申请上海交通大学硕士学位论文

低成本数字集成电路在线故障检测的研究

学　　校：上海交通大学

院　　系：微电子学院

班　　级： Z0621091

学　　号： 1062109058

工程硕士生：祝翔宇

工程领域： 软件工程

导　师Ⅰ： 施国勇（教授）

导　师Ⅱ：

上海交通大学微电子学院

2008 年 12 月

A Dissertation Submitted to Shanghai Jiao Tong University for the Degree of Master of Engineering

A RESEARCH ON LOW COST IMPLEMENTING OF RUNTIME FAULT DIAGNOSIS IN DIGITAL INTEGRATED CIRCUITS

Author：Zhu Xiangyu

Specialty: Microelectronics

Advisor Ⅰ: Prof. Shi Guoyong

Advisor Ⅱ:

School of Microelectronics

Shanghai Jiao Tong University

Shanghai, P.R.China

December 5th, 2008

上海交通大学
学位论文原创性声明

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：

日期：　　年　　月　　日

上海交通大学

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权上海交通大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

保密□，在___年解密后适用本授权书。

本学位论文属于

不保密□。

（请在以上方框内打“√”）

学位论文作者签名：　　指导教师签名：

日期：　　年　　月　　日　　日期：　　年　　月　　日

低成本数字集成电路在线故障检测的研究

摘 要

随着集成电路工艺的进一步发展，CMOS 器件的特征尺寸和内部互联线的宽度越来越小，数字集成电路的规模和设计复杂度成倍增加，数字电路故障的来源越来越多，并且发生率也迅速提高。为应对数字集成电路中日益严重的可靠性问题，迫切需要提高电路的容错能力。目前多数研究集中在数字电路的并发故障检测和恢复技术上，但现有的并发故障检测方法代价过高，不适用于对成本敏感的应用中。为解决这一问题，本文研究了数字集成电路运行时故障检测的低成本实现策略，主要包括以下几个方面的内容：

首先简单介绍了课题的研究背景和电路故障的基本概念，对目前常用的电路容错设计技术分类举例说明，并介绍了并发故障检测的研究现状。

文章的第二部分介绍了可重构系统的基本概念和 FPGA 的动态局部重构特性，论证可重构系统，尤其是支持动态局部重构的 FPGA，是开发和实现系统容错设计的最佳平台。

文章的第三部分提出了分时故障诊断的主导思想和设计流程。分时故障诊断借助系统的可重构能力，在系统运行过程中交替改变检测区域的硬件配置，实现逻辑资源的分时复用，籍此在系统的故障检测能力没有明显下降的前提下，以平均故障检测延时的适当增加为代价，降低系统的冗余面积开销。此外还利用一个自定义的概率模型分析了分时故障诊断的检测能力和面积代价与设计参数之间的近似函数关系，并研究了电路实现中的若干具体问题。文章的最后借助一个应用实例验证了分时故障诊断的可行性以及之前理论分析的结论。

关键词：分时故障检测，并发故障检测，冗余面积开销，平均故障检测延时，动态局部重构

A RESEARCH ON LOW COST IMPLEMENTING OF RUNTIME FAULT DIAGNOSIS IN DIGITAL INTEGRATED CIRCUITS

ABSTRACT

As Integrated Circuits technology goes further into deep sub-micron, there is an impending demand to enhance the fault-tolerant ability of digital integrated circuits. The failure rates as well as failure sources is increasing significantly due to an exponential decrease in the minimal feature size of CMOS devices and the width of interconnections, besides an increase of design size and design complexity. So far, many up-to-date technologies have focused on runtime fault diagnosis such as concurrent error detection and self-recovery, but the area overhead for these technologies is too large to be used in cost-sensitive applications. The main core of this dissertation is the low-cost implementing strategy of runtime fault diagnosis in digital integrated circuits. This dissertation can be divided into four parts:

Part 1 includes a brief introduction to the research background of this project as well as basic concepts of faults in circuits, and an overview to the up-to-date researches on fault-tolerant system.

Part 2 expounds what are reconfigurable systems and what is dynamic partial reconfigurable FPGA and why it is so powerful in implementing fault-tolerant systems.

Part 3 raises the strategy of “Time-division-multiplexing Fault Diagnosis”, as well as the principle, the design flow and features to be trade-off in designing and implementation. The key to TFD is to share one checker space between several different checkers in time domain based on the technology of dynamic partial reconfiguration of FPGA.

Part 4 gives a demonstration on how to implement TFD in practical applications. Besides, the feasibility, area overhead and fault detection latency is also studied.

Keywords: Time-division-multiplexing Fault Diagnosis，Concurrent Error Detection，Area Overhead，Fault Detection Latency，Dynamic Partial Reconfiguration

目 录

附录 1 图片目录

附录 2 表格目录

第一章 引言

1.1 研究背景

自1958年集成电路发明以来，集成电路产业的发展规律基本上符合摩尔定律的预言，即每隔18到24个月，单位面积芯片上集成的晶体管数目增加一倍，器件特征尺寸缩小为原来的70%。据此推断，在21世纪前半叶，集成电路产业仍将以器件特征尺寸不断缩小的硅基CMOS工艺技术为主流。目前，65nm工艺已经成为主流技术，45nm工艺也已经在高端产品中得到推广[1]。

CMOS 工艺的进步对集成电路芯片的影响直接的体现在以下两个方面。首先，由于器件特征尺寸减小，CMOS 晶体管的本征扩散电容相应减小，进而导致基本逻辑门的本征延时相应减小，于是对于相同的电路设计，采用较先进 CMOS 工艺的芯片工作频率更高。其次，由于单个 CMOS 晶体管的尺寸缩小，在相同面积的芯片上可以集成更多的晶体管；但从另一角度上说，由于器件电源电压的减小速度比不上器件特征尺寸的减小速度，器件的功率密度呈现二阶增长的趋势，同时更高的工作频率也意味着更快的开关活动性，使功率密度呈现更快的增长趋势。

然而 CMOS 工艺的进步对于芯片设计和测试的影响并不仅限于此。长期以来，芯片设计者在分析 CMOS 晶体管和逻辑门的特性时采用大量的一阶近似以简化器件模型，忽略了很多次要参数。随着 CMOS 器件特征尺寸不断缩小，晶体管中的某些二阶效应对器件特性的影响越来越明显，导致传统器件模型的分析结果与晶体管的实际工作状态之间的偏差过大。因此必须采用更加精确的模型，对这些二阶效应——如短沟道效应、热载流子效应、CMOS 闩锁效应、强场效应、量子效应、寄生参数和工艺参数涨落等——加以考虑。复杂的器件模型不仅难以分析，并且导致器件工作状态的不确定性变大，给芯片设计和芯片测试带来了巨大的挑战。

另一方面，由于系统集成度越来越高，芯片中的互联结构也越来越复杂，布线延迟在一条时序路径中所占的比例也越来越高。更重要的是，在 GHz 以上频率的高速电路中，互连线开始呈现出传输线特性，严重威胁信号电平的正确传输。同时，互联

结构的分布式电容和分布式电阻对高速电路信号完整性的影响也日益彰显，逐渐成为影响电路最终性能的重要因素。对于芯片设计者和 EDA 工具来说，处理这些日益复杂的器件和互联线模型进而设计可靠的电路变得越来越困难。

与此同时，随着电路规模和复杂度增大，芯片测试所需要的测试模式呈几何级数增涨，所需要的测试手段也更加复杂，从而导致测试周期明显变长，成本增加。迫于成本和市场投放时间（Time-to-market）的压力，芯片生产厂商并不能对每一块芯片均做可靠性测试，客观上导致了芯片的高故障率。例如，未经过高温老化测试的芯片与通过了 150 摄氏度高温老化测试的芯片相比，因为电路故障导致初期损坏的几率高出 1000 倍。

此外，芯片功耗密度增大容易导致芯片局部过热，诱发更多的现场故障（In Field Failure），导致芯片的平均无故障时间（Mean Time To Failure, MTTF）进一步下降。这类故障中最典型的就是“软错误（Soft Error）”。制造芯片的材料中包含放射性元素，当某个放射性原子衰变释放出的带正电荷的 α 射线撞击某个 CMOS 晶体管时，可能导致被撞击的晶体管切换到相反的开关状态，导致逻辑电路出错，这个过程被称为“单粒子翻转（Single Event Upset, SEU）”，也就是所谓的软错误。

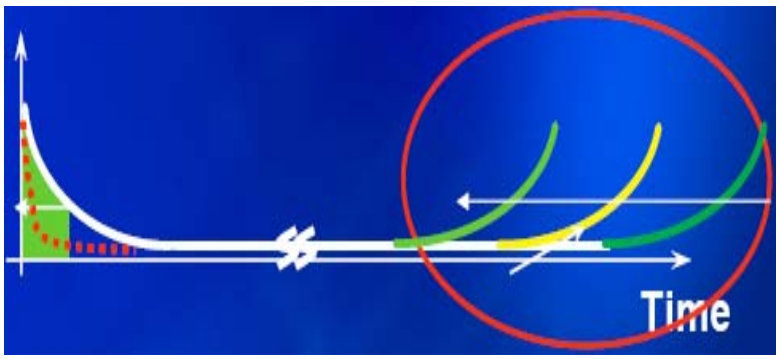

图 1 芯片故障率的时间分布曲线
Fig 1 Chip Failure Rate vs Time

上述问题导致数字集成电路芯片的可靠性严重下降。图 1 来自 Intel 公司的一份统计报告，U 型曲线代表芯片在它的一个生命周期内不同时间段的损坏率，从中可以看到芯片故障率在其生命周期内呈现整体性的上升趋势。曲线左端的弧度增大代表芯片的初期损坏率（Infant Mortality）提高，曲线右端向左平移代表芯片老化加剧用坏

（Worn-out）时间提前，而曲线中段向上平移代表了更多的现场故障。统计数据显示，微软XBOX360设备中的集成电路芯片在三年内出现故障导致无法正常工作的几率是16.4%，而所有消费类电子产品的这一数字大约为15%。

可以预见，随着工艺的进步，传统的电路设计、验证和芯片测试方法必将表现出更大的局限性，芯片的可靠性问题将更加严重。因此，深亚微米工艺下数字集成电路的可靠性设计已经成为一个亟待改进的问题。本论文从电路容错设计的角度，对数字集成电路的可靠性设计做了一定的分析和研究。

1.2 电路故障的基本概念

如果数字电路中的某个器件、模块或者整个系统存在物理缺陷，导致该器件、模块或者系统不能实现设计规范中定义的逻辑功能，则认为发生了故障。在大规模数字集成电路中，故障的种类多种多样，特性复杂，其来源主要包含以下三个方面：

- 设计错误，主要指由于硬件描述语言（Hardware Description Language, HDL）或原理图（Schematic）设计不合理导致的电路功能错误，如竞争冒险、毛刺和亚稳态等。由于电路复杂度越来越高，设计错误出现的几率急剧增加，传统的验证方法已经不足以检测和消除全部设计错误。设计错误通常导致芯片的初期损坏。
- 随机缺陷，主要指芯片掩膜过程中由于工艺偏差导致金属互联结构的局部出现多余或者缺失，进而导致短路故障或开路故障。由于器件和互连线尺寸越来越小，版图密度越来越高，此类缺陷发生的几率也越来越高。而庞大的测试模式数目和高昂的测试成本，导致芯片厂商无法确保随机缺陷在测试过程中被发现。随机缺陷是可能导致芯片初期损坏或出现现场故障的因素之一。
- 非确定性因素，主要指影响芯片工作过程状态的各种环境因素，例如：由于器件的电源电压随着特征尺寸减小而逐渐降低，噪声容限随之减小，受串扰和噪声影响而表现出错误的开关状态的几率也随之上升；在小尺寸器件中，工艺涨落带来了不可忽视的器件参数偏差；不均衡的功率密度分布造成芯片内出现温度梯度，导致器件参数随温度的漂移无法估计；寄生参数和软错误也都是芯片工作中的非确定性因素。非确定性因素通常是芯片出现现场故障的原因。

根据故障的活跃时间，可以将故障分为永久性故障（Permanent Fault）、瞬态故

障（Transient Fault）和间歇性故障（Intermittent Fault）。永久性故障的活跃时间是可以预料的，可以归纳为几个简单的模型：固定（Stuck-at）故障、桥接（Bridge）故障和时延（delay）故障[2]。固定故障是指对于电路中的某一节点，无论驱动此节点的信号如何变化，节点逻辑值始终保持为“0”或“1”不变。桥接故障是指某两根互连线之间被错误的短路，导致此互连线上的信号受线与（Wired-AND）或线或（Wired-OR）的驱动。时延故障是指信号在通过某些逻辑门或者逻辑路径时，延时发生变化，导致时序不能收敛。瞬态故障和间歇性故障的活跃时间通常是不可预料的，仅遵从一定的概率分布，无法用若干有限模型简单描述。

如果一个电路系统能够在发生不可预知的硬件或者软件错误时保持正常的工作状态，则可以说此电路系统具有容错性能。本文主要关注电路系统中针对硬件错误，或者说故障的容错设计。目前经过实验检验的电路容错设计方法可以分为静态容错设计和动态容错设计两种，本文将在下一节中分别具体介绍，并对两者的优劣做简单的比较。

1.3 国内外研究现状

1.3.1 静态容错设计

电路的静态容错设计是指在系统中的某一部分A增加某种特殊的电路结构B，当A部分出现故障时，B部分能够保证传输给系统其他部分的信号是正确的，故障不会传播到系统中正确工作的部分，因而可以有效的屏蔽故障。实现容错结构B通常需要在电路中引入冗余结构，因此这种容错设计方法也成为静态冗余法。根据冗余资源的存在形式不同，静态冗余法又分为时间冗余法、空间冗余法、信息冗余法和组合冗余法。顾名思义，组合冗余法是通过组合使用其他三种方法达到更高的系统可靠性，因此下文只对其他三种方法做简单介绍。

对于系统中要求较高可靠性的模块A，空间冗余法需要在电路中增加A的多个冗余处理模块及相应的多数表决电路，根据表决结果决定输出值。显然冗余处理模块需要占据额外的面积，所以这种方法是以芯片面积为代价，换取系统可靠性的提升。由于多个冗余模块同时出现故障的概率远小于模块A发生故障的概率，因此当少数冗余模块发生错误时，多数表决器仍能输出正确的结果，从而保证了较高的系统可靠性。

空间冗余设计的实现不受电路抽象级的约束，可以在单个逻辑门上实现，也可以是模块级或者系统级实现。

空间冗余设计的典型方案是三模冗余法（Triple Module Redundancy, TMR）[12]，如图2所示。三模冗余法利用三个并行的功能相同的模块实现容错设计，以提高系统的鲁棒性。当三个模块的输出不同时，多数表决器自动选择占多数的输出作为最终输出。假设每个模块出现故障的概率为10^{-6}，则采用三模冗余设计的系统出现故障的概率为10^{-12}。三模冗余法的主要缺点是系统的鲁棒性过于依赖多数判决器本身的鲁棒性，若这一部分出现故障，整个系统均有可能出错，文献[13]中提出了一种针对性的改进方案。另一方面三模冗余法不能屏蔽两个或多个冗余模块同时出现故障的情况，改进的N模冗余法[14]则将系统出现故障的概率进一步降低到了$10^{-6(N+1)}$。

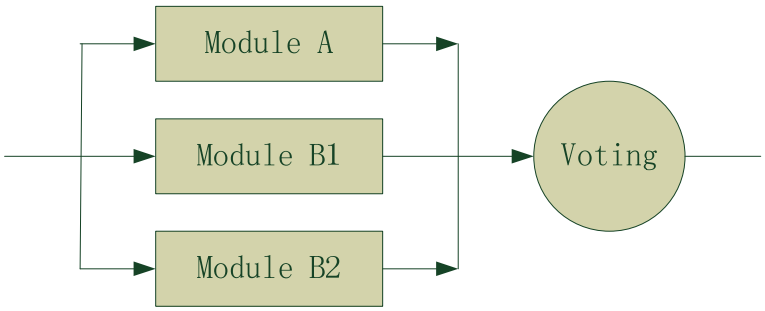

图 2 三模冗余电路示意图
Fig 2 Triple Module Redundancy Scheme

时间冗余法的基本原则是多次重复利用模块A执行逻辑功能，并将每次得到的输出数据提交给多数表决器以求得到正确的结果，从而达到容错的目的。同空间冗余法相比，这种方法需要的冗余逻辑很少，因此芯片的面积代价很低；但由于需要进行多次计算才能得到一次输出，系统的数据处理能力严重下降，因此是以系统性能为代价，换取可靠性的提升。

由于多次重复运算在时间域上具有良好的平滑效果，这种方法能够很好的屏蔽电路的瞬态故障；但对于电路的永久性故障，由于每次输出均已经受到故障的影响，因此无法起到屏蔽的作用；同样这种方法对于间歇性故障的屏蔽能力也是不可靠的。

信息冗余法采用在存储或传输的数据中加入额外的一个或多个比特数据，用于检

验数据的完整性，通常也被称为错误校验编码（Error Correcting Codes, ECC）方法。ECC方法一般分为独立编码方法和非独立编码方法，其中独立编码在加入附加纠错码后无需考虑有用数据本身，可以直接进行下一步的处理；而非独立编码在进一步处理之前需要利用额外的解码电路恢复有用数据，电路实现过于繁琐，所以其应用不如独立编码广泛。

奇偶校验编码[15]是最简单的独立编码信息冗余容错方法，常见于存储器容错设计中。这种方法对电路中的每个数据字加入一个冗余比特，编码原则是使数据字中逻辑“1”的位数为奇数（Odd-parity）或者偶数（Even-parity），通过分别计算存储模块各行和各列中逻辑“1”的个数为基数还是偶数，确定故障的具体位置。与之类似的是汉明（Hamming）校验编码[15]和双轨（Dual Rail）校验编码[16]，这两种方法的编码规则更加复杂，需要的冗余比特也更多，因其具有良好的抗噪声和串扰的性能，常见于数据通路的容错设计中。

以上三种编码方式的共同问题是只能检测和纠正一个比特的错误或者说单个故障，而在深亚微米和纳米级CMOS工艺下，故障往往是连续出现的。Reed-Solomon校验编码[17]是一种可以检测和纠正多个错误的编码方式，对于一个包含N位冗余的数据字，这种方法最多可以检测和纠正N/2个故障。信息冗余容错电路实现起来较为复杂，同时要求电路有规范的数据结构，因此只适用于阵列结构的电路系统如存储器，可编程阵列逻辑（Programmable Array Logic, PAL），现场可编程逻辑门阵列（Field Programmable Gate Array, FPGA）等。

1.3.2 动态容错设计

静态容错设计只是被动的屏蔽故障，其电路实现相对复杂，代价过大，并且其可靠性或者依赖于特殊的电路结构，或者受限于故障的种类。与之相对的动态容错设计可以主动的检测、定位和修复故障，从而可以更好的克服电路中的永久性错误和多个连续错误。

在动态容错设计中，所有其他操作均依赖于故障检测的结果，因而故障检测是最关键的部分，又可以分为离线（Offline）检测和在线（Online）检测两种方法。离线检测周期性的中断被测电路的正常工作，采用预先定义好的测试向量对被测电路执行一次内建自测试（Built In Self Test, BIST），然后恢复被测电路的工作状态。离线测

试的典型应用是文献[4][5][6]中提出的可重配置FPGA的内建自测试方法——Roving STAR[4]。这一方法不需要额外消耗FPGA逻辑资源，而是构造BIST电路使之恰好能够完全利用FPGA的单个可配置逻辑块（Configurable Logic Block, CLB），利用FPGA的可重配置特性，使用此BIST电路按照行列顺序周期往复地遍历FPGA的每个CLB，以实现整个FPGA的故障检测，如图3所示。

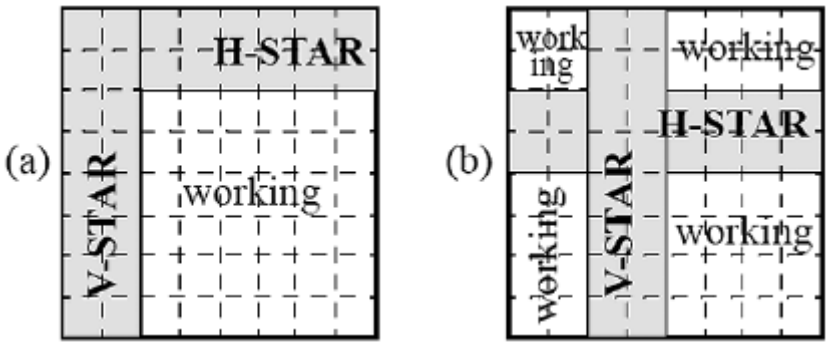

图 3 Roving STARs: (a)初始位置 (b)漫游中的某一位置[4]
Fig 3 Roving STARs: (a) initial position and (b) during roving [4]

BIST方法仅需要很少的测试向量和很少的冗余面积就能够达到非常高的故障覆盖率，但无法检测到电路工作时出现的瞬态故障。BIST在各个CLB之间漫游时，被测系统的配置需要同时切换，会中断正常电路的运行。同时作为一个细粒度（Fine Grain）算法，其算法复杂度过高，不仅需要一个高性能高可靠性的运算控制单元，而且导致错误检出延时过长，不适用于现场故障检测。

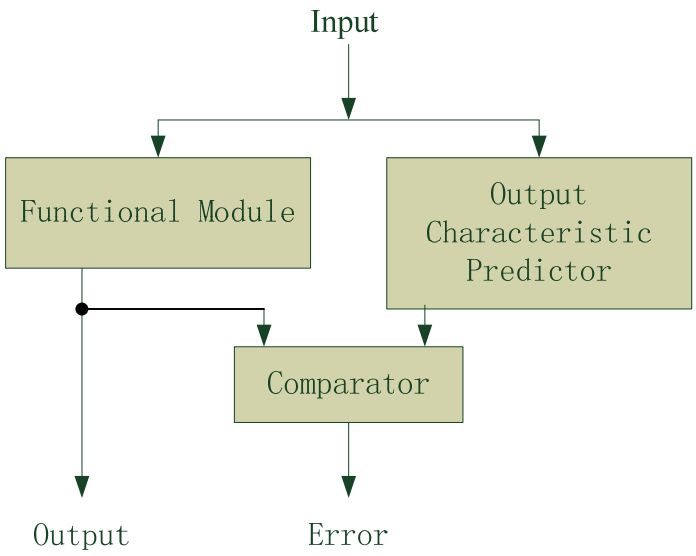

图 4 并发故障检测示意图

Fig 4 Concurrent Error Detection Scheme

在线检测又叫并发错误检测（Concurrent Error Detection, CED），其原理如图4所示。假设对于某个输入向量*i*，待测系统*f*的输出为*f(i)*，并发故障检测利用额外的冗余逻辑计算相同输入向量下待测系统输出的某种特征*p(i)*，然后与由*f(i)*得到的同一特征比较，若两者不符合，则输出一个表示信号表示检测到故障。这里提到的特征，可以是*f(i)*本身，*f(i)*的奇偶性，*f(i)*中逻辑“1”或逻辑“0”或反转的个数，等等。

在检测到系统中出现故障之后，需要对出现故障的逻辑进行修复，目前常用的故障修复方法是自动重复请求（Automatic Repeat of Request, ARQ）方法和重构修复方法。ARQ方法不改变原有电路系统，而是利用额外的时间重复发生故障的数据处理过程，适用于修复瞬态故障和间歇性故障，但显然ARQ无法修复永久性故障。重构修复法把出错部分的正常逻辑功能重新配置到空闲的后备逻辑资源、模块或芯片上，并代替出错的电路结构以保证系统功能正常，适用于修复间歇性故障和永久性故障。由于不同功能模块需要插入对应的备用模块，因此通过增加备用模块的方式会很容易导致系统规模扩大，占用大量的芯片面积，会降低系统的工作效率同时增加系统的功耗。

1.3.3 降低容错设计的代价

由前面的介绍可以看出，容错设计的额外面积开销或者时间开销非常大，使得容错设计很难应用在对成本敏感（Cost Sensitive）的芯片中。例如，根据文献[10]对MCNC91基准电路组（Benchmark Suite）中一些典型的组合逻辑电路的测试结果，一个简单的一位奇偶校验电路将导致平均77%的面积代价和19%的性能损失。而对时序逻辑电路来说，即使是最优化的并发错误检测模块也会造成大约100%的面积代价[11]。之前的研究提出了一些旨在降低容错设计代价的故障检测和恢复策略。

文献[7]提出一种基于分区映射的在线故障诊断和恢复方法。首先将FPGA的逻辑资源平均划分为N个区域，并将整个电路设计划分为N-1块，分别映射到FPGA的一个区域中，如图5所示。预先编译N种布局方案并将得到的配置文件储存在ROM中，若系统输出错误，即依次调用存储的配置文件执行全局重配置，直至找到一个正确的配置。这种方法的面积代价约为1/N，故障恢复的时间代价约为全局重构时间的N/2倍。这种方法的问题在于故障无法在传播至系统最终输出之前被检测到，也无法应付多个分区同时出现故障的情况。更重要的是，当N较大时，需要很大的ROM用于存储配置文件。文献[8]对分区策略做了改进，一定程度上改善了上述问题，但又导致系统的布局布线复杂度显著增加，设计复杂度提高。

图 5 左：初始配置；右：一种备用配置
Fig 5 Left: Initial Configuration; Right: An Alternative Configuration

文献[9]提出了一种基于并发错误检测（Concurrent Error Detection, CED）的方法。这种方法沿用了分区映射的思路，但对每个分区的电路模块采用检测模块实时监控其状态，并将每个分区的局部配置文件存储在ROM中，如图6所示。若某一分区检测到错误，则执行局部重构，利用空闲分区实现此分区的逻辑功能。此方法对永久性故障、

间歇性故障和瞬态故障都有良好的检测能力，但多个并发错误检测模块会导致非常大的额外面积开销（超过110%）和相当程度的性能损失，导致成本成倍增加。

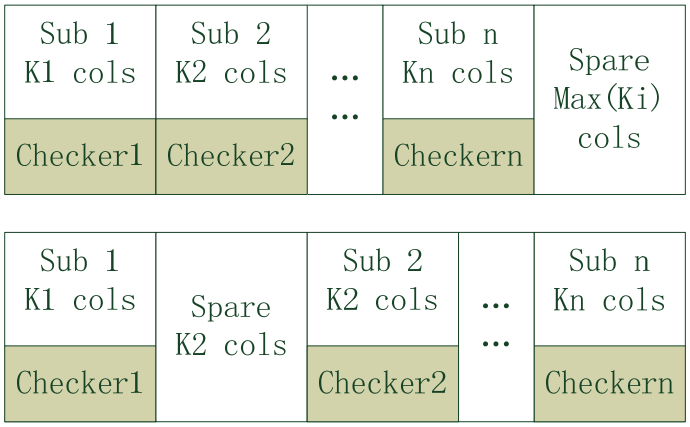

图 6 上：初始配置；下：分区 2 报错后的配置

Fig 6 Up: Initial Configuration; Down: Alternative Configuration after Error detected

1.4 本文的工作

上一节提到的几种数字集成电路容错设计方法或者以额外的面积开销，或者以额外的时间开销造成的性能损失为代价，换取系统稳定性的提高。在绝大多数应用如消费电子、汽车电子及通信设备中，芯片正常工作状态的频繁中断将造成系统可用性降低，而成本和功耗也是需要考虑的重要因素，因此需要一种面积开销较小的系统运行时的故障检测与恢复策略。

本文针对以上应用，提出一种新的基于并发错误检测的容错系统实现方法——称为分时故障诊断（Time Division Multiplexing Error Diagnosis）。分时故障诊断继承了文献[9]的分块思想，但改进了其故障检测模块的分配策略，在系统中加入一个硬件可重构单元，利用其可重构特性在时间域上循环交替的实现针对不同功能模块的并发故障检测模块，以实现电路故障的运行时检测和修复。这种方法在不影响系统性能的前提下，以平均故障检测时延的适当增加为代价，换取额外面积开销的明显下降，同时保证故障检测能力的大致持平。分时故障诊断方法可以应用于基于FPGA的系统，或是内嵌了FPGA核的片上系统/系统级封装（SOC/SIP）如Atmel's FPSLIC II SoC，甚至是包含多个处理器核的片上系统（MPSOC）。为便于研究，本文利用Xilinx FPGA

的动态局部重构特性，在单个FPGA片内对此系统的可行性和性能做了初步验证。

论文的第二章首先介绍了FPGA的逻辑结构和互联结构，重点是基于SRAM的FPGA的岛式结构；进而介绍了可重构系统的基本概念，以及典型的FPGA重构模型；最后总结了Xilinx FPGA的动态局部重构特性。第三章提出了基于并发故障检测和运行时重构的分时故障诊断的基本原则；详细分析了基于分时故障诊断的容错设计流程，并对此流程中需要考虑的因素做了定性的分析；系统的提出了一个概率模型，分析了分时故障诊断对于永久性故障、间歇性故障和瞬态故障的检测能力。第四章在分时故障诊断的框架下，具体讨论故障检测模块的实现，分析了目前常用的检测模块算法，并重点研究了基于异构复制电路的算法，提出了检测模块的寄存器状态同步问题并给出了可行的解决方案。第五章详细介绍了分时故障诊断应用在一个视频监控系统上的实现方案，证明了此方法的可行性并对其实际性能做了初步的测试和分析。第六章是对全文的总结和概括，回顾了本文的研究成果和创新点，分析了不足，并对进一步的工作提出了展望。

第二章 FPGA 重构技术

2.1 FPGA 结构介绍

FPGA是基于通过可编程互联结构连接的可配置逻辑块矩阵的可编程半导体器件，它是在可编程逻辑阵列（PLA）、可编程阵列逻辑（PAL）、门阵列逻辑（GAL）、可擦除逻辑器件（EPLD）等器件的基础上进一步发展的产物。虽然有基于反熔丝（Anti-fuse）的一次可编程FPGA，目前的主流仍然是基于SRAM的可重编程FPGA。基于FPGA的设计无需多次流片（Tape-out），因而具有非常低的不可重用设计成本和非常短的开发周期，并且便于调试和修改电路。作为专用集成电路（Application Specific Integrated Circuits, ASIC）领域中的一种半定制电路，FPGA解决了全定制电路的不足，又提供了比基于微处理器的软件设计强大得多的性能，因而在通信、网络、仪器、高性能计算、工业控制、军事和航空航天等众多领域得到了广泛的应用。

基于SRAM的FPGA主要包含三类资源：可配置逻辑块（CLB）、输入输出缓冲（IOB）和互联网格（Interconnection Grid），部分FPGA同时内嵌RAM或者一些常用的硬核模块。基于SRAM的FPGA根据其资源组织结构可以分为三类：晶格结构（Cell-based）、岛式结构（Island-style）和层次结构，其主要区别在于：逻辑单元能够实现的逻辑功能的多少，互联网络的组织结构。

在晶格结构中，每个CLB仅包含数个两输入基本逻辑门，如NAND、XOR等。CLB之间的数据传输通常通过相邻CLB之间的布线通道完成，很少有跨越多个CLB的互联线资源。这种互联结构适合实现规整的电路结构，如二维滤波矩阵，但对于不规则电路，其效率很低。这种结构的典型代表是Atmel公司的FPGA。

岛式结构又称平面结构，是科研人员常用的可编程器件模型。岛式结构中，每个CLB包含数个查找表（Lookup Table, LUT）和其他附加逻辑电路，可以完成组合逻辑和同步时序逻辑电路功能。互联网格包括了不同跨度的连线：短线资源丰富，较为灵活，但构成较长互联路径时需要通过多个开关盒（Switch Boxes），会造成较大的布线

延时，主要用于局部布线；长线绕开了开关盒因而有较小的布线延时，但不够灵活且资源有限，通常用于两个功能模块之间的通信。图7显示了Xilinx Virtex II FPGA的岛式结构。

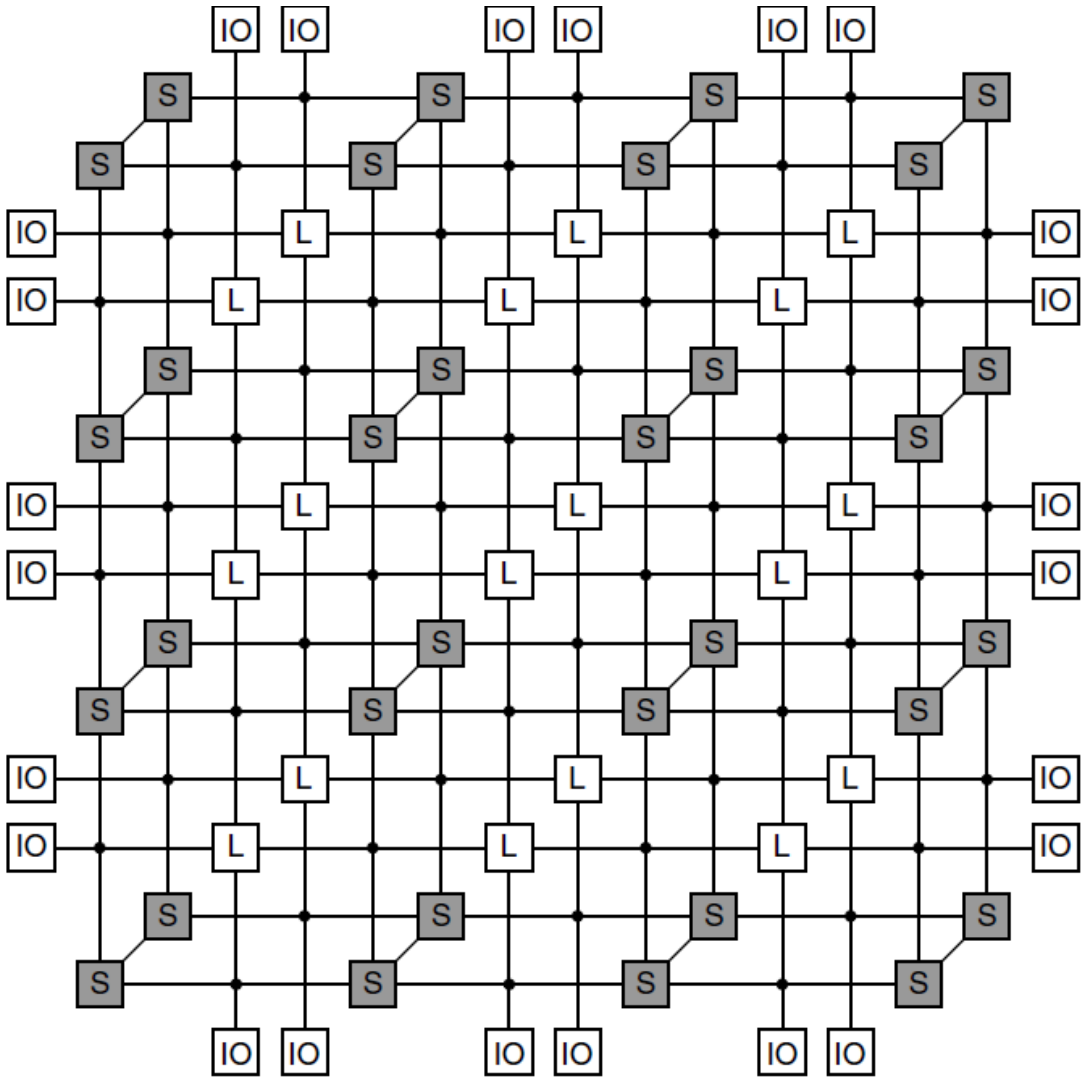

图 7 FPGA 的岛式结构示意图（两层），L 代表 CLB，S 代表开关盒

Fig 7 Island Style Scheme of FPGA, in which “L” stands for CLBs, while “S” for switch boxes

层次结构由岛式结构发展而来。在层次结构中，每个CLB不但包含更多的LUT和触发器，同时也具有内部布线资源，从而可以实现更复杂的逻辑功能。互联网格中包含更加丰富的、有多种长度和用途的布线通道，通常还有专用的全局时钟树。层次结构几乎能够实现任何功能的电路尤其是高扇出电路，是目前商用FPGA的主流结构。

2.2 可重构系统

可重构性是指系统功能根据需求变化相应改变的能力，也称为可编程性。可重构系统包含至少一个可重构硬件单元，通过在时间域上修改可重构单元的配置实现硬件资源的重用。

系统重构技术可以分为静态重构和动态重构。静态重构是指系统的逻辑功能静态重载，即FPGA芯片从复位状态开始，在外部逻辑的控制下，根据外部存储器中的配置数据，重新组合芯片内的各个区域的各种硬件资源，实现逻辑功能的改变。静态重构需要几毫秒到几十毫秒的时间用于芯片复位和配置信息加载，这一时间称为重构时隙。

动态重构是指在系统运行期间，由芯片外部或者内部的逻辑控制，在现有硬件配置的基础上，修改全部或部分逻辑资源和互联网格的配置，以改变系统逻辑功能。动态重构没有芯片复位过程，因此重构之前电路的工作状态——例如计算的中间数据——能够延续到新的电路中。

根据系统逻辑功能的改变发生在芯片的全部区域还是部分区域，重构可以分为全局重构和局部重构。静态重构只能是全局重构，而动态重构既可以是全局重构也可以是局部重构。动态局部重构可以在系统内部逻辑的控制下，根据需要改变部分区域的硬件配置，在此过程中，系统其余部分可以继续工作而不受重构的影响。利用动态局部重构技术，硬件配置可以像软件一样动态调用或修改，从而可以将空间分布的硬件逻辑分化到时间域上，只需要很少的硬件资源，就能实现更多的逻辑功能，达到降低系统的面积、成本和功耗的目的。这种方式减少了重构时需要修改的硬件配置，从而可以大大缩短重构时隙，同时便于重构后的逻辑使用重构前得到的数据。

目前系统重构的方式主要有三种模型：单上下文（Single-context）模型[19]、多上下文（Multi-context）模型[20]和PRTR（Partial Runtime Reconfiguration）模型[21]，如图8所示。

在单上下文模型中，FPGA配置数据缓存的结构类似移位寄存器，配置数据必须按顺序依次存取，因此只能实现全局重构。这意味着即使只需要修改FPGA的一小部分配置，也不得不重写整个芯片的配置。对于运行时重构来说，单上下文模型带来的

开销过大，目前只有Altera FPGA芯片仍在沿用。

多上下文模型是对单上下文模型的改进，其主要思路是准备多个配置文件存储在不同的地址中，未激活的配置文件可随时被重写而不干扰系统运行，同时也降低了系统重构时隙。

在PRTR模型中，FPGA配置数据缓存的结构类似RAM，可以根据其行列地址随机存取任意的数据单元，修改指定位置的硬件配置，因此可以支持动态局部重构，目前在Xilinx和Atmel FPGA中广泛应用。

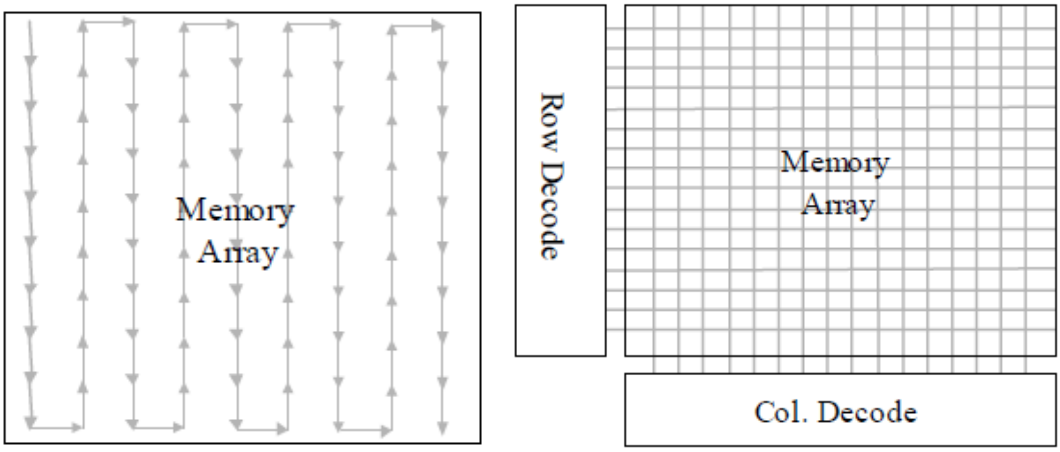

图 8 FPGA 配置模型。左：单上下文模型；右：PRTR 模型

Fig 8 Models of FPGA Configuration, the left for single context and the right for PRTR

2.3 Xilinx 动态局部重构

Xilinx FPGA支持两种动态局部重构模式：基于模块（Module-based）的重构[23]和基于差异（Difference-based）的重构[22]。

基于差异的重构比较重构前后电路的差别，产生一个只描述重构前后硬件配置差别的重构比特流（Bitstream）。这样的比特流比全局重构比特流小很多，因而重构时隙很小，模块功能能够迅速切换。但基于差异的重构只适合对电路做细微的修改，如改变某个LUT的布尔函数或是某个Block RAM存储的数据，而不适合功能模块的交替更换。

基于模块的局部重构将整个系统划分成多个功能模块，模块与模块之间是相互独立的。其中可以实现局部重构的区域称为可重构模块，其余的区域称为静态模块。可重构模块必须具有以下属性：

- 可重构模块的宽度必须是4个Slice的整数倍，最大可以是整个器件的宽度。
- 可重构模块的高度必须是16个CLB的整数倍，最大可以是整个器件的高度。
- 可重构模块边界的水平坐标必须是4个Slice的整数倍，例如左边界可以放置在第0个，第4个，第8个……Slice的位置。
- 可重构区域内包含的所有硬件资源都被看作是可重构模块的一部分，包括Slice，TBUF，Block RAM，IOB，乘法器和全部布线资源。
- 可重构模块不包括任何定时逻辑如BUFGMUX和CLKIOB，因为时钟在比特流中占据单独的帧。
- 可重构模块的边界一经确定即不能改变，任何一个可重构模块所占据的区域都是固定的。
- 可重构模块和其他模块之间，包括可重构模块和静态模块之间以及两个可重构模块之间，都需要借助特殊的总线宏（Macro Bus）实现通信，如图9[22]所示。

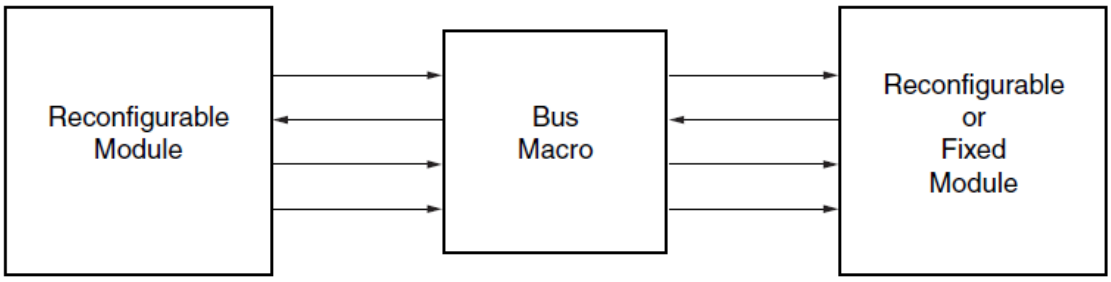

图 9 模块间通信中总线宏的使用

Fig 9 Using Bus Macro in inter-module communication

- 可重构系统的设计必须确保当局部重构发生时，静态区域的运行不依赖于可重构区域的状态，因此需要精确的握手逻辑传递可重构模块能否正常运转的信息。
- 当局部重构发生后，可重构模块内部的存储单元能够保留重构的状态，便于重构

后的模块利用重构前的数据。

图10[22]是一个包含两个可重构模块的动态局部可重构系统的布局示意图，可以看到可重构模块和其他模块之间都是通过总线宏来通信的。

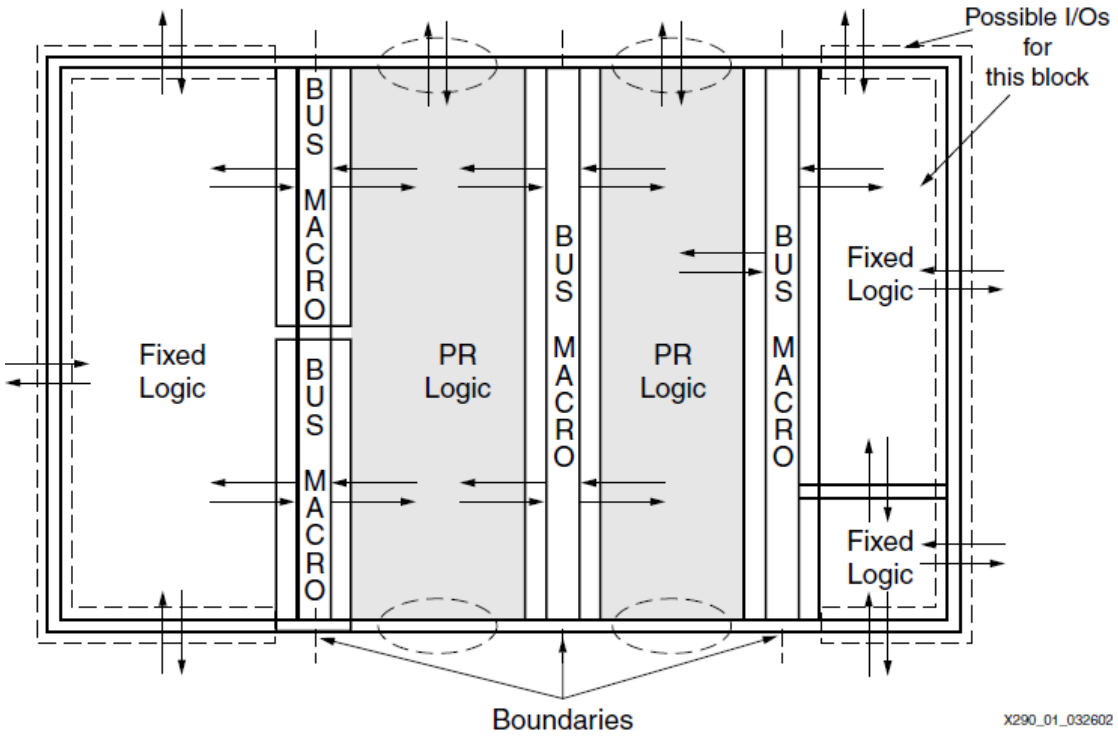

图 10 具有两个可重构模块的设计布局图
Fig 10 Design Layout with 2 Reconfigurable Modules

局部重构会改变可重构区域内的布线结构。为了避免重构后可重构模块和其他模块之间的数据传输出错，需要保证穿过可重构模块边界的布线资源在重构前后保持不变。总线宏是一个预编译的宏模块，不会随着电路重新编译而发生变化，被用于确定两个模块之间的布线结构，保证数据传输的可靠性。

Xilinx提供的总线宏由三态缓冲器（TBUF）及长线连接而成，允许信息的半双工传递，每一位数据使用一个TBUF和一根贯穿整个器件的长线。以Virtex II器件为例，FPGA的每一行支持一个4比特的总线宏，如图11所示。总线宏的位置精确的跨骑在模块A和模块B之间，其中四栅三态缓冲器在A内，另外四栅三态缓冲器在B内。两个模

块之间的通信就是通过带有三台缓冲器的长线来保证的。总线宏允许从模块A到模块B或者相反方向的数据传输，但对于某一个特定设计，数据传输的方向是确定的。总线宏的数量受FPGA中水平方向上可用的长线资源的数量限制。

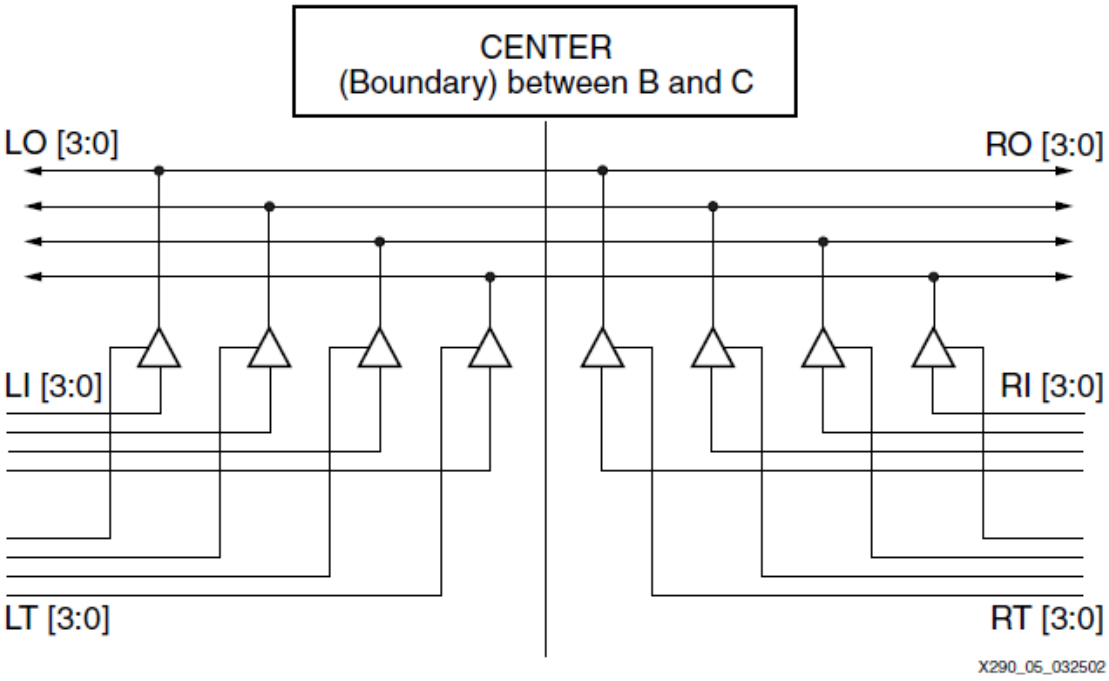

图 11 总线宏的物理实现

Fig 11 Physical Implementation of Bus Macros

2.4 本章小结：动态局部重构应用于可靠系统

芯片级或者板级的容错设计需要一块额外电路板或芯片，在系统的成本、面积、功耗方面都会造成很大的代价。而 FPGA 具有丰富的逻辑资源和良好的可编程、可重构特性，利用其空闲逻辑资源进行片内的冗余容错设计以提高系统可靠性，在硬件资源冗余度上有着巨大优势。基于 SRAM 的 FPGA 的岛式结构具有逻辑资源可编程、互联结构可编程和 IO 接口可编程的多重可编程特性，是通过冗余容错设计实现可靠系统的最佳硬件平台。Xilinx FPGA 的动态局部重构特性能够显著降低 FPGA 的配置数据量和重构时隙，将其应用到冗余容错设计当中，可以同时节省系统的存储器空间、系统配置时间和配置数据通道位宽，因此局部可重构 FPGA 是高可靠性系统的最佳实

现平台。

第三章 分时故障诊断

3.1 基本原则

在本文中，原始设计被定义为单纯基于系统功能规范（Design Specification）得到的、能实现系统全部逻辑功能的、最小化的电路设计，其中不包含任何基于系统可靠性考虑的冗余容错设计。

正如作者在第一章的末尾提出的那样，分时故障诊断策略首先将电路原始设计按照某种分割原则划分为多个模块，并根据每个模块的特点，针对性的设计其并发故障检测模块。

整个设计的硬件实现平台必须是可重构的，也就是说，系统硬件资源中必须包含一个或者多个现场可编程逻辑块。这些逻辑块与实现系统正常逻辑功能无关，而是被用于实现之前设计的并发故障检测模块——此时被称为“检测区域”，或是作为备用的空闲硬件资源用于在必要时实现故障修复。

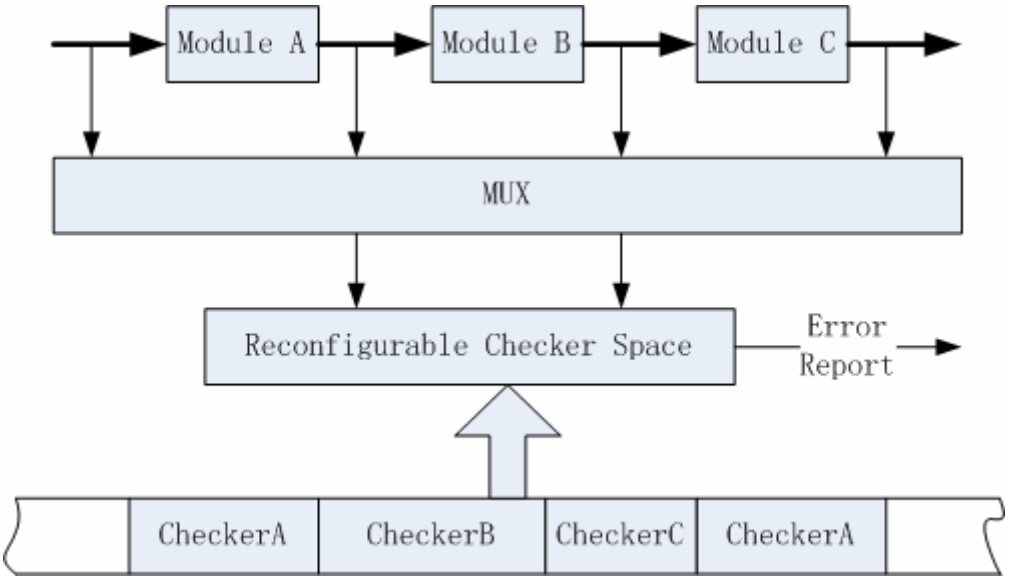

图 12 分时检测策略示意图

Fig 12 Time-division-multiplexing Online Diagnosis Scheme

所有模块的检测模块首先被分成几组，每组分别指派到一个检测区域。利用对检测区域的重构，同一组内的各个检测模块以循环交替的方式依次被配置在指定的区域中，每个检测模块在轮换队列中的次序和持续时间取决于重构发生的时间节点，如图12所示。

利用上述分时复用策略，系统在一个周期内依次监视原始设计中的各个模块，保证每个模块每隔一段时间被检测一次。在图12的示例中，原始设计被分割为三个模块：A、B和C，他们各自的并发故障检测模块分别为CK_A、CK_B和CK_C。系统包含一个现场可编程逻辑块CS，作为检测区域。MUX包含一个N选2的多路选择器及相关的布线结构，除了用于数据通路的切换之外，还起到了隔离原始设计区域和CS区域的信号及时钟域的作用。三个检测模块在时间域上周期性地复用CS区域，当CS区域被配置为CK_A时，MUX切换使得模块A两端的数据通路被接入CS区域，若此时模块A出现故障，则有可能被系统检测到。同理，当CS区域被配置为CK_B时，系统开始监控模块B，模块B中出现的故障有可能被系统检测到，而模块A中出现的故障只有当CS区域再次被配置为CK_A时才能被检测。CS区域配置的切换不会导致原始设计区域电路工作的中断。

3.2 与原始设计有关的要素及其定性分析

根据上一节规定的基本原则，在一个原始设计中应用分时故障诊断策略时，以下三个要素需要根据具体的原始设计来折中考虑：模块分割策略，并发故障检测模块的方案和检测区域时分复用的时间片调度策略。

3.2.1 模块分割

这里所说的模块分割策略，不但包含了原始设计的分割方式，同时也包括了检测区域的布局，以及原始设计各模块与检测区域的映射关系。

模块分割的首要问题是分割后模块的数量，以及检测区域的数量。显然，原始设计分割的越精细，每个模块中包含的逻辑门数目越少，相应的检测模块所要占用的逻辑资源和芯片面积越小，在检测模块分组和检测区域分配时的灵活度也更大。但由于

每个模块都需要分别设计检测模块并生成配置文件，模块数量过多将导致用于存放配置文件的存储器太大，同时设计复杂度增加。

其次是原始设计各模块与检测区域的映射关系。映射到同一块检测区域的模块越多，所需要的检测区域的总面积就越小，整个系统的冗余面积也越低。然而，映射到同一块检测区域的模块过多将导致其布线非常复杂，造成额外的布线面积开销。同时，这也会造成每个模块被监控的时间与未被监控的时间之比太小，这将导致瞬态故障更容易逃脱，永久性故障和间歇性故障的检测延时更大。

最后是采用何种算法分割电路。基于电路功能的划分方法和基于RTL结构的划分方法都可以使用，两者各有利弊[24]。基于RTL结构划分电路容易得到大小均匀的模块，映射到检测区域时更加灵活。但这种做法往往把电路的关键路径分割到很多模块中，导致系统时钟周期延长，性能降低。

基于功能划分电路可以使模块之间的连线数量较少，从而降低布线复杂度。同时各个模块输出的特征容易获得，便于设计最优化的故障检测模块。但是这种方法会导致系统消耗的逻辑门数目和占用的面积更大，因为两个功能模块不能通过逻辑结构的共享减少资源消耗，同时控制各个功能模块的大小也比基于结构的划分困难得多。

在分时故障诊断系统中，各个模块的大小可以存在差异，只需要将大小相近的检测模块映射到同一块检测区域中，就可以使检测区域的总面积最小，进而使总的冗余面积代价较低。为了尽量不影响系统性能，降低布线复杂度，可以首先基于功能划分电路，如果存在某个模块与其他模块的大小悬殊，可单独对此模块采用结构化分割的方法。

总的来说，为了使冗余面积代价和性能损失最小，原始设计分割的粒度，检测区域的数量和两者之间的映射关系必须根据具体的原始设计折中考虑；而对于绝大多数设计来说，基于功能的划分比基于结构的划分更有优势。

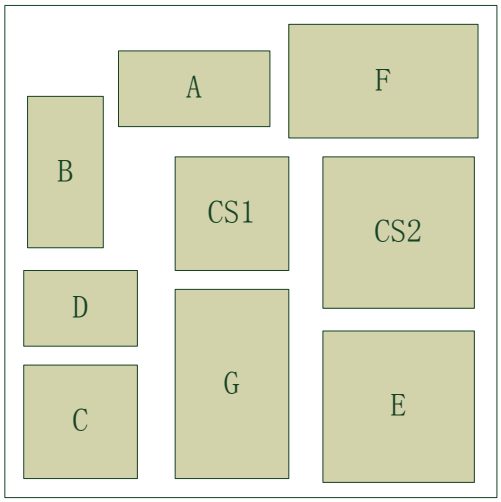

图 13 模块分割示意图
Fig 13 An Example of Partitioning Scheme

上图是分割后的模块布局的一个示例。A-F 代表原始设计分割后得到的七个模块，而 CS1 和 CS2 是两块检测区域。根据七个模块的检测模块的大小，把 CS1 分配给 A-D 四个模块，CS2 分配给 E-G 三个模块。

3.2.2 并发故障检测模块的方案

在分时故障诊断系统中，每个待测模块都有独立的故障检测模块，因此可以针对各个模块的结构和功能特征，并考虑对故障覆盖率和冗余面积开销的要求，分别采用最优化的方案实现。不论采用哪一种方案，都需要预先编译检测模块的配置数据，并将其存放在系统的存储器内，以节省检测区域重构的时间。本文将在第四章中详细介绍并发故障检测模块的实现。

3.2.3 时间片调度策略

系统的时间片调度策略包含两方面的内容：轮转周期和一个轮转周期内每个模块的检测模块被分配的时间片，如图13所示。每个时间片中又包含两部分时间：重构时隙和当前模块真正处于被监控状态下的时间。

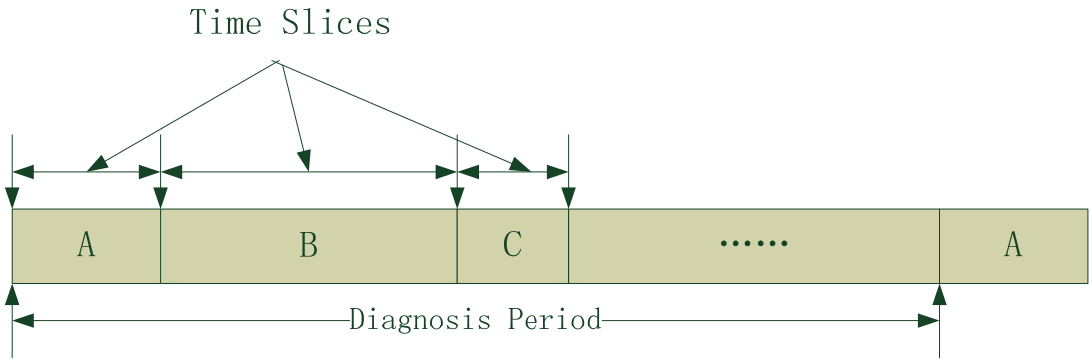

图 14 时间片示意图

Fig 14 Time Slices Queuing Management Scheme

前两节中提到的因素都与系统的硬件配置密切相关，因此一旦在设计阶段确定后就无法修改。而时间片调度通常由嵌入式处理器中运行的软件控制检测区域重构的时间来实现，可以根据需要现场修改。

时间片调度策略是影响平均故障检测时延的重要因素。模块的时间片在一个轮转周期中所占比例越大，此模块的平均故障检测时延越小，瞬态故障逃脱的概率越低。对于已知的容易出错的模块，或者是对系统正常运行极为关键的模块，其时间片占轮转周期的比例应该较大，甚至可以为这类模块分配专用的检测区域。考虑到重构时隙，轮换周期和每个时间片都不能太短，否则大量的时间被浪费在重构时隙上，降低了检测能力和检测效率。

3.2.4 设计流程

综上所述，分时故障诊断的设计流程总结如图15所示。

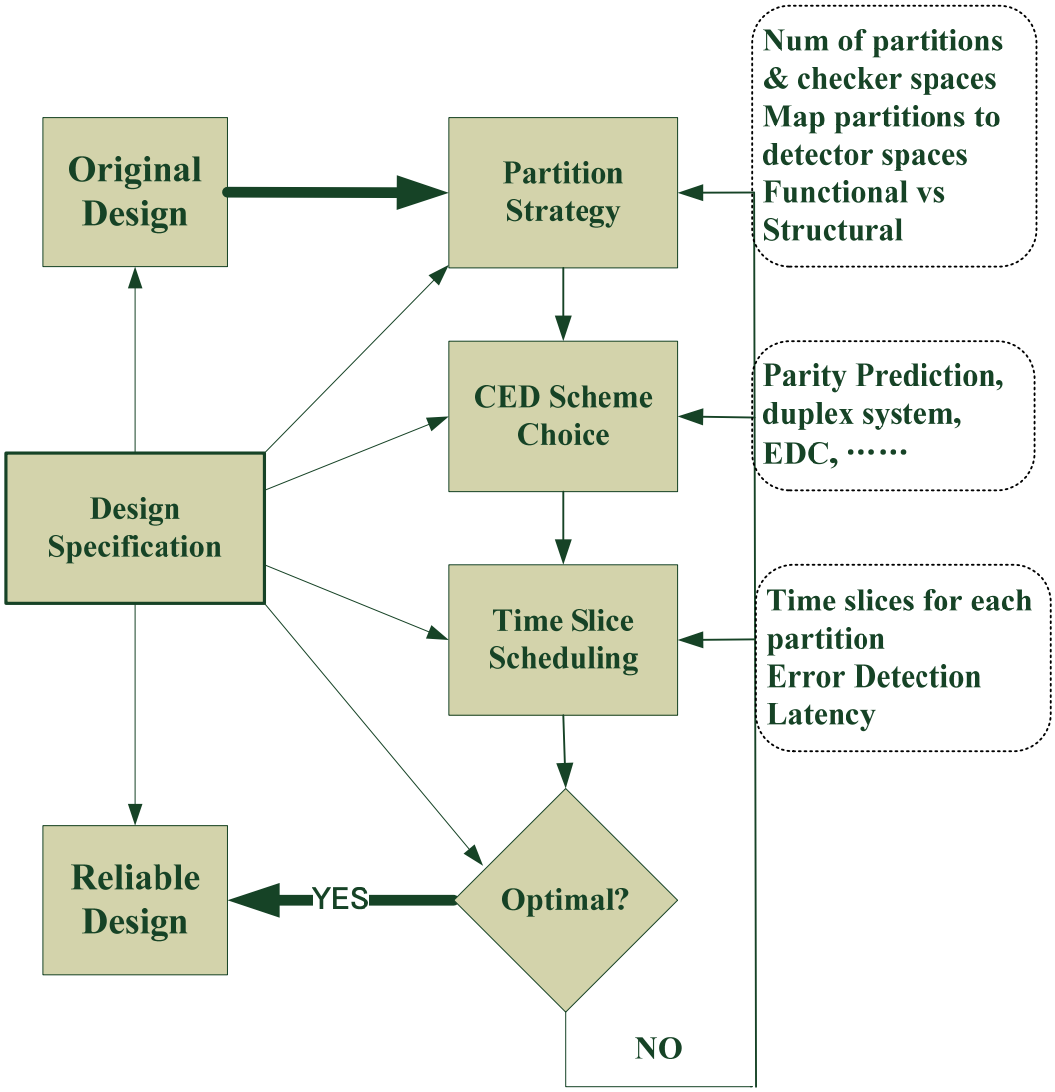

图 15 分时故障诊断设计流程

Fig 15 TDM Error Diagnosis Design Flow

3.3 故障检测能力及代价分析

首先需要明确的是，分时故障诊断既不占用原始系统的运行时间，也不会造成其运行的中断。但分时故障诊断有可能造成系统最高工作频率稍有下降，其原因包括以下两个方面：第一，检测模块周围的接口逻辑增加了布线复杂度，可能造成线网延时的上升；第二，额外的互连线增加了部分逻辑门的扇出系数（Fanout），造成门延时

略微增大。通常这两个因素对系统性能影响很小，并且可以通过仔细的综合和布局布线加以改善，因此在本文中不做考虑。

为了便于与文献[9]中提出的基于Column的并发故障检测方法对比，在本节中建立了一个理想化的概率模型，用于分析分时故障诊断设计的面积代价，以及对于三类故障——永久性故障、间歇性故障和瞬态故障的检测能力，并建立了面积代价和检测能力与设计参数之间的函数关系，以便根据预先确定的容错设计规范估算设计参数。

文献[25]中提出了一个针对间歇性故障的概率模型，把间歇性故障的持续时间按照故障是否对系统运行产生影响分成活跃期和休眠期，所有活跃期的累加作为一次永久性故障，而所有休眠期作为无故障状态。本文借鉴其思想，但把间歇性故障的活跃期作为一系列相互独立的瞬态故障，而休眠期作为两次瞬态故障之间的间隔。这样，永久性故障可以看作第一个活跃期无限长的间歇性故障，而瞬态故障可以看作第一个休眠期无限长的间歇性故障，从而可以用一个统一的模型分析三种故障下分时故障诊断的性能。

为了尽量简化分时故障诊断系统中的不确定因素，本文做如下假设：

- 原始设计可以被分割成面积相等的N_1个模块；
- 每个模块的故障检测模块的面积是相等的；
- 分配给每块检测区域的模块数量是相等的；
- 分配给每个模块的时间片长度是相等的；
- 时间片调度采用Round-robin算法；
- 系统中只可能存在单重故障；

根据以上假设，定义模型中的参数如下（本文中提到的所有关于时间长度的概念和参数均以时钟周期数为单位）：

- 原始设计中每个模块的面积为A_{module}，相应的故障检测模块的面积为$A_{checker}$；
- 系统中检测区域的数量为N_2，复用同一块区域的模块数量为 $N=\frac{N_1}{N_2}$，每块区域的多路选择器接口和额外布线通道的面积为A_{mux}；

- 每个模块分配的时间片长度为T_U，其中重构时隙和被测模块与检测模块的寄存器状态同步花费的时间为T_{RS}，被测模块真正被监控的时间为$T_M=T_U-T_{RS}$；对于Round-robin调度算法，轮转周期为$N*T_U$；
- 间歇性故障的第i个活跃期的长度为$T_{AC}(i)$，第i个休眠期的长度为$T_{SL}(i)$，$T_{AC}(i)$和$T_{SL}(i)$是两个独立同分布的随机变量序列，数学期望分别为E_{AC}和E_{SL}，且所有的$T_{AC}(i)$和$T_{SL}(i)$两两相互独立；
- 如果当前正在被监控的模块中存在故障，对于每个输入向量，故障检测模块能检测到故障的概率为p，p的典型值为10^{-7}；

故障检测模型如图16所示：

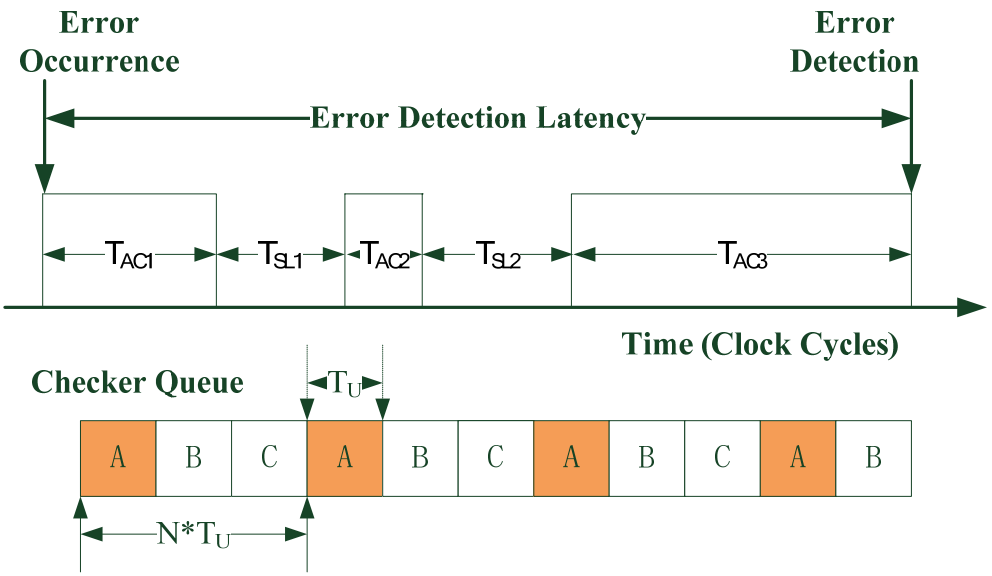

图 16 分时故障诊断模型示意图

Fig 16 TDM Error Diagnosis Model

3.3.1 面积代价

定义分时故障诊断的冗余面积代价A_{OH}为额外增加的逻辑所占面积与原始设计电路所占面积之比，则有：

$$A_{OH} = \frac{N_2 * A_{checker} + N * A_{mux}}{N_1 * A_{module}} = \frac{1}{N}\left(\frac{A_{checker} + A_{mux}}{A_{module}}\right) \approx \frac{K}{N} \quad (1)$$

跟据文献[18]对 8 个 MCNC 基准电路的仿真结果可知，$A_{checker}$ 与 A_{module} 的比例在 0.88～1.77 之间，而 A_{mux} 通常远小于其他两项，因此当 N 不太大时，K 近似为常量。由公式(1)可知，分时故障诊断的面积代价 A_{OH} 与复用同一块区域的模块数量 N 近似成反比。

但是当复用同一块区域的模块增多时，多路选择器接口的面积也会相应增加，因此 A_{mux} 是 N 的一阶函数，比例系数主要取决于被测模块的数据通路的位宽。假设 $A_{module} = A_{checker}$，则有：

$$A_{OH} = \frac{1}{N} + \frac{A_{mux}}{N * A_{module}} \quad (2)$$

在实际应用中，一旦 N2 确定，不论 N 如何变化，$N*A_{module}$ 都是一个常数。如果 N 太大，公式(2)中第二项的增加会抵消第一项的减少，甚至可能出现 A_{OH} 随着 N 的增加而增大的情况，如图 17 所示。图 17 中 $N*A_{module}$ 设为 10000，多路选择器的数据为 16 位。由图 17 可以看出，分时故障诊断策略可以很容易的将系统的冗余面积代价降低至 50%以内，与文献[9]方法相比有着明显的优势。

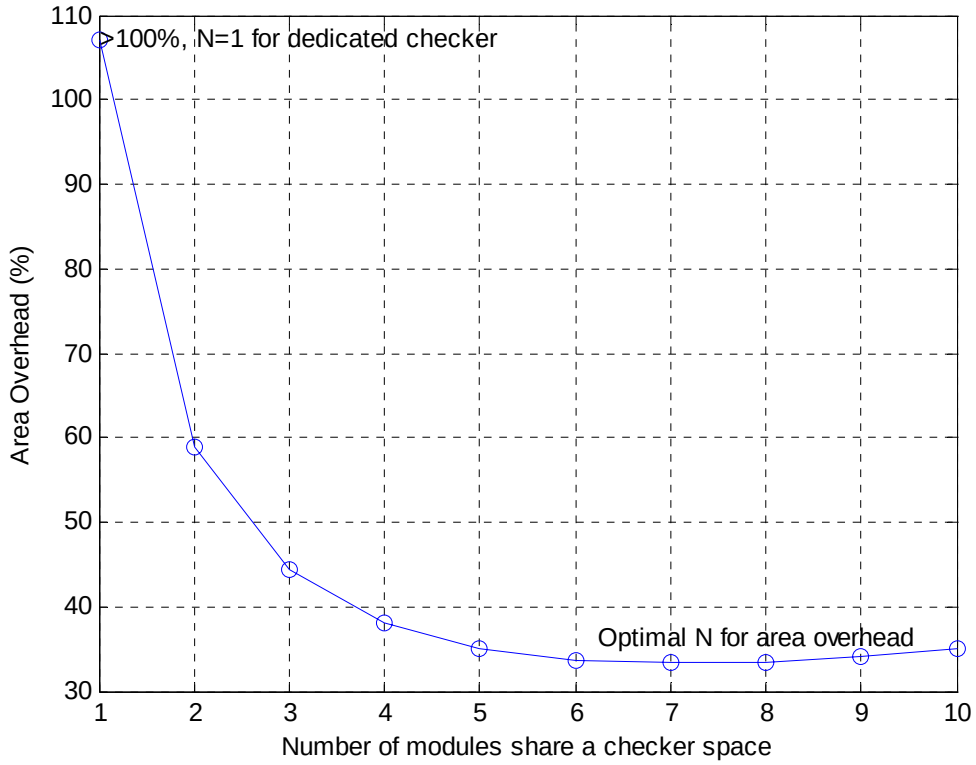

图 17 面积代价与 N 的关系

Fig 17 Area Overhead vs N

3.3.2 故障检测能力

故障检测能力通常用以下两个指标衡量：

- 平均故障检测延时 T_{DL}，定义为故障被检测到的时刻与故障产生的时刻之间的平均间隔；
- 故障检测率 P_D，定义为某个故障产生后，系统在其消失之前检测到此故障的概率；

假设系统中的某个模块 A 出现故障，此故障的当前活跃期长度为 T_{AC}，则在 T_{AC} 内故障模块真正处于被监控状态下的平均时间为：

$$c = \frac{T_{AC}}{N} * \frac{T_M}{T_U} \tag{3}$$

此故障在 T_{AC} 内被检测到的概率为：

$$P_D = \sum_{n=1}^{c} p(1-p)^{n-1} = 1-(1-p)^c \tag{4}$$

假设故障能够在 T_{AC} 内被检测到，则所需要的检测向量的平均数量为：

$$T_O = \frac{\sum_{n=1}^{c} np(1-p)^{n-1}}{\sum_{n=1}^{c} p(1-p)^{n-1}} \tag{5}$$

于是，平均故障检测延时可以用公式(6)表示：

$$\begin{aligned} T_{DL} &= \frac{1}{2}(N-1)T_U + \frac{T_O}{T_M * \frac{E_{AC}}{E_{AC}+E_{SL}}} * NT_U \\ &= \frac{1}{2}(N-1)T_U + \frac{T_O * (E_{AC}+E_{SL})}{T_M * E_{AC}} * NT_U \end{aligned} \tag{6}$$

当模块 A 出现故障时，有可能并未处于被监控状态，需要等待 A 的检测模块被配置才能开始检测。在最不利的情况下，需要等待的时间为$(N-1)T_U$。公式(6)的第一项反映了这个等待时间的平均值。 对于间歇性故障，T_M 同时包括故障活跃期和休眠期，因此一个轮换周期内真正有效的输入向量个数为$T_M * \frac{E_{AC}}{E_{AC}+E_{SL}}$，公式(6)的第二项反映了从开始检测到故障被检测到所需要的平均时间。

3.2.2.1 永久性故障检测

对于永久性故障来说，$T_{AC}=\infty$，$T_{SL}=0$，所以 $c=\infty$，$E_{AC}=\infty$，$E_{SL}=0$。因此可以推出故障检测率为：

$$P_D=\lim_{c\to\infty}\sum_{n=1}^{c}p(1-p)^{n-1}=\lim_{c\to\infty}\left[1-(1-p)^{c}\right]=1 \tag{7}$$

公式(7)说明，采用分时故障诊断策略不会导致永久性故障的检测率降低。检测到此故障所需的输入向量个数为：

$$T_O=\lim_{c\to\infty}\frac{\sum_{n=1}^{c}np(1-p)^{n-1}}{\sum_{n=1}^{c}p(1-p)^{n-1}}=\frac{1}{p} \tag{8}$$

故障检测延时为：

$$T_{DL}=\frac{1}{2}(N-1)T_U+\frac{N*T_U}{p*T_M}=\left(\frac{1}{pT_M}+\frac{1}{2}\right)*N*T_U-\frac{T_U}{2} \tag{9}$$

公式(9)说明永久性故障的检测时延随着复用同一块区域的模块数量增加而线性增加。以 Xilinx 的 XC2VP30 FPGA 芯片为例，系统时钟为 100MHz，可重构区域占芯片面积的 10%时，重构时隙小于 3ms[27]。因此假设 $T_{RS}=3\times10^5$，$T_U=1.5\times10^6$，则永久性故障的 T_{DL} 与 N 的关系如图 18 所示。当 N=1 时，每个模块被一个专用的检测模块监控，相当于文献[9]中的方法，此时永久性故障的 T_{DL} 最小。

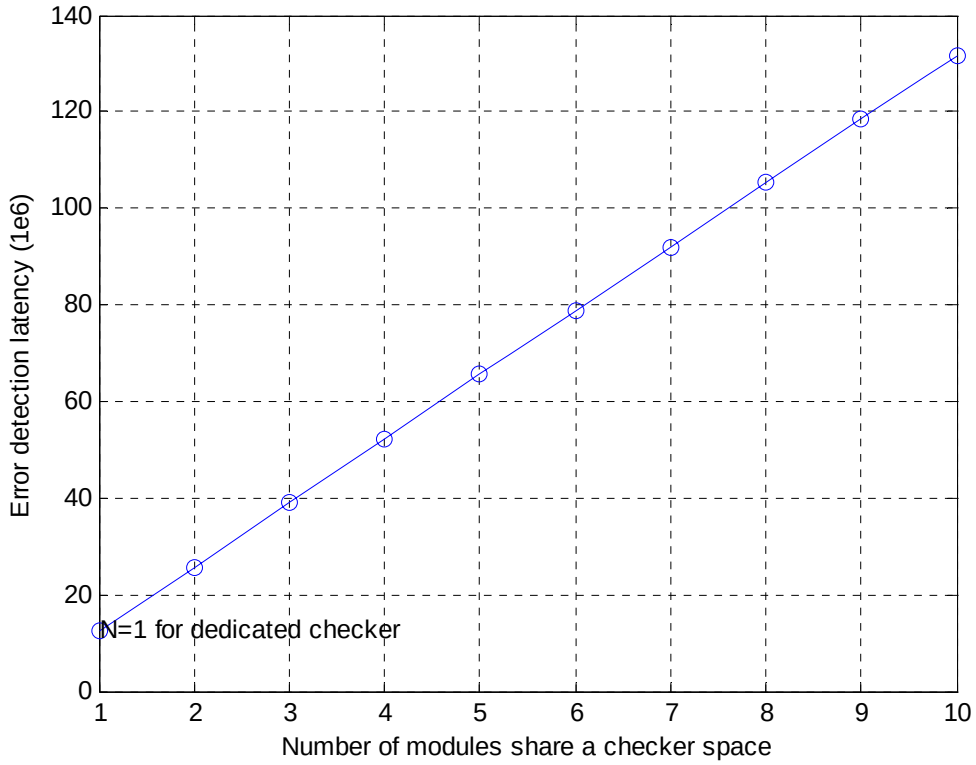

图 18 永久性故障检测时延与 N 的关系

Fig 18 T_{DL} vs N for Permanent Faults

3.3.2.2 瞬态故障检测

对于瞬态故障来说，$T_{SL}=0$，$T_{AC}<\infty$。假设随机变量 T_{AC} 的概率密度函数为 $f(x)=P\{T_{AC}=x, x>=1\}$，由公式(4)推出故障检测率的数学期望为：

$$E(P_D)=\int_{x=1}^{+\infty}\left(1-(1-p)^c\right)*f(x)dx=1-\int_{x=1}^{+\infty}(1-p)^c*f(x)dx \qquad (10)$$

其中 $c=\frac{T_M}{N*T_U}x$。由于 p 的值非常接近 0，当 x 不太大时，$(1-p)^c$ 约等于 1。考虑到瞬态故障的持续时间是有限的，公式(10)中积分项的值略小于 1，因此 $E(P_D)$的值

很小。当 N=1 时，c=x，P_D 最大且只和 T_{AC} 有关。当 N 增加时，c 减小，$(1-p)^c$ 更加趋近于 1，积分项的值增大，导致故障检测能力下降。

当 T_U 增加时 c 增大，从而 P_D 增大。但由于 $T_M=T_U-T_{RS}$，当 T_U 非常大时，$c \approx \frac{x}{N}$，T_U 继续增加时 P_D 基本保持不变。公式(10)化简为：

$$E(P_D)=1-\int_{x=1}^{+\infty}(1-p)^{\frac{x}{N}} * f(x)dx \tag{11}$$

通常瞬态故障的持续时间很短，假设 T_{AC} 分别等于 5×10^4(0.5ms)、 1×10^5(1ms)、2×10^5(2ms)、4×10^5(4ms)，则 N 与 P_D 的关系如图 19 所示。

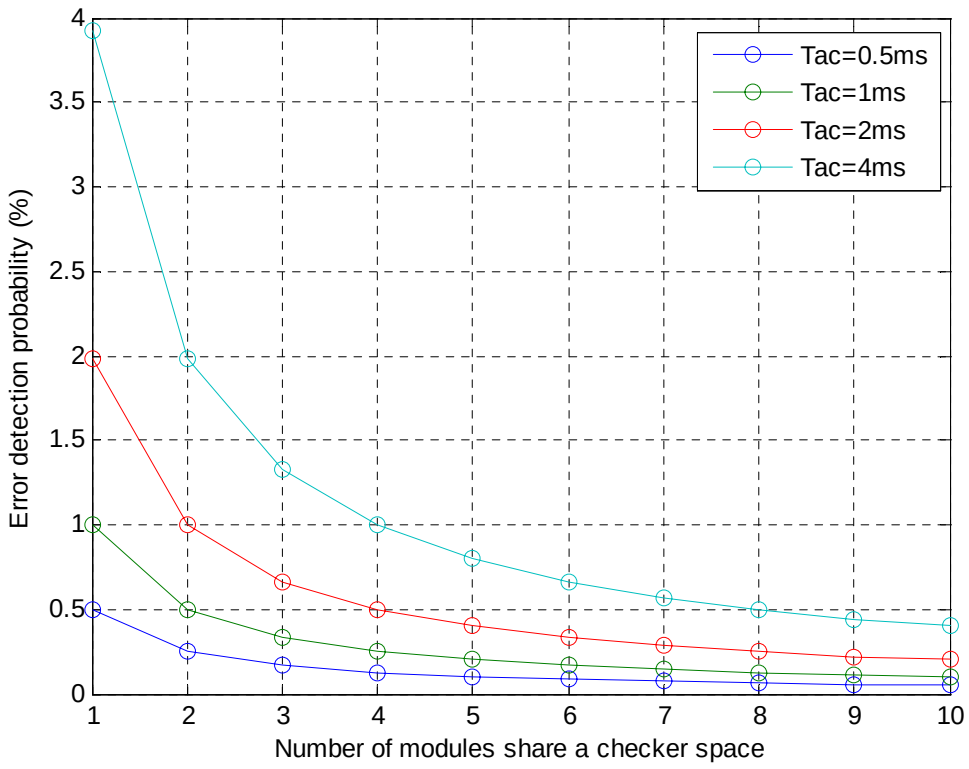

图 19 瞬态故障的检测率与 N 的关系

Fig 19 Detection Probability vs N for Permanent Faults

考虑所有在消失之前被检测到的瞬态故障，平均检测延时 T_{DL} 可以表示为：

$$T_{DL}=\left(\frac{T_O}{T_M}+\frac{1}{2}\right)*N*T_U-\frac{1}{2}T_U \tag{12}$$

$$E\left(T_{DL}\right)=\int_{x=1}^{+\infty}\left(\frac{T_O}{T_M}+\frac{1}{2}\right)*N*T_U*f\left(x\right)dx-\frac{1}{2}T_U \tag{13}$$

其中 T_O 符合公式(5)，而 $c=\frac{T_M*T_{AC}}{N*T_U}$ 。瞬态故障的平均检测延时特性复杂，由于其检测率很低，本文不再讨论其 T_{DL} 与 N 的关系。

3.3.2.3 间歇性故障检测

对于间歇性故障来说，$T_{SL}>0$，$T_{AC}<\infty$。根据之前的假设，T_{AC} (i)和 T_{SL}(i)都是独立同分布的随机变量序列。显然在每个活跃期内，故障被检测到的概率均符合公式(10)。设第 i 个活跃期的故障检测率为 P_{Di}，其数学期望为 E_{PD}，则总的检测率为：

$$P_D=\sum_{j=1}^{\infty}\left[P_{Di}*\prod_{i=1}^{j-1}\left(1-P_{Di}\right)\right] \tag{14}$$

$$E\left(P_D\right)=\sum_{j=1}^{\infty}\left[E\left(P_{Di}\right)*\left(1-E\left(P_{Di}\right)\right)^{j-1}\right]=1 \tag{15}$$

公式(15)说明，采用分时故障诊断策略不会导致间歇性故障的检测率降低。假设在第 i 个活跃期监测到故障，延时为 T_{DLi}，其数学期望为 E_{PD}。则总的平均故障检测延时为：

$$\begin{aligned}T_{DL} &= P_{D1}*T_{DL1}+P_{D2}*\left(1-P_{D1}\right)*\left(T_{DL2}+T_{AC}\left(1\right)+T_{SL}\left(1\right)\right)\\ &\quad +P_{D3}*\left(1-P_{D2}\right)\left(1-P_{D1}\right)*\left(T_{DL3}+T_{AC}\left(2\right)+T_{SL}\left(2\right)+T_{AC}\left(1\right)+T_{SL}\left(1\right)\right)\\ &\quad +\cdots\cdots\\ &= \sum_{j=1}^{\infty}\left\{P_{Dn}*\left[\prod_{i=1}^{j-1}\left(1-P_{Di}\right)\right]\left[T_{DLn}+\sum_{i=1}^{j-1}\left(T_{AC}\left(i\right)+T_{SL}\left(i\right)\right)\right]\right\}\end{aligned} \tag{16}$$

$$\begin{aligned}E\left(T_{DL}\right) &= \sum_{j=1}^{\infty}\left\{E_{PD}*\left(1-E_{PD}\right)^{j-1}*\left[\left(E_{AC}+E_{SL}\right)*\left(j-1\right)+E_{DL}\right]\right\}\\ &= \left(E_{AC}+E_{SL}\right)*\sum_{j=1}^{\infty}\left[j*E_{PD}*\left(1-E_{PD}\right)^{j-1}\right]-E_{AC}-E_{SL}+E_{DL}\\ &= \left(E_{AC}+E_{SL}\right)*\left(\frac{1}{E_{PD}}-1\right)+E_{DL}\end{aligned} \tag{17}$$

其中 E_{PD} 和 E_{DL} 分别符合公式(10)和公式(13)。

$$\begin{aligned}E_{PD} &\approx 1-\left(1-p\right)^{\frac{E_{AC}}{N}*\frac{T_M}{T_U}}\\ E_{DL} &\approx \left(\frac{N}{pT_M}+\frac{N-1}{2}\right)T_U\end{aligned} \tag{18}$$

若 $E_{AC}=10^6$(10ms)，$E_{SL}=10^7$(100ms)，$T_{RS}=10^6$(10ms)，则平均故障检测延时与时间片长度的关系如图 20 所示。

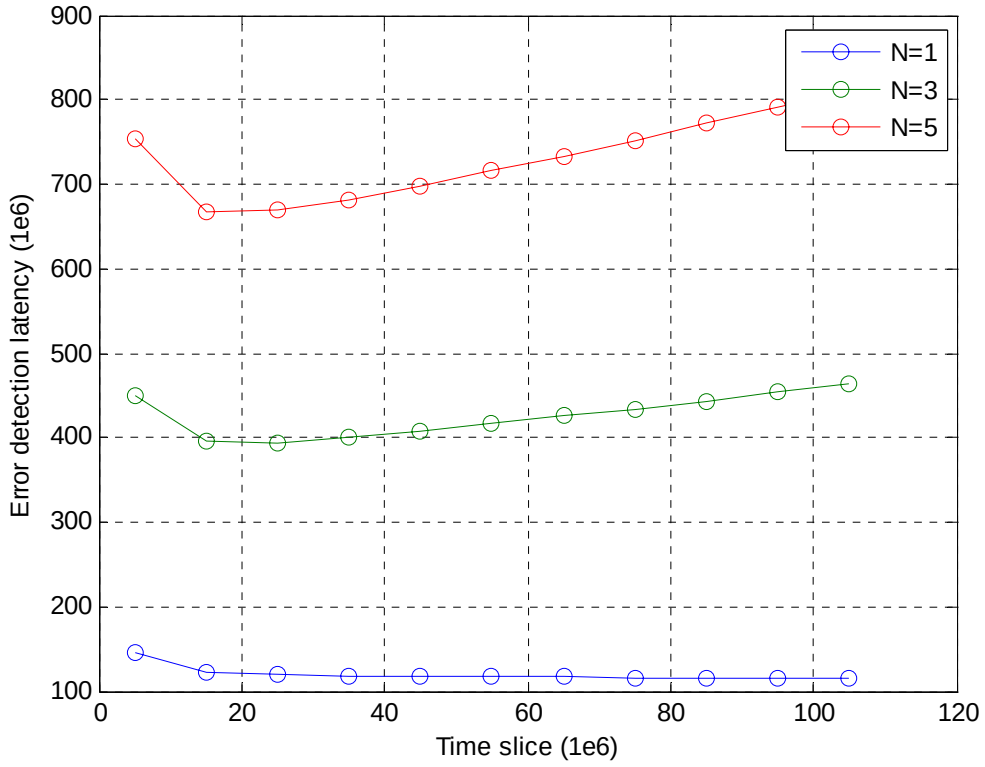

图 20 平均故障检测延时与时间片长度的关系

Fig 20 Average Error Detection Delay vs Time Slices

在上图中，无论 N 为何值，使平均故障检测延时最小的时间片长度均为约 200ms。其原因在于，当 T_U 大于此值时，故障模块进入被监控状态之前的等待时间随着 T_U 增大而增加的趋势在 $E(T_{DL})$中起主导作用。而当 T_U 小于此值时，由于 T_{RS} 影响了检测效率，故障模块进入被监控状态之后的检测时间随 T_U 减小而增加的趋势在 $E(T_{DL})$ 中起主导作用。

3.4 本章小结

本章主要包括以下三个方面的内容：

- 提出了基于并发故障检测和运行时重构的分时故障诊断的基本原则；

- 详细分析了基于分时故障诊断的容错设计流程，并对其中的若干关键问题——模块分割、检测模块的选择、时间片调度——作了定性的分析；
- 系统的提出了一个概率模型，分析了分时故障诊断对于不同种类故障的检测能力，粗略地归纳了设计参数与故障监测能力之间的函数关系，以便在实际应用中根据性能要求估计设计参数。

第四章 故障检测模块的实现

就每一个单独的时间片来看，分时故障诊断系统相当于一个只有部分模块处于被监控状态的并发故障检测系统，因此其故障检测模块的设计可借鉴并发故障检测模块的设计。

在并发故障检测中，预报器由被测模块或系统的输入信号驱动，计算其输出信号的某些特征如奇偶性，预报器的逻辑结构由所要提取的特征和被测电路的传递函数或状态转移方程确定。根据所要提取的特征的不同，目前常用的并发故障检测模块可以分为以下三类：

- 基于奇偶性预测（Parity Prediction）的方案；
- 基于复式系统（Duplex System）的方案；
- 基于单向错误检测编码（Unidirectional Error Detection Codes）的方案。

以上三种方案均适用于任何数字电路。除此之外的一些方法，如文献[28]和文献[29]中的两种方法，仅适用于某些特定结构和功能的电路，因而不具有方法学意义上的普遍性，本文不予考虑。

4.1 并发故障检测方案介绍

4.1.1 复式系统

复式系统指的是两个具有相同逻辑功能的模块组成的系统，最初以多模冗余容错（NMR/NMR）设计的形式出现在经典的静态容错电路中[30][31][32]。基于复式系统的并发故障检测模块框图如图 21 所示。从逻辑功能上来看，预报器就是原始模块的复制，预报器提取的输出数据特征就是原始模块的输出数据本身。原始模块和预报器中的任何一个出现故障，都会导致进入比较器的两路输出信号不一致，此时系统会报告一次错误。

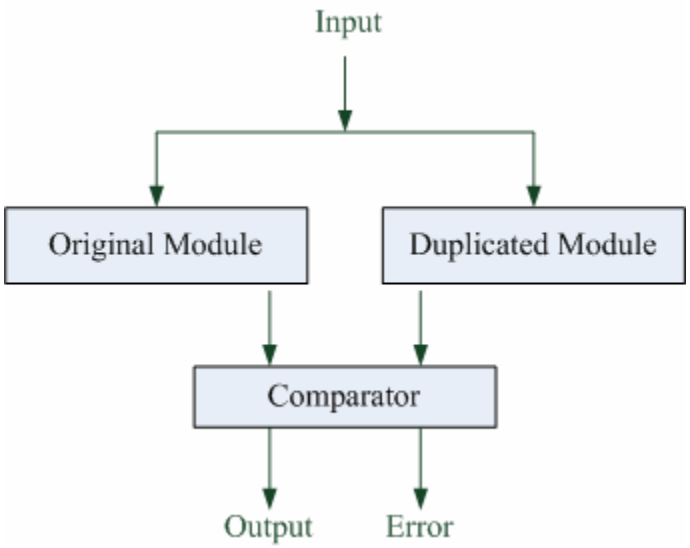

图 21 基于复式系统的并发故障检测
Fig.21 Concurrent Error Detection Based on Duplex System

假设每个时钟周期原始模块出现单重故障（Single Faults）的概率是 $10^{-\alpha}$，则两个模块同时出现相同的单重故障的概率仅为 $10^{-2\alpha}$，这意味着故障传播到后续模块的概率降低了 α 个数量级，因此模块输出数据的可靠性大大提高。

在复式系统中，原始模块逻辑功能的复制有两种方式：同构复制和异构复制（Identical Duplication & Diverse Duplication）。两者的区别在于，同构复制模块不仅与原始模块功能相同，而且 RTL 结构也和原始模块完全相同，而异构复制模块的 RTL 结构与原始模块有明显区别。对于单重故障，两种复制方式具有相同的检测能力。

4.1.2 奇偶性预测

基于奇偶性预测的方法是在高可靠性系统中应用最为广泛的并发故障检测方法[30][32][33][34]，特别适合于数据通道和寄存器堆的故障检测。此处数据的奇偶性指的是多位数据中逻辑‘ 1 ’的数目为奇数还是偶数。

在基于奇偶性预测的并发故障诊断中，预报器根据原始模块的输入数据 X 预测其输出数据的奇偶性，其结果 P 与原始模块的输出数据 Y[n-1:0]一起被送入偶校验电路，如图 22 所示。如果数据{Y[n-1:0], P}中逻辑‘1’的数目为奇数，则说明{Y[n-1:0], P}

中的某一位出现故障。为了避免某个单重故障同时影响原始模块输出的多个数据位导致部分故障不能被检测到，原始模块需要重新设计，确保输出数据的每一位依靠独立的逻辑电路产生，禁止不同位之间共享逻辑单元。

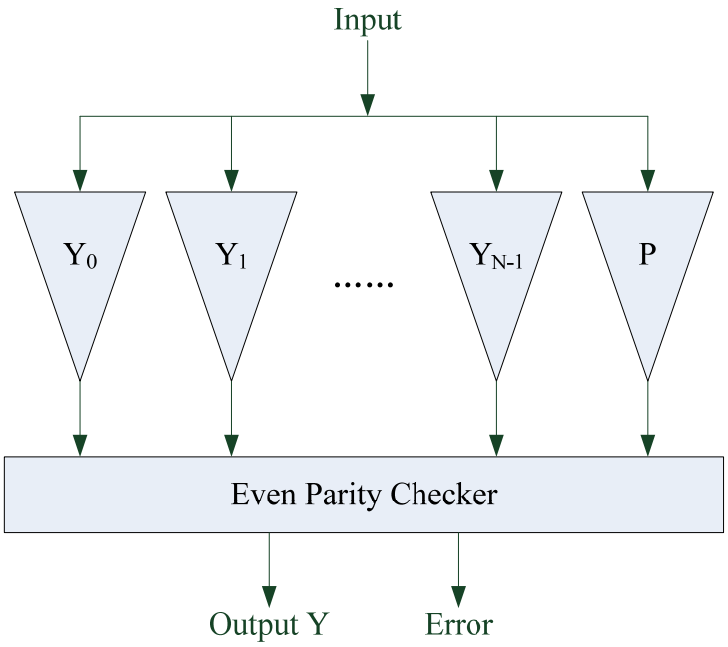

图 22 采用一位奇偶性预测和校验的并发故障检测

Fig.22 Concurrent Error Detection Based on 1-bit Parity Predictor

上述限制导致原始模块的面积显著增大。为了降低故障检测的冗余面积代价，可以采用多位奇偶性预测和校验的方法，如图 23 所示。原始设计输出的各数据位被分成几组，对于每一组数据位，预报器分别预测其奇偶性，并采用图 22 中的方法校验。多位奇偶性预测和校验仍然不允许同一组的数据位之间共享逻辑单元，但允许在不同组的数据位之间共享逻辑单元。

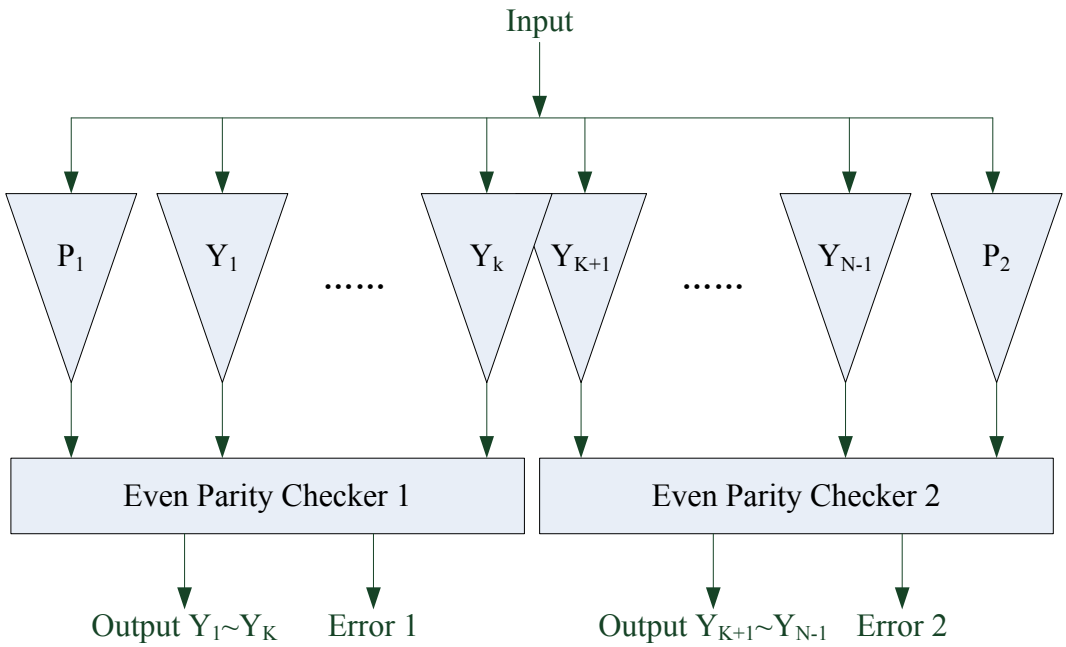

图 23 采用多位奇偶性预测和校验的并发故障检测

Fig.23 Concurrent Error Detection Based on Multi-bit Parity Predictor

例如在图 23 中，原始模块输出各数据位被分为 G_1 和 G_2 两组，其中 $G_1=\{Y_1\sim Y_K\}$，$G_2=\{Y_{K+1}\sim Y_{N-1}\}$，与之相关的奇偶性预测位分别为 P_1 和 P_2。此时 G_1 中的任意位和 G_2 中的任意位均可以共享逻辑单元，除此之外不允许任何不同位之间的逻辑单元共享。通过恰当的分组可以有效的降低冗余面积开销。

4.1.3 单向错误检测编码

采用单向错误检测编码（EDC）的并发故障检测最早出现在文献[35]中。所谓单向，是指这种方法假设所有的错误都是单向的。也就是说，电路中同时出现的错误只能表现为同一种形式，或者是本应为逻辑‘1’的电平变成逻辑‘0’，或者是本应为逻辑‘0’的电平变成逻辑‘1’，但这两种情形不能同时存在。在并发故障检测中常用的错误检测编码有两种：Berger 编码[36]和 Bose-Lin 编码[37]，其结构如图 24 所示。

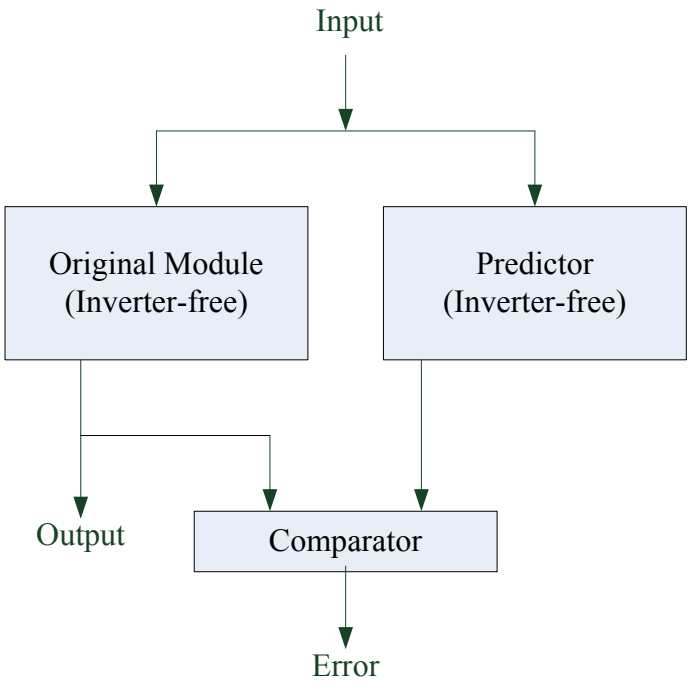

图 24 基于 EDC 的并发故障检测

Fig.24 Concurrent Error Detection Based on EDC

Berger 编码最初被用在通信信道的错误检验中，每个码字由两部分组成：对于一个 n 比特的待编码数据字 Y，其 Berger 编码的前半部分是 Y 本身，后半部分是 Y 的各数据位中逻辑‘0’的个数，这一部分需要额外的$\lceil \log_2 n \rceil$比特。在基于 Berger 编码的并发故障检测中，预报器根据输入数据预测输出数据中逻辑‘0’的个数，或者是预测逻辑‘1’的个数并取其二进制补码（2’ complement），然后与原始模块输出数据中逻辑‘0’的个数比较。如果两者不一致，则报告一次错误。Berger 编码能够检测到电路中任意多个比特的单向错误。

由于 Berger 编码是单向错误检测编码，必须确保电路中的一次单重故障对输出数据的影响是单向的，这意味着原始模块中所有的反相器只能出现在电路的输入端，这种电路综合方式被称为 Inverter-free[35]。Berger 编码的比较器的电路实现可以参考文献[36]。

Bose-Lin 编码是一种参数化的编码方式。参数为 t 的 Bose-Lin 编码最多能够检测到 t 个比特的单向错误[37]。文献[38]给出了基于 Bose-Lin 编码的并发故障检验的电路实现。与 Berger 编码类似，Bose-Lin 编码同样要求原始模块中所有的反相器只出现在电路的输入端，以确保任意的单重故障只会导致输出数据的单向错误。由于 Bose-Lin 编码最多能够检测到 t 个比特的单向错误，原始模块中的任何逻辑单元最多只能被输出数据中的 t 位共享。Bose-Lin 编码的比较器的电路实现可以参考文献[39]。

4.1.4 比较与结论

考虑故障检测能力、冗余面积代价和设计复杂度，根据文献[18]的仿真结果，以上三种方式都有明显的缺陷。基于同构复式系统的方法设计复杂度最低，但其冗余面积代价通常超过 100%，且对于电路中的共模故障的检测能力最差。

共模故障（Common Mode Failures, CMF）是一类特殊的多重故障（Multiple Failures），它代表了由于同一个物理缺陷或环境因素变化导致电路中同时出现作用相似的多个错误的情形。对于同构复式系统来说，CMF 容易在相同的 RTL 结构中表现为方向相同的错误。而异构复式系统的原始模块和预报器有不同的 RTL 结构，CMF 通常不会在输出数据的每个比特上都表现为方向相同的错误，因而这种方法能够改善 CMF 检测的问题，但考虑到系统异构的设计及异构程度的评价较为困难，这种方法的设计复杂度太高。

基于单向错误检测编码的方法以较低的设计复杂度改善了 CMF 检测的问题，但带来了过大的冗余面积代价。基于一位奇偶性预测的方法在故障检测能力上有着明显的缺陷，如果原始模块中的某一个故障导致输出数据的偶数个比特同时出现错误，校验电路就无法正确识别。为解决这一问题，可以采用前文提到的多位奇偶性预测，但这需要仔细的修改原始模块的结构以消除逻辑单元共享，或者采用随机游走算法构造最小奇偶预测树，如文献[46]所述。这两种方法都会导致设计复杂度和冗余面积代价增加。另一方面，修改原始模块的逻辑结构在部分情况下是不可行的，尤其是在采用 IP 实现的设计中。因此，对于低成本的应用，需要研究一种设计复杂度较低的、冗余面积开销较小的、非插入式的（non-intrusive）并发故障检测器。

4.2 故障检测器的简化

根据信息论的原理，对于一个最优化的原始电路模块，若要在预测器中保留其输出信号的全部信息，预测器的逻辑结构不可能比原始电路更简单，此时基于复式系统的故障检测器是并发故障检测的最佳方案，若要简化预测器的逻辑结构以降低冗余面积代价，必须损失原始模块输出信号的部分信息。

从逻辑综合的角度看，综合工具总是根据的电路输入/输出得到布尔函数，通过化简布尔函数得到最优化的逻辑结构。然而布尔函数的化简是一个 NP 问题，输入/输出信号越多，布尔函数越复杂，化简就越困难，导致电路的逻辑结构停留在一个次优解上。

考虑到电路中的一个故障可能会影响到输出信号的多个位，而故障检测器只需要检测到其中一个位与预测器的相应位不一致即可判定故障，因此故障检测器只需要预测和检测原始模块输出的部分位，即可判定原始模块中是否出现故障。根据这种方法，预测器的逻辑函数本身大大简化，其输出信号数目也有所减少，因而故障检测器的逻辑结构与复式系统相比得到简化，从而降低了冗余面积开销。通过合理的选择被预测和比较的位，即可避免故障覆盖率的损失。

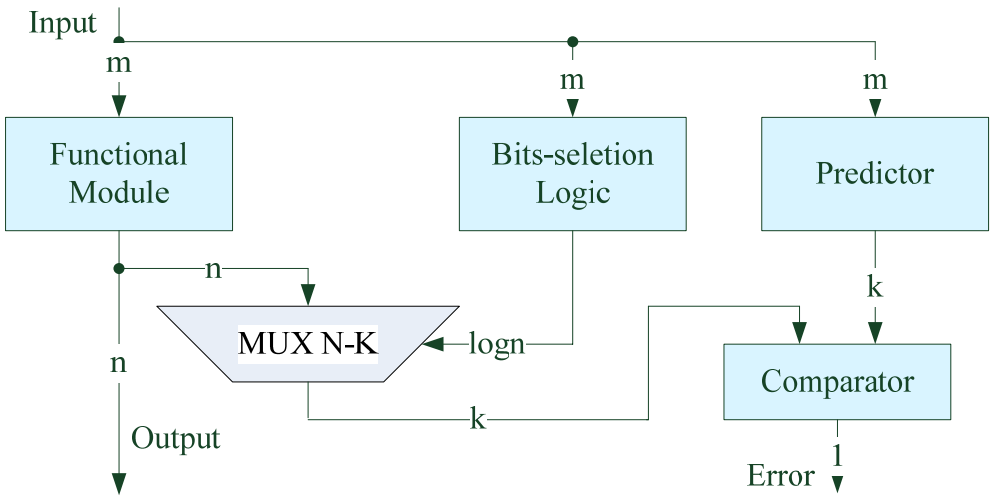

图 25 简化的故障检测器

Fig. 25 Reduced Checker Scheme

然而固定的选择原始模块输出的某些位显然是不可行的，这导致未被选择的位上的 stuck-at 故障可能会被漏过。根据以上讨论，本文提出了如图 25 所示的简化方案。对于任意的 m 位输入向量，通过位选择逻辑控制多路选择器，从原始模块的 n 位输出中选择 k 位，并与预测器产生的 k 位预测值比较。举例来说，对于两输入两输出电路，假设故障列表{F1,F2,…,F10}中的每个故障对电路输出的影响如表格 1 所示，其中 1 代表当前输入向量会激发故障，导致输出的相应位出错。为了覆盖整个故障列表，只需要在输入向量为 V0 或 V2 时选择 O1，V1 或 V3 时选择 O0 作为预测位和检测位即可。此时冗余逻辑包括一个 2 选 1 的多路选择器，，一个 1 位比较器和一个 1 位输出输出信号的逻辑函数，与复制系统中的一个 2 位输出信号的逻辑函数和一个 2 位比较器相比，冗余逻辑更少，冗余面积代价也更低。

	V0=00		V1=01		V2=10		V3=11	
	O1	O0	O1	O0	O1	O0	O1	O0
F1	1	0	1	0	1	0	1	0
F2	1	0	0	0	0	0	1	0
F3	1	0	1	0	0	0	0	0
F4	0	0	0	1	0	1	0	1
F5	0	0	0	1	0	1	0	0
F6	0	0	0	1	0	0	0	0
F7	0	1	1	0	1	1	1	0
F8	0	0	0	0	1	0	0	0
F9	0	1	0	0	0	0	0	1
F10	0	1	0	0	0	0	0	1

表格 1 故障响应矩阵示例

更加一般化的说，对于 m 输入 n 输出的原始模块，简化的预测器要实现 k 个 m 输入的逻辑函数，而 n 选 k 多路选择器可以看作 k 个 n 选一多路选择器，每个的选通信号为 $\log_2 n$ 位，因此位选择逻辑要实现 k× $\log_2 n$ 个 m 输入函数。由于复式系统的预测器需要实现 n 个 m 输入的逻辑函数，所以必须使 $k < \frac{n}{1+\log_2 n}$ 才能达到简化故障检测器的目的。

根据以上讨论，使图 25 中的故障检测器的冗余面积尽可能小的关键是确定最优化的 k 值并从 n 位输出中选择合适的 k 位，本文提出的方法叙述如下：

1、读入原始模块的 RTL 代码，用综合工具（Design Compiler）生成门级网表，用 DFT 工具生成 STIL 测试协议文件。由网表和 STIL 测试协议，采用自动测试向量产生（Automatic test pattern generation, ATPG）工具（TetraMAX）生成原始模块的测试向量集和故障列表，执行故障仿真以确定故障覆盖率，消除故障列表中的等价故障，确定故障列表 F 和测试向量列表 V。依次将 F 的一个故障注入原始模块，并用 V 执行仿真，以确定 F 中的每一个故障对原始模块输出的影响，据此建立类似于表格 1 所示的故障响应矩阵 A。

2、此时问题归结为求矩阵 A 的最小覆盖问题的变种，这是一个 NP 问题。首先按照 V 中各测试向量高$\lceil \log_2 n \rceil$位的值，将 V 中的向量分为 $M = 2^{\lceil \log_2 n \rceil}$ 个子集 $\{S_1, S_2, ..., S_M\}$，子集 S_i 的最小覆盖记录在表 R_i 中。这样对于输入变量的每个子集，所选择的被检测位是统一的，测试向量的高$\lceil \log_2 n \rceil$位可以直接作为 n 选 k 多路选择器的控制信号，消除了额外的位选择逻辑。同时，矩阵 A 也被分割为 M 个子矩阵 $\{A_1, A_2, ..., A_M\}$。

3、对于矩阵 A，如果某一行中只有 S_k 子集中 V_i 向量的第 j 位对应列的元素为 1，说明此故障只能被这一位检测到，因此把 V_i 和 j 添加到 R_k 中，并在矩阵 A 中删除所有被 V_i 向量的第 j 位覆盖的行，删除 V_i 向量的第 j 位对应的列。重复以上过程直到剩余的所有行都被多于两列覆盖。

4、对于矩阵 A，两两比较矩阵的各行，如果对于 F_i 中所有为 1 的元素，F_j 中相应列的元素都为 1，则能够检测到 F_i 的测试向量和位必然能够检测到 F_j，称为 F_i 行“支配（dominate）”F_j 行，因此可以把 F_j 行从矩阵 A 中删除。遍历结束后，矩阵 A 的各行不存在支配关系。

5、对于每个子集 S_k，遍历矩阵 A，找出仅被 S_k 中的列覆盖的所有行，采用 Queen-McClusky 消去法求其最小覆盖，并将结果加入表 R_k 中，删除相应的列和被覆

盖的行。对于当前的矩阵 A，采用 Queen-McClusky 消去法求其最小覆盖，并将结果分别加入表 R_k 中。

以上算法采用 C 语言实现，程序输入为故障响应矩阵，输出为测试向量和相应的被预测位列表。此列表反映了预测器对于不同输入向量的输出，综合后得到预测器的逻辑电路。

4.3 寄存器状态同步

在传统的并发故障检测中，原始设计的每个模块 A 都有一个专用的检测模块 CK_A，系统一旦启动，CK_A 的工作状态不会中断，因此其寄存器状态与 A 的寄存器状态始终是同步的。但在分时故障诊断中，检测区域需要在运行时切换其硬件配置，而重构后的异构复制检测模块处于初始状态，其寄存器状态与 A 的寄存器状态不匹配，因此在开始检测之前需要使其同步，以免比较器报告不存在的错误。

4.3.1 状态复制

本文尝试了两种方法实现同步：状态复制和状态自动刷新。状态复制方法在重构完成后直接将被测模块中全部寄存器的值复制到检测模块的相应寄存器中，如图 26 所示。重构完成后，锁存器的初始值为逻辑“0”，多路选择器选通原始模块，于是原始模块寄存器的状态被复制到检测模块中。此时尽管比较器输出逻辑“1”，但锁存器为逻辑“0”，因此系统不会误报。当两个模块的寄存器状态一致时，表示完成同步，比较器输出逻辑“0”，锁存器置位信号有效，被置为逻辑“1”，多路选择器选通检测模块，于是故障检测开始。当出现故障时，比较器输出逻辑“1”，系统报告错误。

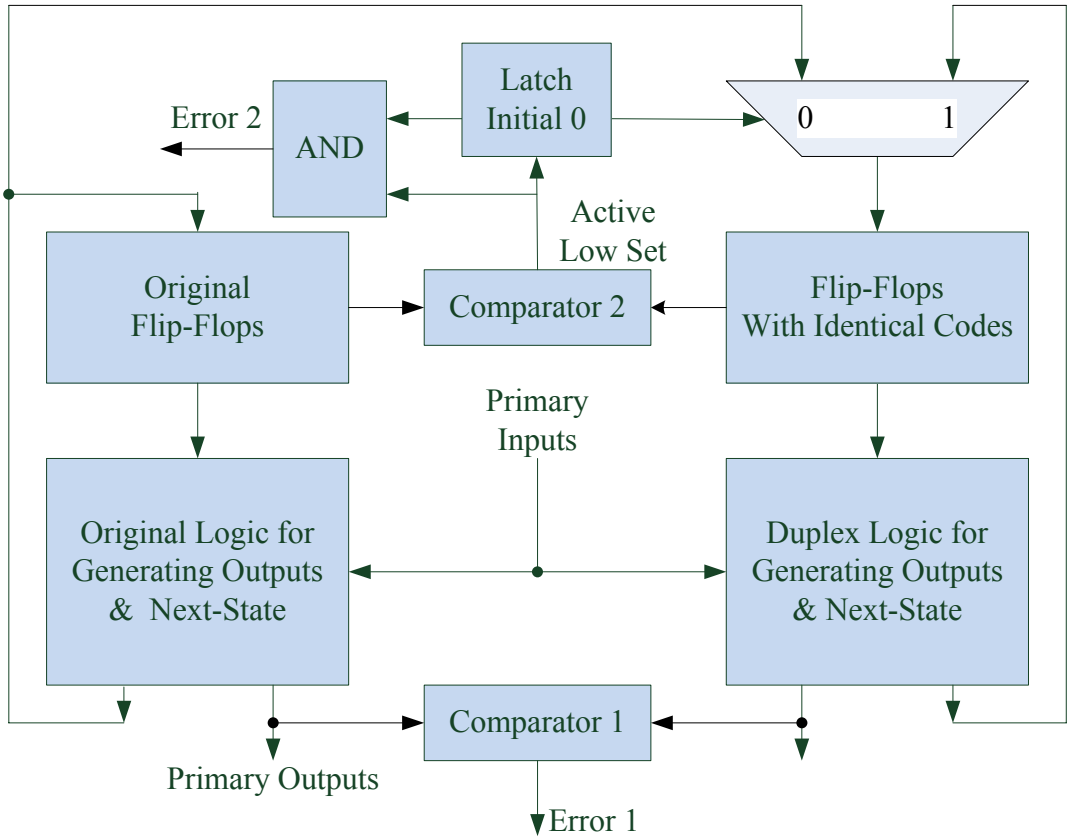

图 26 状态复制示意图

Fig.26 State Copy Scheme

状态复制方法可以在 FPGA 平台上得到验证。首先设计两个两比特计数器，初始值均为 0，其中一个（c1）实现递增技术，另一个（c2）实现递减计数，并将其计数值相加（sum）输出。根据图 26 分别设计这两个计数器的故障检测模块，并利用重构使两者在同一块区域内交替实现。图 27 中的仿真波形显示，检测区域重构后，寄存器状态同步可以在一个时钟周期内完成，在此期间系统不会报告错误（err）。当系统中未出现故障时，计数值之和始终为 0，表明原始系统的运行未受到重构的影响。采用强迫置位的方法在系统中模拟 stunk-at-1 故障（f），系统能够正确的报告错误。通过开发板上的硬件验证，可以观察到加法输出信号和错误报告信号符合预期。以上实验充分说明，状态复制方法能够实现预想的同步功能。根据综合报告，采用状态复制的方法时，原始设计和加入故障检测模块后的设计的大小分别为 65 个和 116 个等效逻辑门，其冗余面积开销约为 78%，与传统并发故障检测方法在时序逻辑电路中最

优化的冗余面积开销为 110%相比，占有明显优势。在更大规模的设计中，冗余面积开销还将进一步下降。

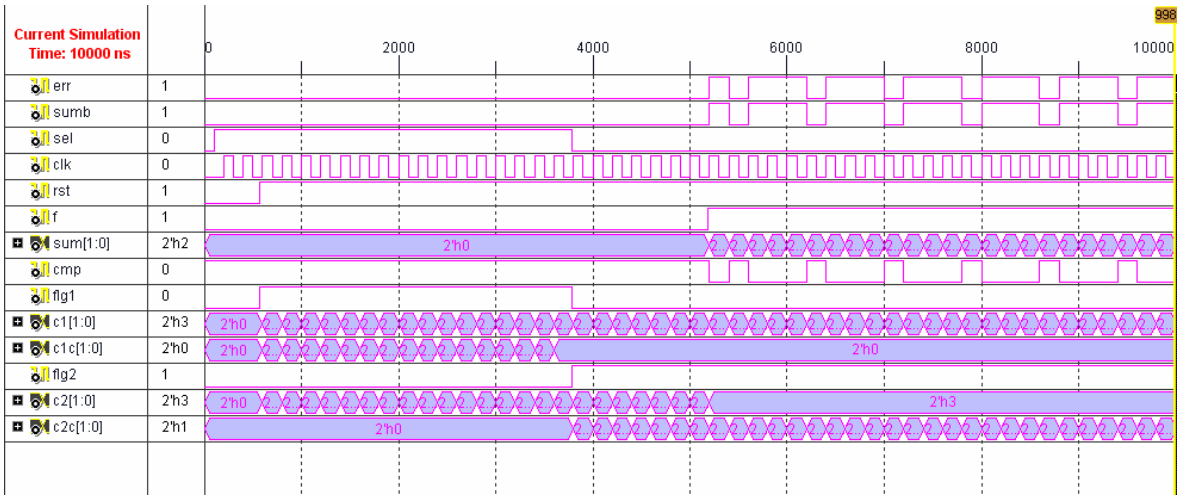

图 27 状态复制示例时序图

Fig. 27 Timing Waveform of An Example of State Copy

这种方法的优点是寄存器状态的同步可以在一个时钟周期内完成，有助于提高故障检测效率，且实现起来较为简单。但其问题在于，对每个寄存器都需要额外的逻辑资源和布线资源实现复制，当寄存器数目较多时，可能造成很大的冗余面积开销。因此，状态复制适用于寄存器数目较少，但在系统中十分关键的模块。同时，对于包含加密的 IP 核的原始设计，其具体的逻辑实现是不可见的，因而无法应用状态复制方法。

4.3.2 状态自动更新

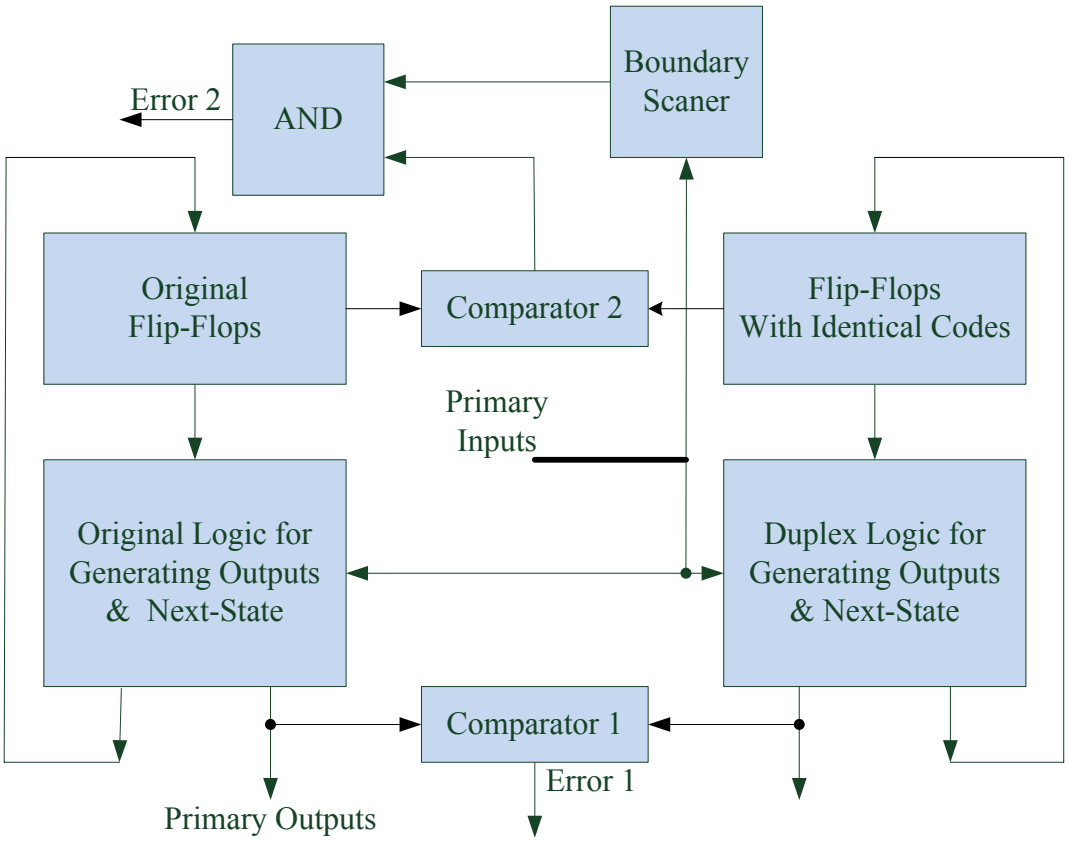

图 28 状态自动更新示意图

Fig. 28 State Auto Update Scheme

针对以上问题，本文尝试了状态自动刷新的方法。在绝大多数应用中，数据的处理都是按照一定的格式分成多个单元依次进行的，如在网络和通信系统中以数据包为单位，而在视频编解码系统中以场或者帧为单位。在这样的模块中，前后两个数据单元的寄存器状态并没有直接的继承关系，对于后一个数据单元来说，模块中寄存器的前一个状态是无用的。如果在一个新的数据单元开始时，原始设计模块和重构后的故障检测模块同时开始接收和处理数据，寄存器状态就能实现自动同步。

采用状态自动更新方法几乎不需要在寄存器状态同步上消耗冗余逻辑资源。为了防止故障检测系统误报，需要在寄存器状态同步完成之前屏蔽比较器输出，如图 28 中的边界扫描器所示。边界扫描器需要根据原始设计所处理的数据格式分别设计，用

于识别数据单元中表示头的数据字。每次故障检测模块重构后，边界扫描器的输出即被复位为逻辑‘0’。一旦检测到数据单元边界，其输出即变为逻辑‘1’，以允许比较器结果输出。

状态自动更新方法带来的冗余面积开销非常小，尤其是对于规模较大的系统，几乎可以忽略不计。尤其是在较大规模的异构复制电路中，实现寄存器状态的复制是一项非常复杂的任务，而自动更新避免了此问题。同时，这种方法也能基于黑盒模型很好的处理包含加密的 IP 核的模块。但这种方法的缺点在于，状态自动更新过程往往需要多个时钟周期，尤其对于较大的数据单元，可能导致故障检测效率的严重下降。同时，边界扫描器的设计与电路结构密切相关，增加了设计复杂度。本文在下一节的开发实例中采用了状态自动更新方法，因而此处不再给出验证。

4.4 本章小结

本章主要讨论了分时故障诊断中检测模块实现的相关具体问题。4.1 节中介绍了三种常用的并发故障检测算法，并说明和比较了三者的性能。4.2 节提出了一种简化的故障检测器实现算法，通过选择性的预测和检测原始模块的部分比特，在不损失故障覆盖率的前提下，简化了预测器逻辑，达到了降低冗余面积开销的目的。4.3 节提出了故障检测模块重构后寄存器状态的同步问题，设计了两种解决方案，并比较了其优缺点和适用范围。

第五章 开发实例：视频监控系统

为了进一步验证分时故障诊断的可行性，并检验其面积代价和故障检测延时，本章将分时故障诊断策略应用到一个视频监控系统当中，根据图 15 完成了从原始设计到具有较强故障检测能力的可靠设计的整个流程，并在一块 Xilinx XUPV2P 开发板[42]上实现了此系统。

5.1 原始设计

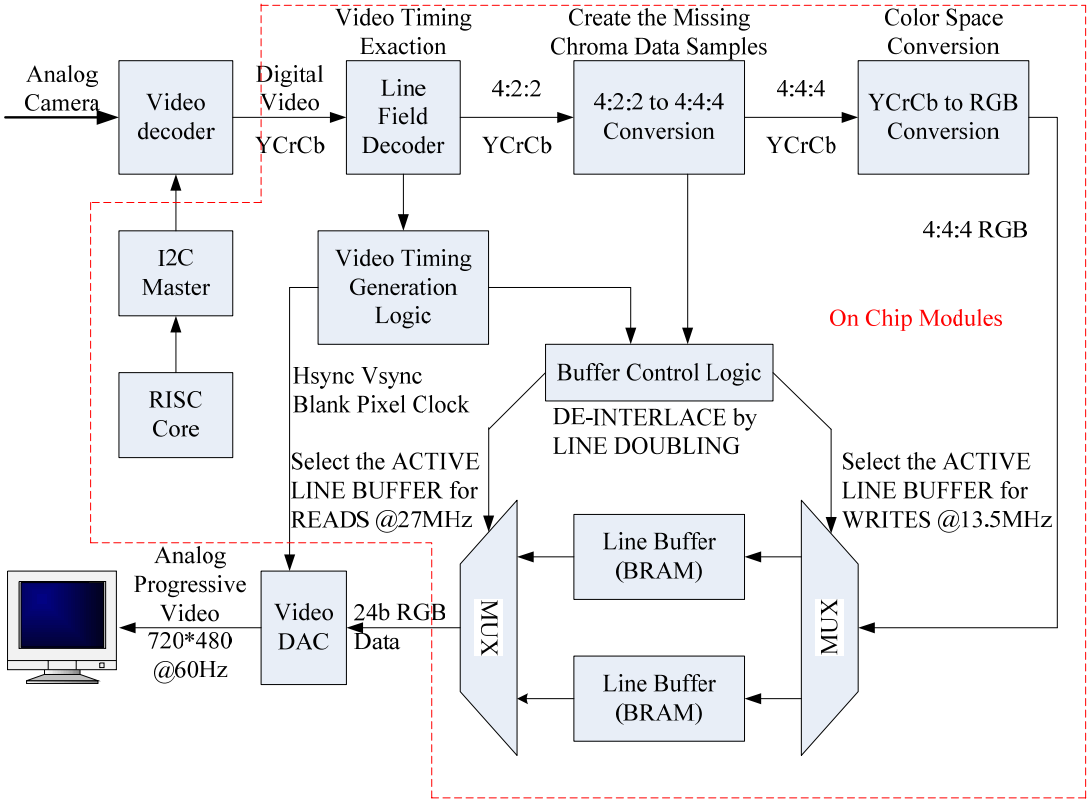

图 29 视频监视系统的原始设计

Fig.29 Original Design of Video Capture

视频监控系统的原始设计如上图所示，它来自 Xilinx 为 XUPV2P 开发板提供的参考设计[43]，本文对其略加修改，使其成为一个更加规整的模块化设计以便应用分时故障诊断。外部模拟摄像头的输入经过编码转换为数字图像后送入 FPGA 芯片内，

在片内执行图像格式转换后输出到片外，并由 DAC 转换回模拟信号并显示在监视器上。系统中 FPGA 芯片内的部分如图 29 红色虚线框所示，除了用于控制片外设备的 RISC 核和 I2C 总线控制器之外，主要包含以下几个模块：

- 行—场解码模块（Decoder）；
- YCrCb 4:2:2 到 4:4:4 色差信号补偿模块（4:2:2 to 4:4:4 Conversion）；
- YCrCb 到 RGB 色彩空间转换模块（YCrCb to RGB Conversion）；
- 视频时序产生模块（Timing Generation）；
- 隔行扫描解交错模块（De-interlacing）。

其中隔行扫描解交错模块主要由 RAM 组成。考虑到 RAM 中通常包含 BIST 电路，此处不再针对隔行扫描解交错模块做分时故障诊断。在接下来的设计中，分时故障诊断仅被应用于其余四个模块，并且对冗余面积代价的讨论也以此为基础。

5.2 系统模块的分割

本例中，原始设计本身即是模块化的，四个模块分别执行相对独立的功能，因此可以采用基于功能的划分方法分割电路，即 N_1=4。原始设计中各个模块的大小如表格 2 的第二列所示。

表格 2 原始设计模块大小(等效门数包括 IO BUFFER)

Module Name	Module Size (Equivalent Gates)	Checker Size (Equivalent Gates)	N=4	N=2
Decoder	746	787	G1	G1
4:2:2 to 4:4:4 Conversion	2,849	2,847	G1	G2
YCrCb to RGB Conversion	3,973	3,975	G1	G2
Timing Generation	1,264	936	G1	G1

根据图 17 和图 18 的分析结果，当复用同一块检测区域的模块数量 N 超过 3 时，随着 N 增大，系统的冗余面积开销下降并不明显，而永久性故障的平均检测时延呈线性增长趋势。为验证这一点，本例中设计了两套检测方案：

- 方案一：四个模块共享同一块故障检测区域，即 N=4；

- 方案二：四个模块分成两组，其中 G1 包括时序产生模块和解码模块，G2 包括两个格式转换模块，每组分别共享一块故障检测区域，即 N=2。

5.3 故障检测模块设计

首先采用基于复式系统的方法，为四个模块分别设计故障监测器，用于分时故障检测系统。考虑到四个模块中均含有大量的寄存器，例如色差信号补偿模块包含 144 个寄存器，如图 30 所示，采用状态复制比较复杂，同时导致显著的冗余面积开销。因此采用状态自动更新的方法实现原始设计模块和故障检测模块的寄存器状态同步。

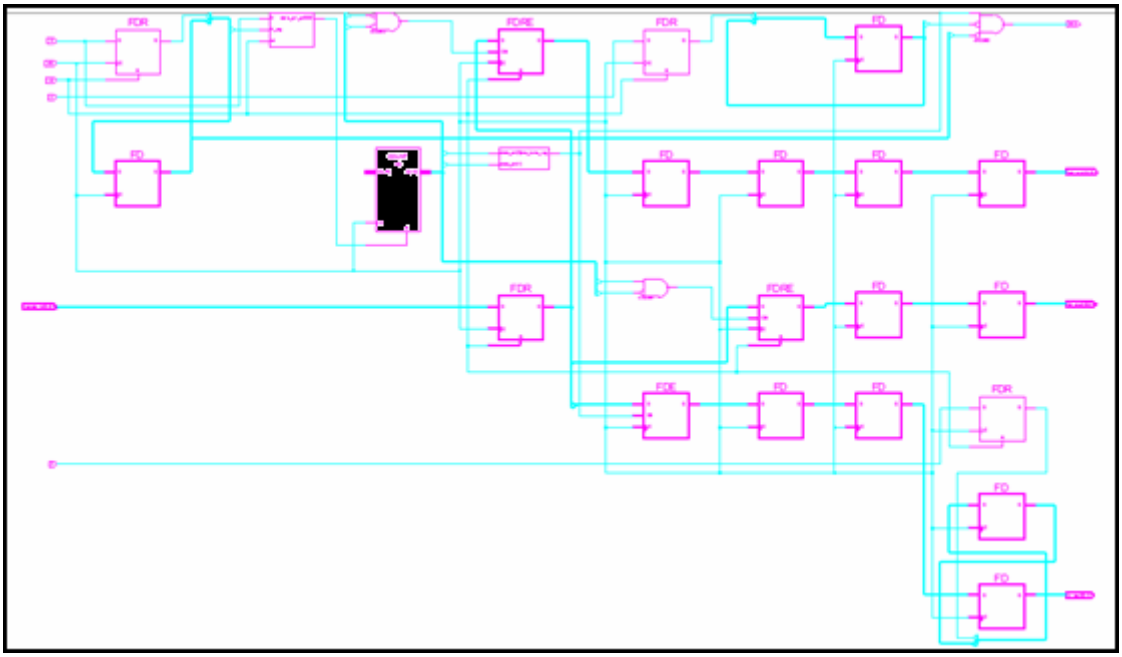

图 30 色差信号补偿模块 RTL 结构图，其中 FD、FDR、FDRE 均代表寄存器组
Fig.30 RTL Schematic of Chrome Compensation Module, FD, FDR, FDRE stand for registers

仍以图 31 中的色差信号补偿模块为例，为设计边界扫描器，首先分析此模块的数据格式。YCrCb 4:2:2 图像格式在水平方向上对每个像素点的灰度采样，而只对间隔的像素点分别采样两个色差，图像数据序列为 Y1Cr1Y2Cb1Y3Cr2Y4Cb2…，因此每个像素的需要两个字节表示。此模块的功能是保持输入的灰度信号不变，利用相邻像素的色差信号补偿当前像素缺失的色差信号，补偿后的数据序列为 Y1Cr1Cb1Y2Cr1Cb1Y3…。

由输入数据序列可知，色差信号补偿模块以 4 个连续的输入数据为一个处理单元，RTL 结构图显示，控制其状态转换的是两位计数器 state_cnt，如图 31 中黑色单元所示。只需在处理一个新单元之前根据色差信号补偿模块的 state_cnt 计数值复位故障检测模块，并根据模块的流水级数延时相应的时钟周期，即可实现状态同步，如图 31 所示。其中 rst1 为色差信号补偿模块的 state_cnt 产生的故障检测模块复位信号，y_out 和 Y_rg4 分别为色差信号补偿模块和故障检测模块输出的灰度信号。

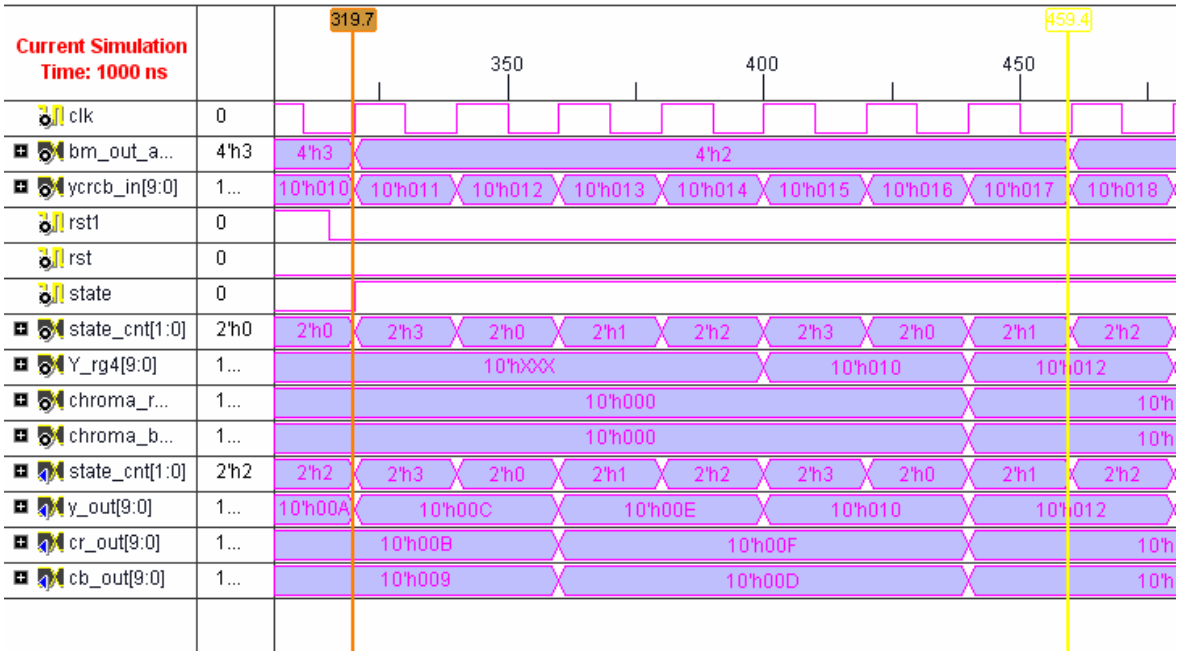

图 31 状态自动刷新时序图

Fig. 31 Timing Waveform of State Auto Update

由上例可以总结，状态自动更新的实现关键在于确定数据边界和状态转换逻辑，以及确定流水级数。

接下来针对色彩空间转换模块，采用第四章中提出的基于部分选择的方法，实现简化的故障检测器，并与采用传统的复式系统方法实现的故障检测器比较。简化的故障检测器的设计流程已经在第四章中详细讨论，这里不再重复，关键参数及结果列举如下：

- 采用的工具：Synopsys Design Compiler, TetraMAX ATPG

- 故障列表中的故障类型及数量：953 个 stack-at 故障（已消除等效故障）
- 模块输入输出位宽：12bits
- 故障仿真方式：non-scan functional test
- 测试向量数量：64
- 故障响应矩阵大小：953 * 64 * 12
- 被选择的预测位组合如下：

输入向量组	被预测位序号	输入向量组	被预测位序号
00XX0XXX0XXX	3，6，7，1	10XX0XXX0XXX	9，5，8，2
00XX0XXX1XXX	4，8，9，3	10XX0XXX1XXX	7，6，9，4
00XX1XXX0XXX	9，3，6，10	10XX1XXX0XXX	7，0，2，11
00XX1XXX1XXX	5，11，6，2	10XX1XXX1XXX	5，6，8，1
01XX0XXX0XXX	8，4，0，9	11XX0XXX0XXX	4，8，3，10
01XX0XXX1XXX	2，10，6，5	11XX0XXX1XXX	2，0，4，1
01XX1XXX0XXX	9，6，10，0	11XX1XXX0XXX	10，2，0，1
01XX1XXX1XXX	2，5，7，9	11XX1XXX1XXX	8，9，0，2

表格 3 简化的故障检测器的部分选择位列表（色彩空间转换模块）

根据表格 3 综合得到的故障检测器大小为 2457 个等效逻辑门，大约为原始模块的 61%。利用 TetraMAX 的故障仿真器对原始模块和故障检测器模块的组合执行故障注入仿真，工具自动添加了 1326 个 stack-at 故障，都能够被检测到。故障检测器的仿真波形如下，其中最下面的两条波形分别代表故障注入和检测器报告错误。

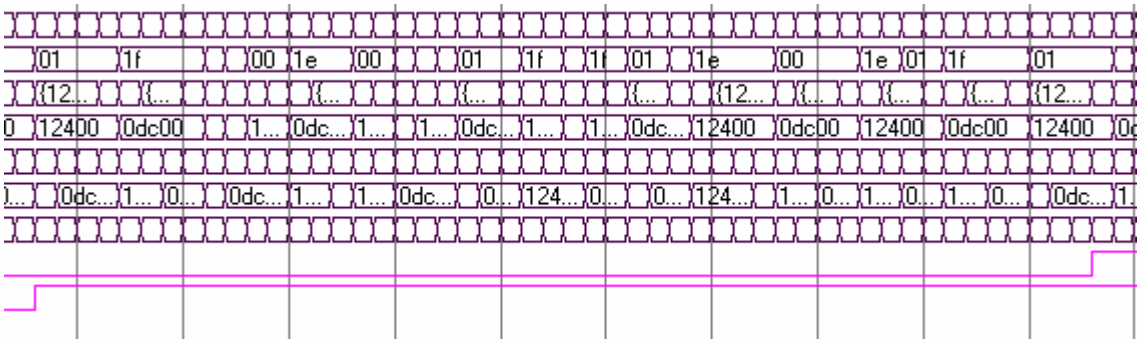

图 32 简化的故障检测器仿真波形（色彩空间转换模块）

5.4 硬件实现

本例在一块 Xilinx XUPV2P 开发板上实现硬件设计，利用 Xilinx FPGA 的动态局部重构能力执行故障检测模块的运行时重构，以实现分时故障诊断。XUPV2P 开发板搭载一块 Xilinx Virtex II Pro 30 FPGA，集成 PowerPC405 硬核，支持高速 SelectMAP 配置模式[44]和动态局部重构。在 Xilinx ISE 和 PlanAhead 工具支持下，可以很方便地采用模块化设计流程开发动态局部重构系统，实现的步骤如下：

- 功能模块设计：分别综合原始设计模块和和5.3节设计的故障检测模块的HDL代码以得到网表（Netlist）文件，其中原始设计模块作为静态模块，而故障检测模块作为可重构模块；
- 顶层模块设计：使用黑盒（Black Box）实例化所有的静态模块和一组可重构模块，定义全局逻辑和模块之间的互连线；由于可重构区域在不同的时间片上需要连接到静态区域的不同模块，全部静态模块的输入输出信号通过多路选择接口连接到可重构区域，重构的同时改变接口的位选信号以切换信号线连接，并在可重构模块和静态模块之间插入总线宏；
- 初始预估：在PlanAhead中定义时序约束和输入输出端口位置约束，评估每个模块的大小，确定模块布局，定义可重构区域，放置全局逻辑如DCM、BUFG、总线宏等；
- 模块实现与组合：分别实现可重构模块和静态模块，将需要初始下载的可重构模块和静态模块进行组合，生成初始配置比特流和多个重构比特流；
- 验证设计：利用仿真工具进行静态时序分析和功能仿真，通过FPGA Editor进行可视化检测，保证布线没有超出模块边界，如图33所示。
- 下载比特流：下载初始配置比特流到目标板可以启动系统，下载重构比特流以实现动态局部重构；

至此即在Xilinx Virtex II Pro 30 FPGA上实现了带有分时故障诊断设计的视频监控系统，如图34所示，其中可重构区域用红色标识。

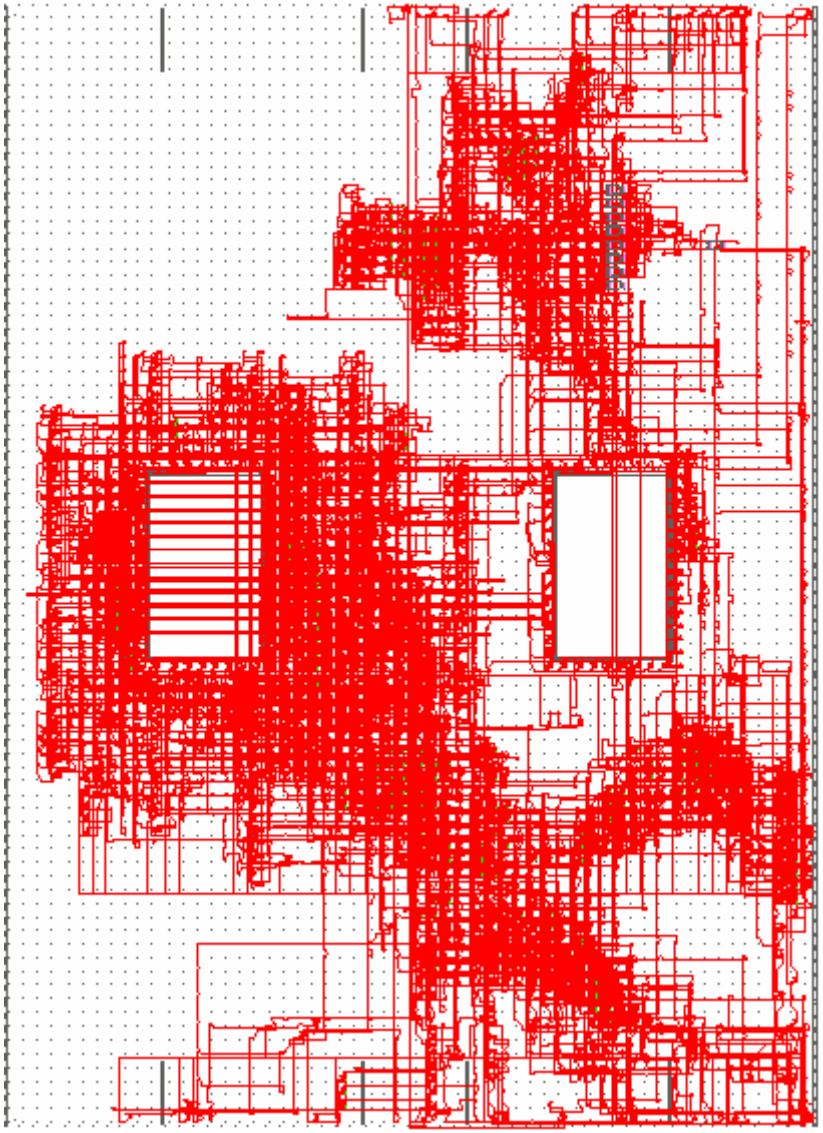

图 33 FPGA 布线图（N=1）

Fig.32 FPGA Routing

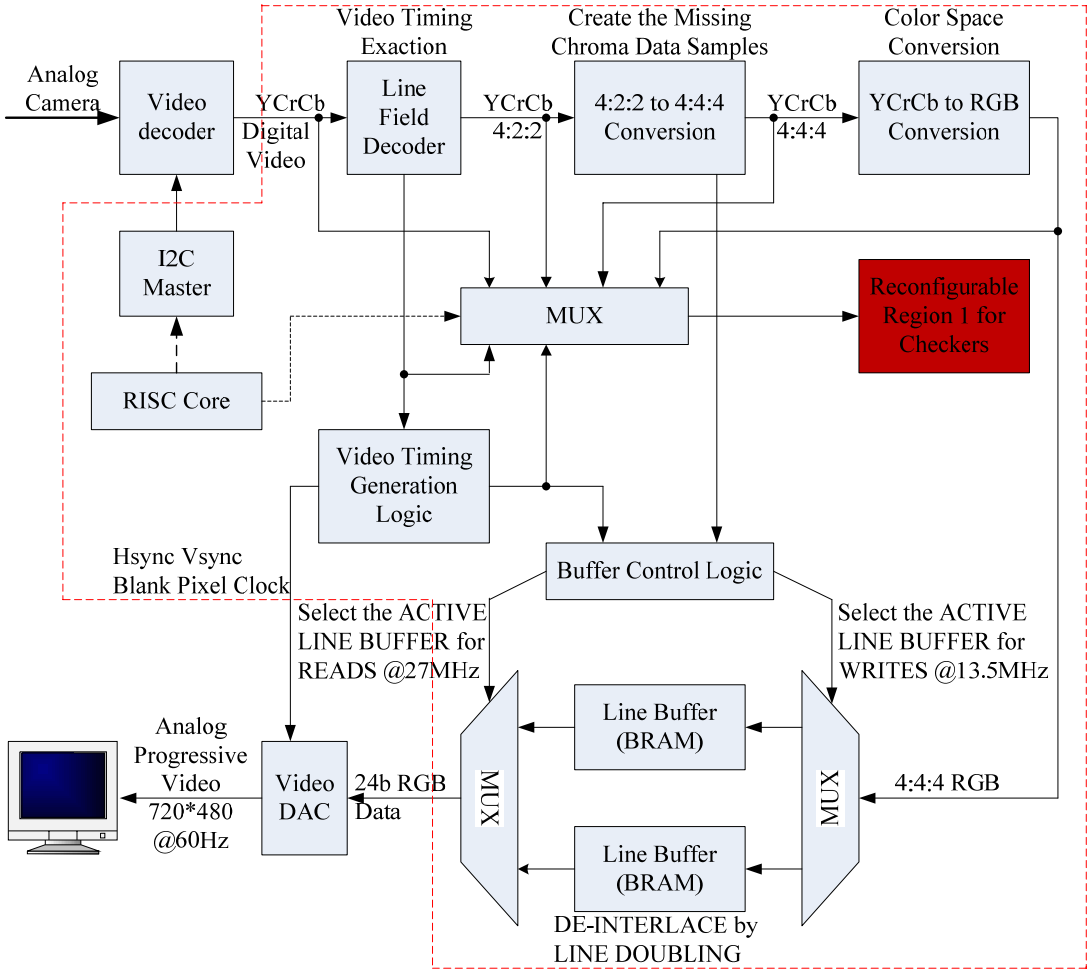

图 34 应用了分时故障诊断的视频监控系统

Fig.33 Video Capture with TFD Applied

5.5 时间片调度

本例中，重构比特流存储于 FPGA 片外的 Flash 中。当一次局部重构被调度时，FPGA 在片内的 RSIC 核控制下，从 Flash 中读取配置数据，采用 SelectMAP 模式配置可重构区域，实现故障检测模块的替换，如图 35 所示。对于不包含 RISC 核的应用，也可以以很低的代价借助 PROM 或 CPLD 或自定制的电路配置 FPGA，具体实现参考文献[45]。

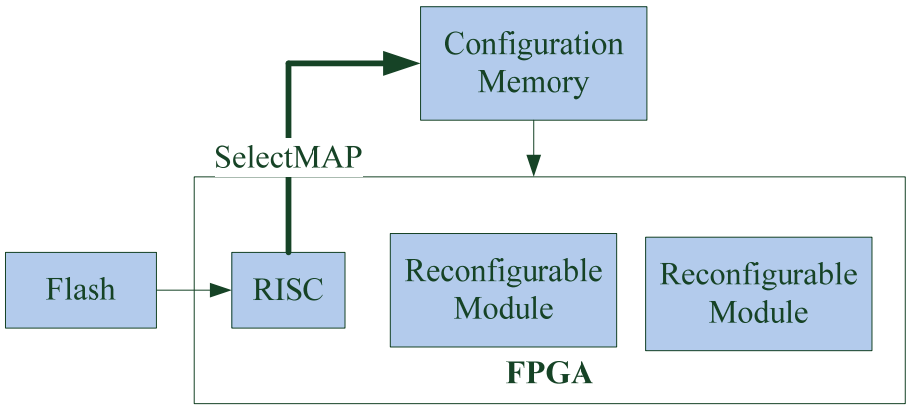

图 35 FPGA 配置示意图

Fig.34 FPGA Configuration Scheme

故障检测模块在时间域上的调度由运行于 RISC 核上的软件控制，因而可以在硬件配置确定后更改。为便于与理论分析结果比较，本例中采用 Round-Robin 调度算法，并对每个故障检测模块分配相同长度的时间片，分别测试 T_U 为 2×10^5，2×10^6，2×10^7 三种情况下的平均故障检测延时。

5.6 测试结果及分析

本节将从系统的冗余面积开销和对间歇性故障的平均检测延时两个方面评估分时故障诊断，并与原有的并发故障检测策略对比。

表格 4 冗余面积开销（分时故障检测系统）

Area Overhead	N=4(excluding MUX)	N=4(Reduced Checker)	N=4	N=2	N=1
equivalent gate count	10864/7185	10077/7185	11591/7185	12341/7185	15692/7185
	51.2%	40%	61.3%	71.8%	118.4%

5.6.1 冗余面积开销

如表格 4 所示，当采用基于复式系统的故障检测器时，与原有的并发故障检测策略（N=1）相比，分时故障诊断在冗余面积开销上有明显优势。在本例中，N=4 时的冗余面积开销约等于 N=1 时的一半，且明显优于 N=1 的系统的理论最优值 110%。

当 N 较小时，冗余面积开销随着 N 的增加而下降，但并不与 N 的倒数成正比（N=4 *vs* N=2），显然多路选择接口限制了冗余面积开销的进一步下降。由表格 4 的第二列看出，当 N=4 时，多路选择接口的影响已经不能忽略。由于本例的系统规模较小，此影响表现的尤为明显（大约 10%的冗余面积开销）。

当采用简化的故障检测器时，冗余面积开销进一步降低至 40%。

5.6.2 平均故障检测延时

为验证系统的故障检测性能，可以利用外部输入信号将电路中某个数据位或控制信号强制置位为逻辑“0”或者“1”，模拟“stuck-at-0”或“stuck-at-1”故障。本例中在进入色彩空间转换模块的灰度信号的最高位 Y[9]上导入 stuck-at-0 故障，如图 36 所示。验证中采用的故障模型有两种，一种是永久性故障，另一种是以 2×10^6 个时钟为周期，活跃时间占 50%的间歇性故障。由于 Y[9]是模块输入端的数据通路，每个输入向量有 50%的概率导致模块输出数据出现错误，因此对于每个输入向量，能够检测到故障的几率为 50%。

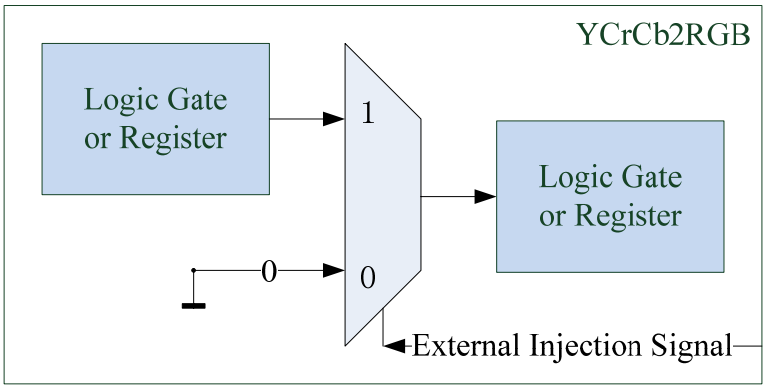

图 36 YCrCb 模块中插入 stuck-at-0 故障
Fig. 35 Inject a stuck_at_0 fault to YCrCb

根据图 36 修改 5.4 节实现的电路，并下载到 FPGA 芯片中，通过开发板上的拨动开关控制故障的导入，并将故障报警信号引出到 LED 上，经过多次实验，系统在显示终端的输出始终能够真实反映摄像头所监控的场景，证明动态重构过程对系统的

正常工作状态完全没有形象。而无论是永久性故障还是间歇性故障，系统总是能够在故障导入后不太长的时间内报警，证明了分时故障诊断系统具有可靠的检测能力。

为测试故障检测延时，本例在电路中加入硬件的计时器（如图 37 所示），自永久性故障被导入后开始计数，直到报警信号有效时计数停止，根据开发板的 LED 状态观察到预警信号有效后，利用 Xilinx 提供的 RISC 核的 debug 工具 XMD 读取计数值，即可得到以时钟周期数表示的故障检测延时，如表 5 所示。

表 5 平均故障检测延时

T_{DL}(clock cycles)	$T_U=2\times10^5$	$T_U=2\times10^6$	$T_U=2\times10^7$
N=1	5	5	5
N=2(average on 100 times)	127,596	1,008,450	9,951,674
N=4(average on 100 times)	332,822	3,166,916	28,232,956

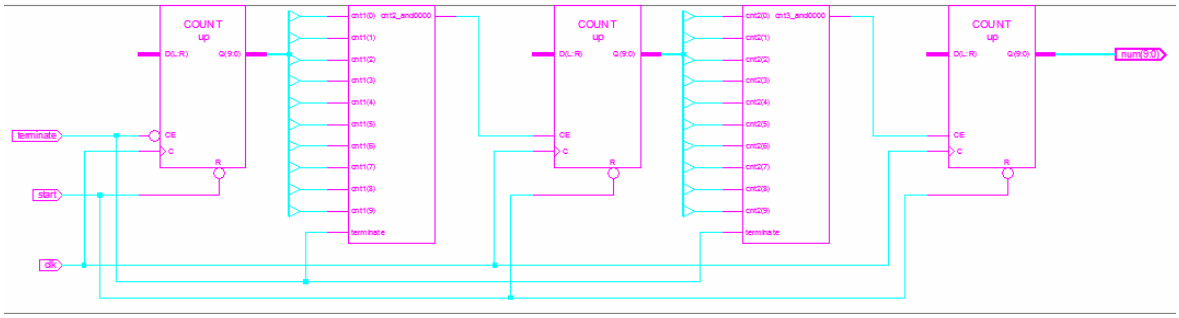

图 37 硬件计时电路

Fig 36. Hardware Timer

表 5 的第一行为 N=1 时的平均故障检测延时，代表了检测模块本身需要多少个有效的输入向量才能检测到故障，对应于公式(6)和(9)中的第二项。由于系统对主要数据通路上的 stuck-at-0 故障非常敏感，并且此模块的逻辑级数较少，因此检测延时很小，如图 38 所示。其中 inject 为故障导入控制信号，低电平代表导入故障，error 为故障报警信号，低电平代表检测到故障，从图中可以看出 stuck-at-0 故障导致的错误在模块内部传播的过程。

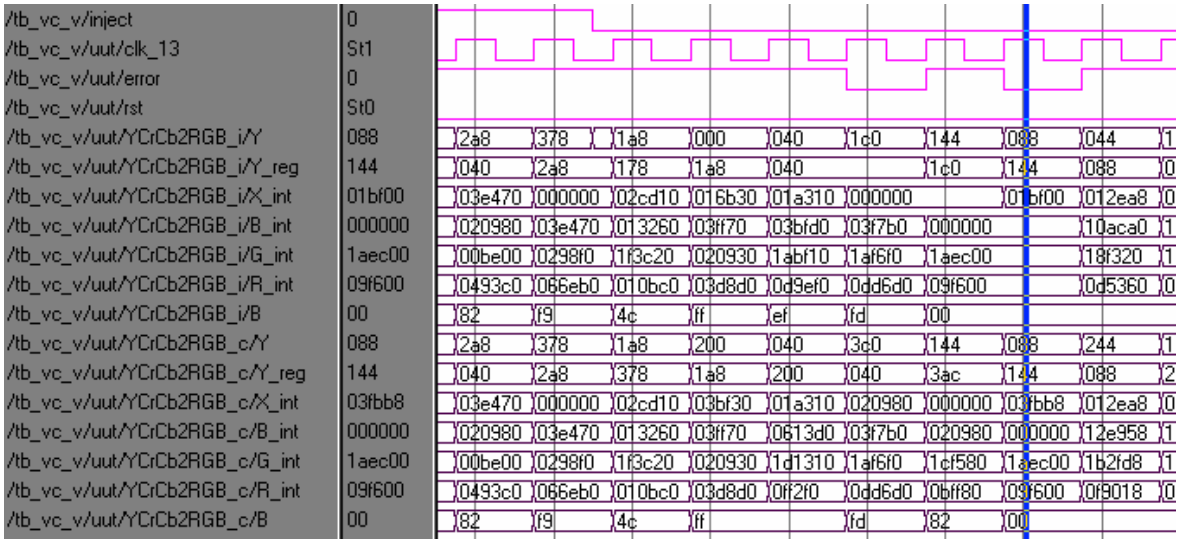

图 38　N=1 时的故障检测延时

Fig. 37 Fault Detection Latency when N=1

表 5 的第二行和第三行是采用分时故障诊断方法的检测延时，主要由等待时间组成，对应于公式(6)和(9)的第一项。由于测试的样本数只有 100 个，得到的平均检测延时与理论分析结果尚有一定的偏差，但基本上能够验证公式(9)所描述的函数关系。同时，对于常见的时钟频率为 100MHz 量级的系统，分时故障诊断的平均检测延时不足 0.5 秒，绝大多数应用都可以容忍这个延时。

5.7 本章小结

本章在之前各章介绍的背景知识基础上，选取了一个小规模的视频监控系统，根据分时故障诊断的开发流程，利用第三章提出的概率模型预估故障检测系统的设计参数，借助 Xilinx FPGA 芯片的动态局部重构特性，在一块 Virtex II Pro 30 FPGA 芯片上实现了分时故障诊断在实际系统中的应用。经过软件仿真和上板调试，证明分时故障诊断在冗余面积开销、故障检测能力及平均故障检测延时方面大致符合理论分析的结果，并且均达到了较好的性能。

第六章 总结与展望

随着集成电路工艺的进一步发展和数字集成电路的系统规模成倍增加，数字集成电路中的故障来源越来越多，故障发生率也迅速提高。为应对数字集成电路中日益严重的可靠性问题，电路的并发故障检测能力的重要性与日俱增。但现有的并发故障检测方法成本代价过高，不适用于对成本敏感的应用中。为解决这一问题，本文研究了数字集成电路中并发故障检测的低成本实现策略。

6.1 主要创新点

本文主要以具有动态局部重构能力的 FPGA 芯片为系统的研究和实现平台，但本文提出的分时故障诊断方法可以推广到所有具有可重编程能力的器件和系统中，如 Multi-FPGA、SIP、MPSOC 等。本文的主要创新点包括以下几点：

本文将 FPGA 的动态局部重构特性应用到电路的在线故障检测中，提出了一种新的、适合对成本敏感的应用的在线故障检测策略，它在没有明显降低系统的故障检测能力的前提下，显著降低了故障检测的冗余面积开销，同时其平均故障检测延时的增加在可接受的范围内。

本文分析了分时故障诊断系统设计中需要折衷考虑的因素，提出了一个针对分时故障诊断系统的概率模型，用以估算在不同的故障特性和设计参数下系统的故障检测能力和平均检测延时，给出了设计参数与检测性能之间的近似函数关系，并在此基础上提出了详细的开发流程。

本文将分时故障诊断应用到实际系统中，在一块 Virtex II Pro 30 FPGA 芯片上实现了对视频监控系统的分时故障诊断，验证了之前做出的理论分析和数学推导，证明了分时故障诊断的可行性和现实性，并通过实际的测试数据，进一步说明了分时故障诊断在对成本敏感的应用中的巨大优势。

6.2 分析和展望

同时，本文也存在一定的不足之处，主要有以下两点：

首先，开发实例中采用的系统规模较小，仅能粗略的验证模块分割策略对系统性能和实现代价的影响，未能得到更加精确的公式或曲线。若要采用大规模的系统，则需要借助 EDA 工具完成，而目前未能获得此类工具。

其次，当前数字集成电路中的多重故障尤其是 CMF 出现的几率越来越频繁。CMF 的产生机制比较复杂，其建模和仿真工作非常困难且复杂，因此本文的验证工作只能针对单重故障进行。但考虑到其特性，对 CMF 的检测能力理论上只与故障检测模块所采用的算法有关，因此不会对分时故障诊断系统的性能产生额外的影响。

综上所述，分时故障诊断的应用实例已经在支持动态局部重构的 FPGA 芯片实现，其代价和检测性能也得到了初步的验证。下一步的工作可以从两个方面入手，其一是在更大规模的系统上应用分时故障诊断，重点验证不同的模块分割方式对系统代价和性能的影响，并且采用更加完善的故障注入仿真验证系统的故障检测能力；其二是将分时故障诊断应用到 MPSOC 平台上，并进一步分析其在系统功耗方面的代价，同时便于在更多的实际产品中应用。

参 考 文 献

[1] International Technology Roadmap for Semiconductors, 2008 edition, http://public.itrs.net.

[2] Eric Chmelar, The Test and Diagnosis of FPGAS, Dissertation for the degree of Doctor of Philosophy, Stanford University, 2004.

[3] 杨士元，数字系统的故障诊断与可靠性设计（第二版），北京：清华大学出版社，2000.

[4] M. Abramovici, C. E. Stroud, and J. M. Emmert, Online BIST and BIST-Based Diagnosis of FPGA Logic Blocks, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Dec. 2004, VOL. 12, NO. 12, pp. 1284-1294.

[5] M. Abramovici and C. E. Stroud, On-Line BIST and Diagnosis of FPGA Interconnect Using Roving STARs, Proceedings of the 7th IEEE International Workshop for On-Line Testing, 2001, pp. 27-33.

[6] M. Abramovici, C. E. Stroud, B. Skaggs, and J. M. Emmert, Improving On-Line BIST-Based Diagnosis for Roving STARs, Proceedings of the 6th IEEE International Workshop for On-Line Testing, 2000, pp. 31-39.

[7] W. J. Huang and E. J. McCluskey, Column-Based Precompiled Configuration Techniques for FPGA Fault Tolerance, Proceedings of the 9th Annual IEEE Symposium on Field Programmable Custom Computing Machines, 2001, pp. 137-146.

[8] J. Lach, W. H. Mangione-Smith, and M. Potkonjak, Enhanced FPGA Reliability Through Efficient Run-Time Fault Reconfiguration, IEEE Transactions on Reliability, Sep. 2000, VOL. 49, NO. 3, pp. 296-304.

[9] W. J. Huang, S. Mitra, and E. J. McCluskey, Fast Run-Time Fault Location in Dependable FPGAs, Proceedings of the 16th IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems, 2001.

[10] M. K. Stojcev, G. L. Djordjevic, and T. R. Stankovic, Implementation of Self-checking Two Level Combinational Logic on FPGA and CPLD Circuits, Microelectronics Reliability, 2004, VOL. 44, pp.173-178.

[11] C. Zeng, N. Saxena, and E. J. McCluskey, Finite State Machine Synthesis with Concurrent Error Detection, Proceedings of the 1999 IEEE International Test Conference, 1999, pp. 672-679.

[12] D. P. Siewiorek, and R. S. Swarz, Reliable Computer Systems: Design and Evaluation, 2nd Edition, Digital Press, 1992.

[13] S. Habinc, Functional Triple Modular Redundancy: VHDL Design Methodology for Redundancy in Combinatorial and Sequential logic, NASA Office of Logic Design, 2002.

[14] J. Biernat, The Effect of Compensating Faults Models on NMR System Reliability, IEEE Transactions on Reliability, June 1994, VOL. 43, NO. 2, pp. 294-300.

[15] R. Kohno, S. Pasupathy, H. Imai, and M. Hatori, Robust ADPCM System Using an Error Correcting Code, IEEE International Conference on Acoustics, Speech, and Signal Processing, 1986.

[16] D. Sokolov, J. Murphy, A. Bystrov, and A. Yakovlev, Design and Analysis of Dual-rail Circuits for Security Applications, IEEE Transactions on Computers, 2005, VOL.54, NO. 4, pp. 449-460.

[17] G. C. Cardarilli, S. Pontarelli, M. Re, and A. Salsano, Design of a Self-checking Reed-Solomon Encoder, Proceedings of the 11th IEEE International Symposium on Online Testing, 2005, pp. 201-202.

[18] S. Mitra, and E. J. McCluskey, Which Concurrent Error Detection Scheme to Choose, Proceedings of the 2000 IEEE International Test Conference, 2000, pp. 985-994.

[19] Altera Inc., Altera Mega Core Functions, http://www.altera.com/html/tools/megacore.html, San Jose, CA, 1998.

[20] S. Trimberger, D. Carberry, A. Johnson, and J. Wong, A Time-Multiplexed FPGA, Proceedings of IEEE Symposium on FPGAs for Custom Computing Machines, 1997, pp. 22-28.

[21] S. Hauck, T. Fry, M. Hosler, and J. Kao, The Chimaera Reconfigurable Functional Unit. IEEE Transactions on Very Large Scale Integration Systems, 2004, VOL. 12, NO. 2, pp. 206-217.

[22] Xilinx Inc., Xilinx Application Notes 290: Difference-based Partial Reconfiguration, http://www.xilinx.com/support/documentation/application_notes/xapp290.pdf, San Jose, CA, 2007.

[23] P. Sedcole, B. Blodget, T. Becker, J. Anderson and P. Lysaght, Modular Dynamic Reconfiguration in Virtex FPGAs, Proceedings of the 2006 IEEE Symposium on Computing and Digital Technology, 2006, VOL. 153, No. 3, pp. 157-164.

[24] F. Vhid, T. D. Le, and Y. Hsu, A Comparison of Functional and Structural Partitioning, Proceedings of IEEE International Symposium on System Synthesis, 1996, pp. 121-126.

[25] S. Kamal and V.V. Page, Intermittent Faults: A Model and a Detection Procedure, IEEE Transactions on Computers, July, 1974, VOL. C-23, No. 7, pp. 713-719.

[26] A.A. Ismaeel and R. Bhamagar, Test for Detection and Location of Intermittent Faults in Combinational Circuits, IEEE Transactions on Reliability, June, 1997, VOL. 46, No. 2, pp. 269-274.

[27] P. Lysaght, B. Blodget, J. Mason, J. Young, and B. Bridgford, Invited Paper: Enhanced Architectures, Design Methodologies and CAD Tools for Dynamic Reconfiguration of Xilinx FPGAS, IEEE International Conference on Field Programmable Logic and Applications, 2006.

[28] W. J. Huang, N. R. Saxena, and E. J. McCluskey, A Reliable LZ Data Compressor on Reconfigurable Coprocessors, Proceedings of IEEE Symposium on Field Programmable Custom Computing Machines, 2000.

[29] J. Y. Jou, and J. A. Abraham, Fault-Tolerant FFT Networks, IEEE Transactions on Computers, May 1988, Vol. 37, No. 5, PP. 548-561.

[30] F. Sellers, M. Y. Hsiao, and L. W. Bearnson, Error Detection Logic for Digital Computers, McGraw-Hill Book Company.

[31] R. M. Sedmak, and H. L. Liebergot, Fault-Tolerance of a General-Purpose Computer Implemented by Very Large Scale Integration, Proceedings of IEEE Symposium on Fault Tolerance Computing, 1978, PP. 137-143.

[32] G. D. Kraft, and W. N. Toy, Micro-programmed Control and Reliable Design of Small Computers, Prentice Hall, 1981.

[33] K. De, C. Natarajan, D. Nair, and P. Banerjee, RSYN: A System for Automated Synthesis of Reliable Multi-level Circuits, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, June 1994, VOL. 2, pp. 186-195.

[34] N. A. Touba, and E. J. McCluskey, Logic Synthesis of Multi-level Circuits with Concurrent Error Detection, IEEE Transactions on Computer Aided Design, July 1997,

VOL. 16, pp. 783-789.

[35] N. K. Jha, and S. J. Wang, Design and Synthesis of Self-Checking VLSI Circuits, IEEE Transactions on Computer Aided Design, June 1993, VOL. 12, pp. 878-887.

[36] M. A. Marouf, and A. D. Friedman, Design of Self-checking Checkers for Berger Codes, Proceedings of IEEE Symposium on Fault Tolerance Computing, 1978, pp. 179-184.

[37] B. Bose, B. and D. J. Lin, Systematic Unidirectional Error-Detecting Codes, IEEE Transactions on Computers, Nov. 1985, pp. 1026-1032.

[38] D. Das, N. A. Touba, Synthesis of circuits with low-cost concurrent error detection based on Bose-Lin codes, Proceedings of the 16th IEEE Symposium on VLSI Test, Apr. 1998, pp. 309-315.

[39] N. K. Jha, Totally self-checking checker designs for Bose-Lin, Bose and Blaum codes, IEEE Transactions on Computer Aided Design, Jan. 1991, VOL. 10, pp. 136-143.

[40] E. J. McCluskey, Logic Design Principles, Prentice-Hall, 1986.

[41] S. Mitra, and E. J. McCluskey, Combinational Logic Synthesis for Diversity in Duplex Systems, Proceedings of the 2000 IEEE International Test Conference, 2000, pp. 179-188.

[42] Xilinx Inc., Xilinx University Program Virtex-II Pro Development System Hardware Reference Manual, http://www.xilinx.com/univ/XUPV2P/Documentation/ug069.pdf, San Jose, CA, 2008.

[43] Xilinx Inc., Xilinx XUPV2P Reference Designs and Demonstrations: Video Decoder using VDEC-1, http://www.xilinx.com/univ/xupv2p_demo_ref_designs.htm, San Jose, CA, 2008.

[44] Xilinx Inc., Using a Microprocessor to Configure Xilinx FPGAs via Slave Serial or SelectMAP Mode, http://www.xilinx.com/support/documentation/application_notes/xapp502.pdf, San Jose, CA, 2008.

[45] Xilinx Inc., Xilinx Application Notes 138: Virtex FPGA Series Configuration and Readback, http://www.xilinx.com/support/documentation/application_notes/xapp138.pdf, San Jose, CA, 2008.

[46]S. Almukhaizim, P. Drineas, Y. Makris, Cost-Driven Selection of Parity Trees, Proceedings of the 22nd IEEE VLSI Test Symposium, 2004, pp. 319-324.

致　谢

随着论文工作的完成，研究生学习也即将结束了，在此谨向辛勤培育过我的老师，和所有关心过我的同学、朋友、亲人致以最崇高的敬意和诚挚的感谢。

首先感谢我的导师施国勇教授在论文过程中的指导和帮助。施老师不仅学识渊博，具有广博的理论和实践知识，而且治学严谨、平易近人、关心学生，在平时的学习和研究中提出方向性的指导意见十分深刻，从施老师身上得到的不仅是知识，而且是治学的态度。感谢祝永新老师在开题答辩中提出的宝贵意见。感谢所有为我们的学习创造良好环境而辛勤工作的老师们。

感谢 Xilinx 大学计划部的谢凯年博士在开题前期对我的指导，在课题研究过程中提供软硬件支持。

感谢 EDA 项目组的所有同学，和他们在一起让我学到了各方面的知识，在遇到困难时大家都会热心帮助，给我以关怀。感谢 402 宿舍的全体成员，感谢所有的同学，陪我一起走过了美好的两年半。

最后，感谢我的父母、亲人和朋友在学习过程中一直的支持和鼓励，是他们的信任和关心让我在学习中有无穷的动力。

最诚挚的祝福给所有给予我帮助的人！

攻读硕士学位期间已发表或录用的论文

[1]第一作者，“基于动态局部重构的 HCRP 协议的研究”，电子技术应用，已录用。