IEEE Trans. VLSI*, vol. 6, pp. 31-38, 1998.

[94] K. R. Crounse and L. O. Chua, “The CNN universal machine is as universal as a Turing machine,” *IEEE Tran. Circuit Sys. I*, vol. 43, pp. 353-355, 1996.

[95] P. Julián, R. Dogaru, M. Itoh, M. Hänggi and L. O. Chua, “On the RTD implementation of simplicial cellular nonlinear networks,” in *Proc. 7th IEEE Int. Workshop CNNs Appl.*, pp. 140-147, 2002.

[96] C. Gerousis, S. M. Goodnick and W. Porod, “Toward nanoelectronic cellular neural networks,” *Int. J. Circ. Theor. Appl.*, vol. 28, pp. 523-535, 2000.

[97] T. Yang, R. A. Kiehl and L. O. Chua, “Tunneling phase logic cellular nonlinear networks,” Int. J. of Bifurcation and Chaos, vol. 11 (12): 2895-2911, DEC 2001.

[98] G. Toth, C. S. Lent, P. D. Tougaw, Y. Brazhnik, W. Weng, W. Porod, R.-W. Liu and Y.-F. Huang, “Quantum cellular neural networks,” *Superlattices and Microstructures*, vol. 20, pp. 473-478, 1996.

[99] R. P. Feynman, R. B. Leighton and M. Sands, *The Feynman Lectures on Physics.* Reading, Massachusetts: Addison-Wesley Publishing Company, 1966.

[100] D. Deutsch, “Quantum theory, the Church-Turing principle and the universal quantum computer,” *Proc. R. Soc. Lond. A*, 400 (1818), pp. 97-117, 1985.

[101] D. Deutsch and A. Ekert, "Quantum computation," *Physics World*, pp. 47-52, March 1998.

[102] P. Shor, "Algorithms for quantum computation: discrete log and factoring," in *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, IEEE Computer Society Press, Los Alamitos, CA, USA, pp.124-134, 1994.

[103] L. K. Grover, "Quantum mechanics helps in searching for a needle in a haystack," *Phys. Rev. Lett.*, vol. 79, pp. 325-328, 1997.

[104] S. Hallgren, "Polynomial-time quantum algorithms for Pell's equation and the principal ideal problem," in *Proc. The 34th ACM Symposium on Theory of Computing (STOC2002)*, Montreal, Canada, May 19-21, 2002.

[105] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, Cambridge University Press, 2000.

[106] L. M. K. Vandersypen, M. Steffen, G. Breyta, C. S. Yannoni, Mark H. Sherwood and I. L. Chuang, "Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance," *Nature*, Vol. 414, pp. 883-887, 2001.

[107] C. H. Bennett and D. P. DiVincenzo, "Quantum information and computation," *Nature*, Vol. 404, pp. 247-255, 2000.

[108] D. Vion, A. Aassime, A. Cottet, P. Joyez, H. Pothier, C. Urbina, D. Esteve, M. H. Devoret, "Manipulating the quantum state of an electrical circuit," *Science*, Vol. 296, pp. 886-889, 2002.

[109] A. J. Leggett, "Superconducting qubits — a major roadblock dissolved," *Science*, Vol. 296, pp. 861-862, 2002.

[110] S. Lloyd, "A potentially realizable quantum computer," *Science*, Vol. 261, pp. 1569-1571, 1993.

[111] N. Pippenger, "Reliable computation by formulas in the presence of noise," *IEEE Trans. Inform. Theory*, vol. 34, pp. 194-197, 1988.

[112] T. Feder, "Reliable computation by networks in the presence of noise," *IEEE Trans. Inform. Theory*, vol. 35, pp. 569-571, 1989.

[113] R. L. Dobrushin and S. I. Ortyukov, "Upper bound on the redundancy of self-correcting arrangements of unreliable functional elements," *Prob. Inf. Trans.*, vol. 13, pp. 203-218, 1977.

[114] R. L. Dobrushin and S. I. Ortyukov, "Lower bound for the redundancy of self-correcting arrangements of unreliable functional elements," *Prob. Inf. Trans.*, vol. 13, pp. 59-65, 1977.

[115] N. Pippenger, G. D. Stamoulis and J. N. Tsitsiklis, "On a lower bound for the redundancy of reliable networks with noisy gates," *IEEE Trans. Inform. Theory*, vol. 37, pp. 639-643, 1991.

[116] N. Pippenger, "On networks of noisy gates," in *Proc. 26th Ann. Symp. Foundations Comput. Sci.*, pp. 30-38, 1985.

[117] N. Pippenger, "Invariance of complexity measures for networks with unreliable gates," *J. ACM*, vol. 36, pp. 531-539, 1989.

[118] N. Pippenger, "Developments in "The synthesis of reliable organisms from unreliable components"," in *Proc. Symposia in Mathematics*, vol. 50, pp. 311-324, 1990.

[119] W. Evans and N. Pippenger, "On the maximum tolerable noise for reliable computation by formulas," *IEEE Trans. Inform. Theory*, vol. 44, pp. 1299-1305, 1998.

[120] D. Lu, *Stochastic Process and Applications*, Tsinghua University Press, 1986.

[121] R. H. Klunder and J. Hoekstra, "Programmable logic using a SET electron box," in *Proc. ICECS 2001*, pp.185-188, 2001.

[122] J. M. Tour, W. L. Van Zandt, C. P. Husband, S. M. Husband, L. S. Wilson, P. D. Franzon and D. P. Nackashi, "Nanocell logic gates for molecular computing," *IEEE Trans. Nanotechnology*, vol. 1, pp. 100-109, 2002.

[123] P. J. Kuekes and R. S. Williams, "Demultiplexer for a molecular wire crossbar network (MWCN DEMUX)," *U. S. Patent 6256767*, July 2001.

[124] J. G. Tryon, "Quadded logic," *Redundancy Techniques for Computing Systems*, pp. 205-228, 1962. Spartan Books, Washington D.C., USA.

[125] P. A. Jensen, "Quadded NOR logic," *IEEE Trans. Reliability*, vol. 12, pp. 22-31, 1963.

[126] W. H. Pierce, *Failure-Tolerant Computer Design,* Academic Press, 1965.

[127] J. A. Abraham, "A combinatorial solution to the reliability of interwoven redundant logic networks," *IEEE Trans. Computers*, vol. 24, pp. 578-584, 1975.

[128] J. A. Abraham and D. P. Siewiorek, "An algorithm for the accurate reliability evaluation of triple modular redundancy networks," *IEEE Trans. Computers*, vol. 23, pp. 682-692, 1974.

[129] A. E. Barbour and A. S. Wojcik, "A general, constructive approach to fault-tolerant design using redundancy," *IEEE Trans. Computers*, vol. 38, pp. 15-29, 1989.

[130] C. E. Stroud, "Reliability of majority voting based VLSI fault-tolerant circuits," *IEEE Trans. VLSI*, vol. 2, pp. 516-521, 1994.

[131] S. Spagocci and T. Fountain, "Fault rates in nanochip devices," in *Electrochemical Society Proc.*, vol. 98-19, pp. 582-593, 1999.

[132] M. Forshaw and K. Nikolić, "Algorithms and architectures for nanoelectronic computers," in *Technical report MEL-ARL Project ANSWERS*, June 2000.

[133] D. Mange, M. Sipper, A. Stauffer and G. Tempesti, "Toward robust integrated circuits: the embryonics approach," in *Proc. IEEE*, vol. 88, pp. 516-541, 2000.

[134] I. Koren and Z. Koren, “Defect tolerance in VLSI circuits: techniques and yield analysis,” in *Proc. IEEE*, vol. 86, pp. 1819-1836, 1998.

[135] C. H. Stapper, “A new statistical approach for fault-tolerant VLSI systems,” in *Proc. 22nd Int. Symp. Fault-Tolerant Computing*, pp. 356-365, 1992.

[136] J. Lach, W. H. Mangione-Smith and M. Potkonjak, “Low overhead fault-tolerant FPGA systems,” *IEEE Trans. VLSI*, vol. 6, pp. 212-221, 1998.

[137] F. Distante, M.G.Sami and R. Stefanelli, “Fault-tolerance and reconfigurability issues in massively parallel architectures,” in *Proc. CAMP, Computer Architecture for Machine Perception 1995*, pp. 340-349, 1995.

[138] W. G. Brown, J. Tierney and R. Wasserman, “Improvement of electronic-computer reliability through the use of redundancy,” *IRE Trans. Electronic Computers*, vol. 10, pp. 407-416, September 1961.

[139] T. Oya, T. Asai, T. Fukui and Y. Amemiya, “A majority-logic device using an irreversible single-electron box,” *IEEE Trans. Nanotechnology*, vol. 2, no. 1, pp. 15-22, 2003.

[140] M. J. B. Duff and T. J. Fountain, *Cellular logic Image Processing*, 1986. Academic Press, London, UK.

[141] D. P. DiVincenzo, “Quantum Computation,” *Science*, Vol. 285, pp. 255, 1995.

[142] Y. Shimazu, “Two four-junction qubits coupled by a superconducting flux transporter”, an internal report in the Quantum Transportation group at the Delft University of Technology (4/5/2000).

[143] FastHenry, a multipole-accelerated inductance analysis program developed by the RLE Computational Prototyping Group at Massachusetts Institute of Technology, U.S.A. http://rleweb.mit.edu/vlsi/codes.htm

[144] T. van Duzer and C. W. Turner, *Principles of Superconductive Devices and Circuits*, 2nd ed. Upper Saddle River, New Jersey: Prentice Hall PTR, 1998.

[145] A. Eket and R. Jozsa, “Quantum Computation and Shor’s Factoring Algorithm,” *Reviews of Modern Physics*, Vol. 68, No. 3, pp. 733, 1996.

[146] F. G. Paauw, “Spectroscopy experiments on two coupled Josephson persistent current qubits,” a Master thesis in the Quantum Transportation group at the Delft University of Technology, November 2002.

[147] I. L. Chuang, L. M. K. Vandersypen, X. Zhou, D. W. Leung, S. Lloyd, “Experimental Realization of a Quantum Algorithm,” *Nature*, Vol. 393, 14 May 1998.

[148] D. Deutsch and R. Jozsa, “Rapid solutioin of problems by quantum computation,” *Proc. R. Soc. Lond. A439*, pp. 553-558, 1992..

[149] A. Barenco, C. Bennett, R. Cleve, D. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. Smolin, and H. Weinfurter, “Elementary gates for quantum computation,” *Physical Review A*, Vol. 52, 3457, 1995.

[150] V. Vedral, A. Barenco, and A. Ekert, "Quantum Networks for Elementary Arithmetic Operations," *Physical Review A*, Vol. 54, No. 1, July 1996.

[151] D. S. Crankshaw, E. Trías and T. P. Orlando, "Magnetic Flux Controlled Josephson Array Oscillators," *IEEE Tran. Applied Superconductivity*, vol. 11, pp. 1223-1226, 2001.

[152] R. C. Rey-de-Castro, M. F. Bocko, A. M. Herr, C. A. Mancini and M. J. Feldman, "Design of an RSFQ Control Circuit to Observe MQC on an rf-SQUID," *IEEE Tran. Applied Superconductivity*, vol. 11, pp. 1014-1017, 2001.

[153] D. S. Crankshaw, J. L. Habif, X. Zhou, T. P. Orlando, M. J. Feldman and M. F. Bocko, "An RSFQ Variable Duty Cycle Oscillator for Driving a Superconductive Qubit," *IEEE Tran. Applied Superconductivity*, vol. 13, pp. 966-969, 2003.

[154] T. P. Orlando, S. Lloyd, L. S. Levitov, K. K. Berggren, M. J. Feldman, M. F. Bocko, J. E. Mooij, C. J. P. Harmans and C. H. van der Wal, "Flux-based Superconductive Qubits for Quantum Computation," *Physica C*, 372-376, pp. 194-200, 2002.

[155] J. P. Hornak, *The Basics of MRI*, online at: http://www.cis.rit.edu/htbooks/mri/inside.htm.

[156] C. Bennett, "Logical Reversibility of Computation," *I.B.M. J. Res. Dev.*, Vol. 17, pp. 525-535, 1973.

[157] A. De Vos, "Reversible Computing," *Progress in Quantum Electronics*, Vol. 23, pp. 1-49, 1999.

[158] S. Roy, V. Beiu and M. Sulieman, "Majority Multiplexing: Economical Redundant Fault-Tolerant Designs for Nano Architectures," in Proc. Int. Joint Conf. on Neural Networks, 2004. [online] http://www.eecs.wsu.edu/~vbeiu/workshop_nips03/Presentations/S_Roy.pdf

[159] R. I. Bahar, J. Mundy and J. Chen, "A probabilistic-based design methodology for nanoscale computation," in *Proc. Int. Conf. on CAD*, pp. 480-486, 2003.

[160] "Adaptive systems," research program in the NASA Institute for Nanoelectronics and Computing. [online] http://inac.purdue.edu/wps/portal/_s.155/1020

[161] Ö. Türel, J. H. Lee, X. Ma and K. K. Likharev, "Neuromorphic architectures for nanoelectronic circuits," Int. J. of Circuit Theory and Applications, 32, pp. 277-302, 2004.

[162] P. P. Jonker, "Why Linear Arrays Are Better Image Processors," in *Proceedings of The 12th IAPR International Conference On Pattern Recognition*, IEEE, Computer Society, pp. 334-338, 1994.

[163] T. J. Fountain, "The Design of Highly-Parallel Image Processing Systems Using Nanoelectronic Devices," in *Proceedings of CAMP'97*, IEEE Computer Society Press, Los Alamitos, CA, USA, pp.210-219, 1997.

Acknowledgement

I would like to gratefully acknowledge:

Pieter Jonker, of course, for having been a great colleague, a mentor and a friend;

Ted Young, for having been a mentor and an enthusiastic leader;

Mike Forshaw, for the hospitality and productivity during my stay in London, and being a guide in the realm of nano-architecture issues;

Hans Mooij, Peter Hadley, Patrick Dewilde, Henk Corporaal, Yun He and Lucas van Vliet, for the scientific discussions we had and the precious time you spent on this thesis;

Albert Vossepoel, Bob Duin, Piet Verbeek, Ed Frietman and Yuval Garini, for your smiles;

Klara Schaafsma, Ronald Ligteringen and Wouter Smaal, for your assistance;

Gerold Kraft, Jurjen Caarls, Wouter Caarls, Frans Vos, Geert de Vries, Koen Vermeer and Edward Valstar, for being great roommates;

Mike van Ginkel, Cris Luengo Hendriks, Dick de Ridder, David Tax, Pavel Paclik, Ela Pekalska, Marina Skurichina, Judith Dijk, Marjolein van der Glas, Michiel de Bakker, Frank de Jong, Stelian Persa, Frank Faas, Iwo Serlie, Suprijanto, Kees van Wijk, Tuan Pham, Bart Vermolen, Margreet Docter and Amy Ahluwalia, for the enjoyable time on or off the campus;

Wanda en Gilbert, voor jullie hulp als wij in Holland, of niet, zijn.

My parents and my family, I know that every step I make is still under your watch;

My wife, Mingyue Chen, for her company, support and understanding all the time.

Curriculum Vitae

Jie Han was born in Pingshan, Hebei province, China, on November 11, 1975. He moved with his parents to Beijing in 1980. He studied at the Beijing No. 94 Middle School from 1988 to 1991 and at the Beijing No. 4 High School from 1991 to 1994. In 1994 he entered the Tsinghua University in Beijing and started to study electronics. In 1999 he obtained a B.Sc. degree in electronic engineering with a thesis on the project: "Signal transmission via power line carrier communication."

After his graduation in 1999 Jie Han moved to The Netherlands and became a Ph.D. student at the Quantitative Imaging Group (previously the Pattern Recognition Group) of the Faculty of Applied Sciences, Delft University of Technology. From 1999 to 2003, he worked on a Delft Inter-faculty Research Center (DIRC) program *Novel Computation Structures based on Nanoelectronic and Quantum Devices* (NanoComp), under the guidance of Dr.ir. P.P. Jonker and Prof.dr. I.T. Young. His subject was "Fault-tolerant architectures for nanoelectronic and quantum devices."

In 2004, while remaining at the Delft University of Technology, he continued his work through a European Union program *Nanoelectronic and Nanophotonic Computing Architecture Review* (NanoArch Review), with colleagues from the U.K. and Germany (Project leader: Dr. M. Forshaw, UCL).

From August 2004, Jie Han works as a postdoctoral scholar at the Advanced Computing and Information Systems Laboratory, Department of Electrical and Computer Engineering, University of Florida, Gainesville, Florida, U.S.A.