中图分类号：TN407　　论文编号：1028704　22-SZ067
学科分类号：085209

硕士学位论文

SRAM 型 FPGA 应用相关互连测试与容错的关键技术研究

研究生姓名　　毛志明
专 业 类 别　　工程硕士
专 业 领 域　　集成电路工程
指 导 教 师　　张颖 讲师

南京航空航天大学

研究生院 电子信息工程学院

二〇二一年十二月

Nanjing University of Aeronautics and Astronautics
The Graduate School
College of Electronic and Information Engineering

Research on Application-Dependent Testing and Tolerant of SRAM-based FPGA Interconnects

A Thesis in
Integrated Circuit Engineering

by

Mao Zhiming

Advised by

Assistant Prof. Zhang Ying

Submitted in Partial Fulfillment

of the Requirements

for the Degree of

Master of Engineering

December, 2021

承诺书

本人声明所呈交的硕士学位论文是本人在导师指导下进行的研究工作及取得的研究成果。除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得南京航空航天大学或其他教育机构的学位或证书而使用过的材料。

本人授权南京航空航天大学可以将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、汇编学位论文。

（保密的学位论文在解密后适用本承诺书）

作者签名：_____

日　　期：_____

摘 要

随着集成电路的快速发展，数字系统的规模越来越大，为了抢夺产品市场，拥有开发周期短、可重复编程以及内部资源丰富等诸多特点的现场可编程门阵列（Field Programmable Gate Array，FPGA）器件逐渐成为开发者的首选。而随着 FPGA 的应用越来越广，对其本身的可靠性要求也在不断提高，其中由于 FPGA 不同工艺的特点使得 SRAM（Static Random Access Memory）型 FPGA 的可靠性被广泛关注与重视。而且随着电路的高度集成化，SRAM 型 FPGA 内部的互连资源出现故障的概率越来越大，因此研究 SRAM 型 FPGA 互连资源的可靠性，对提高 FPGA 的可靠性以及保障相应应用系统的稳定性具有重要意义。

本文主要基于应用相关的思想对 FPGA 互连资源的测试、诊断以及容错方法进行了研究，本文主要工作包含以下几方面：

1）提出了一种基于改进故障模型的应用相关互连测试方案。该方案约束桥接故障只发生在同一查找表（Look Up Table，LUT）的信号线之间，并将反馈桥接故障细分为影响输出型和反馈输入型，最后使用布尔可满足性理论（Boolean Satisfiability，SAT）对优化后的故障模型进行约束，生成相应的测试配置进行测试。

2）在测试的基础上，提出了一种基于故障粗细粒度的非自适应互连诊断方案。该方案通过分析电路的互连情况，并结合测试阶段的错误端口输出信息判断出故障所在路径，再针对不同路径对故障所在范围进行粗粒度诊断，最后复用测试阶段的配置对故障粗粒度范围内的信号线进行细粒度精准诊断定位。

3）在诊断的基础上，提出了一种基于细粒度冗余的修复型互连容错方案。该方案将诊断定位到的互连故障映射到相邻位置的逻辑资源上，对该逻辑资源进行基于 LUT 的细粒度复制，再使用两级异或电路实现对故障的检错纠错，修复电路中存在的互连故障。

在 ISCAS'89 基准电路中模拟注入互连故障并进行仿真，实验结果验证了上述方案的有效性。而后对所提出方案的相应参数进行分析评估，并与现有方案进行对比，测试方案中生成的测试配置在覆盖电路中所有互连故障的前提下实现了测试配置数的最小化；诊断方案在相同的诊断精度下实现了诊断复杂度的降低以及相应诊断配置数的减少；容错方案直接处理存在的互连故障，只需增加少量的面积就能恢复电路的正常工作。由此可见，本文所提出的方案具有有效性和优越性，有利于提高 FPGA 的可靠性及其应用系统的稳定性。

关键词：SRAM 型 FPGA，互连测试，应用相关测试，故障诊断，故障容错

ABSTRACT

The rapid development of integrated circuits has resulted in an increase in the scale of digital systems. In order to seize the product market, Field Programmable Gate Arrays (FPGAs) have rapidlly become the main choice of developers because to their numerous advantages such as a shorter design cycle, configurablility and extensive internal resources. However, as FPGAs become more widely used, the requirements for their dependability increase, and the reliability of SRAM based FPGA is especially concerned and valued due to the unique properties of different FPGA processes. With the high integration of circuits, the probability of interconnect failures in SRAM based FPGA is also increasing, it is critical to analyze the reliability of the interconnect networks in SRAM based FPGA in order to improve the FPGA's reliability and assure the stability of the related digital system.

In this thesis, the reliability research of FPGA interconnect networks is studied using the application-dependent concept, the main contents of this thesis include the following:

1. An application-dependent interconnect testing strategy which based on an enhanced fault model is proposed. The strategy limits the occurrence of bridging faults to the nets of a single LUT, and subdivides the model of feedback bridging faults into types of influence output and feedback input. Finally, the Boolean Satisfiability (SAT) constraint is employed to build the optimized fault model and generate corresponding test configurations.

2. Following the interconnects testing in the preceding stage, a coarse-grain and fine-grain non-adaptive interconnect fault diagnosis scheme is proposed. Firstly, the faulty path is determined by examining the interconnection of the circuit under test and merging the information from error output port of the test phase. The path is next subjected to a coarse-grained diagnosis in order to to narrow down the fault location. Finally, the fault detection configurations are reused to precisely locate the faulty net within the coarse granularity.

3. After detecting and diagnosing the issue, a fault tolerant method based on fine-grained redundancy is proposed for repairing the located interconnect fault. Firstly, the located interconnection fault is mapped to the neighboring logical resource, followed by a fine-grained redundant replication of the nearby logical resources using the Look-up Table. Finally, a correction circuit comprised of two XOR operations is introduced to repaire the interconnect fault and restore the normal functionality of the circuit.

Simulating the insertion of interconnect faults into ISCAS'89 benchmark circuits is carried out to verify the effectiveness of the related solutions. The data of the proposed schemes are then studied and evaluated, and they are compared to existing schemes. The results indicate that the test configurations generated in the proposed testing scheme minimizes the number of test configurations required to cover all interconnect faults. The proposed diagnosis scheme simplified diagnosis and reduced the number of diagnosis configurations while maintaining the same diagnostic precision, and the proposed fault tolerant scheme directly addresses the interconnect fault, requiring only a small amount of area consumption to restore normal operation. Therefore, the proposed schemes are effective and advantageous, which benefits the FPGA's reliability and the stability of the related digital system.

Keywords: SRAM-based FPGA, Interconnect testing, Application-dependent testing, Fault diagnosis, Fault tolerance

目 录

图表清单

缩略词

缩略词	英文全称
FPGA	Field Programmable Gate Array
SRAM	Static Random Access Memory
LUT	Look Up Table
SAT	Boolean Satisfiability
ISCAS	International Symposium on Circuits and Systems
SSI	Small Scale Integration
MSI	Medium Scale Integration
ASIC	Application Specific Integrated Circuit
PLD	Programmable Logic Devices
PAL	Programmable Array Logic
GAL	Generic Array Logic
CPLD	Complex Programmable Logic Device
ASSP	Application Specific Standard Product
DSP	Digital Signal Processor
DCM	Digital Clock Management
IP	Intellectual Property
IR	Interconnect Resources
BIST	Built In Self-Test
ATE	Auto Test Equipment
CLB	Configurable Logic Block
SM	Switch Matrix
SMT	Satisfiability Modulo Theory
TPG	Test Pattern Generator
ATPG	Automatic Test Pattern Generation
IOB	Input/Output Block
CMOS	Complementary Metal-Oxide-Semiconductor Transistor
NCD	Native Circuit Description
XDL	Xilinx Design Language

ICAP	Internal Configuration Access Port

第一章 绪论

1958 年，集成电路之父杰克·基尔比（Jack Kilby）研制出世界上第一块集成电路，开创了电子技术历史的新纪元，英特尔创始人罗伯特·诺伊斯（Robert Noyce）在基尔比的基础上发明了可商业生产的集成电路，使半导体产业由“发明时代”进入了“商用时代”，从此集成电路迅速发展，由小中规模集成电路，发展到超大规模，甚至是专用集成电路。而无论是小规模集成电路（Small Scale Integration，SSI），中规模集成电路（Medium Scale Integration，MSI），或者是专用集成电路（Application Specific Integrated Circuit，ASIC），其功能都是固定的，可编程逻辑器件（Programmable Logic Devices，PLD）则别开生面，在制造以后可以通过编程呈现出不同的逻辑功能[1]。早期的可编程逻辑器件主要有可编程阵列逻辑（Programmable Array Logic，PAL）和通用阵列逻辑（Generic Array Logic，GAL），随着集成电路器件的更新换代，出现了 FPGA 与复杂可编程逻辑器件（Complex Programmable Logic Device，CPLD），由于工艺的难度差异，CPLD 集成度较低，其结构更适合完成复杂的组合逻辑，而 FPGA 集成度更高，且更适合实现复杂的时序逻辑，现如今电路设计规模越来越大，动辄上万或几十万的寄存器规模，因此久而久之 FPGA 便成为了开发者的首选器件。

1.1 研究背景与意义

从 1985 年赛灵思（Xilinx）公司发明了世界上第一片 FPGA 开始，经过 36 年的高速发展，FPGA 因其独特的特性在半导体市场占据一席之地，甚至与 ASIC、ASSP（Application Specific Standard Product）三分天下，据市场统计数据，2020 年，全球 FPGA 的年销售额已超过 60 亿美元，预计到 2025 年，全球市场将超过 120 亿美元[2]。FPGA 能够逐步侵蚀 ASIC 的传统市场，主要在于 FPGA 做为一种半定制电路，既解决了定制电路的不足，又克服了原有可编程器件门电路有限的缺点。如图 1.1 所示，FPGA 开发周期短，开发流程简化，开发人员能通过软件更改片上程序，无需承受替换或重新设计芯片的巨大时间成本，而全定制电路 ASIC 开发周期长，工序复杂精密，芯片功能固化后可重配置功能有限，使用 FPGA 进行开发的时间比 ASIC 设计周期平均减少 55%，在激烈的市场竞争下，往往产品的上市时间比产品的价格更重要，产品上市时间越晚，市场份额损失得就越多。

FPGA 目前在国内被大量使用，利用 FPGA 做原型设计及前期样品生成的做法已逐渐成为“必要”的手段，高性价比、可重配置、开发周期短以及内部资源丰富等优点使得 FPGA 拥有广泛的市场，如手机通信所需的基站、旅游出行所需的汽车、健康体检所需的仪器等，在国防、航空航天中也都有广泛使用。FPGA 作为可编程逻辑器件的杰出代表，越来越受人们的重视，因而对其可靠性的要求也就越来越高。一旦基于 FPGA 的数字系统出现故障，无论是通信、汽车电子、医疗领域，还是国防、航空航天领域，都将会造成难以估计的伤害

与损失，因此研究 FPGA 的可靠性是非常有必要的。

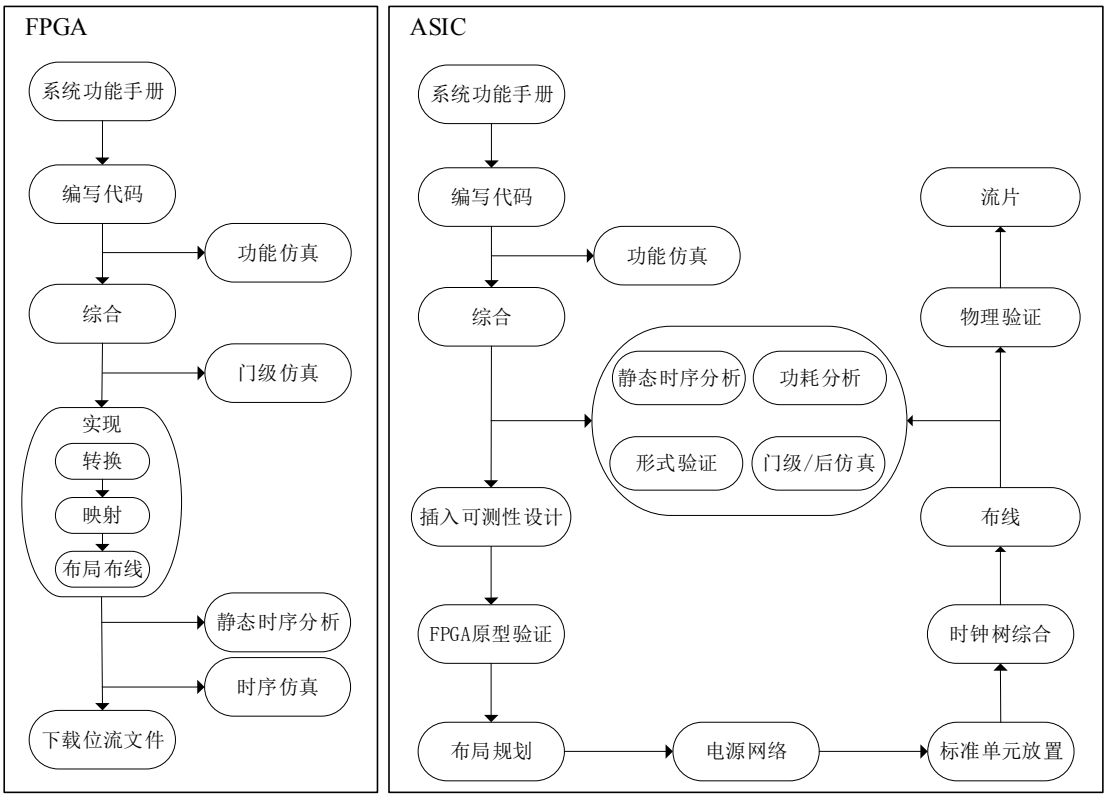


图 1. 1 FPGA 与 ASIC 设计开发流程

FPGA 存在着不同的工艺种类，反熔丝（Antifuse）型 FPGA、闪存（Flash）型 FPGA 和 SRAM 型 FPGA，不同的工艺具有不同的特点。其中反熔丝型 FPGA 只能编程一次，编程之后成为固定器件，灵活性不足，稳定性较高；Flash 型 FPGA 可反复擦写，且有掉电后非易失性的特点，灵活性和可靠性兼得，但成本较高，因而未被广泛应用；SRAM 型 FPGA 可反复编程，每次上电后需重新加载配置，灵活性好，稳定性略有不足，较低的成本以及高性价比使其在各领域被广泛使用[3]，因此本文主要对 SRAM 型 FPGA 展开研究。

SRAM 型 FPGA 内部资源丰富，通常由可编程逻辑块、可编程输入输出块、可编程互连资源以及常用的存储器、DSP（Digital Signal Processor）、DCM（Digital Clock Management）等硬核 IP（Intellectual Property）模块组成，其中互连资源（Interconnect Resource，IR）占芯片面积的 60%以上，并且随着电路的高度集成化，元器件以及布线的密度越来越大，使得 FPGA 互连资源在制造和使用过程中出现故障的概率也越来变高，因此研究 SRAM 型 FPGA 互连资源的可靠性具有极其重要的意义，本文将针对 SRAM 型 FPGA 互连资源的故障测试、诊断定位以及容错的关键技术进行研究，保障互连资源的可靠性，有效地提高 FPGA 的稳定性，进而确保其应用系统的稳定性。

1.2 相关技术的研究现状

从发明之初的 2um 制造工艺到如今的 7nm 工艺；从最初的近百数量的片上集成资源，到现在的数千万乃至近亿的片上资源，FPGA 在这 36 年取得了惊人的发展，随着 FPGA 本

身的集成度不断提高，内部结构更加复杂，FPGA 可靠性的要求也越来越受重视，大量文献对 FPGA 的可靠性展开了研究。国外对 FPGA 的可靠性研究起始于上世纪八九十年代，拥有成熟的可靠性技术，所铸造的专利和技术壁垒极高，国内从事该领域的研究比国外更晚，但在国内迫切发展集成电路产业的驱动下，对 FPGA 可靠性的研究也取得了一些实际成果，且国内在短期内有望赶上国际的前沿研究水平[4]。

1.2.1 FPGA 互连故障测试

故障测试是检验 FPGA 可靠性的有效手段。一般的测试流程如图 1.2 所示，首先分析故障模型，然后针对相应故障模型进行测试配置生成，之后施加测试激励，再收集测试响应，通过比较测试响应得出测试结果。对于设计人员来说，FPGA 芯片使用灵活，但其重复可编程的特性导致的应用的不确定性，却增加了故障测试的难度，目前还没有一种故障测试方案能满足所有故障测试要求，因此根据不同的测试需要，可将现有的 FPGA 故障测试技术分为两类：应用无关测试和应用相关测试。

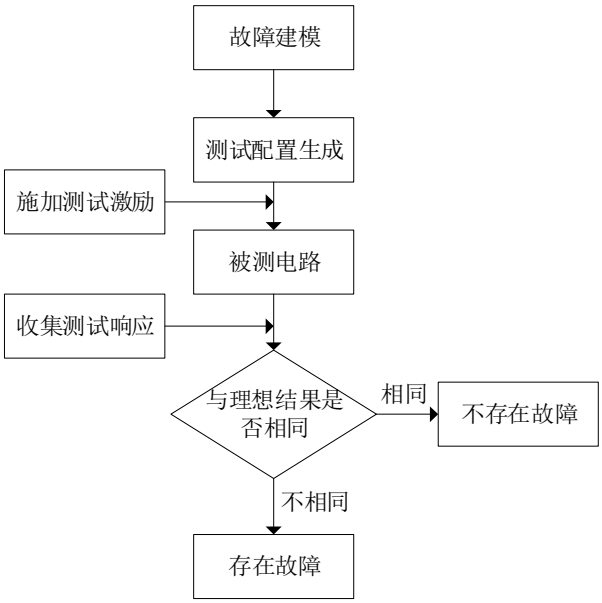


图 1. 2 测试流程图

1）应用无关测试

应用无关测试是从制造厂商的角度对FPGA进行测试，又称为工厂测试，目的是对FPGA内部所有资源进行性能完备的测试[5-9]，测试步骤繁冗复杂，效率较低。应用无关测试将FPGA内部资源分块进行测试，其中对于 IR 资源，主要分为非内建方法和内建自测试（Built In Self-Test，BIST）方法。

非内建自测试，其测试向量来自于外部，大多是基于自动测试设备（Auto Test Equipment，ATE）进行测试，因而许多文献都致力于优化 IR 资源的测试结构和测试配置生成策略，以减少测试配置数，缩短测试时间。FPGA 内部 IR 资源丰富，开关矩阵（Switch Matrix，SM）的存在使得内部的互连线得以灵活的转接，Michel Renovell 等人通过优化 SM 的测试结构，

在最少的测试配置数下实现了相应互连故障的覆盖测试[10,11]。Liu Pe 等人利用测试可编程逻辑块（Configurable Logic Block，CLB）阵列的方法，将 CLB 配置为输入等于输出的形式，从而级联整个阵列，最终实现 CLB 之间互连线的测试[12]。Shukla Banik 等人提出一种基于可满足性模理论（Satisfiability Modulo Theory，SMT）的 FPGA 互连测试配置生成策略，通过对目标故障设计约束，算出满足约束的测试向量，实现不降低故障覆盖率的情况下最小化配置数量[13]。四川电子科技大学的周涛搭建了软硬件协同仿真测试平台对 FPGA 内部互连资源进行测试[14]。合肥中国科学院大学罗俊杰研究基于 ATE 的检测软件，利用配置存储器存储配置程序，实现自动配置 FPGA 进行测试[15]。

内建自测试，通过使用 FPGA 内部逻辑资源，实现整个测试流程，其测试向量来自于内部。传统的基于 ATE 进行测试存在 FPGA 输入输出端口数量有限的顾虑，Charles Stroud 等人第一次提出使用 BIST 方法测试 IR 资源，将整个测试过程建立在 FPGA 内部，有效地解决了输入输出端口限制问题[16]。Sun Xiaoling 等人将差错控制编码与 BIST 测试相结合，在整个测试过程中加入奇偶校验码，从而简化测试响应分析的过程[17]。Charles Stroud 等人又第一次提出在线 BIST 测试，将 FPGA 分为工作区域和测试区域，使用测试区域对整个 FPGA 进行巡回测试[18]。JackSmith 等人研究出自启动的测试生成模块（Test Pattern Generator，TPG），并结合自配置开关矩阵，实现对路由资源的充分测试，减少外部比特流的下载次数[19]。四川电子科技大学的杨会平实现了基于 BIST 的多资源联合测试，可同时对 FPGA 内部逻辑资源和互连资源进行测试[20]。

2）应用相关测试

不同于应用无关测试，应用相关测试是面向用户的测试，只关心用户电路映射在 FPGA 上所使用到的硬件资源的正确性[21-23]，通常用户电路使用不到 FPGA 内部所有资源，因而相比于应用无关测试，应用相关测试的效率更高；并且用户在使用 FPGA 进行设计时，往往不会考虑插入 BIST、扫描链以及测试点等可测性结构，基于 ATPG（Automatic Test Pattern Generation）等技术又较难实现对用户设计的完全测试，所以应用相关测试是用户对 FPGA 进行检测的最佳策略。

Mehdi B. Tahoori 等人最早将应用相关测试应用于 FPGA 的 IR 测试中，并提出保留用户设计的互连配置，将所有逻辑块配置修改为“与逻辑”和“或逻辑”，完成对互连故障的固定 0/1 的覆盖测试[24]；随后又考虑到互连故障中的桥接故障，利用单项函数，并结合沃尔什（Walsh）码生成测试配置，完成对固定故障和桥接故障的覆盖测试[25]。H. Almurib 等人对互连故障进行进一步的补充，细化了桥接故障的类型，并提出使用一种测试配置，在该配置下施加多次测试向量的方法进行互连测试，实现了测试配置数的最小化[26]。T. Nandha Kumar 等人随后又提出在用户设计的电路存在较大扇出数时，将单个激活输入连接到多条线，以解决单次施加测试向量数量过多的问题[27]。在文献[28]中指出现有应用相关测试技术未覆盖到反馈桥接故障的测试问题之后，Shukla Banik 等人借鉴文中反馈桥接的定义，并使用 SAT 生

成满足故障约束的测试配置，从而覆盖相应故障的测试[29]。四川电子科技大学的罗毅将双模冗余与 BIST 方法相结合，将用户设计电路中的关键模块设计成带 BIST 功能的模块，并进行冗余复制，比较两模块的输出，存在差错时，启用模块的 BIST 功能进行故障测试[30]。北京化工大学的张娜提出将 FPGA 的配置数据进行回读比较的方法，解析出配置数据所包含的硬件资源状态信息，从而实现互连资源的故障测试[31]。

应用无关测试主要对 FPGA 进行出厂测试，需要覆盖 FPGA 所有的资源，测试步骤较为繁琐；应用相关测试则只关心用户电路所使用到的 FPGA 资源，测试效率较高，而现有测试文献中的应用相关测试方案却大多只考虑到某一类型的互连故障测试，因此本文将采取应用相关互连测试方法对现有文献中的互连故障覆盖问题进行分析研究。

1.2.2 FPGA 互连故障诊断

故障诊断是提高 FPGA 可靠性的重要一环。FPGA 内部互连资源丰富，为准确定位存在于互连资源中的故障需要花费大量时间，因此互连故障诊断总是比互连故障测试更困难[32]。根据诊断对象进行区分，互连故障诊断方法存在应用无关和应用相关的类别，但该分类与互连测试方法相同，因此本节根据互连故障诊断原理将现有互连诊断技术分为自适应诊断和非自适应诊断。

1）自适应诊断

在自适应诊断方法中，下一步诊断配置的选择基于上一步的诊断配置的结果。自适应诊断方法是一个不断逼近故障的过程，在过程中根据诊断结果调整诊断配置，大多用于故障数量有限或故障范围有限的情况下以最小代价定位故障。

Mehdi B. Tahoori 提出一种两步诊断技术对 FPGA 互连资源中的单个开路故障进行精准识别，粗粒度步骤将互连故障定位到某一信号线上，细粒度步骤对该路径上的故障进行精准定位，其中细粒度步骤使用二分法对故障进行隔离，分步启用信号线中的金属线段，以实现故障的定位[33]。随后 Mehdi B. Tahoori 等人在诊断电路中单个桥接故障时，使用二分法对故障进行激活，逐步定位到存在故障的信号线对[34]。T. Nandha Kumar 等人对在输出端口处检测到的故障进行回溯，将故障路径上的查找表 LUT 配置为透明状态（输入等于输出），找到故障路径上的故障分支，再针对故障分支进行诊断[35]。Haider A.F. Almurib 等人先对电路的路由分支进行激活和去激活处理，再使用配置确定电路中可能存在的互连故障类型，然后针对不同的故障类型施加不同的诊断策略[36]。

2）非自适应诊断

在非自适应诊断方法中，先执行所有的诊断配置，并记录结果，然后再分析结果判定出故障所在位置。非自适应诊断的时间是固定的，而自适应诊断的时间是根据故障所在位置随机的，可能是最坏的情况，也可能是最好的情况。但自适应方法在诊断过程中需要根据上一次的诊断结果调整下一次配置，总的花费时间通常大于非自适应方法所需要的时间，所以非自适应方法一般优于自适应方法。

Yu Yinlei 等人使用非自适应方法，在该方法中只使用六次配置就可实现对 FPGA 任意单个互连故障的准确位置和类型的诊断[37]。Debaleena Das 等人将用户设计的电路分为活跃分支和惰性分支，分别对活跃分支和惰性分支施加完激励后，依靠时钟信息，即故障错误信息第一次出现在输出端的时钟周期，实现 FPGA 互连资源中故障的定位[38]。Charles Stroud 等人提出使用 BIST 结构诊断互连故障，对故障路径应用不同的 BIST 结构，施加激励，并收集所有结果，最后与故障字典中故障特性进行比对，从而确定故障所在位置[39]。Mehdi B. Tahoori 等人在诊断固定故障时所使用的配置与故障位置信息相关联，在施加完多次诊断配置之后，根据所有的配置结果进行编码即可定位到故障所在位置[34]。T. Nirmalraj 等人对互连测试阶段所使用的测试配置以及测试结果进行分析，结合不同类型故障的特点对可能存在的故障进行一一排除，从而实现故障的定位[40]。四川电子科技大学的项传银将互连故障映射到相邻 LUT 输出上，结合映射后 LUT 的输出值对故障进行诊断[41]。上海复旦大学的马珂洁提出使用局部重配置的方法减少配置时间，对 FPGA 互连故障进行诊断[42]。

自适应诊断方法需要根据诊断结果不断调整诊断配置，所花费的时间成本较大；非自适应诊断方法则先收集所有的诊断配置结果，再根据配置结果分析出故障所在位置，所需的诊断时间较少，然而现有诊断文献中的非自适应诊断方案都直接对待测电路中所有的互连线进行诊断，所需诊断的范围较大，因此本文将采取非自适应诊断方法对现有诊断文献中的故障诊断范围进行优化。

1.2.3 FPGA 互连故障容错

故障容错是保证 FPGA 可靠性的关键技术。在 FPGA 存在故障的情况下，为了保证基于 FPGA 的数字系统仍然能够正常工作，必须采用相应的容错技术，就像人体细胞发生有害变异时，白细胞会对有害变异细胞进行吞噬，修复身体机能。FPGA 内部故障可分为两种，软错误和硬错误，软错误主要是 FPGA 由高能粒子辐射和噪声干扰造成的非永久性故障，不在本文讨论范围内；硬错误主要是 FPGA 在制造过程存在的缺陷，或使用过程中导致的永久性损伤，国内对于 FPGA 的容错研究主要集中在星载芯片的软错误容错，对于硬错误的容错研究较少。现有的 FPGA 硬错误容错技术主要是利用 FPGA 内部的冗余硬件资源实现容错，其中 FPGA 互连资源的硬错误容错技术存在着预防型容错和修复型容错的区别。

1）预防型容错

预防型容错是用户在设计之初就考虑到容错策略，并将相应的容错技术插入到设计电路中，以应对电路可能发生的故障。常用的 FPGA 互连故障预防型容错技术都是基于双模冗余、三模冗余进行设计的，首先找到用户电路中的关键模块和路径，再使用空闲资源对相应的关键模块和路径进行复制，最后结合多数表决器进行表决，忽略错误路径的输出，实现容错[43-47]。

2）修复型容错

修复型容错则是待测电路在经过故障测试和故障诊断过程之后，发现并定位到电路中存

在的故障，再采取措施对相应故障进行容错。互连故障的修复型容错又分为硬件级容错和配置级容错[48]，其中硬件级的互连故障容错大多是基于 FPGA 硬件逻辑行/列的移位，通过多路选择器绕过原有的故障行/列，切换到阵列末端的备用单元行/列，实现对故障的容错[49,50]；配置级的互连故障容错则是基于布局布线算法，主要分为替代配置以及增量映射两类，替代配置方法是将未使用到故障资源的预编译配置替换原始配置[51,52]，而增量映射方法是将原始配置重新布局布线，以避开故障资源实现容错[53-55]。

预防型容错方法适用于电路设计之初，需要考虑电路正常工作时可能发生的任意故障；修复型容错方法则是在故障测试和故障诊断的基础上进行的，只需针对诊断定位到的故障进行处理，与本文研究步骤相契合，但现有修复型容错文献中所需要增加的资源面积消耗以及相应的工作复杂度都较高，因此本文考虑结合预防型容错方法和修复型容错方法，对现有容错文献中的故障处理措施进行改进。

1.3 本文主要研究工作

FPGA 开发较于 ASIC 开发有着一定的优势，所以 FPGA 越来越受欢迎，并被广泛应用于各领域，因而对 FPGA 的可靠性要求越来越高，相应高效的可靠性技术也越来越受重视。本文针对 FPGA 内部发生故障概率较大的互连资源进行了深入的可靠性研究，分析了现有 FPGA 互连故障测试、诊断以及容错技术，并归纳了相应技术的优缺点以及适用范围，决定采用应用相关测试技术、非适应性诊断技术和修复型容错技术对 FPGA 互连资源进行可靠性研究。具体研究工作如下：

1）针对现有应用相关互连测试文献中的桥接故障测试以及反馈桥接故障覆盖难题，本文提出了一种基于故障模型优化的应用相关互连测试方法。在用户电路映射到 FPGA 上之后，一些物理距离相距较远的互连线之间发生桥接故障的概率较小，将桥接故障约束在同一 LUT 信号线之间以优化对桥接故障的测试；并将反馈桥接故障的类型细分为影响输出型和反馈输入型，以解决反馈桥接故障的覆盖问题；再结合单项函数以解决测试过程中的故障传播问题，然后使用 SAT 对优化后的故障模型进行约束，并根据用户电路中的互连情况对可能产生的赋值冲突进行约束，最后生成相应的测试配置进行故障测试。

2）考虑到现有互连故障诊断技术的复杂度以及诊断配置数过多的问题，本文沿用测试阶段优化的故障模型，提出了一种基于故障粗细粒度范围的非自适应故障互连诊断策略。首先分析待测电路的互连情况，明确电路中的单输出路径和多输出路径情况；然后根据测试阶段的错误输出端口信息，判断电路中故障所在的路径；再根据不同路径的粗粒度诊断方法对故障所在路径进行粗粒度诊断，以缩小故障诊断范围；最后复用测试阶段的测试配置对故障粗粒度范围内的信号线进行比对，排除无故障信号线，实现细粒度的单故障精准定位。

3）在完成 FPGA 互连资源的故障测试以及故障诊断工作之后，便要采取措施对故障进行修复型容错，而为了避免修复型故障容错技术普遍存在的资源消耗过大、实现难度较大以及无法直接处理互连故障等缺点，本文提出了一种基于双模冗余的修复型互连故障纠错容错

方案。该方案将预防型容错的双模冗余技术与修复型容错的概念相结合，把测试和诊断所定位到的互连故障映射到相邻的逻辑资源上，并对该原始逻辑资源进行基于 LUT 的细粒度双模冗余，所增加的面积消耗较低，再插入两级异或电路实现对故障的检错纠错，第一次异或操作检测互连故障是否导致电路发生错误变化，第二次异或操作将互连故障导致电路发生的错误变化进行纠正，以恢复电路的正常工作，完成对互连故障的修复。

4）本文使用 ISCAS'89 基准电路对提出的可靠性方案进行了验证，通过模拟注入互连故障进行仿真，仿真结果证实了所提出的互连故障测试、诊断以及容错方法的有效性。然后对所提出方法的故障测试配置数、故障诊断配置数以及容错技术所造成的面积开销进行综合评估，并与其他论文中的数据结果进行比对。比对结果表明，测试方案所生成的测试配置在完成对用户电路中所有互连故障的覆盖的前提下实现了最小的测试配置数；诊断方案减少了所需诊断的信号线数量，实现了诊断复杂度的降低，并且所使用的诊断配置数相比于已有文献也有所减少，有效地提高了故障诊断的效率；容错方案直接对互连故障进行处理，实现的难度较低，且能以更少的面积消耗完成对互连故障的容错，以上比对结果证实了本文所提出的方法的优越性。

1.4 论文的组织结构

本文组织结构如下：

第一章为绪论部分，讨论了本课题的研究背景和选题意义，对现有可靠性技术发展现状进行了分析，归纳了相应技术的优缺点和适用范围，并阐述了本文主要的研究工作以及创新点，最后总结了论文的组织结构。

第二章首先介绍了 SRAM 型 FPGA 内部资源的结构，然后具体阐述了 FPGA 互连资源的故障模型，为后续推进可靠性研究工作进行铺垫。

第三章首先介绍了单项函数，以解决互连故障在测试过程中需要考虑的故障传播问题，然后对反馈桥接故障模型进行了优化，并结合单项函数分析优化后的反馈桥接故障现象，最后使用 SAT 对优化后的故障进行了测试生成。

第四章首先分析了待测电路的互连情况，结合错误端口的信息判断故障所在路径，然后介绍了单输出路径以及多输出路径的粗粒度诊断方法，最后阐述了使用复用配置法对故障进行细粒度精确定位的步骤。

第五章首先介绍了对互连故障所影响到的逻辑资源进行细粒度冗余的策略，然后阐述了使用两级异或电路在双模冗余的基础上对故障进行检错纠错的具体过程。

第六章使用 ISCAS'89 电路分别对所提出的测试、诊断以及容错方案进行了仿真验证，并将所提出方案的数据结果与现有的技术进行分析和对比。

第七章总结了本文的研究工作成果，并根据所做工作中存在的不足之处对下一步的研究方向提出了建议和展望。

第二章 FPGA 结构与互连故障模型概述

2.1 SRAM 型 FPGA 器件结构

从第一片FPGA诞生到如今，国外的FPGA技术已经发展了36年，这也就是为什么FPGA市场长时间被国外厂商 Xilinx、Altera（后被 Intel 收购）、Lattice、Microsemi 所垄断的原因。四大巨头中，Xilinx 公司所占市场份额最大，因此为了研究 SRAM 型 FPGA 的特点，本文以 Xilinx 公司的 FPGA 来进行介绍。

随着工艺水平的提高，为提升芯片的整体性能，FPGA 上集成了越来越多的功能模块（如：块随机存取存储器，DSP，嵌入式软核），而这些 FPGA 结构的参考模型都源于经典的 SRAM 架构。经典的 SRAM 型架构是由 CLB、IR 以及输入/输出模块（Input/Output Block，IOB）组成的二维阵列，其中 CLB 是 FPGA 的基本功能单元，用来实现任意逻辑电路；IR 资源包括金属线段和开关矩阵，实现内部资源之间的连接；IOB 实现芯片与外部的通信，主要分布在芯片的四周。如图 2.1 所示。

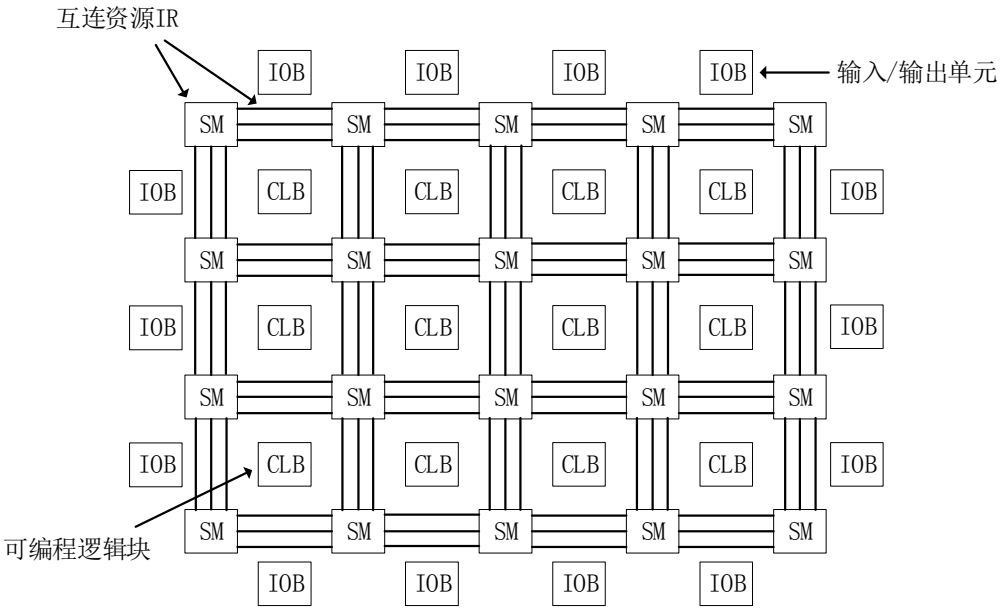


图 2. 1 经典的 SRAM 型 FPGA 结构框图

2.1.1 可编程逻辑块（CLB）

可编程逻辑块（CLB）是实现用户电路功能的基本单元，不同型号的 FPGA 中所含的 CLB 数量不同，相应的特性也不同，下面以 Virtex4 系列 FPGA 为例进行介绍。Virtex4 系列 FPGA 的 CLB 内部包含 4 个切片（Slice），不同 Slice 之间相互独立，相应的 CLB 结构图如 2.2 所示。在 Slice 内部包含 LUT、进位链、多路选择器、触发器以及互连金属线等资源，其中 LUT 是 Slice 内部主要的逻辑功能实现单元；进位链是 Slice 内部的专用算术逻辑，可

快速产生进位和借位信号，能大大提高复杂运算的效率；多路选择器可级联 LUT 以实现更多的逻辑功能；触发器是 Slice 内部重要的时序单元，支持同步/异步使能以及复位，也可配置为锁存器，在图 2.3 中展示了 Virtex4 系列 FPGA 的 Slice 内部电路。

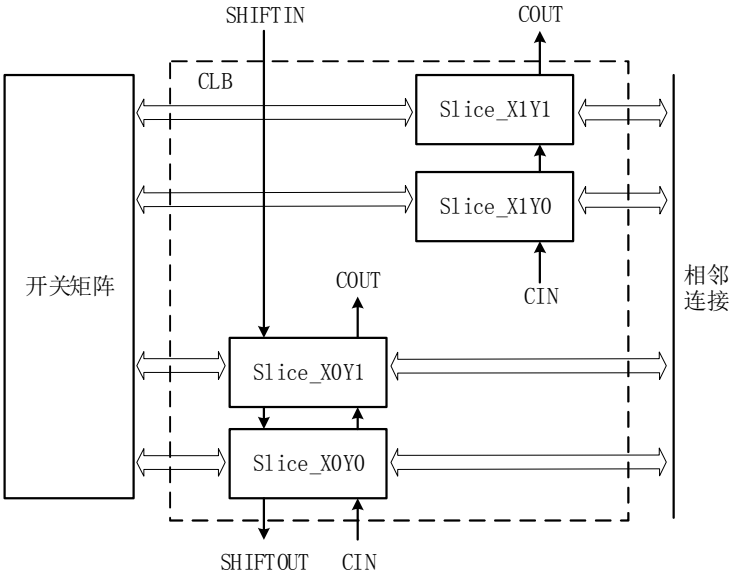


图 2. 2 Virtex4 系列 FPGA 的 CLB 结构图

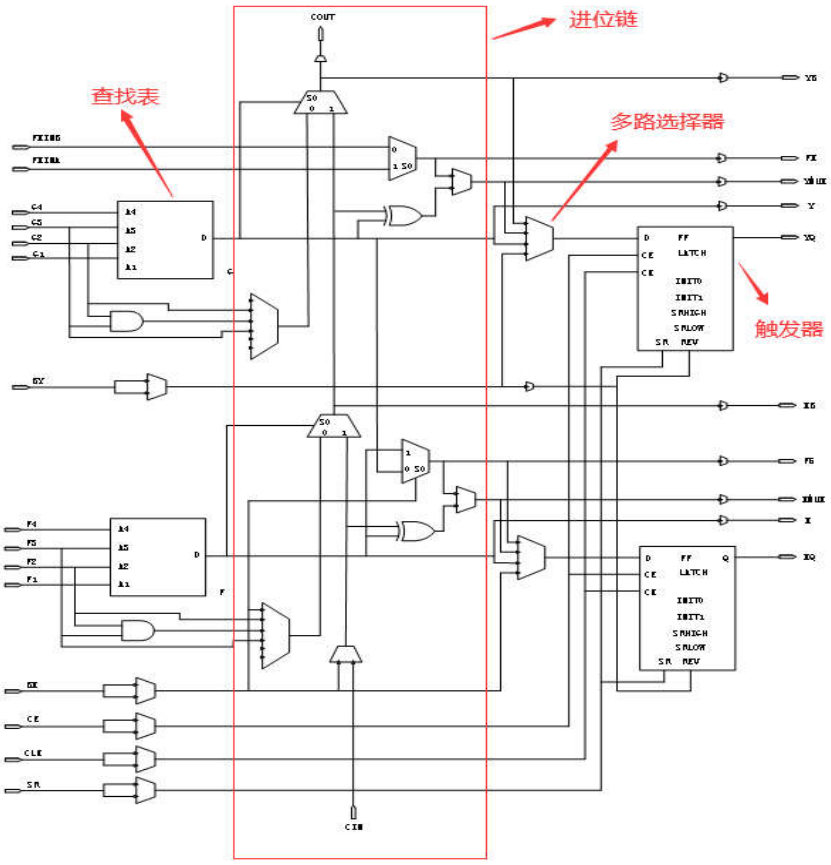


图 2. 3 Virtex4 系列 FPGA 的 Slice 内部电路图

对于最重要的 LUT 资源，规格有 4 输入和 6 输入的不同，本质上也就是一个 16 位或

32 位的 SRAM 存储器。通过对 LUT 进行编程，能够实现任意 4 输入或 6 输入的逻辑函数，LUT 的输入就是 SRAM 的地址，LUT 存储的初始值就是 SRAM 阵列中所存逻辑功能的真值，LUT 实现某一逻辑功能的过程就是通过输入端的地址查询对应逻辑功能的真值表，并将查询后的对应值进行输出。

2.1.2 可编程互连资源（IR）

可编程互连资源（IR）用于实现 CLB 与 CLB、CLB 与 IOB 之间的通信，主要由相互独立的金属线段、可编程开关点以及开关矩阵构成，不同金属线段之间的连接与断开是由可编程互连点进行控制的，而这些可编程互连点在开关盒中形成的阵列就是开关矩阵。局部互连资源结构如图 2.4 所示。

1）互连金属线段

信号在互连金属线上的驱动能力和传输速率取决于互连金属线段的长度和工艺，根据其长度、工艺以及不同的分布位置，可将互连金属线分为不同的类型，如单长线、双长线、长线等。

单长线存在于每个 CLB 附近，其长度为 2 个 CLB 之间的距离，凭借着可编程互连点的存在，使得单长线与 CLB 之间的布线非常灵活，提供了相邻 CLB 之间的快速通信；但若使用单长线进行长距离传输，中间必定要经过许多开关矩阵，而信号经过开关矩阵会存在一定的时延，因此单长线并不适合长距离传输信号。

双长线的长度是单长线的两倍，在进入开关矩阵之前，双长线横跨两个 CLB，主要提供中距离的快速布线通道。

长线是从芯片的一头横跨到另一头的互连金属线，不经过任何开关矩阵，信号延迟小，主要用于长距离的信号传输。

2）开关矩阵

水平和垂直的单长线或双长线交汇在一个用来控制布线方向的可编程开关矩阵中，以提供 FPGA 内部资源进行灵活的布线。从图 2.4 中右上角可看到，开关矩阵内的可编程互连点提供了东西南北四个方向的连通性，无论互连金属线段从哪个方向进入开关矩阵，都可以通过编程实现灵活的转接。

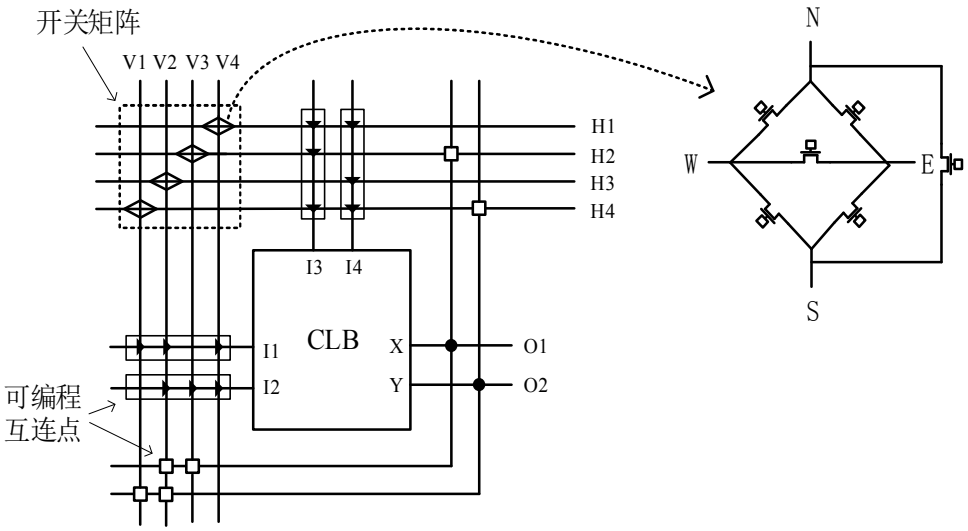


图 2. 4 局部互连资源结构图

3）可编程互连点

可编程互连点本质上就是一些传输晶体管，主要采用 CMOS 工艺，通过 SRAM 配置单元控制传输晶体管的开启或关闭，实现对每个单元的编程。SRAM 结构如图 2.5 所示。

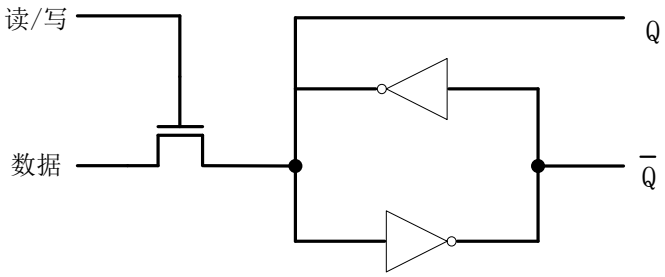


图 2. 5 SRAM 配置单元结构图

从图 2.5 中可看出，两个对称的反相器实现数据的存储，通过读写信号控制存储单元的门控访问，读的过程就是将保存在 Q 的值传递给数据位，写的过程就是以数据位的值驱动 Q，从而覆盖反相器先前的状态。

2.2 FPGA 互连资源的故障模型

FPGA 互连资源的故障主要分为开路和短路，开路故障大多是由可编程互连点的常开或互连金属线的断路引起的;短路故障则是由可编程互连点的常闭或两条互连金属线之间的短路导致的。为了对故障进行检测，需要对故障进行模型化，而故障模型则是对所有的故障结构和故障现象进行分析之后构造出来的结果，FPGA 常用的互连故障模型分为固定故障和桥接故障，其中固定故障主要是针对开路故障以及可编程互连点的常闭故障进行建模，而桥接故障则是针对互连金属线之间的短路故障进行建模。互连故障类型如图 2.6 所示。

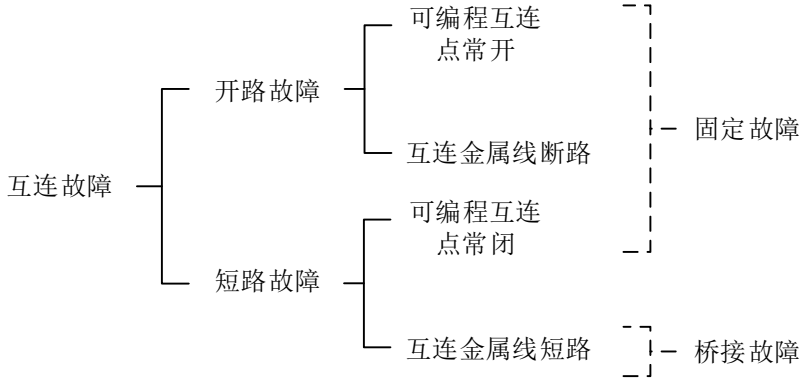


图 2. 6 互连故障类型

1）固定故障

可编程互连点常开、互连金属线断路以及可编程互连点常闭，这些都将导致电路中相应信号线上传输的信号固定为某一个逻辑值，若固定在低电平，则为固定 0 故障；若固定在高电平，则为固定 1 故障。

2）桥接故障

两互连金属线之间的短路，会造成电路中两个信号线发生意外连接，在图 2.7（a）中描述了 X 信号线与 Y 信号线存在无反馈桥接故障的示意图，而若恰好两信号线一前一后，处于同一支路时，则会形成反馈环路，在图 2.7（b）中描述了 X 信号线与 Y 信号线存在反馈桥接故障的示意图，因此根据故障结构可将桥接故障分为无反馈桥接故障和反馈桥接故障。

同时，当两互连金属线发生短路，且两金属线上传输的值不相同时，两互连金属线信号驱动能力的不同决定了最终的故障现象。根据两互连金属线信号驱动能力的强弱所导致的不同故障现象进行建模，可将无反馈桥接故障和反馈桥接故障细分为“主导”型、“主导与”型以及“主导或”型。

“主导”桥接故障的现象是其中一条金属线无论传输什么值，都为强驱动，致使另一条金属线的值始终被牵制，随着强驱动金属线的值改变而改变，若金属线 X 和 Y 发生“主导”型桥接故障，且 X 为强驱动，则无论金属线 Y 传输什么值，最终都会与强驱动金属线 X 的值相同，在图 2.7（c）和图 2.7（d）中分别描述了强驱动金属线 X 传输 0 和 1 时，金属线 Y 传输的逻辑值变化。“主导与”桥接故障的现象是传输逻辑值 0 的金属线驱动能力强于传输逻辑值 1 的金属线，从而导致传输逻辑值 1 的金属线最终传输逻辑值 0，即“主导与”桥接的逻辑值变化由主导信号线的逻辑值 0 主导，在图 2.7（e）中描述了 X“主导与”Y 桥接故障的等效逻辑模型。“主导或”桥接故障的现象是传输逻辑值 1 的金属线驱动能力强于传输逻辑值 0 的金属线，从而导致传输逻辑值 0 的金属线最终传输逻辑值 1，即“主导或”桥接的逻辑值变化由主导信号线的逻辑值 1 主导，在图 2.7（f）中描述了 X“主导或”Y 桥接故障的等效逻辑模型。

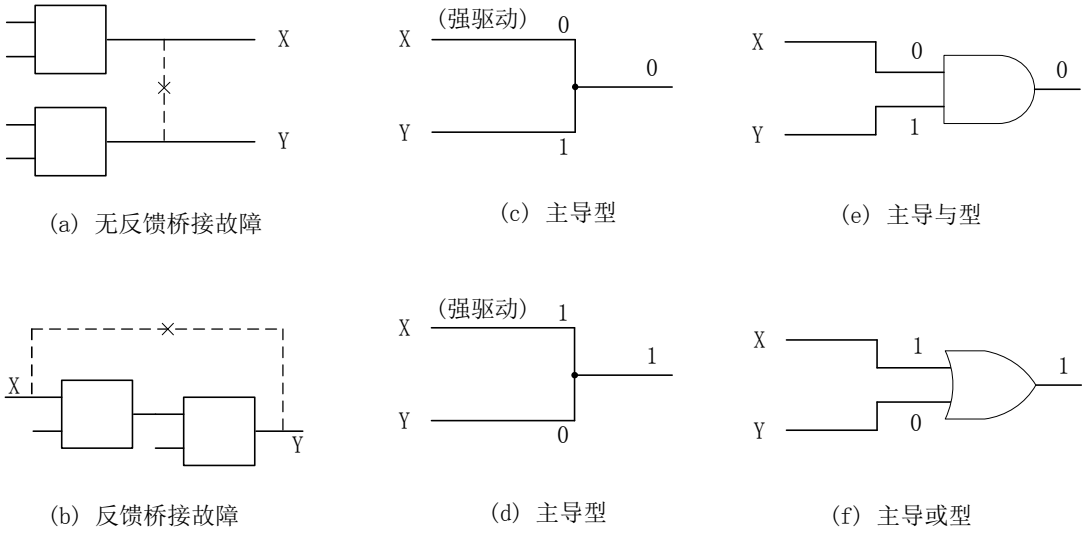


图 2.7 桥接故障示意图

根据上述互连资源不同故障模型的原理和故障现象，可得出相应的故障真值表，如表 2.1 所示。在表 2.1 第 1 行展示的是无故障时的真值情况，表中其他行则列举了存在不同互连故障时对应真值的变化情况，其中加粗且带下划线的逻辑值为相应故障所导致的变化。

表 2.1 互连故障真值表

故障类型	真值情况			
XY（无故障）	00	01	10	11
XY（X 固定 0）	00	01	**0**0	**0**1
XY（Y 固定 1）	**1**0	**1**1	10	11
X 主导 Y	00	0**0**	1**1**	11
Y 主导 X	00	**1**1	**0**0	11
X 主导与 Y	00	0**0**	10	11
Y 主导与 X	00	01	**0**0	11
X 主导或 Y	00	01	1**1**	11
Y 主导或 X	00	**1**1	10	11

上述介绍的互连故障模型将作为后续故障测试的研究对象，且在后续的互连测试方案中将主要围绕着桥接故障进行优化分析。

2.3 本章小结

本章首先介绍了 SRAM 型 FPGA 的典型架构，然后介绍了内部重要的逻辑资源和互连资源，对逻辑资源内部的构成以及互连资源的种类进行了阐述，最后着重分析了 FPGA 内部互连资源的故障种类，根据不同故障的现象和结构，对故障的通用模型进行了介绍，并给出了相应的互连故障真值表，为后续互连资源的测试、诊断以及容错工作的展开进行了铺垫。

第三章 基于改进故障模型的 FPGA 应用相关互连故障测试

由第二章可知，FPGA 中的互连资源分为两部分：一部分为 CLB 内的互连资源，一部分为 CLB 间的互连资源。CLB 内的互连资源位于每个 CLB 内部，为 CLB 内部的资源提供了布线能力，其测试可在逻辑测试时同时进行；CLB 间的互连资源实现了 CLB 与 CLB 之间的路由。本文只讨论 CLB 间的互连资源测试，主要考虑 CLB 间的互连资源实现 LUT 与 LUT 的连接，若用户电路中使用了进位链、多路复用器等资源，将绕过这些部分。

CLB 间的互连资源出现错误，即相应互连金属线段或可编程互连点出现错误，在用户电路中则表现为对应的信号线所传输的逻辑值发生错误，因此反过来测试用户电路中信号线传输的正确与否，即可证明电路所使用的硬件互连资源是否存在故障。为检测用户电路中信号线传输的正确与否，许多文献都通过保持路由资源不变，修改逻辑资源的配置进行测试，而其中大多数都只考虑到某一类型的互连故障，如文献[24]中只针对了固定 0/1 故障进行测试，文献[25]虽然考虑到桥接故障，却没有对桥接故障的类型进行细分，文献[26]中解决了桥接故障的相应问题，但在文献[28]中被指出未覆盖到反馈桥接故障的测试，随后文献[29]借鉴相关反馈桥接故障的定义进行测试，却未给出生成的测试向量示例，因此本文针对现有文献存在的问题，提出基于改进故障模型的 FPGA 应用相关互连故障测试方案，对包含反馈桥接故障在内的所有互连故障进行测试。

3.1 单项函数

在测试用户电路信号线的过程中，使用单项函数解决故障的传播和故障的激活问题。单项函数是一个只含有最大项或最小项的逻辑函数式，其真值表中有且只有一组向量的输出与其他组向量的输出不同，这组向量也被称为激活输入。在图 3.1 中展示了 4 输入 LUT 的单项函数示例，该单项函数只有一个最大项，其激活输入为 1001，只有当输入为激活输入 1001 时，其输出才为 0，其他任何输入所对应的输出都为 1。将所有的 LUT 都配置为对应的单项函数，这样就能防止在测试过程中出现故障湮灭的现象，保证了故障的传播。

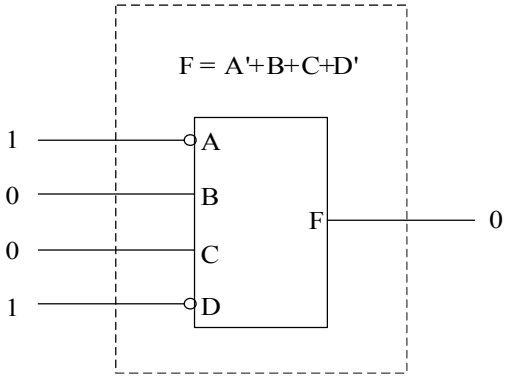
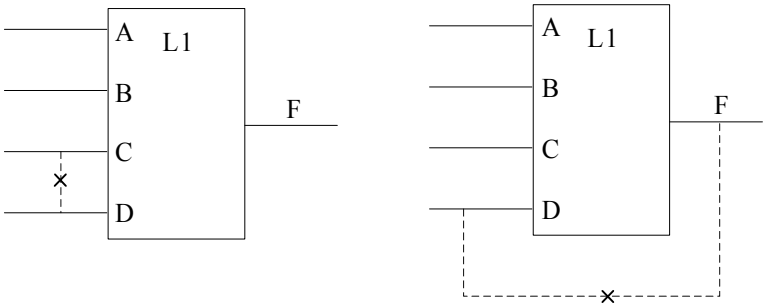


图 3. 1 应用激活输入的单项函数

单项函数还能保证故障的激活。当输入为激活输入时，其输入向量会激活相应的故障，如图 3.1 中的 1001 向量就会激活信号线 A 的固定 0、信号线 B 的固定 1、信号线 C 的固定 1、信号线 D 的固定 0 等故障，若此时电路存在相应的故障，例如信号线 A 存在固定 0 故障，那么输入信号线 A 上所传输的值就会被故障影响，由 1 变为 0，此时输入则由 1001 变为了 0001，输入不再是激活输入，根据单项函数的特性，LUT 输出端的值也会发生错误改变，由 0 变为 1，从而检测到故障的存在。

3.2 桥接故障模型的优化

在解决了故障传播问题之后，继而进行互连故障的改进和优化。现有文献研究桥接故障时考虑了任意信号线之间的可能性，而在 FPGA 设计中，考虑所有信号线之间的桥接故障是不合适的，因为这样不仅使得故障的数量变得十分庞大，而且整个电路的状态分析也会变得十分复杂，增加了测试的难度，并且基于 FPGA 的布局布线信息，一些物理距离相距较远的信号线之间是不可能发生桥接故障的。为了优化对桥接故障的测试，本文将只考虑单个 LUT 信号线上的无反馈桥接故障和反馈桥接故障，在图 3.2（a）和图 3.2（b）中分别展示了 4 输入 LUT 上存在无反馈桥接和反馈桥接故障的示意图。



(a) 无反馈桥接　　(b) 反馈桥接

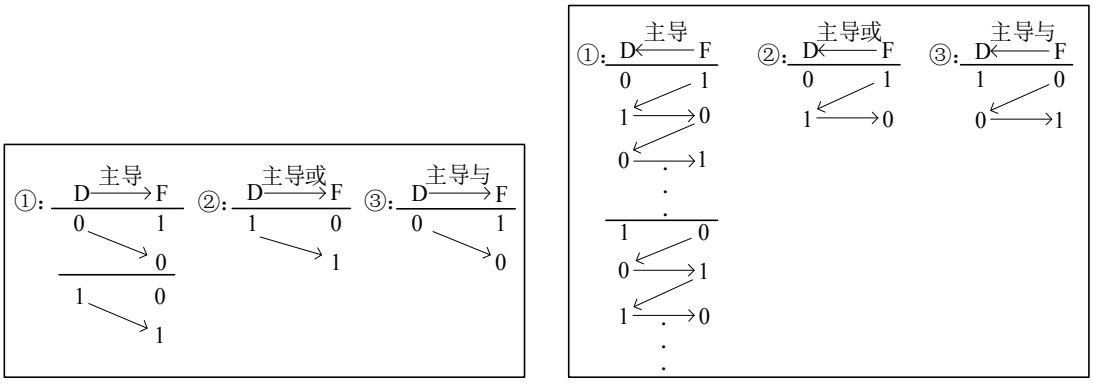
图 3. 2 LUT 上的桥接故障示意图

单个 LUT 信号线上的反馈桥接故障，如图 3.2（b）所示，发生在输入信号线 D 与输出信号线 F 之间，而现有的测试工作中都只考虑了输入信号线 D 通过反馈桥接的意外环路影响输出信号线 F 的情况，并未考虑到输出信号线 F 通过意外环路影响输入信号线 D。因此本文将反馈桥接故障细分为影响输出型桥接和反馈输入型桥接，下面结合单项函数对这两种类型的反馈桥接故障进行故障测试现象分析，如图 3.3 所示。

影响输出型桥接表现为输入信号线 D 通过意外环路导致输出信号线 F 发生变化，在图 3.3（a）中对影响输出型桥接的“主导”、“主导与”以及“主导或”三种类型进行分析。当存在信号线 D“主导”信号线 F 的影响输出型桥接故障时，若 D 和 F 逻辑值相同，则无法观察到故障现象，也就无法检测出 D“主导”F 桥接，若 D 和 F 的逻辑值不相同，如图 3.3（a）中描述的例①，当 D 为 0（1），F 为 1（0）时，因为存在 D“主导”F 的桥接，输入信号线 D 通过意外桥接环路使得输出线 F 的逻辑值发生变化，即由正确的 1（0）变为错误的 0（1），而这种变化可以通过单项函数传播到最终输出端，因此可以检测到影响输出型“主

导”反馈桥接故障的存在。图 3.3（a）中的例②描述了信号线 D“主导或”信号线 F 的桥接，因为“主导或”桥接的逻辑值变化由主导信号线的逻辑值 1 主导，所以当 D 为 1，F 为 0 时，D 会导致 F 由正确的 0 变为错误的 1，结合单项函数的故障传播机制，能够在最终输出端检测到故障的存在。图 3.3（a）中的例③描述了信号线 D“主导与”信号线 F 的桥接，因为“主导与”桥接的逻辑值变化由主导信号线的逻辑值 0 主导，所以在 D 为 0，F 为 1 时，D 会导致 F 由正确的 1 变为错误的 0，同样也能检测到故障的存在。

反馈输入型桥接表现为输出信号线 F 通过意外通路导致输入信号线 D 发生变化，在图 3.3（b）中同样将反馈输入型桥接分为“主导”型、“主导或”型以及“主导与”型进行分析。“主导”型桥接故障的现象只有在两信号线逻辑值不等时才能被激活，图 3.3（b）中例①描述了 F“主导”D 桥接故障的两种情况。当 D 为 0，F 为 1 时，因为 F“主导”D 桥接故障的存在，输出信号线 F 通过意外桥接环路使得输入 D 的逻辑值发生变化，即 D 会由 0 变为 1，但输入 D 发生改变后，在使用单项函数的情况下，此时的输入向量与单项函数的激活输入不同，因此输出 F 会发生改变，由 1 变为 0。这时 D 为 1，F 为 0，D 和 F 的值又不相同，“主导”桥接故障激活，F 又会使 D 发生变化，由 1 变为 0，这时输入恢复为原来的激活输入，导致 F 又会发生改变，重复以上变化，将形成振荡。当 D 为 1，F 为 0 时，同理可得，故障现象也表现为逻辑值振荡，虽然反馈输入型“主导”桥接会导致电路发生振荡，但是由于单项函数的存在，故障的振荡现象会向后传播，因而在最后的观测点可以观测到与正确值不相同的输出，所以能检测反馈输入型“主导”桥接故障的存在。图 3.3（b）的例②和例③分别分析了 F“主导或”D 的桥接故障以及 F“主导与”D 的桥接故障，其中例②故障发生的逻辑变化由主导信号线 F 的逻辑 1 主导，例③故障发生的逻辑变化由主导信号线 F 的逻辑值 0 主导，对应故障现象的分析步骤与上述相同，最终都能在输出端检测到故障的存在。



(a) 影响输出型　　(b) 反馈输入型

图 3. 3 反馈桥接故障测试现象分析

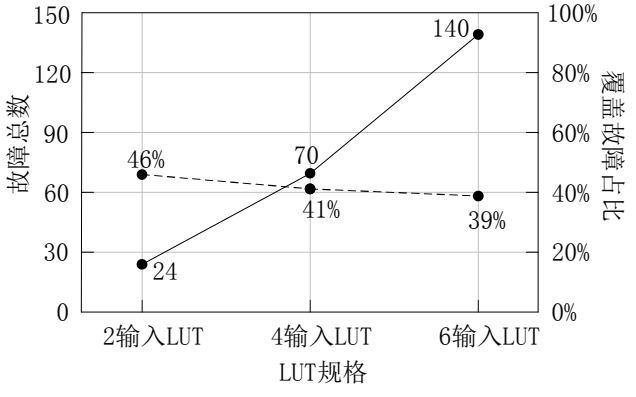
在上述对优化的故障模型进行分析之后，可以看出使用单项函数并施加特定的测试向量能够很好地检测到反馈桥接故障的存在，相比于以往文献，本文优化的故障模型既解决了数量庞大的故障列表问题，又优化了反馈桥接故障存在的测试难题。根据优化后的故障模型，

可以罗列出单个 4 输入 LUT 可能存在的所有互连故障，相应的故障列表如表 3.1 所示。

表 3.1 单个 4 输入 LUT 的故障列表

输入信号	固定故障		桥接故障			
A	固定 0	固定 1	A 主导 B	A 主导 C	A 主导 D	A 主导 F
			A 主导与 B	A 主导与 C	A 主导与 D	A 主导与 F
			A 主导或 B	A 主导或 C	A 主导或 D	A 主导或 F
B	固定 0	固定 1	B 主导 A	B 主导 C	B 主导 D	B 主导 F
			B 主导与 A	B 主导与 C	B 主导与 D	B 主导与 F
			B 主导或 A	B 主导或 C	B 主导或 D	B 主导或 F
C	固定 0	固定 1	C 主导 A	C 主导 B	C 主导 D	C 主导 F
			C 主导与 A	C 主导与 B	C 主导与 D	C 主导与 F
			C 主导或 A	C 主导或 B	C 主导或 D	C 主导或 F
D	固定 0	固定 1	D 主导 A	D 主导 B	D 主导 C	D 主导 F
			D 主导与 A	D 主导与 B	D 主导与 C	D 主导与 F
			D 主导或 A	D 主导或 B	D 主导或 C	D 主导或 F
F	固定 0	固定 1	F 主导 A	F 主导 B	F 主导 C	F 主导 D
			F 主导与 A	F 主导与 B	F 主导与 C	F 主导与 D
			F 主导或 A	F 主导或 B	F 主导或 C	F 主导或 D

本文所优化的故障模型适用于任意规格的 LUT，但不同规格的 LUT 由于输入信号线数量的不同，所存在的互连故障数量会不相同，在单次测试配置下所能覆盖到的互连故障数量也不相同。通过计算不同规格的 LUT 存在的所有互连故障数，以及计算不同规格的 LUT 在单次测试配置下所能覆盖到的最大互连故障数量，根据相应的统计数目在图 3.4 中描绘了单个不同规格 LUT 的故障情况。图中实线描绘了单个 LUT 所存在的互连故障总数，虚线描绘了单次配置所能覆盖到最多的互连故障数占单个 LUT 故障总数的百分比情况。



注：实线为单个LUT的故障总数；
虚线为单个配置所覆盖的最多故障占比。

图 3.4 单个不同规格 LUT 的故障情况

从图 3.4 中可看出 LUT 的规格越大，所存在的互连故障总数越多，而随着 LUT 规格的增大，施加单次向量所能覆盖到最多的故障数占相应故障总数的比重逐渐减小，因此可得出，对于具体电路映射在不同型号的 FPGA 上，测试该电路所需的互连测试配置数与该 FPGA 的 LUT 规格呈反比趋势。

3.3 测试向量的生成

在对反馈桥接故障模型进行改进优化之后，再对电路中的整个互连故障列表进行测试向量生成，即给出覆盖到列表中所有故障应该满足的条件，用布尔等式或不等式对这些条件进行描述，并使用 SAT 判断是否存在满足上述布尔等式约束的测试向量，最终实现以最少数量的测试配置覆盖到所有的故障。

SAT 问题是一类用来确定是否存在满足给定布尔公式的解释的问题，来自不同领域的许多问题的表述都可以使用 SAT 来完成，使用布尔公式表示的问题可用布尔可满足性 SAT 来解决，而使用非布尔公式表述的问题需要使用可满足性模理论 SMT 来解决[13]。

在 FPGA 互连测试中的配置生成问题可以用一组布尔约束公式进行表述，因此可以使用 SAT 来求解满足约束条件的测试向量。为了构建测试电路的配置，将 SAT 求解器算出的满足约束条件的测试向量，即分配给每个 LUT 信号线的值，充当对应 LUT 的激活输入，这些值也决定了在每个 LUT 上实现的单项函数功能。

1）布尔值约束：为测试电路中的信号线，需要施加特定测试向量，而测试向量是由逻辑值 0 和逻辑值 1 组成的，因此在生成测试向量之前，需要对测试向量进行布尔值约束，即约束每条信号线的值为 0 或 1。

2）固定故障约束：为了检测信号线上的特定故障，测试向量首先应该激活故障。检测固定故障，就需要施加激活固定故障的测试向量，即检测固定 0 故障，需要施加逻辑值 1，检测固定 1 故障，则需要施加逻辑值 0。在一个应用设计中存在许多互连信号线，这些信号线无法在一次测试配置中完全覆盖所有信号线的固定故障，因此需要多次测试配置，所以固定故障的约束条件为至少存在一种配置，使得某一条被测互连线被施加逻辑值 0，存在另一种配置使得同一条被测互连线被施加逻辑值 1，如公式（3-1）所示。

$$\exists(x,y),\text{使得}V_{n_i}^{x}\neq V_{n_i}^{y}\text{且}x\neq y \tag{3-1}$$

其中 V_{n_i} 代表一个 LUT 的第 n_i 条输入信号线的值，x 和 y 代表配置号。

3）无反馈桥接故障约束：无反馈桥接故障主要发生在 LUT 输入端的成对信号线中，只有在同一个测试配置中对相应的成对信号线施加相反的逻辑值时，才能检测到无反馈桥接故障的存在。而无反馈桥接故障又分为“主导”型、“主导与”型 、“主导或”型，要激活对应的故障，只能在测试配置中对该对信号线施加不同的逻辑值。因此，对于某一个 LUT 的无反馈桥接约束如以下公式所示：

$$\text{主导型：}\exists(i,j),\text{使得}V_{n_i}^{x}\neq V_{n_j}^{x}\text{且}i\neq j \tag{3-2}$$

$$\text{主导与型：}\exists(i,j),\text{使得}V_{n_i}^{x}\neq V_{n_j}^{x},i\neq j,\text{且}V_{n_i}^{x}=0,V_{n_j}^{x}=1 \tag{3-3}$$

$$\text{主导或型：}\exists(i,j),\text{使得}V_{n_i}^{x}\neq V_{n_j}^{x},i\neq j,\text{且}V_{n_i}^{x}=1,V_{n_j}^{x}=0 \tag{3-4}$$

其中 n_i 和 n_j 代表该 LUT 的两个不同输入信号线，n_i 为主导方，n_j 为被主导方，V_{n_i} 和 V_{n_j} 代表相应的逻辑值，x 为配置号。

4）反馈桥接故障约束：本文只考虑单个 LUT 内的反馈桥接故障，并将反馈桥接分为影响输出型和反馈输入型，对应的约束条件相比于以往研究中所使用的一系列等式约束更加简捷方便，使用简单的测试向量就能够覆盖反馈桥接故障的各种类型。其原理与无反馈桥接故障约束类似。因此，对于某一个 LUT 的影响输出型桥接的约束如以下公式所示：

$$\text{主导型：}\exists(i,f),\text{使得}V_{n_i}^{x}\neq V_{n_f}^{x}\text{且}i\neq f \tag{3-5}$$

$$\text{主导与型：}\exists(i,f),\text{使得}V_{n_i}^{x}\neq V_{n_f}^{x},i\neq f,\text{且}V_{n_i}^{x}=0,V_{n_f}^{x}=1 \tag{3-6}$$

$$\text{主导或型：}\exists(i,f),\text{使得}V_{n_i}^{x}\neq V_{n_f}^{x},i\neq f,\text{且}V_{n_i}^{x}=1,V_{n_f}^{x}=0 \tag{3-7}$$

其中 n_i 代表该 LUT 的第 n_i 条输入信号线，n_f 代表该 LUT 的输出信号线，V_{n_i} 和 V_{n_f} 代表其相应的逻辑值，n_i 为主导方，n_f 为被主导方，x 为配置号。

反馈输入型桥接与影响输出型桥接的区别在于主导和被主导的信号线相反，其约束与影响输出型桥接约束同理，不再赘述。

5）赋值冲突约束：对于特定的设计电路，其 LUT 之间的错综复杂连接，会使得同一条信号线分别连接到多个 LUT，而在上述对不同故障类型的激活约束过程中，可能会使得同一条信号线被赋值为多个逻辑值，即某条信号线在同一测试配置中被同时赋值“0”和“1”两种逻辑值，所以对于此类的赋值冲突需要进行避免。赋值冲突约束如公式（3-8）所示。

$$\text{若}\exists n_i^{lut1}=n_i^{lut2},\text{则必须使得}V_{n_i}^{lut1}=V_{n_i}^{lut2}\text{成立} \tag{3-8}$$

其中 n_i^{lut1} 和 n_i^{lut2} 表示相同的信号线连接到两个不同的 LUT，$V_{n_i}^{lut1}$ 和 $V_{n_i}^{lut2}$ 代表相应的信号逻辑值。

为了覆盖电路中所有的故障，生成的测试配置必须满足上述所有的约束条件。以图 3.5 所示电路为例，使用上述的约束条件进行测试向量生成，在表 3.2 中展示了所提出的方法生成的测试配置和对应的单项函数，并在表 3.3 中列出了相应测试配置所覆盖的故障，考虑到文章的篇幅，在表 3.3 中只列出了查找表 L1 的情况为例，并且省略了不同配置下重复覆盖的故障。

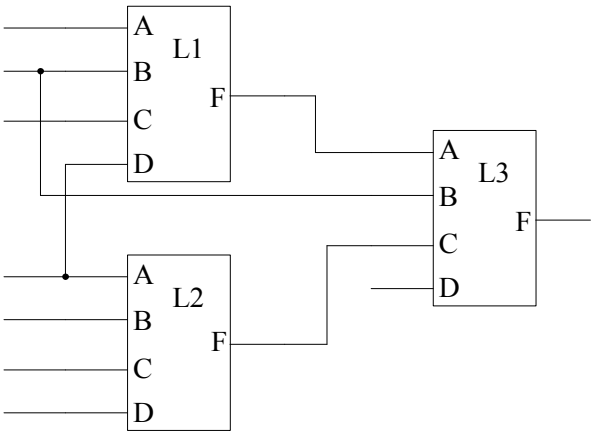


图 3. 5 示例电路

表 3. 2 针对示例电路生成的测试配置及对应的单项函数

LUT	LUT 输入	测试向量	生成的测试配置以及对应的 LUT 单项函数			
			配置 1	配置 2	配置 3	配置 4
L1	A	0011	00101	01110	10010	11001
	B	0101	F=A'·B'·C·D'	F=A+B'+C'+D'	F=A'+B+C+D'	F=A·B·C'·D'
	C	1100				
	D	0110				
	F	1001				
L2	A	0110	01110	10100	10011	01001
	B	1001	F=A+B'+C'+D'	F=A'+B+C'+D	F=A·B'·C'·D	F=A'·B·C'·D'
	C	1100				
	D	1010				
	F	0011				
L3	A	1001	10010	01001	00111	11100
	B	0101	F=A'+B+C+D'	F=A'·B·C'·D'	F=A'·B'·C·D	F=A'+B'+C'+D
	C	0011				
	D	1010				
	F	0110				

表 3.3 查找表 L1 的故障覆盖情况

LUT	配置 1	配置 2	配置 3	配置 4
L1	00101	01110	10010	11001
	A/1，B/1，C/0，D/1，F/0，A 主导 C，A 主导 F，B 主导 C，B 主导 F，C 主导 A，C 主导 B，C 主导 D，D 主导 C，D 主导 F，F 主导 A，F 主导 B，F 主导 D，A 主导与 C，A 主导与 F，B 主导与 C，B 主导与 F，D 主导与 C，D 主导与 F，C 主导或 A，C 主导或 B，C 主导或 D，F 主导或 A，F 主导或 B，F 主导或 D	B/0，D/0，F/1，A 主导 B，A 主导 D，B 主导 A，C 主导 F，D 主导 A，F 主导 C，A 主导与 B，A 主导与 D，F 主导与 B，F 主导与 C，F 主导与 D，B 主导或 A，B 主导或 F，C 主导或 F，D 主导或 A，D 主导或 F	A/0，C/1，B 主导 D，D 主导 B，B 主导与 A，B 主导与 D，C 主导与 A，C 主导与 D，F 主导与 A，A 主导或 B，A 主导或 C，A 主导或 F，D 主导或 B，D 主导或 C	C 主导与 B，C 主导与 F，D 主导与 A，D 主导与 B，A 主导或 D，B 主导或 C，B 主导或 D，F 主导或 C

从表 3.3 中可看出，所生成的测试配置不仅覆盖到所有固定故障，而且还覆盖到优化后的无反馈桥接故障和反馈桥接故障，成功解决了以往研究中的反馈桥接覆盖难题。

以表 3.3 中相同的形式统计示例电路中其他查找表的故障覆盖情况，然后在忽略掉不同配置下重复覆盖的故障的前提下，统计每个配置所覆盖到的故障数量，再计算出每个配置所覆盖的故障数占电路中互连故障总数的百分比。图 3.5 所示的示例电路中有 3 个 4 输入 LUT，因此示例电路中的互连故障总数为 210 个，在忽略掉重复覆盖的故障的前提下，分析出所生成的测试配置一覆盖到了 87 个互连故障，占总故障数的 41%，测试配置二覆盖到了 58 个故障，占总故障数的 69%，测试配置三覆盖到了 41 个故障，占总故障数的 89%，测试配置四覆盖到了剩余的 24 个故障，完成了 100%的故障覆盖，图 3.6 中展示了相应的百分比折线图。

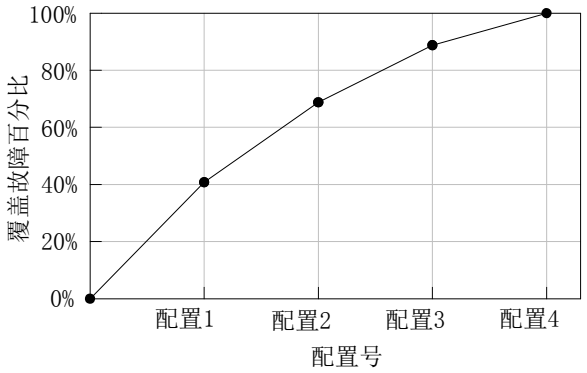


图 3.6 测试配置所覆盖的故障数占故障总数的百分比

上述的测试配置生成步骤是在对原始电路进行综合之后，使用 Python 脚本对综合后的电路网表文件提取电路的连接关系，再根据电路的连接关系对电路中的互连线进行故障约束以及赋值冲突约束，并调用 Python 的 SAT 库：Z3[56]，生成满足上述约束的测试配置，将生成的测试向量作为 LUT 的激活输入，计算出对应的单项函数。生成测试配置后，需要将原始电路修改为测试电路，即将原始电路中 LUT 的逻辑功能修改为测试配置中的单项函数。为了修改 LUT 的内容，使用“xdl -ncd2xdl **.ncd”命令将原始电路经过布局布线之后所生成的本地电路描述（Native Circuit Description，NCD）文件转成赛灵思设计语言（Xilinx Design Language，XDL）文件，其中 XDL 文件是可读文本文件。选择布局布线之后的 NCD 文件是为了防止电路中的资源映射位置发生改变。在将 NCD 文件转换成 XDL 文件之后，再使用脚本语言对 XDL 文件中的 LUT 逻辑功能进行批量修改，将修改好的 XDL 文件通过“xdl -xdl2ncd **.xdl”命令转回 NCD 文件，便得到了对应的测试电路，最后施加对应的测试向量进行测试。整个测试的步骤如图 3.7 所示。

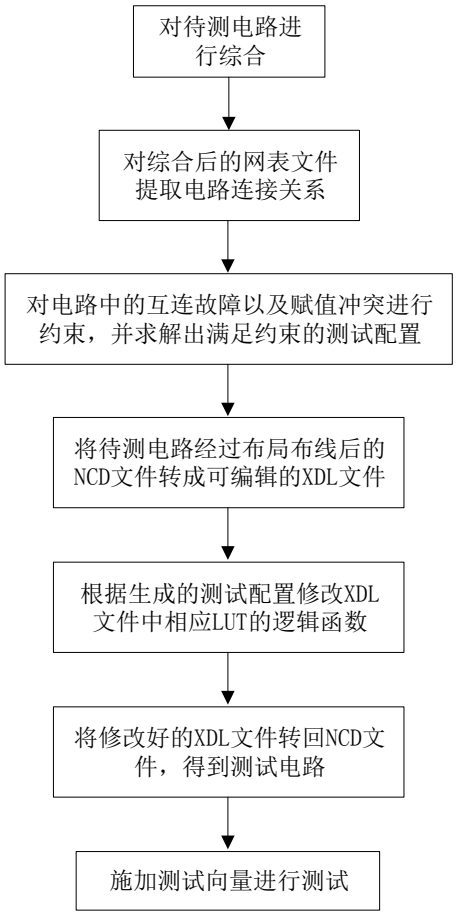


图 3. 7 测试流程图

在 FPGA 互连测试过程中不仅需要考虑故障的覆盖率，还需要考虑测试时间的成本，而测试中所花费的时间主要取决于测试配置的数量。结合先前的研究对本文所提出方案的测试配置数进行研究，因为文献[25]中提出激活 n 条信号线的所有固定和桥接故障只需要 $\log_2(n+2)$ 次测试配置，之后文献[28]中指出使用 $\log_2(n+2)$ 次配置无法覆盖“主导与”、

“主导或”型桥接故障，并提出在$2 \le n \le 7$时，需要多 1 次配置才能实现故障列表的全覆盖，而本文所提出方案只考虑了同一个 LUT 信号线之间的桥接故障，而单个 LUT 的信号线数量在 2 到 7 之间，所以本文的测试方案需要$\log_2(n+2)+1$次配置才能覆盖单个 LUT 的所有互连故障，对多个相互独立的 LUT 亦是如此，但电路中的 LUT 并非相互独立，LUT 与 LUT 之间存在连接关系，这种连接可能会使得$\log_2(n+2)+1$次配置产生赋值冲突，针对较为复杂的电路连接关系就需要额外的 1 个配置处理赋值冲突，因此理论上本文所优化的互连故障模型最多只需要$\log_2(n+2)+2$次配置就能实现故障列表的 100%覆盖，其中n为单个 LUT 的信号线数。在后续的测试验证过程中，不同基准电路实际所需的测试配置数与此处理论分析的测试配置数相同，且相比于现有文献中的测试配置数，本文所需的测试配置数是具有一定优势的。

3.4 本章小结

为解决现有应用相关互连测试方法中的反馈桥接故障覆盖问题，本章提出一种基于故障模型优化的互连测试方案。首先考虑到故障在测试过程中的传播问题，使用单项函数既能保证测试过程中故障的传播，还能保证故障的激活。然后根据现有互连故障测试文献中的不足之处，对互连故障模型进行优化改进，约束无反馈桥接故障和反馈桥接故障只发生在单个 LUT 的信号线之间，并将反馈桥接故障细分为影响输出型和反馈输入型，再结合单项函数对优化后的反馈桥接故障现象进行了分析，使用简单的测试向量就能够检测到相应故障的存在，证明了本文对故障模型的优化改进是有意义的。最后介绍了所提出方案中的测试配置生成步骤，使用布尔可满足性 SAT 对包括优化后的桥接故障在内的所有互连故障进行约束，生成满足约束条件的测试向量，在生成测试配置之后，将原始电路中的 LUT 功能修改为测试配置中的单项函数，得到相应的测试电路，再施加测试向量即可进行测试。针对本文的故障模型所生成的测试配置能很好地覆盖相应的互连故障，并且最多只需要$\log_2(n+2)+2$次配置就能实现故障列表的 100%覆盖（n为单个 LUT 的信号线数）。

第四章 基于粗细粒度故障的 FPGA 非自适应互连故障诊断

第三章所提出的应用相关互连测试方案用来检测用户电路中的互连资源是否存在故障，若其所有的测试配置都成功，说明电路中不存在互连故障；而若某个测试配置失败，存在输出响应不匹配的现象，说明电路中存在互连故障，这时就需要准确地对故障进行诊断定位，以便后续对故障进行容错，恢复电路的正常运行。

对于互连故障诊断，其主要的目的是提高故障诊断定位的精准度，在诊断精度相同时，诊断方案的诊断时间（诊断配置数）以及复杂度便成了主要的优化对象。文献[34]针对不同的故障类型采取不同的诊断方法，复杂度较大，所需的诊断配置数也较多。文献[35]对输出端检测到的互连故障进行反向回溯，根据路径的深度不断进行诊断步骤的迭代，方法过于繁琐，不适用于桥接故障，且诊断中需引入理想状况下的无故障空闲互连资源。文献[36]需要先对电路的路由分支进行激活和去激活处理，然后使用配置确定电路中可能存在的故障类型，最后再施加不同的诊断策略，步骤较为繁琐。文献[40]复用互连测试阶段生成的测试配置，结合相应的测试结果对电路中每根互连线都进行比对，所需诊断的故障范围较大。为优化现有诊断技术中存在的问题，本文在第三章的测试基础上，提出了一种基于故障粗细粒度范围的非自适应诊断方法，对测试阶段所检测到的单个互连故障进行诊断。

4.1 电路互连情况分析

被测电路中的连接关系交错复杂，为了一步步缩小单个故障的诊断范围，首先需要摸清电路的路径情况，通过对电路的每个输出端进行回溯，记录影响各自输出端的信号线，再分析影响每个输出端的信号线集合之间是否存在交集，从而判断出电路中的路径情况。

为了方便理解，将以图 4.1 电路为例进行说明。首先对图 4.1 电路的每个输出端口进行回溯，找到影响各自输出端的信号线集合，如影响输出端口 1 的信号线集合为 N1:{n1,n2,n3,n4,n9,n15,n16,n17,n27}；影响输出端口 2 的信号线集合为 N2:{n5,n6,n7,n8,n10,n11,n12,n13,n14,n18,n19,n20,n28}；影响输出端口 3 的信号线集合为 N3:{n5,n6,n7,n8,n10,n11,n12,n13,n14,n21,n22,n23,n29}；影响输出端口 4 的信号线集合为 N4:{n5,n6,n7,n8,n10,n24,n25,n26,n30}。再对影响各自输出端口的信号线集合进行分析，发现 N1 与 N2、N3、N4 之间都不存在任何交集，而 N2、N3、N4 之间存在共同交集 C1:{n5,n6,n7,n8,n10}。因此可判断，该电路由两条相互独立的路径组成，一条是输出端口 1 所在的单输出路径，另一条是输出端口 2、3、4 所在的多输出路径。

分析完电路的互连情况，再结合测试结果对故障所在路径进行判断。在测试过程中，如果某个测试配置成功，说明在该配置下所有输出端的值都是正确的；如果某个测试配置失败，说明存在相应输出端的值与正确输出响应不匹配，而不同位置和数量的输出端口不匹配，透

露着不同的故障位置信息。下面讨论在不同输出端口观测到不匹配现象时单个故障所在路径的情况。

1)单个输出端口不匹配：对于电路中存在的单个故障，若该故障只造成单个输出端口出现故障不匹配的情况，那么有两种可能。

a.故障存在于单输出路径上：如图 4.1 示例电路，若在某个失败测试配置下，只有输出端口 1 出现错误，根据电路互连情况的分析可知该故障存在于输出端口 1 所在的单输出路径上。

b.故障存在于多输出路径上：同样以图 4.1 所示电路为例，若在某个失败测试配置下，只有输出端口 2 出现错误，根据电路互连情况的分析可知故障存在于输出端口 2、3、4 所在的多输出路径上。

2）多个输出端口不匹配：针对电路中存在的单个故障，如果该故障导致多个输出端口出现故障不匹配的情况，那么只有一种可能，即故障存在于多输出路径上。以图 4.1 电路为例，若在某个失败测试配置下，输出端口 2 和输出端口 3 出现错误，结合上述对电路互连情况的分析可知故障存在于输出端口 2、3、4 所在的多输出路径上。

上述的所有可能情况都是基于单个失败的测试配置进行分析的，由于电路中只存在单个互连故障，其所能影响的输出端口不会发生改变，即在不同的失败测试配置下只会出现相同的不匹配输出端口情况，因此只需对任意一个失败的测试配置进行分析即可。

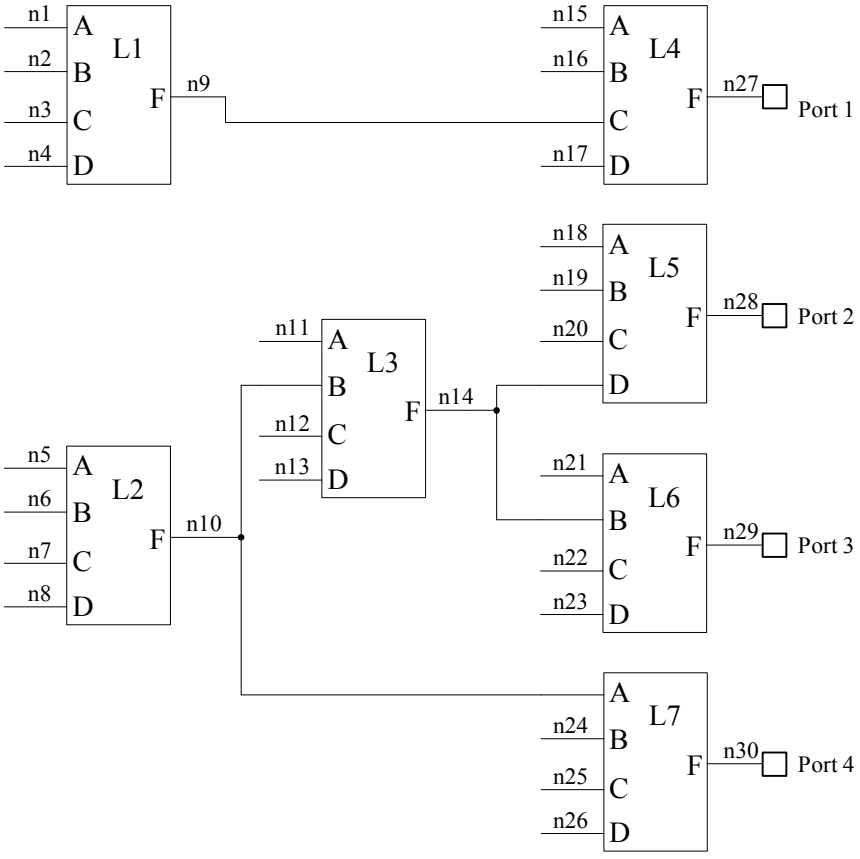


图 4. 1 简单示例电路

4.2 故障粗粒度范围诊断

在上一小节中结合电路的互连情况对任意单个失败的测试配置进行分析，可知单故障所在的路径，因此后续的步骤都无需考虑电路中其他无故障路径，缩小了单故障的诊断范围。下面根据故障所在路径，对故障范围进行粗粒度诊断定位，为后续细粒度诊断做铺垫。

4.2.1 单输出路径

在图 4.1 所示电路中，若故障存在于输出端口 1 所在的单输出路径，因为该路径只存在单个输出端口，所以该单输出路径上的任何一条信号线存在故障都会影响到输出端口 1 处的值，由于该单输出路径的 LUT 数量较少，连接关系较简单，可直接对该路径上的信号线进行细粒度精准诊断。

但如果某一单输出路径过于复杂，可根据该单输出路径的不同输入分支的组合深度进行二分法缩小故障的范围。例如图 4.1 中可将输出端口 1 所在的单输出路径分为查找表 L1 所在的左边区域，以及查找表 L4 所在的右边区域。然后将查找表 L1 配置为“乘积的和”（Sum Of Product，SOP）（$F=A\cdot\overline{A}+B\cdot\overline{B}+C\cdot\overline{C}+D\cdot\overline{D}$）或“和的乘积”（Product Of Sum，POS）（$F=(A+\overline{A})\cdot(B+\overline{B})\cdot(C+\overline{C})\cdot(D+\overline{D})$）。在 LUT 输入端输入任意值，配置为 SOP 的 LUT 的输出值都为 0，配置为 POS 的 LUT 的输出值都为 1。将查找表 L1 配置为 SOP 或 POS，即可使得 n9 信号线上的值与查找表 L1 的输入端无关，这时观察输出端口 1，若在输出端口 1 处未发现输出错误，则说明单输出路径上 L4 所在的右边区域不存在故障，即故障存在于单输出路径上 L1 所在的左边区域；若在输出端口 1 发现输出错误，则说明故障存在于单输出路径上 L4 所在的右边区域。

4.2.2 多输出路径

若故障存在于多输出路径上，该故障所在粗粒度范围可根据电路互连情况以及错误输出端口信息进行判断。如图 4.1 所示电路，若故障存在于输出端口 2、3、4 所在的多输出路径上，通过对电路互连情况的分析，可找到只影响相应端口的信号线，例如同时只影响输出端口 2 和 3 的信号线为{n11,n12,n13,n14}，同时只影响输出端口 2、3、4 的信号线为{n5,n6,n7,n8,n10}。再结合错误输出端口信息，即可判断故障的粗粒度范围，若在测试过程中只观察到输出端口 2、3 出现不匹配现象，则可确定故障存在于{n11,n12,n13,n14}信号线上；而若在测试过程中观察到输出端口 2、3、4 出现不匹配现象，可确定故障存在于{n5,n6,n7,n8,n10}信号线上。

上述步骤可确定单故障所在的粗粒度范围，故障诊断区域大大缩小，所需诊断的信号线数量也大量减少，为后续细粒度的故障精准定位提供了便利。

4.3 故障细粒度精准诊断

在确定故障所在粗粒度范围后，再使用复用测试配置法对故障所在区域内的故障信号线

进行细粒度精准诊断定位。

如果在对图 4.1 所示电路进行互连测试时，发现任意的失败配置中都是输出端口 2、3、4 出现不匹配，结合电路的互连情况进行分析，可知故障存在于电路中的多输出路径，再对该路径进行粗粒度故障范围诊断，确定故障存在于多输出路径的{n5,n6,n7,n8,n10}信号线上。下面通过使用复用配置法对粗粒度范围内的故障进行细粒度精准诊断定位。

4.3.1 复用配置法

复用配置法，就是通过复用测试配置和测试结果，对不同测试配置中信号线的值进行比对，排除无故障信号线的嫌疑，最终找出失败配置的原因，实现故障所在信号线的精准定位。下面介绍复用配置法诊断单个固定故障和桥接故障的原理。

使用复用配置法诊断单个固定故障，首先从成功的配置中排除无故障信号线，若某一信号线在不同的成功配置中有被赋为“0”和“1”，说明该信号线上的值不会影响到测试结果，即该信号线上不存在固定故障；再分析失败的配置，若某一信号线在不同的失败配置中被赋为“0”和“1”，同样说明该信号线上的值不会影响到测试结果，即该信号线上无固定故障的嫌疑。若复用配置法最终定位到多个固定故障嫌疑时，则需要使用额外的配置对多余的故障嫌疑进行排除。

使用复用配置法诊断单个桥接故障，首先从成功的配置中排除无故障信号线对，若某一对信号线在任意成功配置中被赋为“01”或“10”，说明该对信号线不存在相应类型的桥接故障；再分析失败的配置，若某一对信号线在任意失败配置中被赋为“00”或“11”，说明该对信号线不存在任何桥接故障。同样若最终定位到多个桥接故障嫌疑时，也需要使用额外的配置对多余的故障嫌疑进行排除。

4.3.2 固定故障诊断

因为电路中只存在单个互连故障，要么是固定故障，要么是桥接故障。在不知道故障的类型下，可先假设该故障为最常见的固定故障进行诊断。在表 4.1 中列出了故障粗粒度范围{n5,n6,n7,n8,n10}内信号线的测试配置及结果。

表 4. 1 测试配置结果

配置号	故障所在粗粒度范围					测试结果
	n5	n6	n7	n8	n10	
一	0	1	0	0	1	失败
二	1	1	0	1	0	失败
三	1	0	1	0	0	成功
四	0	0	1	1	1	成功

表 4.2 为复用表 4.1 中的测试配置对固定故障进行精准诊断定位的结果。从表 4.2 中可看到第 3 和 4 个测试配置是成功的，信号线 n5、n8 以及 n10 在不同的成功配置中都有被赋为逻辑值“0”和“1”，即可排除信号线 n5、n8、n10 的固定故障嫌疑。信号线 n6 在成功的配置中都只被赋为逻辑值“0”，信号线 n7 同样在成功的配置中都只被赋为逻辑值“1”，并且信号线 n6、n7 在不同的失败配置中也都只被赋为相同的逻辑值，因此即可怀疑信号线 n6 存在固定 0 故障，信号线 n7 存在固定 1 故障。由于电路中只存在单个故障，所以需要再使用额外的配置排除多余的故障嫌疑，将配置一中 n7 信号线的逻辑值“0”修改为逻辑值“1”，以验证 n7 的固定 1 故障，即使用“01101”配置进行测试，表 4.2 中该配置测试结果成功，说明故障为信号线 n7 的固定 1 故障。

表 4. 2 固定故障诊断结果

配置号	故障所在粗粒度范围					测试结果
	n5	n6	n7	n8	n10	
一	0	1	0	0	1	失败
二	1	1	0	1	0	失败
三	1	0	1	0	0	成功
四	0	0	1	1	1	成功
	n5/0	**n6/0**	n7/0	n8/0	n10/0	
	n5/1	n6/1	**n7/1**	n8/1	n10/1	
额外的配置，排除多余的固定故障嫌疑						
五	0	1	1	0	1	成功
用来确定故障类型的配置						
六	0	0	0	0	0	失败

上述诊断结果是在假设故障为固定故障的前提下完成的，至于故障类型是否为对应的固定故障，则需要使用一个配置最终确定故障类型，在该配置中只需激活相应的固定故障，并使得其他信号线与该信号线的值相同，避免激活桥接故障。表 4.2 中为确定故障是否是信号线 n7 的固定 1 故障，使用了“00000”配置进行诊断，表 4.2 中显示该配置测试结果失败，说明故障就是假设的固定故障，诊断工作结束，最终真正的互连故障发生在信号线 n7 上，且为固定 1 故障。

而若表 4.2 中的配置六成功，说明故障不是所假设的固定故障，故障类型错误，需要重新复用配置对桥接故障进行诊断，此时可复用的配置除了测试配置，还有诊断固定故障时使用的配置五和配置六，即直接复用表 4.2 中的所有配置就可完成桥接故障的诊断，结合复用配置法诊断桥接故障的原理，可诊断出最终真正的故障发生在信号线 n6 与 n7 之间，且为“10”组合。

4.3.3 桥接故障诊断

同样也可先假设故障为桥接故障进行故障诊断工作。复用表 4.1 中的测试配置对{n5,n6,n7,n8,n10}内信号线的桥接故障进行诊断，先分析成功的配置，例如表 4.1 中的第 3 次测试配置，在该成功配置中可看到信号线 n5 与 n6 被赋为“10”，即可排除信号线 n5 与 n6 的“10”组合；其次分析失败的配置，例如表 4.1 中的第 1 次测试配置，在该失败配置中信号线 n5 和 n7 被设置为“00”，即可判断信号线 n5 和 n7 之间不存在任何桥接故障，该配置的失败是另有原因的。对表 4.1 中的测试配置分析完备之后，最终剩下信号线 n8 和 n10 的“01”组合、“10”组合，以及信号线 n6 和 n7 的“10”组合，相应的桥接故障诊断结果如表 4.3 所示。因为电路中只存在单个故障，为了排除多余的故障嫌疑，需要使用额外的配置进行诊断，并根据相应配置的结果确定真正的桥接故障。在表 4.4 中展示了整个诊断过程所需的配置以及测试结果，其中配置五是用来排除多余的故障嫌疑的，在该配置中信号线 n6 和 n7 的“10”组合被激活，而信号线 n8 和 n10 被保持为“11”状态，表中该配置测试失败，说明相应的桥接故障发生在信号线 n6 和 n7 之间，且为“10”组合。

表 4. 3 桥接故障诊断结果

信号线		1				
		n5	n6	n7	n8	n10
0	n5	\	√	√	√	√
	n6	√	\	√	√	√
	n7	√	×	\	√	√
	n8	√	√	√	\	?
	n10	√	√	√	?	\

同样诊断完桥接故障需要对最终的故障类型进行确定，判断故障类型是否是假设的桥接故障，即上述发生在信号线 n6 和 n7 之间的桥接故障有可能是 n6 的固定 0 故障，或 n7 的固定 1 故障，因为桥接故障牵扯到两条信号线，所以最多需要两个配置来确定故障类型是否是桥接故障。在这两个配置中，需要使这两条信号线一直保持相同的“0”和相同的“1”，从而保证相应的桥接故障不被激活，并验证对应的固定故障嫌疑。表 4.4 中为确定故障是否是信号线 n6 和 n7 之间的“10”组合，将配置一中信号线 n6 的逻辑值“1”修改为“0”，保证信号线 n6 和信号线 n7 的值相同，即使用配置六“00001”以验证 n7 的固定 1 故障嫌疑；同样将表 4.4 中配置一的信号线 n7 的逻辑值“0”修改为“1”，保证信号线 n6 和信号线 n7 的值相同，即使用配置七“01101”以验证 n6 的固定 0 故障嫌疑，表 4.4 中相应的诊断配置结果都是成功的，因此可最终确定真正的故障就是桥接故障，发生在信号线 n6 和 n7 之间，且为“10”组合。

表 4. 4 测试配置及桥接故障诊断配置

配置号	故障所在粗粒度范围					测试结果
	n5	n6	n7	n8	n10	
一	0	1	0	0	1	失败
二	1	1	0	1	0	失败
三	1	0	1	0	0	成功
四	0	0	1	1	1	成功
额外的配置，排除多余的桥接故障嫌疑						
五	0	1	0	1	1	失败
用来确定故障类型的配置						
六	0	0	0	0	1	成功
七	0	1	1	0	1	成功

而若表 4.4 中的配置六和配置七存在任意一个失败的结果，即可证明故障不是所假设的桥接故障。若故障不是假设的桥接故障，则需要重新复用配置对固定故障进行诊断，但此时可复用的配置除了测试配置，还有诊断桥接故障时使用的配置五、配置六以及配置七，即直接复用表 4.4 中所有的配置就可完成固定故障的诊断。例如在配置六失败，配置七成功的情况下，结合复用配置法诊断固定故障的原理可精准定位到最终真正的故障为信号线 n7 的固定 1 故障；在配置六成功，配置七失败的情况下，同样可精准定位到最终真正的故障为信号线 n6 的固定 0 故障。

需要注意的是，虽然桥接故障存在主导方和被主导方，但通常情况下，后续的故障容错方案只需要知晓故障所在信号线以及故障在何种情况下被激活，而无需明确桥接故障具体的主导方和被主导方，因此本方案只完成故障所在位置信息以及故障在何种情况下被激活的诊断。整个诊断方案的流程如图 4.2 所示。

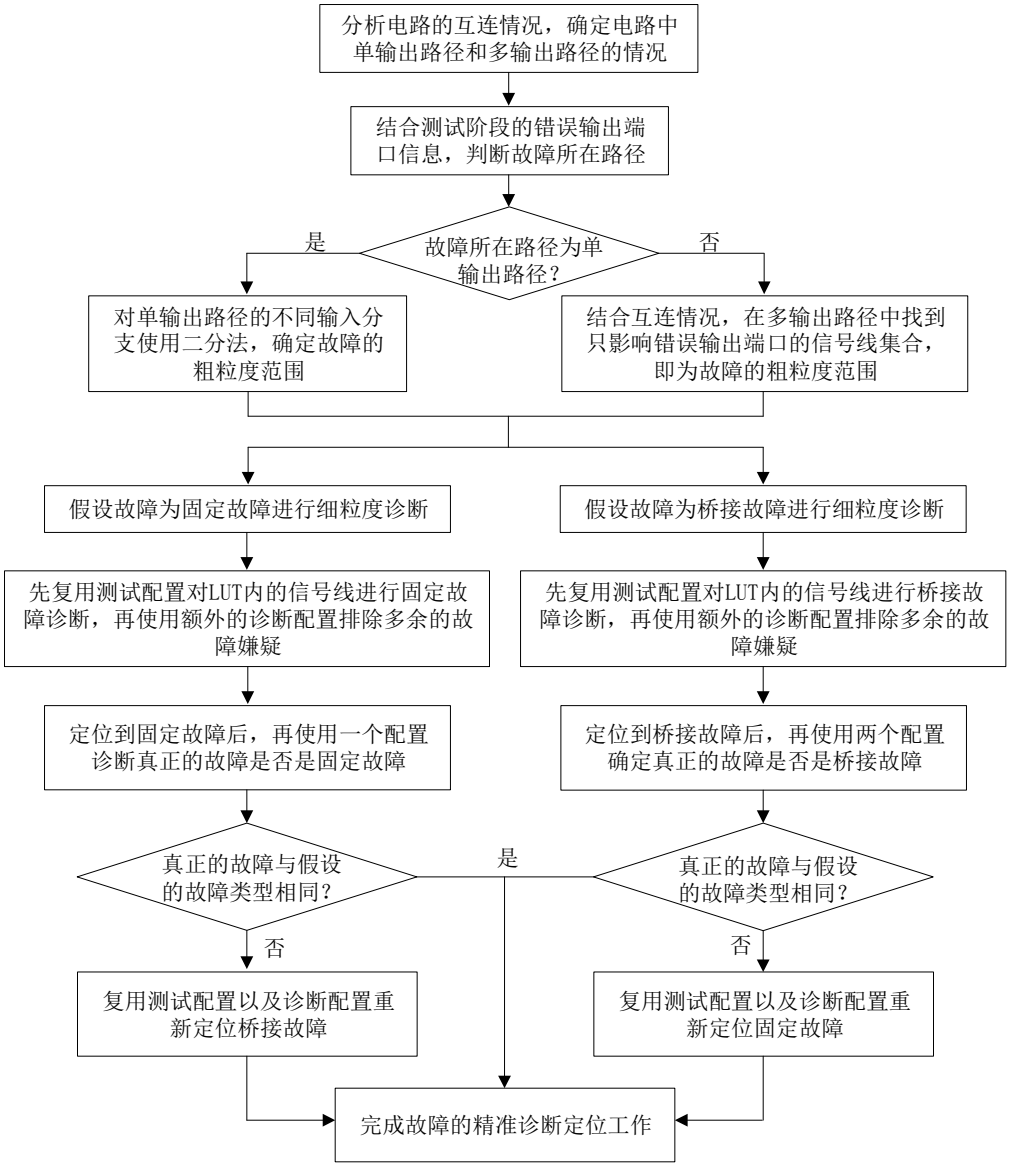


图 4. 2 诊断流程图

4.4 诊断配置数分析

本文所提出的诊断方法在分析完电路的互连情况后，结合测试阶段所观察到的输出端口错误信息，明确故障所在的路径，再根据故障所在路径采用对应粗粒度诊断方法，确定故障的粗粒度范围，最后再复用测试配置，对单个固定故障或桥接故障进行细粒度精准定位。其中本文复用的配置是第三章测试方案中所生成的测试配置，针对不同的电路所需要的最大测试配置次数为 $\log_2(n+2)+2$， n 为单个 LUT 的信号线数量。

细粒度精准诊断故障时，诊断步骤不同，所需要的诊断配置数就会不同。若先假设故障为固定故障，固定故障诊断的最好情况是直接复用 $\log_2(n+2)+2$ 次测试配置即可定位固定故障，最坏的情况则是需要使用额外配置排除多余的故障嫌疑，因为电路中可能存在相互独立的 LUT（无赋值冲突约束），这些相互独立的 LUT 上的信号线在生成的测试配置中可能被赋为相同的值，因而在细粒度精准定位时就会存在多个 LUT 上都存在故障嫌疑，若此时

故障粗粒度范围内存在 m 个 LUT，这时就需要 $\log_2 m$ 次配置以确定故障最终所在的 LUT 信息，然后再使用额外一个配置排除对应 LUT 内多余的固定故障嫌疑，因而需要额外 $(\log_2 m)+1$ 次诊断配置。在诊断完固定故障时，需要最后再使用 1 个配置确定真正的故障是否是假设的固定故障，若是固定故障，则整个诊断工作完成，若不是固定故障，复用已有的测试和诊断配置即可完成最终的桥接故障诊断，因此先假设故障为固定故障的诊断路线最终需要 $\left[\log_2(n+2)+2\right]+(\log_2 m)+1+1$ 次配置，其中 n 为单个 LUT 的信号线数，m 为故障粗粒度范围内的 LUT 数量。

若先假设故障为桥接故障，诊断桥接故障的最好情况同样是直接复用测试配置即可定位桥接故障，此时的配置数为 $\log_2(n+2)+2$，其中 n 为单个 LUT 的信号线数，而最坏情况也需要使用额外配置排除多余的故障嫌疑，其最终所需的配置数与故障粗粒度范围内 LUT 的数量有关，若故障粗粒度范围内存在 m 个 LUT，首先需要 $\log_2 m$ 次配置确定故障所在 LUT，然后再使用额外一次配置排除对应 LUT 内多余的桥接故障嫌疑，额外诊断配置数为 $(\log_2 m)+1$，m 为故障粗粒度范围内的 LUT 数量。在诊断完桥接故障时，需要最后使用 2 个配置确定真正的故障是否是假设的桥接故障，若是桥接故障，则整个诊断工作完成，若不是桥接故障，复用已有的测试和诊断配置也可完成最终的固定故障诊断，因此先假设故障为桥接故障的诊断路线最终需要 $\left[\log_2(n+2)+2\right]+(\log_2 m)+1+2$ 次配置，其中 n 为单个 LUT 的信号线数，m 为故障粗粒度范围内的 LUT 数量。

从上述对诊断配置数的分析中可知，相比于一开始诊断固定故障的路线，先诊断桥接故障的步骤更为复杂，且所需的诊断配置数也更多，并可得出本文所提出的诊断方案最多需要 $\left[\log_2(n+2)+2\right]+(\log_2 m)+3$ 次配置就可检测并定位单个任意故障。在后续的诊断验证过程中，根据该公式分析不同的基准电路所需的最大诊断配置数，并与现有诊断文献进行比较，结果表明本文所需的最大诊断配置数存在一定的优势。

4.5 本章小结

为优化现有互连诊断技术中存在的复杂度较大和诊断配置数过多的问题，本章在测试工作的基础上，提出了一种基于故障粗细粒度范围的非自适应单故障诊断方案。该方案首先对电路的互连情况进行分析，判断出电路中的路径情况，并结合测试过程中的错误输出端口信息确定故障所在路径。然后再根据故障所在路径类型选择对应的粗粒度诊断方法，对故障的粗粒度范围进行诊断，缩小了故障的诊断范围，减少了诊断工作的复杂度。接着使用复用配置法对故障粗粒度范围内的信号线进行细粒度精准定位，在不知故障类型的前提下，介绍了先假设故障为固定故障和先假设故障为桥接故障两种不同的诊断路线，并分析了这两种诊断路线各自所需的诊断配置数，诊断顺序不同，所需要的诊断配置数也会不同，最后比较这两条诊断路线所需的诊断配置数大小，得出本文所提出的方案最多需要 $\left[\log_2(n+2)+2\right]+(\log_2 m)+3$ 次配置就可诊断定位单个任意故障（n 为单个 LUT 的信号线数，m 为故障粗粒度范围内的 LUT 数量）。

第五章 基于细粒度冗余的 FPGA 修复型互连故障纠错容错

在第三章应用相关互连测试过程中检测到用户电路存在故障，再使用第四章中的非自适应互连诊断方案对电路中的单个任意故障进行精准诊断定位，明确互连故障的类型、所在位置以及在何种情况下被激活。在掌握互连故障的相关信息之后，就需要采取容错措施对电路中的互连故障进行修复容错，以恢复电路的正常工作。

互连资源修复型容错技术都是使用 FPGA 内部的冗余资源对互连故障进行容错，其中硬件级容错大多基于备用逻辑行/列的移位，但需要增加的资源面积消耗较大；配置级容错使用布局布线算法对电路进行重新映射，实现的复杂度较大，并且在出现故障时需要不断地对故障进行规避，对 FPGA 内部空闲资源的要求过大。为改进现有修复型容错技术的不足，本文提出一种基于细粒度冗余的 FPGA 修复型互连故障容错方案，对测试和诊断到的互连故障进行检错纠错。

5.1 细粒度双模冗余

在对待测电路进行互连故障测试和诊断定位之后，可明确电路中互连故障所在的位置以及互连故障类型，首先根据诊断到的互连故障类型对互连故障附近的原始逻辑资源进行细粒度的双模冗余复制，下面以一个示例电路为例进行说明，示例电路如图 5.1 所示。

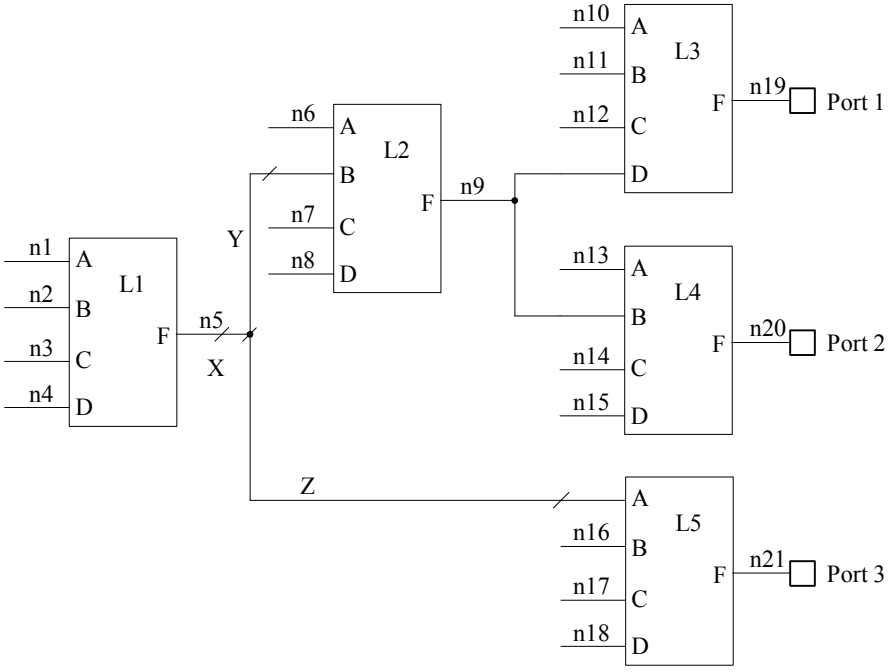


图 5. 1 简单示例电路

1）当互连故障为只涉及到一条信号线的固定故障时，将该固定故障导致的错误映射到所在信号线附近的原始逻辑资源上，再对映射后的原始逻辑资源进行基于 LUT 的细粒度冗

余复制。假设在对图 5.1 示例电路进行互连故障诊断之后，定位到故障存在于信号线 n5 上，为固定故障，因为信号线是由一系列金属互连线以及可编程开关点连接而成的，由图 5.1 中 n5 的连接关系可知信号线 n5 是从查找表 L1 所在 CLB 的输出端连接到金属互连线，再由金属互连线进入开关矩阵改变布线方向，并分两路进行路由，最后分别连接到查找表 L2 所在的 CLB 以及查找表 L5 所在的 CLB，示例电路中信号线 n5 的路由示意图如图 5.2 所示。若在互连故障测试过程中输出端口 1、2、3 都存在输出不匹配现象，可判断该固定故障发生在进入开关矩阵之前的互连资源上，即靠近查找表 L1 所在 CLB 的 X 段上，这种情况下，需要将该固定故障映射到查找表 L1 上，并对查找表 L1 进行细粒度冗余复制。而若在互连故障测试过程中只有输出端口 1、2 存在输出不匹配现象，可判断该固定故障发生在进入开关矩阵之后朝着查找表 L2 所在 CLB 进行路由的互连资源上，即靠近查找表 L2 所在 CLB 的 Y 段上，这种情况下，需要将该固定故障所导致的错误映射到查找表 L2 上，并对查找表 L2 进行细粒度冗余复制。同理，若在互连故障测试过程中只有输出端口 3 存在输出不匹配现象，可判断该固定故障存在信号线 n5 的 Z 段上，需要将该固定故障映射到查找表 L5 上，并对查找表 L5 进行细粒度冗余复制。

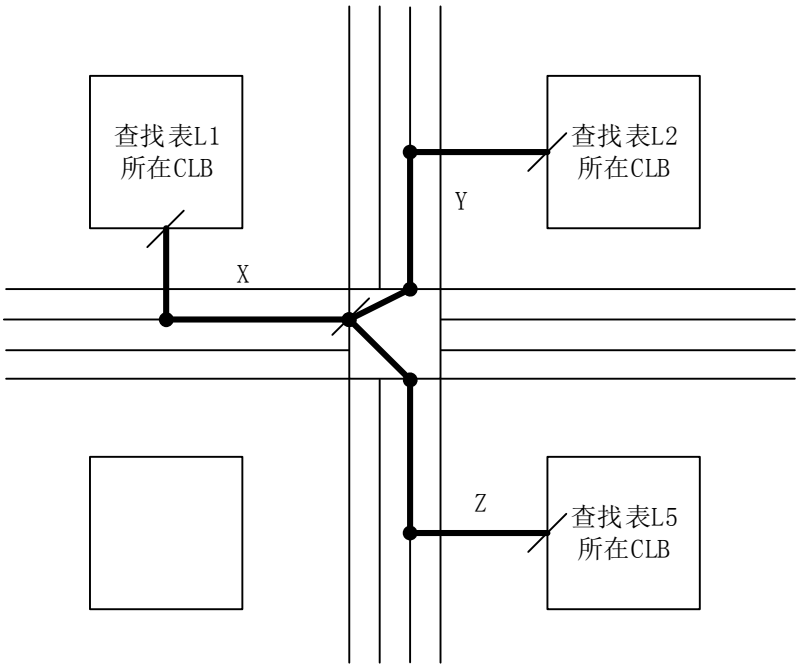


图 5. 2 示例电路的信号线 n5 路由示意图

2）当互连故障为存在于一对信号线上的桥接故障时，两条信号线中的任意一条都可能受到桥接故障的影响，在故障容错时就需要考虑两条信号线。但桥接故障存在“主导”、“主导与”以及“主导或”等类型，若在诊断过程中多使用一次诊断配置对桥接故障的主导方和被主导方进行明确的话，容错则只需对被主导信号线附近的原始逻辑资源进行冗余复制，这将极大方便容错策略的实施，然而多使用一次诊断配置会导致诊断配置数的增加，所带来的代价较大。而本文因为在互连测试和诊断过程中都只考虑了单个 LUT 内信号线的桥接故障，即存在桥接故障的一对信号线是连接了同一个 LUT 的，因此直接对该 LUT 进行冗余复制即

可，方便了相应容错策略的实施。所以对于桥接故障的容错，无需明确桥接故障的主导方和被主导方，直接将桥接故障导致的错误映射到该对信号线共同连接的原始逻辑资源上，并对映射后的原始逻辑资源进行基于 LUT 的细粒度复制冗余。假设在对图 5.1 示例电路进行互连故障诊断之后，定位到故障存在于信号线 n1 和信号线 n5 之间，为桥接故障，因为该桥接故障发生在查找表 L1 的信号线上，所以选择将该桥接故障映射到查找表 L1 上，并对查找表 L1 进行细粒度冗余复制。而若在互连故障诊断之后，定位到故障存在于信号线 n5 和信号线 n6 之间，为桥接故障，因为该桥接故障发生在查找表 L2 的信号线上，所以将该桥接故障映射到查找表 L2 上，对查找表 L2 进行冗余复制即可。

5.2 检错纠错电路

在完成细粒度双模冗余复制之后，需要对被复制的原始资源和冗余资源进行处理，即在电路中插入检错纠错电路对故障进行修复，而在插入检错纠错电路之后，需要改变原始电路的部分路由以完成新电路的连接，因此先对插入检错纠错电路后原始资源和冗余资源的路由情况进行分析，示例电路的原始路由示意图如图 5.3 所示。

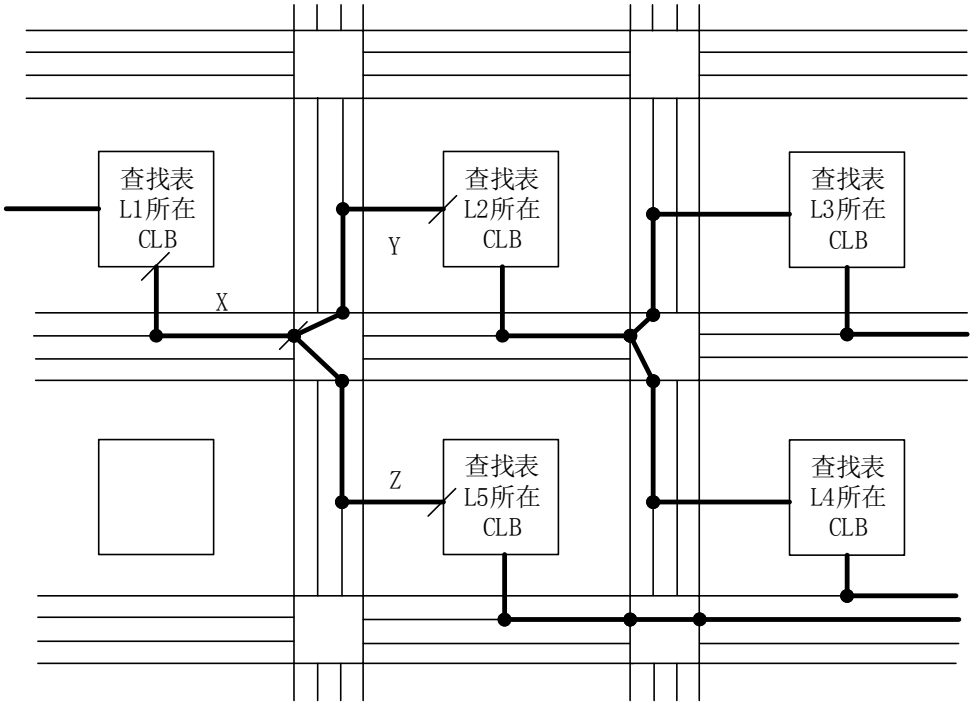


图 5. 3 示例电路的原始路由示意图

在插入检错纠错电路后，对于冗余 LUT，其相关的连接需要避开互连故障节点进行路由，保证复制后的冗余 LUT 不受互连故障的影响，而被复制的原始 LUT 的路由情况则与互连故障节点的位置相关。

1）若被复制的原始 LUT 在互连故障节点之前，对于原始 LUT 所在 CLB 到检错纠错电路之间的路由需要囊括互连故障节点，即需要保证新电路中仍然存在先前诊断定位到的互连故障。例如在对图 5.1 示例电路进行互连故障诊断之后，定位到固定故障存在于信号线 n5

的 X 段上，因此在细粒度冗余时选择复制查找表 L1，此时原始查找表 L1 在互连故障节点之前，且该固定故障并不会影响到查找表 L1 的输出，所以插入容错电路之后，原始查找表 L1 所在 CLB 到检错纠错电路之间的路由需要囊括信号线 n5 的 X 段，然后再在开关矩阵中改变信号线 n5 的布线方向，完成到检错纠错电路的路由，再将经过检错纠错处理的正确输出连接到后续电路，实现容错。插入容错电路后的相关路由示意图如图 5.4 所示。

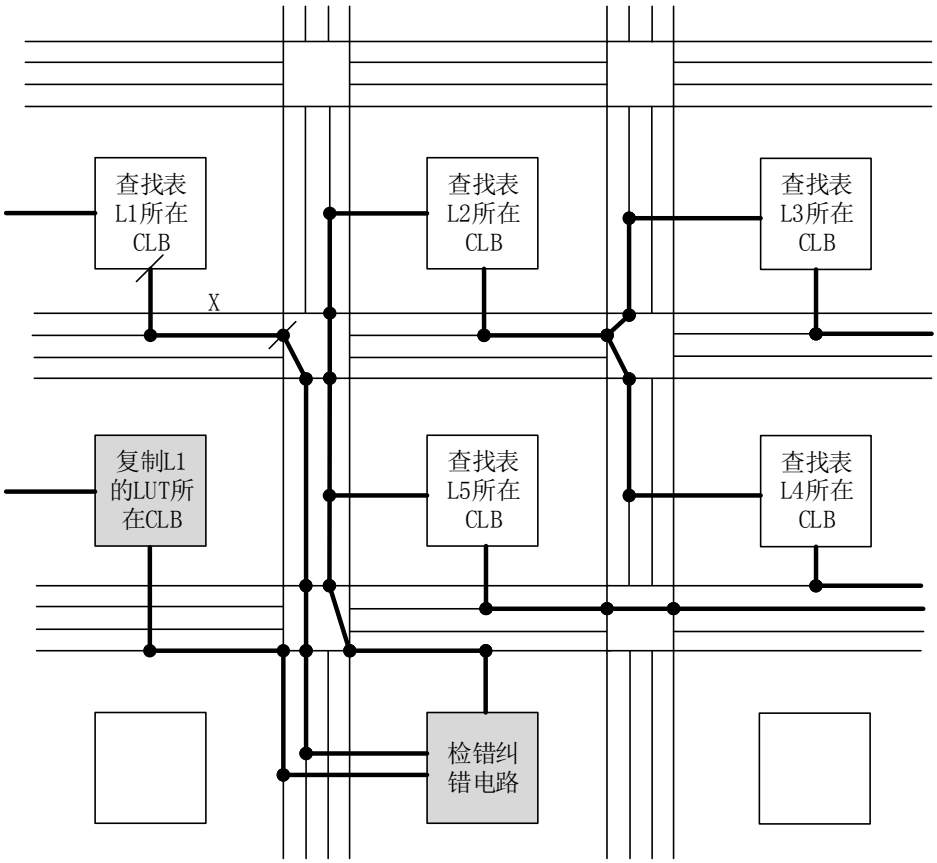


图 5. 4 互连故障在 X 段上的容错电路路由示意图

2）若被复制的原始 LUT 在互连故障节点之后，因为此时的互连故障会影响到原始 LUT 的输出，所以故障节点到原始 LUT 所在 CLB 之间的路由不能更改，即同样需要保证新电路中仍然存在先前诊断定位到的互连故障。例如对图 5.1 示例电路进行互连故障诊断之后，定位到固定故障存在于信号线 n5 的 Y 段上，因此在细粒度冗余时选择复制查找表 L2，而此时原始查找表 L2 在互连故障节点之后，该固定故障会影响到查找表 L2 的输出，所以插入容错电路之后，信号线 n5 到查找表 L2 所在 CLB 之间的原始路由不能更改，只需保证信号线 n5 到冗余 LUT 之间的路由避开 n5 信号线的 Y 段，然后再将原始 LUT 和冗余 LUT 的输出连接到检错纠错电路，并将经过检错纠错之后的正确输出连接到后续电路。插入容错电路之后的相关路由示意图如图 5.5 所示。

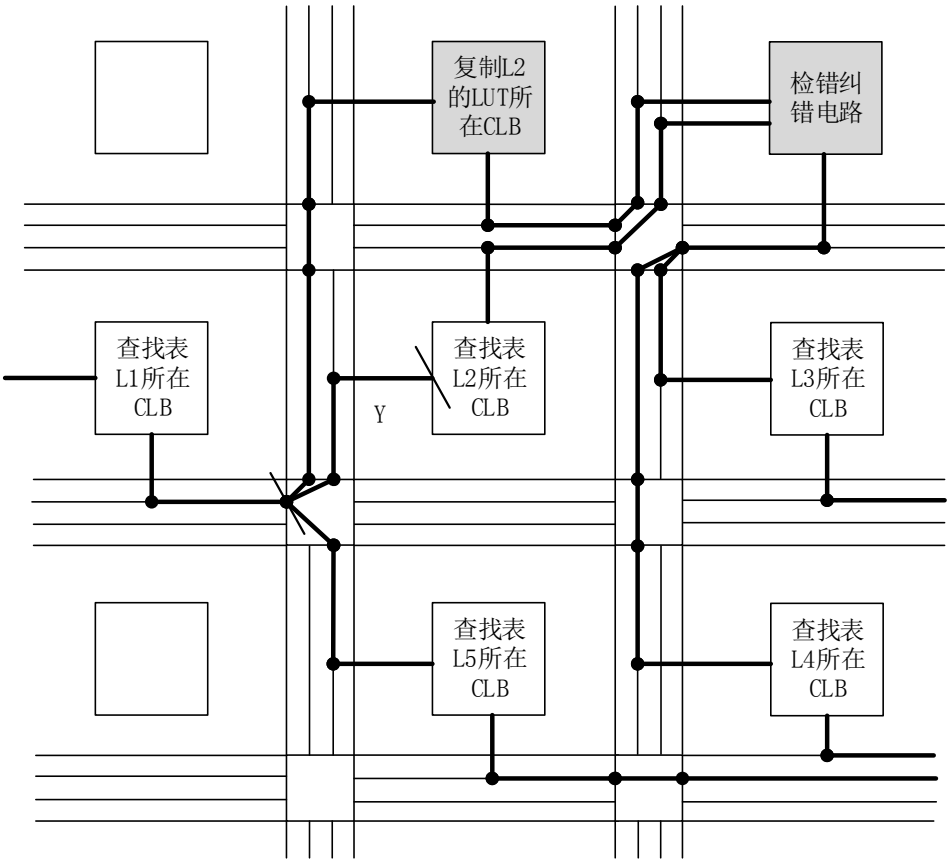


图 5. 5 互连故障在 Y 段上的容错电路路由示意图

分析完插入容错电路之后的路由情况，再分析容错电路的功能，该容错电路能实现对互连故障的检错纠错，容错电路图如 5.6 所示。

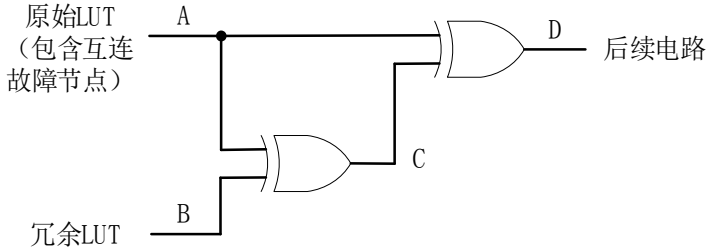


图 5. 6 容错电路

在图 5.6 电路中对被复制的原始 LUT 和冗余 LUT 进行了两次异或处理，该电路的真值表如表 5.1 所示。第一次异或处理为双模冗余资源的输出之间的异或，实现对互连故障的监控，因为在用户电路正常工作中，互连故障不一定被激活，因而不一定会使得电路发生错误变化，而且就算互连故障被激活，根据原始 LUT 逻辑函数的不同，互连故障的错误也不一定会使得原始 LUT 的输出发生改变，所以将原始 LUT 的输出和冗余 LUT 的输出进行异或，可检测互连故障是否导致原始 LUT 的输出发生错误改变。表 5.1 中第三列为第一次异或处理后的结果，当第一次异或结果为“1”时，说明电路中存在的互连故障导致原始 LUT 的输出发生了错误变化，当第一次异或结果为“0”时，说明电路中存在的互连故障未影响到原

始 LUT 的输出。第二次异或处理为第一次异或的结果与原始 LUT 的输出之间的异或，实现对互连故障的纠错，表 5.1 中第四列为第二次异或处理的结果。若电路中的互连故障未使原始 LUT 的输出发生改变，即第一次异或结果为“0”，再将第一次异或结果与原始 LUT 的输出进行异或，由于“0”与任何值的异或结果都是该值本身，所以第二次异或的输出结果与原始 LUT 的正确输出相同，此时将第二次异或的结果连接到后续电路，并不会影响电路的正常运行；若电路中的故障导致原始 LUT 的输出发生错误改变，即第一次异或结果为“1”，再将第一次异或结果与原始 LUT 的输出进行异或，因为“1”与任何值的异或结果都是该值的相反值，所以第二次异或后的输出结果与原始 LUT 的错误输出值相反，纠正了互连故障所导致的原始 LUT 输出错误，再将第二次异或的结果连接到后续电路，恢复电路的正常运行，完成故障的容错。

表 5. 1 容错电路真值表

A：原始 LUT（包含互连故障节点）	B：冗余 LUT	C：第一级异或结果（C=A^B）	D：第二级异或结果（D=A^C）
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	1

本文所提出的容错方案是通过对互连故障附近的原始逻辑资源进行基于 LUT 的细粒度冗余复制，然后再在原始电路中插入检错纠错电路实现容错，该方案使用两次异或操作实现对互连故障的检错纠错，直接处理互连故障引起的错误改变，并且在互连故障未被激活时也不会影响正常电路的工作，造成的面积消耗也非常的少，成功的弥补了现有修复型容错技术的缺点。

5.3 本章小结

在经过第三章互连测试以及第四章互连诊断后，掌握了互连故障的位置以及故障类型等相关信息，需要对互连故障进行修复型容错，而为优化现有互连故障修复型容错技术中存在的资源面积消耗过大以及无法直接处理故障的问题，本章提出了一种基于细粒度冗余的互连故障纠错容错方案。该方案首先根据故障所在信号线以及测试时输出端口的错误信息，判断出故障大致所在节点，针对故障所在节点对附近的原始逻辑资源进行基于 LUT 的细粒度冗余复制。再插入检错纠错电路，插入检错纠错电路之后需要更改原始电路的部分路由，依据被复制的原始 LUT 在互连故障节点之前或之后，对电路的布线进行对应的更改，保证在新电路中仍然存在先前诊断定位到的互连故障。然后在检错纠错电路中对被复制的原始 LUT 和冗余 LUT 进行两次异或处理，第一次异或实现对互连故障的监控，第二次异或实现对互连故障的纠错，最终完成对互连故障的容错。本文所提出的容错方案采用了细粒度冗余策略

以及简单的检错纠错电路，所造成的资源面积消耗较少，并且该方案直接纠正互连故障所引起的错误变化，实现对故障的修复，同时在互连故障未被激活时所提出的容错电路也不会影响正常电路的工作。

第六章 仿真验证与分析

第三章、第四章和第五章中介绍了所提出的互连故障测试、诊断以及容错方案，并详细说明了对应方案的主要内容和实现步骤，本章将使用国际标准时序电路集 ISCAS'89 中的基准电路对上述方案进行仿真验证实验，相应的实验都是在一个 8GB 内存的 Window 系统上进行的，并且为了与现有文献中的研究结果进行比对，本文选择了相关文献中相同型号的 Xilinx Virtex4（XC4VLX100）FPGA 进行实验，其中 Virtex4 的 LUT 规格为 4 输入。

6.1 测试方案验证与分析

6.1.1 测试仿真验证

下面对本文在第三章所提出的测试方案进行仿真验证，主要验证该测试方法中所生成的测试配置的有效性，在仿真过程中需要注入固定故障、无反馈桥接故障和反馈桥接故障等互连故障类型进行测试，而因为本文所做的研究主要集中在 FPGA 互连资源的可靠性分析上，对于故障注入的研究不在本文工作范围内，所以本文将仅通过控制输入激励或修改 LUT 逻辑函数来控制信号线的值，以模拟相应互连故障的现象，从而达到注入故障的效果。

本文选择了 ISCAS'89 中的 S27 时序电路进行测试仿真，S27 时序电路如图 6.1 所示。从图 6.1 中可看出 S27 电路为 4 输入 1 输出电路，包含 5 个 LUT，根据第三章中的方法对 S27 电路进行测试向量生成，相应的测试配置按图 6.1 中查找表的顺序进行了展示，如表 6.1 所示。

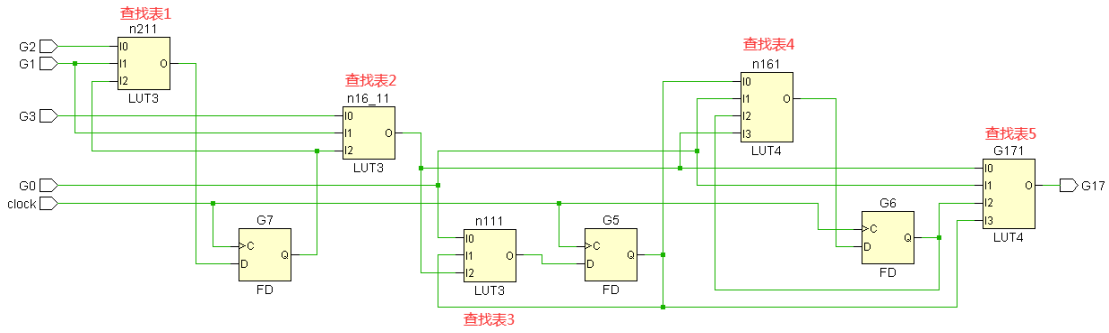


图 6. 1 S27 电路图

表 6. 1 针对 S27 电路生成的测试配置

LUT	LUT 输入	测试向量	生成的测试配置以及对应的 LUT 单项函数			
			配置一	配置二	配置三	配置四
查找表 1	I0	0110	011	100	110	001
	I1	1010	F= I0'·I1	F= I0'+ I1	F= I0'+ I1'	F= I0'·I1'
	O	1001				

表 6.1（续）

LUT	LUT 输入	测试向量	生成的测试配置以及对应的 LUT 单项函数			
			配置一	配置二	配置三	配置四
查找表 2	I0	0011	0111	0001	1100	1010
	I1	1010	F= I0'·I1·I2	F= I0'·I1'·I2'	F= I0'+I1'+I2	F= I0'+I1+I2'
	I2	1001				
	O	1100				
查找表 3	I0	0110	011	110	100	001
	I2	1100	F= I0'·I2	F= I0' +I2'	F= I0' +I2	F= I0'·I2'
	O	1001				
查找表 4	I0	1001	1011	0110	0101	1000
	I1	0110	F= I0·I1'·I3	F= I0+I1' +I3'	F= I0'·I1·I3'	F= I0'+I1+I3
	I3	1100				
	O	1010				
查找表 5	I0	1100	10110	11001	01100	00011
	I1	0110	F= I0'+I1+I2'+I3'	F= I0·I1·I2'·I3'	F= I0+I1'+I2'+I3	F= I0'·I1'·I2'·I3
	I2	1010				
	I3	1001				
	O	0101				
输入激励：G0,G1,G2,G3			0100	1010	1111	0001
输出响应：G17			0	1	0	1

根据相应的测试配置批量修改原始 S27 电路中 LUT 的逻辑函数以构建测试电路，因为本文采取仿真的方式进行验证，只需对 S27 综合后的网表文件进行修改即可，在得到对应的测试电路后，再施加对应的测试向量进行测试。

首先分析配置一，由表 6.1 可知在配置一中只需对电路的输入端{G0,G1,G2,G3}施加“0100”向量，在电路无故障情况下，经过正常的传输延时之后即可在输出端 G17 观察到稳定的正确响应值“0”，无故障情况下的配置一测试仿真结果如图 6.2 中所示。

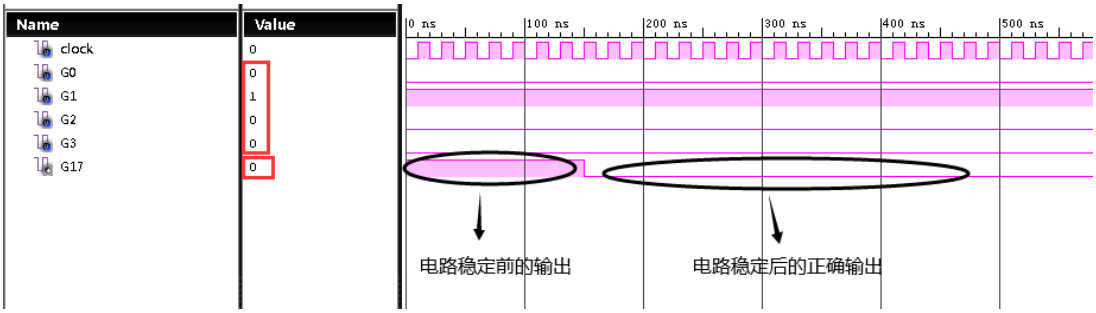


图 6. 2 无故障情况下 S27 的配置一仿真图

然后分析配置二，由表 6.1 可知在配置二中只需对电路的输入端{G0,G1,G2,G3}施加“1010”向量，在电路无故障情况下同样可在输出端 G17 观察到稳定的正确响应值“1”，无故障情况下的配置二测试仿真结果如图 6.3 中所示。

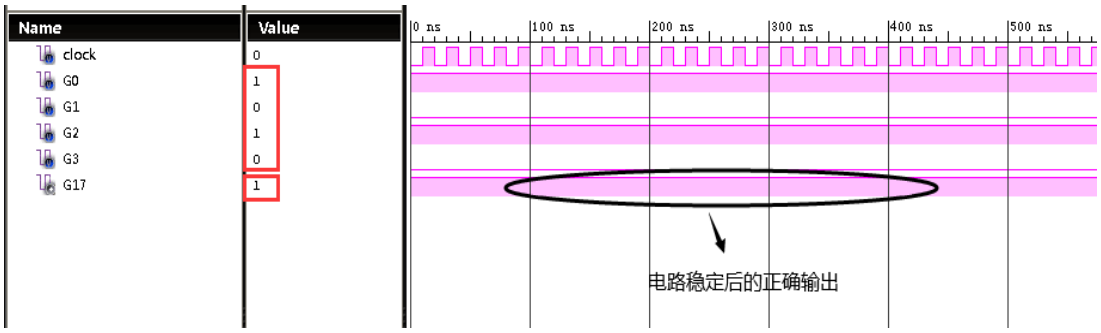


图 6. 3 无故障情况下 S27 的配置二仿真图

再分析配置三，由表6.1可知在配置三中只需对电路的输入端{G0,G1,G2,G3}施加“1111”向量，在电路无故障情况下，可在输出端观察到稳定的正确响应值“0”，无故障情况下的配置三测试仿真结果如图 6.4 中所示。

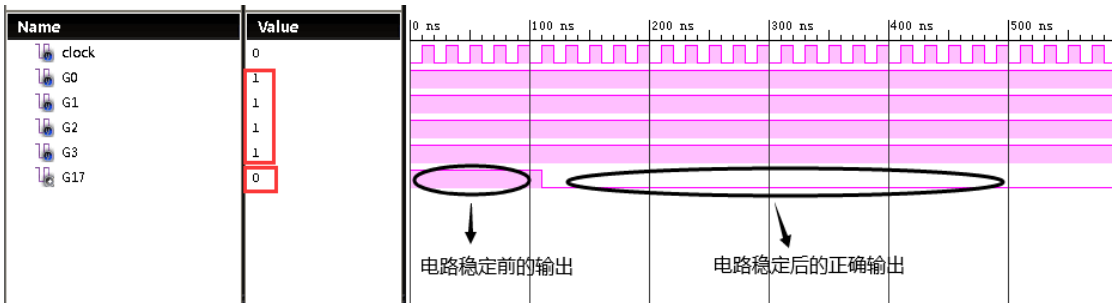


图 6. 4 无故障情况下 S27 的配置三仿真图

最后分析配置四，由表 6.1 可知在配置四中只需对电路的输入端{G0,G1,G2,G3}施加“0001”向量，在电路无故障情况下，可在输出端观察到稳定的正确响应值“1”，无故障情况下的配置四测试仿真结果如图 6.5 中所示。

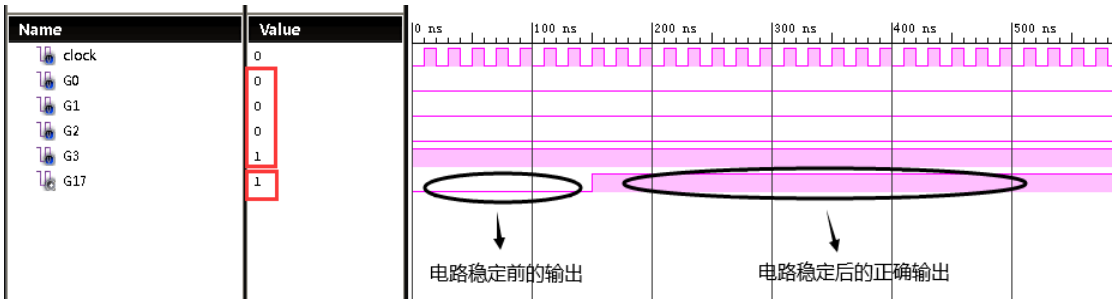


图 6. 5 无故障情况下 S27 的配置四仿真图

6.1.1.1 注入固定故障的测试验证

通过修改图 6.1 中查找表 4 的逻辑函数，使得其输出信号线在任何情况下都保持为“0”，模拟注入固定 0 故障。下面分析各测试配置对该固定 0 故障的测试结果。

在测试配置一中，查找表 4 的输出信号线被配置为“1”，激活了注入的固定 0 故障，因此可在输出端 G17 观察到稳定的输出“1”，该输出与图 6.2 无故障情况下配置一的正确输出“0”不相同，存在不匹配现象，可检测到电路中存在故障。图 6.6 为注入固定 0 故障后配置一的仿真图。

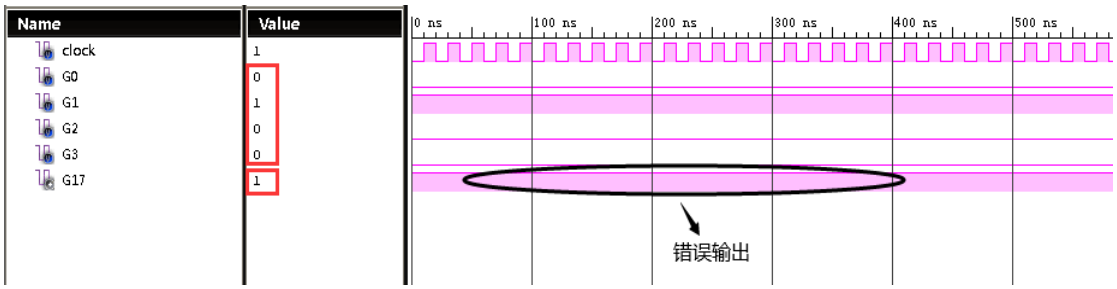


图 6. 6 注入固定故障后 S27 的配置一仿真图

在测试配置二中，查找表 4 的输出信号线也被配置为“0”，未激活注入的固定 0 故障，因此可在输出端 G17 观察到稳定的输出“1”，该输出与图 6.3 无故障情况下配置二的正确输出“1”相同，不存在不匹配现象。图 6.7 为注入固定 0 故障后配置二的仿真图。

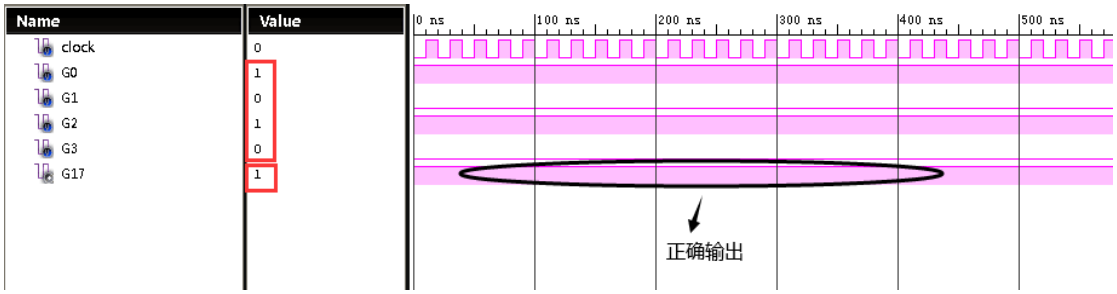


图 6. 7 注入固定故障后 S27 的配置二仿真图

在测试配置三中，查找表 4 的输出信号线被配置为“1”，激活了注入的固定 0 故障，因此可在输出端 G17 观察到稳定的输出“1”，该输出与图 6.4 无故障情况下配置三的正确输出“0”不相同，存在不匹配现象，可检测到电路中存在故障。图 6.8 为注入固定 0 故障后配置三的仿真图。

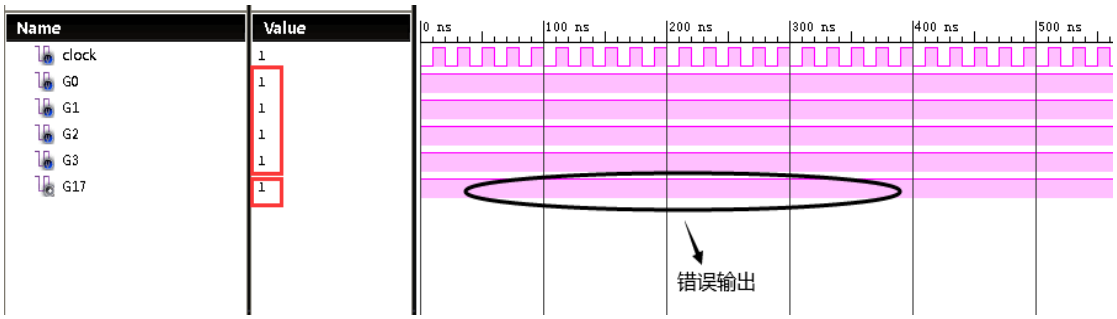


图 6. 8 注入固定故障后 S27 的配置三仿真图

在测试配置四中，查找表 4 的输出信号线同样被配置为“0”，未激活注入的固定 0 故障，因此可在输出端 G17 观察到稳定的输出“1”，该输出与图 6.5 无故障情况下配置四的正确输出“1”相同，不存在不匹配现象。图 6.9 为注入固定 0 故障后配置四的仿真图。

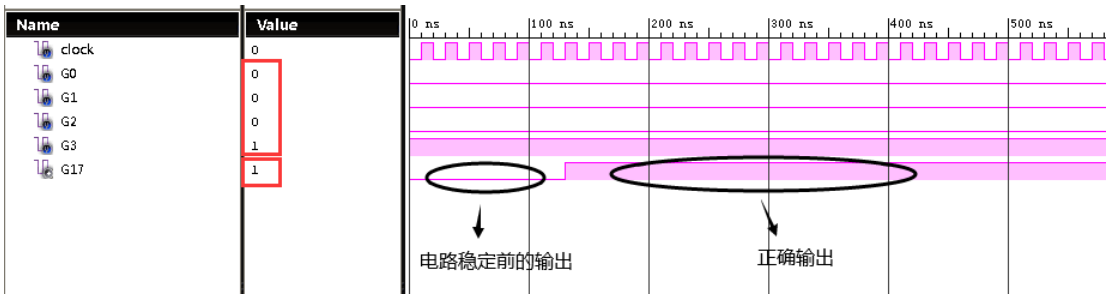


图 6. 9 注入固定故障后 S27 的配置四仿真图

上述对 S27 的测试仿真结果显示在失败的测试配置一和配置三中能够检测到 S27 电路存在故障，因此所提出的测试方案能够覆盖电路中的固定故障测试。

6.1.1.2 注入无反馈桥接故障的测试验证

通过修改输入激励，在图 6.1 中查找表 2 的输入信号线 I0 和输入信号线 I2 之间模拟注入 I2“主导与”I0 的桥接故障。由第二章中表 2.1 展示的互连故障真值表可知，只有在查找表 2 的{I0,I2}被赋为“10”时，才会激活相应的 I2“主导与”I0 故障，导致{I0,I2}发生由“10”变为“00”的错误改变，其他任何情况都不会激活该故障。因此本文逆向操作，若存在测试配置激活了发生在查找表 2 上的 I2“主导与”I0 桥接故障，即在该配置下{I0,I2}为“10”，通过将该配置下输入端 G3 的激励“1”修改为“0”，使得查找表 2 的{I0,I2}变为“00”，以模拟 I2“主导与”I0 桥接的故障现象，达到注入该桥接故障的效果。下面分析各测试配置对该无反馈桥接故障的测试结果。

在测试配置一中，查找表 2 的信号线 I0 和 I2 被配置为“01”，未激活 I2“主导与”I0 桥接故障，因此该故障不会使得电路发生错误改变，即在配置一中该故障不存在任何故障现象，所以无需修改输入激励。此时输入激励{G0,G1,G2,G3}为“0100”，最终可在输出端 G17 观察到稳定的输出“0”，该输出与图 6.2 无故障情况下配置一的正确输出“0”相同，不存在不匹配现象。图 6.10 为注入无反馈桥接故障后配置一的仿真图。

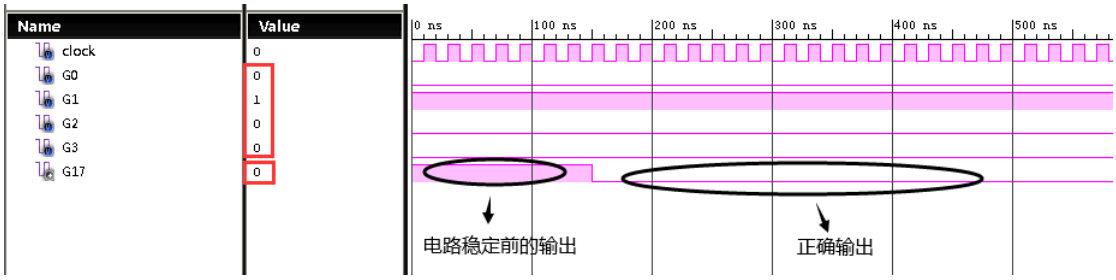


图 6. 10 注入无反馈桥接故障后 S27 的配置一仿真图

在测试配置二中，查找表 2 的信号线 I0 和 I2 被配置为“00”，未激活 I2“主导与”I0 桥接故障，同样该故障不会使得电路发生错误改变，所以无需修改输入激励。此时输入激励{G0,G1,G2,G3}为“1010”，最终可在输出端 G17 观察到稳定的输出“1”，该输出与图 6.3 无故障情况下配置二的正确输出“1”相同，不存在不匹配现象。图 6.11 为注入无反馈桥接故障后配置二的仿真图。

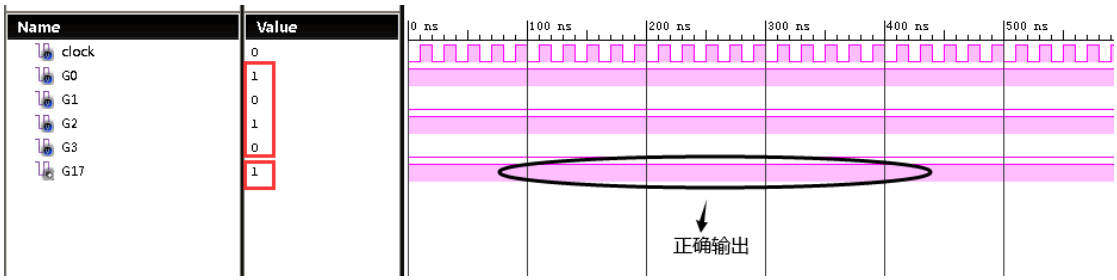


图 6. 11 注入无反馈桥接故障后 S27 的配置二仿真图

在测试配置三中，查找表 2 的信号线 I0 和 I2 被配置为“10”，激活了 I2“主导与”I0 桥接故障，因此通过修改输入端 G3 的激励在查找表 2 上注入 I2“主导与”I0 桥接故障。注入故障后的输入激励{G0,G1,G2,G3}为“1110”，最终可在输出端 G17 观察到稳定的输出“1”，该输出与图 6.4 无故障情况下配置三的正确输出“0”不相同，存在不匹配现象，可以检测到电路中存在故障。图 6.12 为注入无反馈桥接故障后配置三的仿真图。

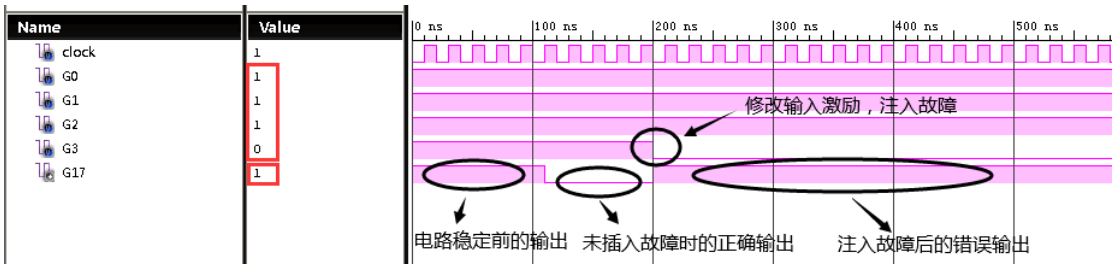


图 6. 12 注入无反馈桥接故障后 S27 的配置三仿真图

在测试配置四中，查找表 2 的信号线 I0 和 I2 被配置为“11”，I2“主导与”I0 桥接故障未被激活，因此该故障在配置四中不会引起任何错误改变，所以也无需修改输入激励。此时输入激励{G0,G1,G2,G3}为“0001”，最终可在输出端 G17 观察到稳定的输出“1”，该输出与图 6.5 无故障情况下配置四的正确输出“1”相同，不存在不匹配现象。图 6.13 为注入无反馈桥接故障后配置四的仿真图。

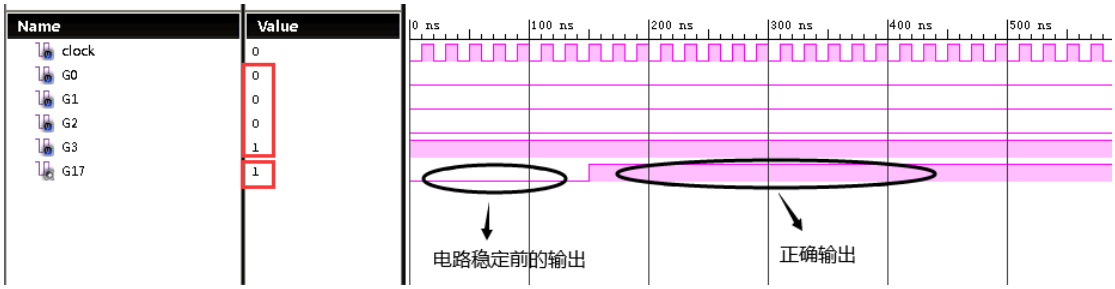


图 6. 13 注入无反馈桥接故障后 S27 的配置四仿真图

上述对 S27 的测试仿真结果显示在失败的测试配置三中能够检测到 S27 电路存在故障，因此所提出的测试方案能够有效覆盖电路中的无反馈桥接故障测试。

6.1.1.3 注入反馈桥接故障的测试验证

同样通过修改输入激励的方式，在图 6.1 中查找表 1 的输入信号线 I0 和输出信号线 O 之

间模拟注入 O“主导或”I0 的反馈输入型桥接故障。同样由第二章中表 2.1 展示的互连故障真值表可知，只有在查找表 1 的{I0,O}被赋为“01”时，才会激活相应的 O“主导或”I0 故障，其他任何情况都不会激活该故障。而从第三章图 3.3 中对反馈桥接故障的现象分析中可知，该故障被激活后会导致查找表 1 的信号线 I0 和信号线 O 各发生一次翻转，即 I0 由“0”变为“1”，O 由“1”变为“0”。因此本文逆向操作，若存在测试配置激活了发生在查找表 1 上的 O“主导或”I0 反馈输入型桥接故障，即在该配置下{I0,O}为“01”时，通过修改该配置下输入端 G2 的激励，使得查找表 1 的信号线 I0 在保持逻辑值“0”一段时间后变为逻辑值“1”，以模拟 O“主导或”I0 反馈输入型桥接的故障现象，达到注入该桥接故障的效果。下面分析在各测试配置对该反馈桥接故障的测试结果。

在测试配置一中，查找表 1 的信号线 I0 和 O 被配置为“01”，激活了 O“主导或”I0 反馈输入型桥接故障，因此通过上述方法在查找表 1 上模拟注入 O“主导或”I0 反馈输入型桥接故障。注入故障后的输入激励{G0,G1,G2,G3}在保持“0100”一段时间后变为“0110”，最终可在输出端 G17 观察到稳定的输出“1”，该输出与图 6.2 无故障情况下配置一的正确输出“0”不相同，存在不匹配现象，可以检测到电路中存在故障。图 6.14 为注入反馈桥接故障后配置一的仿真图。

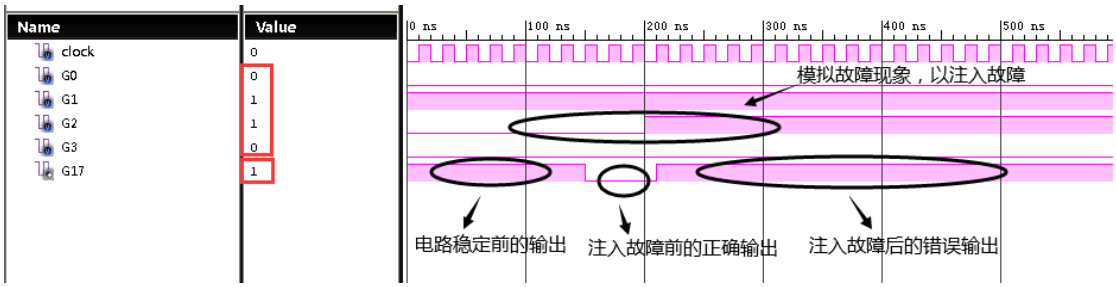


图 6. 14 注入反馈桥接故障后 S27 的配置一仿真图

在测试配置二中，查找表 1 的信号线 I0 和 O 被配置为“10”，未激活 O“主导或”I0 反馈输入型桥接故障，因此该故障在配置一中不会引起任何错误改变，所以无需修改输入激励对故障进行模拟。此时输入激励{G0,G1,G2,G3}为“1010”，最终可在输出端 G17 观察到稳定的输出“1”，该输出与图 6.3 无故障情况下配置二的正确输出“1”相同，不存在不匹配现象。图 6.15 为注入反馈桥接故障后配置二的仿真图。

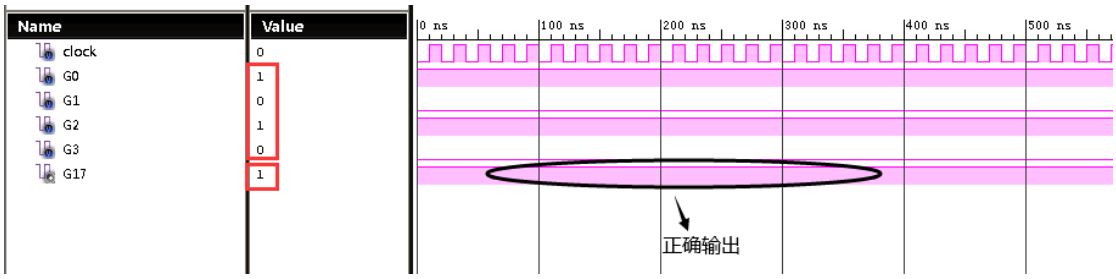


图 6. 15 注入反馈桥接故障后 S27 的配置二仿真图

在测试配置三中，查找表 1 的信号线 I0 和 O 被配置为“10”，同样未激活 O“主导或”

I0 反馈输入型桥接故障，因此该故障在配置一中不会引起任何错误改变，所以无需修改输入激励对故障进行模拟。此时输入激励{G0,G1,G2,G3}为“1111”，最终可在输出端 G17 观察到稳定的输出“0”，该输出与图 6.4 无故障情况下配置三的正确输出“0”相同，不存在不匹配现象。图 6.16 为注入反馈桥接故障后配置三的仿真图。

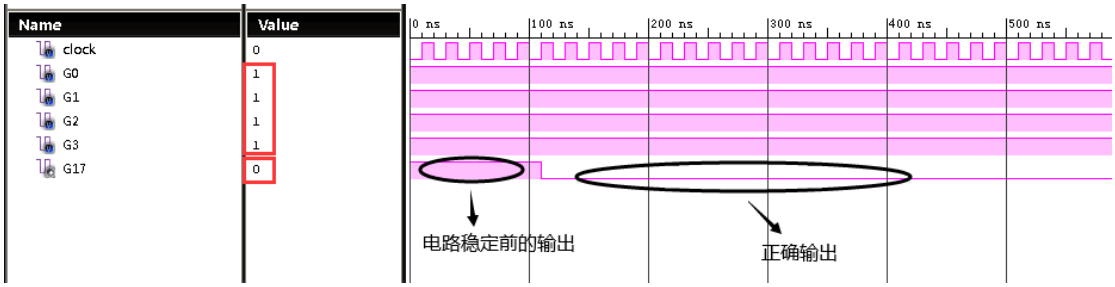


图 6. 16 注入反馈桥接故障后 S27 的配置三仿真图

在测试配置四中，查找表 1 的信号线 I0 和 O 被配置为“01”，激活了 O“主导或”I0 反馈输入型桥接故障，同样通过上述方法在查找表 1 上模拟注入 O“主导或”I0 反馈输入型桥接故障。注入故障后的输入激励{G0,G1,G2,G3}在保持“0001”一段时间后变为“0011”，最终可在输出端 G17 观察到稳定的输出 “0”，该输出与图 6.5 无故障情况下正确的输出“1”不相同，存在不匹配现象，可以检测到电路中存在故障。图 6.17 为注入反馈桥接故障后配置四的仿真图。

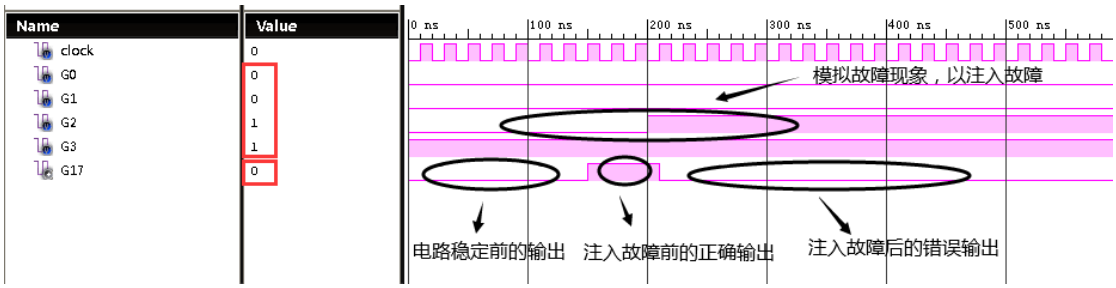


图 6. 17 注入反馈桥接故障后 S27 的配置四仿真图

上述对 S27 的测试仿真结果显示在失败的测试配置一和配置四中能够检测到 S27 电路存在故障，因此所提出的测试方案能够覆盖电路中的反馈桥接故障测试。

6.1.2 测试配置数分析

在上一节中通过对 S27 电路中模拟注入不同类型的互连故障，并使用所提出的测试方案进行仿真验证，结果表明测试方案中所生成的测试配置能够成功覆盖到 S27 电路中存在的固定故障、无反馈桥接故障以及反馈桥接故障等互连故障的测试，证明了本文在第三章中所提出的测试方法的有效性。下面分析该测试方法针对不同的 ISCAS'89 基准电路所需的测试配置数量。

针对第三章互连测试方案中所优化的故障模型，选择了与文献[29]中相同的 20 个基准电路进行测试分析，表 6.2 中展示了这些基准电路映射在 Xilinx Virtex-4（XC4VLX100）FPGA 上的资源使用情况以及所存在的互连故障数量。

表 6. 2 ISCAS'89 基准电路在 Virtex-4 上的资源使用情况

基准电路	LUT 总数	Net 总数	总故障个数
S298	28	48	1272
S344	37	73	2847
S349	37	73	2847
S382	44	75	3000
S400	44	76	3078
S420	37	76	3078
S444	45	78	3237
S510	78	121	7623
S526	43	76	3078
S641	67	120	7500
S713	71	126	8253
S820	107	145	10875
S1196	200	257	33667
S1238	209	277	39057
S1423	195	316	50718
S1488	236	296	44548
S1494	235	292	43362
S5378	383	664	222108
S9234	291	495	123750
S15850	928	1502	1131757

再根据第三章中所介绍的测试步骤，对这些基准电路进行测试配置生成，在表 6.3 中展示了对应基准电路所需的测试配置数。

表 6. 3 ISCAS'89 基准电路的测试配置生成结果

基准电路	配置生成时间（s）	配置次数	配置次数[29]
S298	0.172	5	5
S344	0.235	5	5
S349	0.237	5	5
S382	0.283	5	5
S400	0.331	5	5
S420	0.249	5	5
S444	0.284	5	6
S510	0.481	5	6
S526	0.335	5	6
S641	0.451	5	6
S713	0.470	5	6
S820	0.793	5	7
S1196	1.346	5	7
S1238	1.615	5	9
S1423	1.384	5	9
S1488	1.858	5	9

表 6.3（续）

基准电路	配置生成时间（s）	配置次数	配置次数[29]
S1494	1.891	5	11
S5378	3.575	5	11
S9234	2.128	5	12
S15850	9.942	5	12

在第三章中对所提出方案的测试配置数进行了理论分析，最多需要 $\log_2(n+2)+2$ 次配置就能实现电路故障列表的 100%覆盖（n 为 LUT 的信号线数），因为本文使用的是 Virtex-4 系列的 FPGA，其 LUT 规格为 4 输入，代入公式计算可得出实现电路故障列表的 100%覆盖所需的最大配置数为 5 次，与表 6.3 中第 3 列展示的实际测试配置生成结果相吻合，表明了本文在第三章分析的最大配置数的正确性。

本文所提出的测试方案主要贡献在于所生成的测试配置优化了桥接故障的测试，并解决了反馈桥接故障的覆盖难题，因为许多测试文献都只针对某一类型的互连故障进行了研究，所以为了更加公平的进行数据结果比对，只对同样考虑到反馈桥接故障覆盖的文献[29]进行了分析，并在表 6.3 中第 4 列展示了文献[29]中的测试配置结果。文献[29]中的测试方案针对反馈桥接故障进行了测试配置生成，其中针对反馈桥接故障的约束条件借鉴了文献[28]中的对反馈桥接故障的定义，过于复杂，但该文所使用的测试生成方法与本文相似，因此将本文的测试生成数据与文献[29]中的数据进行比较，最能表明本文优化后的桥接故障和反馈桥接故障模型的优越性。从表 6.3 中可看出本文与文献[29]所生成的测试配置次数在前几个基准电路上持平，在后续较大规模的基准电路上，本文实现了测试配置数量的减少，而在测试 FPGA 过程中所花费的时间主要取决于测试配置的数量，因此所提出的测试方案相比于文献[29]是有优势的，即本文在第三章中所做的研究工作既解决了现有应用相关测试文献中的反馈桥接故障覆盖问题，又实现了测试配置数的减少，对提升 FPGA 应用相关互连资源测试的效率具有重要的意义。

6.2 诊断方案验证与分析

6.2.1 诊断仿真验证

在 6.1.1 节对 S27 基准电路进行的测试仿真实验中确定 S27 电路中存在故障，因此本节将在 6.1.1 节测试实验的基础上，使用所提出的诊断方法对这些故障进行精准诊断定位，以验证所提出的诊断方案的有效性。为了更好的进行诊断定位，对 S27 电路中的信号线进行了标注，如图 6.18 所示。

首先分析电路的互连情况，因为 S27 电路为 4 输入 1 输出电路，只有一个输出端，所以 S27 电路只存在一条单输出路径，从而可判断在测试实验中检测到的故障存在于输出端口 G17 所在的单输出路径上。由于 S27 电路的规模较小，从图 6.18 中可看出 S27 电路只有 9 条信号线，即相应的单输出路径较为简单，因此可直接对该路径上的信号线进行细粒度诊

断，下面分别对测试仿真实验中所模拟注入的固定故障和桥接故障进行诊断验证。

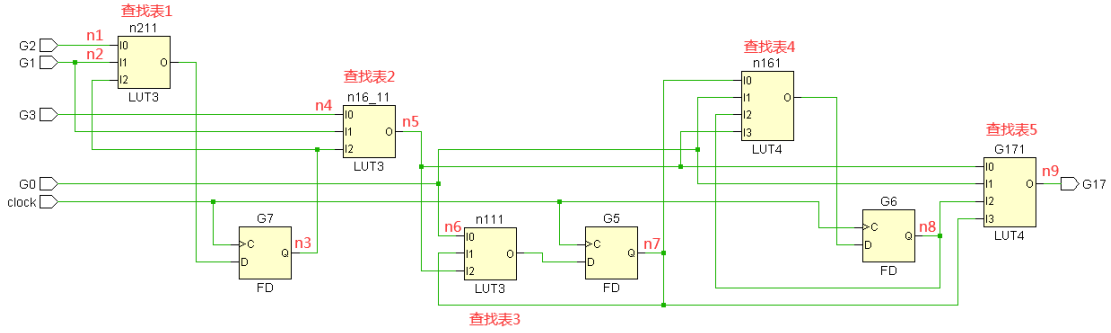


图 6. 18 标注了信号线的 S27 电路图

6.2.1.1 注入固定故障的诊断验证

根据 S27 电路的测试配置以及 6.1.1.1 节中注入固定故障的测试仿真结果，对所注入的故障进行精准诊断定位，相应的测试配置及测试结果如表 6.4 所示。

表 6. 4 S27 电路的测试配置及注入固定故障后的测试结果

配置号	故障所在粗粒度范围									测试结果
	n1	n2	n3	n4	n5	n6	n7	n8	n9	
一	0	1	1	0	1	0	1	1	0	失败
二	1	0	0	0	1	1	0	0	1	成功
三	1	1	0	1	0	1	0	1	0	失败
四	0	0	1	1	0	0	1	0	1	成功

使用复用配置法对该故障进行诊断，因为不清楚该故障的类型，这里先假设注入的故障为最常见的固定故障。在表 6.4 中可看到第 2 个配置和第 4 个配置是成功的，在成功的配置中信号线 n1、n3、n4、n5、n6、n7 都分别被赋为了逻辑值“0”和“1”，因此可排除信号线 n1、n3、n4、n5、n6、n7 的固定故障嫌疑；第 1 个配置和第 3 个配置是失败的，但信号线 n2、n8、n9 在不同的失败配置中都只被赋为了相同逻辑值，因此可怀疑信号线 n2 存在固定 0 故障，信号线 n8 存在固定 0 故障，信号线 n9 存在固定 1 故障。但由于电路中只存在一个互连故障，所以需要使用额外的配置排除多余的故障嫌疑，通过将配置二中 n2 的逻辑值“0”修改为“1”，以验证 n2 的固定 0 故障，即使用“110011001”配置进行诊断，该配置下的输入激励为“1110”，理想输出响应为“1”，仿真结果如图 6.19 所示。

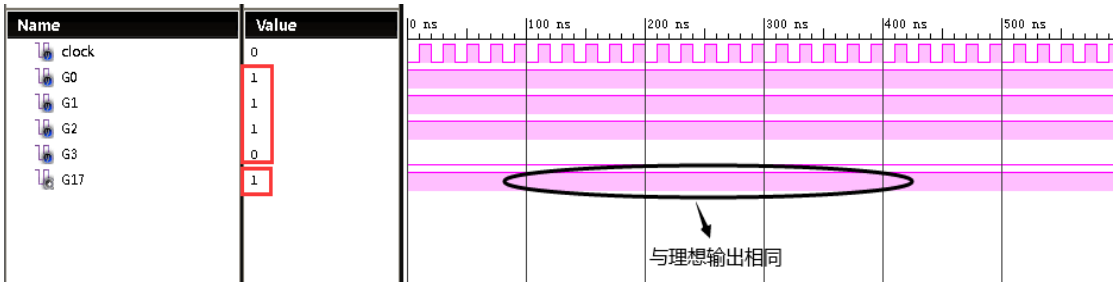


图 6. 19 注入固定故障后 S27 的诊断配置一仿真图

从图 6.19 中的仿真波形可看出配置“110011001”的测试结果正确，可排除信号线 n2 的固定 0 故障的嫌疑。再将配置二中 n8 的逻辑值“0”修改为“1”，验证 n8 的固定 0 故障，即使用“100011011”配置进行诊断，该配置下的输入激励为“1010”，理想输出响应为“1”，仿真结果如图 6.20 所示。

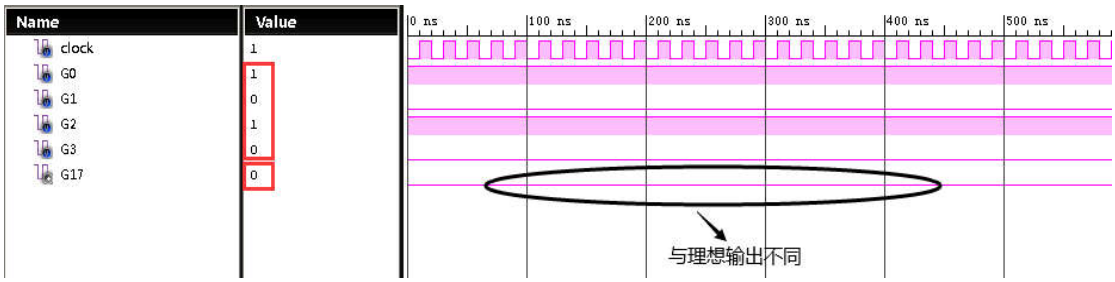


图 6. 20 注入固定故障后 S27 的诊断配置二仿真图

从图 6.20 中的仿真波形可看出配置“100011011”的测试结果失败，因此可确定固定故障为信号线 n8 的固定 0 故障。而上述诊断过程都是假设故障为固定故障的前提下进行的，为了确认故障是否是固定故障，需要最后使用一个配置确定故障的类型。先前的诊断已经判断故障可能是信号线 n8 的固定 0 故障，因此为了激活该固定故障，且避免激活其他的桥接故障，使用“111111111”配置进行诊断，该配置下的输入激励为“1111”，理想输出响应为“1”，仿真结果如图 6.21 所示。

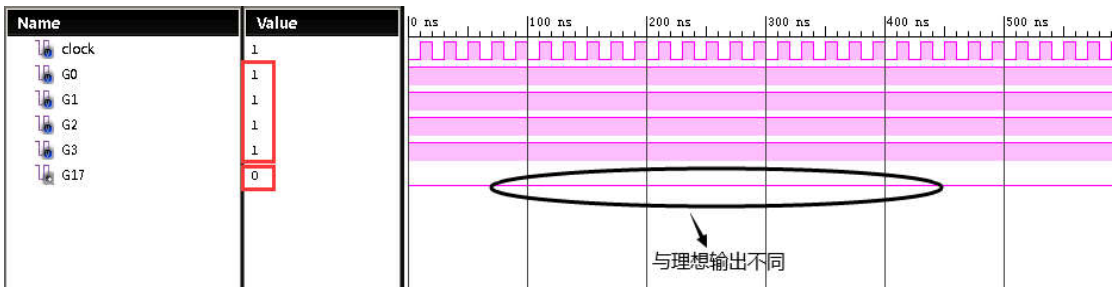


图 6. 21 注入固定故障后 S27 的诊断配置三仿真图

从图 6.21 中的仿真波形可看出配置“111111111”的测试结果失败，因此可确定最终的故障类型就是假设的固定故障，且故障为信号线 n8 上的固定 0 故障，即查找表 4 的输出信号线上的固定 0 故障，与 6.1.1.1 节中所注入的故障相同，证明了所提出的诊断方法的有效性。上述诊断步骤结果如表 6.5 所示。

表 6. 5 对 S27 电路注入固定故障后的诊断配置及结果

配置号	故障所在粗粒度范围									测试结果
	n1	n2	n3	n4	n5	n6	n7	n8	n9	
一	0	1	1	0	1	0	1	1	0	失败
二	1	0	0	0	1	1	0	0	1	成功
三	1	1	0	1	0	1	0	1	0	失败
四	0	0	1	1	0	0	1	0	1	成功
	n1/0	**n2/0**	n3/0	n4/0	n5/0	n6/0	n7/0	**n8/0**	n9/0	
	n1/1	n2/1	n3/1	n4/1	n5/1	n6/1	n7/1	n8/1	**n9/1**	
额外的配置，排除多余的固定故障嫌疑										
五	1	1	0	0	1	1	0	0	1	成功
六	1	0	0	0	1	1	0	1	1	失败
用来最终确定故障类型的配置										
七	1	1	1	1	1	1	1	1	1	失败

6.2.1.2 注入桥接故障的诊断验证

在 6.1.1.2 节和 6.1.1.3 节中注入的故障都是桥接故障，而对其进行诊断的原理是相同的，因此本节将以 6.1.1.2 节中注入的无反馈桥接故障为例，根据 S27 电路的测试配置以及注入无反馈桥接故障的测试仿真结果，对所注入的故障进行精准诊断定位，相应的测试配置及测试结果如表 6.6 所示。

表 6. 6 S27 电路的测试配置及注入无反馈桥接故障后的测试结果

配置号	故障所在粗粒度范围									测试结果
	n1	n2	n3	n4	n5	n6	n7	n8	n9	
一	0	1	1	0	1	0	1	1	0	成功
二	1	0	0	0	1	1	0	0	1	成功
三	1	1	0	1	0	1	0	1	0	失败
四	0	0	1	1	0	0	1	0	1	成功

同样使用复用配置法对该故障进行诊断，因为不清楚该故障的类型，这里也先假设注入的故障为最常见的固定故障。在表 6.6 中可看到第 1、2 和 4 个配置是成功的，在不同的成功配置中所有的信号线都分别被赋为了逻辑值“0”和“1”，因此可排除所有信号线的固定故障嫌疑，即电路中存在的故障不是固定故障，而是桥接故障。

故障不是假设的固定故障，而是桥接故障，因此需要重新使用复用配置法对桥接故障进行诊断，而本文约束桥接故障只发生在同一 LUT 的信号线上，所以只考虑发生在查找表 1 的信号线{n1,n2,n3}、查找表 2 的信号线{n2,n3,n4,n5}、查找表 3 的信号线{n5,n6,n7}、查找

表 4 的信号线{n5,n6,n7,n8}、以及查找表 5 的信号线{ n5,n6,n7,n8,n9}之间的桥接故障。复用表 6.6 中的测试配置对桥接故障进行诊断，最终的诊断结果如表 6.7 所示，表中的虚线框与不同查找表上的信号线相对应。

表 6. 7 S27 电路中的桥接故障诊断

信号线		1								
		n1	n2	n3	n4	n5	n6	n7	n8	n9
0	n1	\	√	√						
	n2	√	\	√	√	√				
	n3	√	?	\	?	√				
	n4		√	√	\	√				
	n5		?	√	√	\	?	√	?	√
	n6					√	\	√	√	√
	n7					√	√	\	?	√
	n8					√	√	√	\	√
	n9					√	?	√	√	\

从表 6.7 中可看出，在对测试配置分析完备后还存在 7 个故障嫌疑，由于电路中只存在一个互连故障，所以需要额外的配置排除多余的故障嫌疑。此时故障粗粒度范围内存在 5 个 LUT，需要使用额外配置先确定桥接故障存在于哪个 LUT 上，先使用“000001010”配置，激活查找表 3、4、5 上的桥接故障嫌疑，并避免激活查找表 1、2 上的桥接故障嫌疑，该配置下的输入激励为“1000”，理想输出响应为“0”，仿真结果如图 6.22 所示。

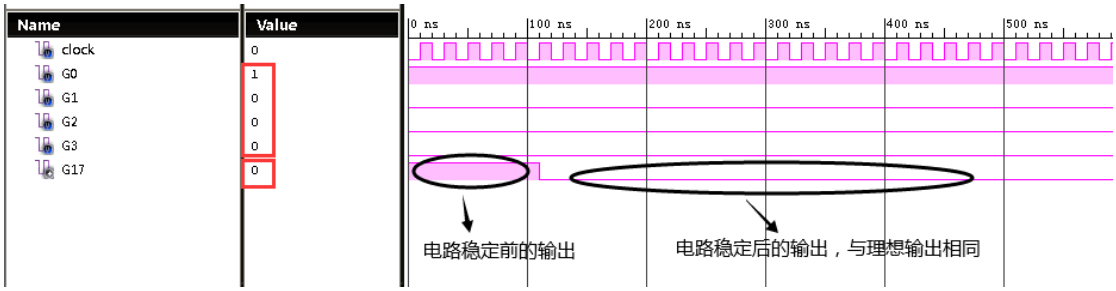


图 6. 22 注入桥接故障后 S27 的诊断配置一仿真图

从图 6.22 中的仿真波形可看出配置“000001010”的测试结果正确，因此可排除查找表 3、4、5 上的桥接故障嫌疑，这时只剩下查找表 1 和查找表 2 上的桥接故障嫌疑。再修改表 6.6 中的失败配置三，将信号线 n4 的值由“1”改为“0”，使信号线 n3 和信号线 n4 之间的“01”组合不被激活，即使用“110001010”配置验证信号线 n2 和 n3 之间以及信号线 n2 和 n5 之间的故障嫌疑，从而判断故障是否在查找表 1 上，该配置下的输入激励为“1110”，理想输出响应为“0”，仿真结果如图 6.23 所示。

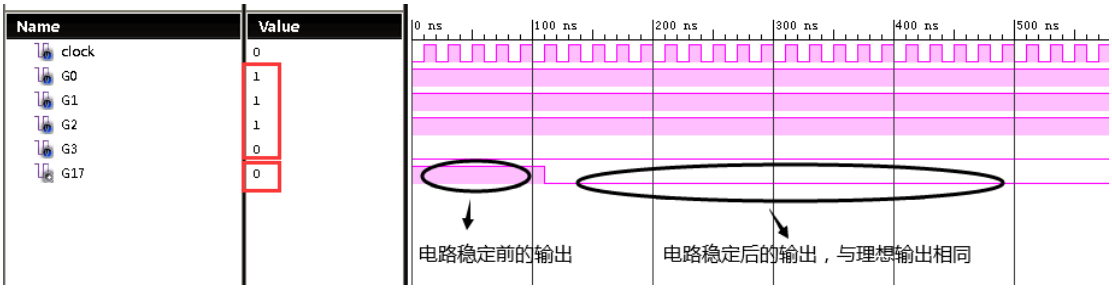


图 6. 23 注入桥接故障后 S27 的诊断配置二仿真图

从图 6.23 中的仿真波形可看出配置“110001010”的测试结果正确，因此可确定桥接故障不在查找表 1 上，而在查找表 2 上，确定完故障存在查找表 2 上之后就只剩下一个故障嫌疑，即桥接故障为信号线 n3 与信号线 n4 之间的“01”组合，而在上述诊断过程中已经判断出真正的故障为桥接故障，所以无需再使用额外的配置确定最终的故障类型。上述诊断过程最终确定故障为信号线 n3 与信号线 n4 之间的桥接故障，且为“01”组合，与 6.1.1.2 节中在查找表 2 上注入的 I2“主导与”I0 桥接故障相同，证明了所提出的诊断方法的有效性。上述诊断配置及结果如表 6.8 所示。

表 6. 8 对 S27 电路注入无反馈桥接故障后的诊断配置结果

配置号	故障所在粗粒度范围									测试结果
	n1	n2	n3	n4	n5	n6	n7	n8	n9	
一	0	1	1	0	1	0	1	1	0	成功
二	1	0	0	0	1	1	0	0	1	成功
三	1	1	0	1	0	1	0	1	0	失败
四	0	0	1	1	0	0	1	0	1	成功
额外的配置，排除多余的桥接故障嫌疑										
五	0	0	0	0	0	1	0	1	0	成功
（配置五诊断故障是否在查找表 3、4、5 上）										
六	1	1	0	0	0	1	0	1	0	成功
（配置六诊断故障是否在查找表 1 上）										

6.2.2 诊断配置数分析

在上一节中使用了所提出的诊断方案对 6.1.1 节所发现的故障进行诊断定位仿真，结果所诊断定位到的故障位置以及故障类型都与模拟注入的故障相同，证明了本文在第四章中所提出的诊断方法的有效性。下面分析该诊断方法针对不同的 ISCAS'89 基准电路所需的诊断配置数量。

针对第四章中的互连诊断方案，选择了与测试阶段相同的 20 个基准电路进行诊断分析，

表 6.9 中展示了这些基准电路映射在 Xilinx Virtex-4（XC4VLX100）FPGA 上的相关参数，其中第 2 和第 3 列是基准电路的资源使用情况，第 4 列和第 5 列展示了基准电路的路径分析情况，本文还结合相应基准电路的互连情况，对故障可能存在的最大粗粒度范围进行了分析，并在第 6 列展示了故障最大粗粒度范围内的 LUT 数量。

表 6. 9 ISCAS'89 基准电路映射在 Virtex-4 上的相关参数信息

基准电路	LUT 总数	Net 总数	单输出路径数量	多输出路径数量	最大故障范围内 LUT 个数
S298	28	48	0	1	12
S344	37	73	0	1	27
S349	37	73	0	1	27
S382	44	75	0	1	23
S400	44	76	0	1	24
S420	37	76	1	0	37
S444	45	78	0	1	25
S510	78	121	0	1	35
S526	43	76	0	1	20
S641	67	120	2	1	34
S713	71	126	2	1	28
S820	107	145	0	1	74
S1196	200	257	0	1	128
S1238	209	277	0	1	125
S1423	195	316	0	1	141
S1488	236	296	0	1	147
S1494	235	292	0	1	151
S5378	383	664	0	1	174
S9234	291	495	5	1	197
S15850	928	1502	28	1	692

在分析完故障的粗粒度范围后，根据所提出的方法进行细粒度诊断，最后结合第四章中总结的最大诊断配置数公式，对诊断 ISCAS'89 基准电路中单个任意故障所需的最大配置数情况进行了分析，对应基准电路所需的诊断配置数如表 6.10 所示，表 6.10 中还列出了已有文献中的最大诊断配置数进行比较。

表 6. 10 诊断 ISCAS'89 基准电路所需的最大配置数

基准电路	诊断所需的最大配置数		
	本文方案	文献[40]	文献[34]
S298	12	13	19
S344	13	14	22
S349	13	14	22
S382	13	14	22

表 6.10（续）

基准电路	诊断所需的最大配置数		
	本文方案	文献[40]	文献[34]
S400	13	14	22
S420	14	14	22
S444	13	14	22
S510	14	15	22
S526	13	14	22
S641	14	15	22
S713	13	15	22
S820	15	16	25
S1196	15	18	25
S1238	15	18	28
S1423	16	18	28
S1488	16	18	28
S1494	16	18	28
S5378	16	20	31
S9234	16	19	28
S15850	18	23	34

首先从表 6.10 中的数据可看出本文所需的最大诊断配置数与文献[34]、文献[40]所需的最大配置数相比是有优势的。然后以具体的方法而言，文献[34]中诊断固定故障使用非适应性方法，诊断桥接故障使用适应性方法，复杂度较高；文献[40]中的方法复杂度低，适用性强，但该方法需对电路中所有的信号线进行分析，所需诊断的信号线数量过多；而本文在文献[40]的基础上结合电路的互连情况以及测试配置结果判断出故障的粗粒度范围，减少了所需诊断的信号线数量，优化了相应方法的短板。最后就覆盖的故障类型而论，文献[34]和文献[40]中并未考虑到桥接故障的“主导与”、“主导或”桥接类型，而本文沿用了第三章所提出的测试方法中的故障模型，简化了故障诊断的复杂度，因此在诊断过程中能够更加精准，更加高效地诊断出互连故障的相应信息。

6.3 容错方案验证与分析

6.3.1 容错仿真验证

在 6.2.1 节对 S27 基准电路进行的故障诊断实验中定位到了 S27 电路中存在的故障，因此本节将在 6.2.1 节诊断实验的基础上，使用所提出的纠错容错方法对故障进行修复，以验证所提出的容错方案的有效性。

无论诊断到的故障是固定故障还是桥接故障，所采取的容错措施是相同的，因此下面将以 6.2.1.2 节中所诊断到的桥接故障为例，对 S27 电路进行容错处理。

因为 6.2.1.2 节中所诊断到的桥接发生在信号线 n3 和信号线 n4 之间，所以下面将对查找表 2 进行细粒度双模冗余，并在双模冗余之后插入两级异或电路实现对故障的检错纠错。对 S27 进行容错处理之后的电路图如 6.24 所示。

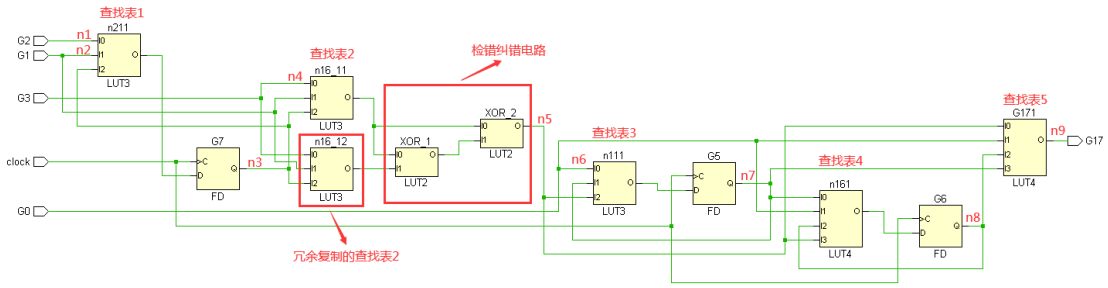


图 6. 24 容错之后的 S27 电路图

S27 电路的原始布局布线图如图 6.25 所示，在图 6.25 的 Slice_X63Y211 中映射了查找表 1 与后续连接的触发器逻辑，Slice_X63Y210 中映射了查找表 3、查找表 4 以及各自连接的触发器，Slice_X62Y210 中映射了查找表 2 和查找表 5，在图 6.25 中还标注了故障信号线 n3、故障信号线 n4 以及信号线 n5 的原始路由情况。

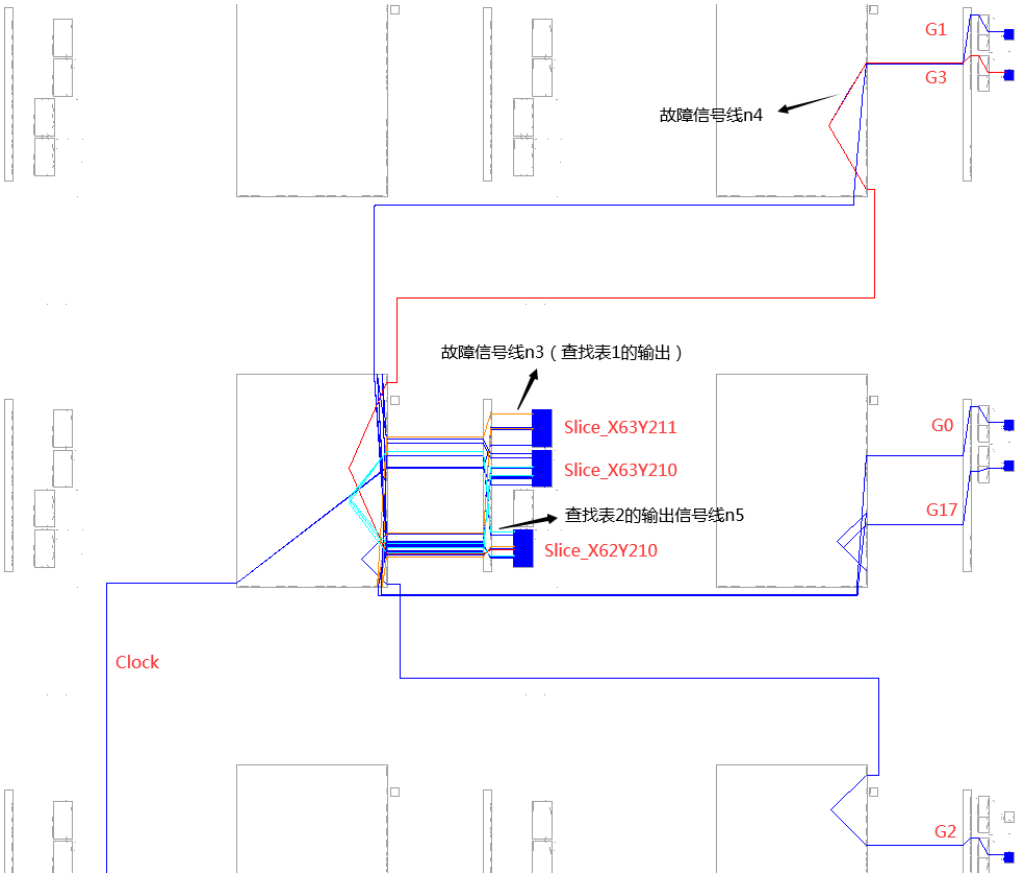


图 6. 25 S27 电路的原始布局布线图

按照第五章所述的步骤对 S27 电路进行容错，在原始路由的基础上，手动对查找表 2

进行复制，并插入检错纠错电路，再手动对冗余复制的 LUT 以及插入的检错纠错电路进行增量路由，因为被复制的原始查找表 2 在故障节点之后，所以相应的增量路由原理与第五章中图 5.5 的容错示意图相同，实现容错之后的 S27 电路的布局布线图如图 6.26 所示。在图 6.26 中，因为是在原始路由的基础上进行容错，所以原始资源的映射位置没有发生改变，其中增加的 Slice_X61Y211 中为复制查找表 2 的冗余资源，Slice_X61Y210 中为插入的检错纠错电路，并且故障信号线 n3 和信号线 n4 到冗余资源之间的路由避开了原始路由，保证了冗余资源不受故障的影响，而后将经过检错纠错后的正确输出 n5 连接到了后续电路，以恢复电路的正常工作。

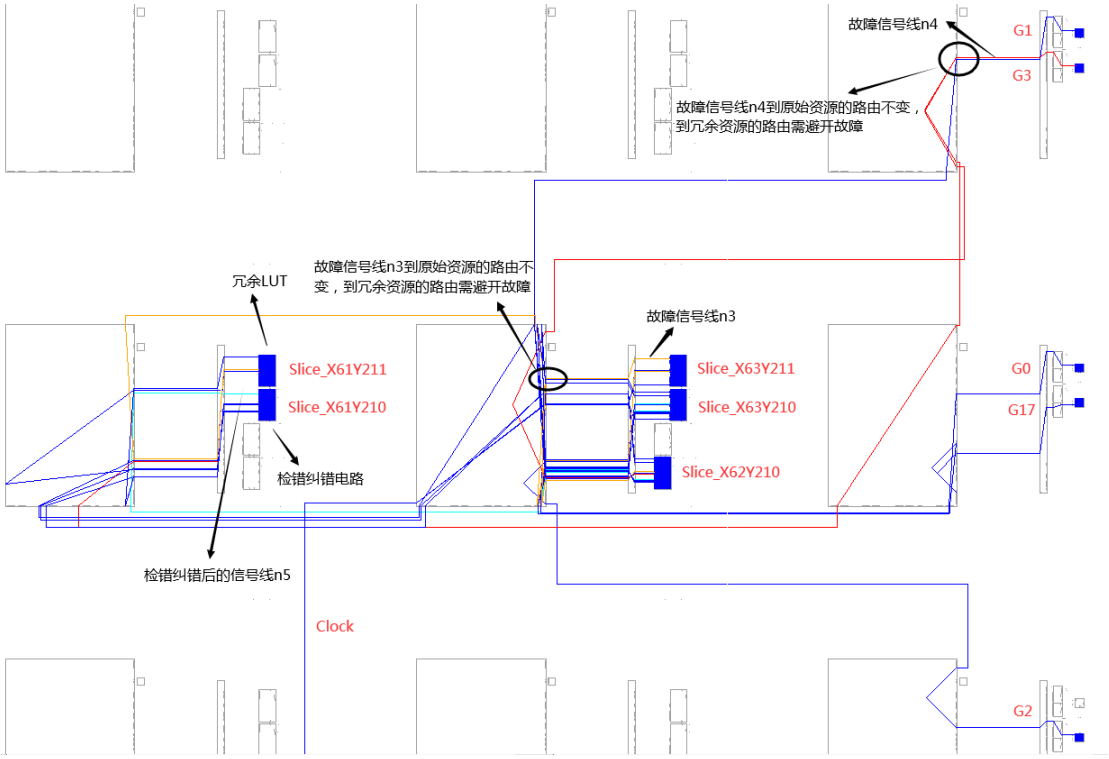


图 6. 26 容错之后的 S27 电路布局布线图

因为在 6.1.1.2 节中对 S27 电路注入无反馈桥接故障后，只有测试配置三检测到 S27 中存在故障，且在后续诊断出该故障存在于信号线 n3 与信号线 n4 之间，而此处使用所提出的容错方案对该故障进行了容错处理，所以为了验证容错方案的有效性，同样使用配置三测试经过容错之后的 S27 电路是否恢复正常工作。本文采用仿真的方式进行验证，在仿真验证过程中因为需要通过输入端 G3 模拟注入故障，为了保证冗余 LUT 不受故障的影响，这里使用 G3_copy 端口为冗余 LUT 输送正确激励。在测试配置三下的正确输入激励为{G0,G1,G2,G3}为“1111”，注入故障后的输入激励{G0,G1,G2,G3}为“1110”,理想输出响应为“0”，仿真结果如图 6.27 所示。

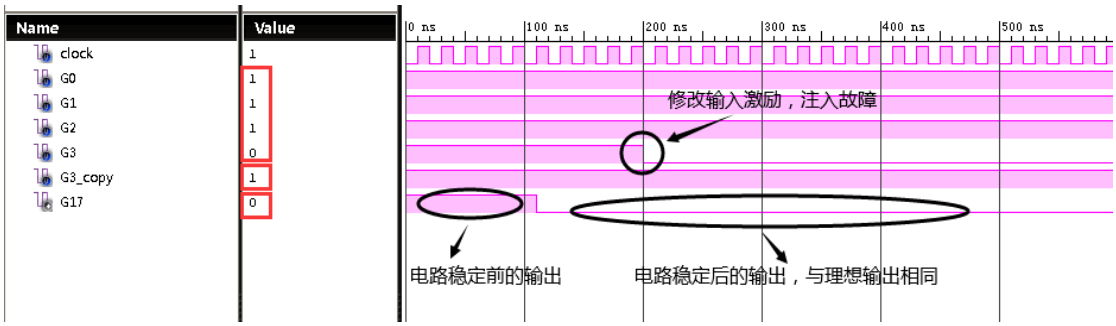


图 6. 27 注入无反馈桥接故障后进行容错的 S27 配置三仿真图

从图 6.27 中的仿真波形可看出测试配置三下的仿真结果正确，S27 电路工作正常，即对存在故障的 S27 电路实施容错方法之后，电路恢复了正常工作，证明了所提出的容错方案的有效性。

6.3.2 数据结果分析

在上一节中使用了所提出的容错方案对 6.2.1.2 节所诊断到的桥接故障进行容错仿真，结果表明所采取的检错纠错方式能够修复电路中存在的互连故障，恢复电路的正常工作，证明了本文在第五章中所提出的容错方法的有效性。下面分析该容错方法针对不同的 ISCAS'89 基准电路所需要增加的资源消耗情况。

针对第五章中的互连容错方案，选择了与测试以及诊断阶段相同的 20 个基准电路进行容错分析，表 6.11 中展示了这些基准电路映射在 Xilinx Virtex-4（XC4VLX100）FPGA 上的相关参数以及实施容错方案所需增加的资源消耗情况。

表 6. 11 ISCAS'89 基准电路在 Virtex-4 上进行容错的相关参数信息

基准电路	LUT 总数	容错所需增加的LUT 数	增加的资源占比情况
S298	28	3	10.71%
S344	37	3	8.11%
S349	37	3	8.11%
S382	44	3	6.82%
S400	44	3	6.82%
S420	37	3	8.11%
S444	45	3	6.67%
S510	78	3	3.85%
S526	43	3	6.98%
S641	67	3	4.48%
S713	71	3	4.23%
S820	107	3	2.80%
S1196	200	3	1.50%
S1238	209	3	1.44%
S1423	195	3	1.54%
S1488	236	3	1.27%

表 6.11（续）

基准电路	LUT 总数	容错所需增加的LUT数	增加的资源占比情况
S1494	235	3	1.28%
S5378	383	3	0.79%
S9234	291	3	1.03%
S15850	928	3	0.32%

现有修复型容错技术中的硬件级容错大多基于逻辑行/列的移位，所增加的面积消耗过大，且大多数文献年代久远，不好公平地进行数据比对；修复型容错技术中的配置级容错则是避开故障，对整个电路重新进行布局布线，无法评估所增加的面积消耗，因此本文未列出现有文献中的数据进行比对，但从表 6.11 中可看出本文所提出的容错方案只需增加 3 个 LUT 的资源消耗，对于规模越大的基准电路所增加的资源消耗占比越小，且本文直接修复电路中的互连故障，实现的复杂度较小，能更简便地恢复电路的正常工作。

6.4 本章小结

本章以 ISCAS'89 基准电路为研究对象，首先通过模拟注入不同的互连故障类型对第三章中的测试方案进行仿真验证，并评估了不同的基准电路所需的测试配置数；而后在测试的基础上，使用第四章中的诊断方案对测试到的故障进行诊断定位验证，同样对不同基准电路所需的最大诊断配置数进行了分析；最后根据诊断到的故障位置信息，对基准电路进行容错方法的验证，给出了第五章中的容错方案的资源消耗情况。本章的 ISCAS'89 基准电路的仿真实验和数据分析结果证明了本文所提出的互连测试、互连诊断以及互连容错方案的有效性和优越性，为提高 FPGA 的可靠性，保证相应数字系统的稳定性做出了贡献。

第七章 总结与展望

7.1 总结

FPGA 技术的迅猛发展使得其在半导体市场中脱颖而出，越来越受欢迎同时也越来越受重视，研究如何提高 FPGA 的可靠性，保证相应数字系统的稳定性是现在的研究热点。随着工艺水平的不断提升，FPGA 内部集成的资源越来越丰富，使得元器件与元器件之间的距离逐渐变小，内部路由布线的难度也逐渐提升，从而在互连资源上发生故障的概率也就越来越高，因此研究 FPGA 互连资源的可靠性具有重要的意义。

本课题实现了对 SRAM 型 FPGA 互连资源的测试、诊断以及容错方案的设计，并完成了相应的分析和验证工作。现对本文在研究过程中的主要工作进行总结。

1）提出了一种基于改进故障模型的应用相关互连测试方案，该方案考虑到用户电路中物理距离相距较远的互连线之间发生桥接故障的概率很小，通过约束桥接故障只发生在同一 LUT 的信号线之间，从而优化对桥接故障的测试，随后又将反馈桥接故障的类型进行细分，结合单项函数的使用，只需简单的测试向量即可检测到相应故障的存在，最后使用 SAT 对用户电路中存在的所有互连故障进行约束，生成对应的测试配置进行测试，解决了现有文献中反馈桥接故障的覆盖难题。

2）提出了一种基于故障粗细粒度范围的非自适应互连故障诊断方案，该方案在测试的基础上对用户电路中的任意单个互连故障进行诊断，首先对用户电路的互连情况进行分析，确定电路的单输出路径和多输出路径；然后结合测试阶段的错误输出端口信息判断故障所在的路径，再针对该路径进行粗粒度诊断，缩小故障诊断范围，减小诊断复杂度；继而，复用测试阶段的测试配置和测试结果对故障粗粒度范围内的信号线进行细粒度精准定位，分别从成功的测试配置和失败的测试配置中对无故障信号线进行排除，最终完成对单故障的故障类型以及位置的诊断。

3）提出了一种基于细粒度冗余的修复型互连故障纠错容错方案，该方案在诊断的基础上，对定位到的互连故障进行修复容错，首先根据定位到的互连故障所在节点位置，对相邻的原始逻辑资源进行基于 LUT 的细粒度冗余复制，然后插入检错纠错电路，在检错纠错电路中对原始逻辑资源和冗余逻辑资源进行两次异或操作，第一次异或操作检测存在的互连故障是否引起电路发生错误改变，第二次异或操作纠正该错误改变，从而恢复电路的正常工作，实现对互连故障的修复。

4）通过在 ISCAS'89 基准电路上模拟注入互连故障，对所提出的测试、诊断以及容错方案进行仿真验证，结果证明了所提出的方案的可行性。同时对相应方案所需的测试配置数、诊断配置数以及容错所增加的资源消耗情况进行了分析评估，与现有论文中的方案相比，测试方案中的测试配置既覆盖了电路中所有的互连故障，所需的测试配置数也有所减少；诊断

方案既降低了诊断工作的复杂度，所需的诊断配置数也有一定的优势；容错方案直接修复互连故障以恢复电路正常工作，所增加的资源消耗也非常少。因此本文所提出的方案对提高FPGA 的可靠性，保证相应数字系统的稳定性具有重要意义。

7.2 展望

本文对 SRAM 型 FPGA 互连资源提出的应用相关测试、诊断以及容错方案能够提高FPGA 的可靠性，但相应的方案仍然存在一些需要改进和完善的地方：

1）本文针对所提出的互连测试、诊断以及容错方案进行了完备的可行性和优越性验证，但若存在成熟且便捷的故障注入技术进行辅助，则能够大幅提高 FPGA 可靠性方案的研究效率，加快可靠性方案的迭代验证以及优化，从而为提高 FPGA 的可靠性做出更多的贡献。

2）本文所提出的故障测试以及诊断方案能够结合 FPGA 的部分可重构功能进行优化，使用 FPGA 的内部配置访问接口（Internal Configuration Access Port，ICAP）能够大幅的优化测试以及诊断 FPGA 所需的重配置步骤，减少相应的时间消耗。

3）本文所提出的可靠性方案完成了对 FPGA 内部互连资源的故障测试、故障诊断以及故障容错，但基于 FPGA 的数字系统规模越来越庞大，所包含的 FPGA 数量也越来越多，需要将相应的可靠性方案与自动化技术结合，采取流水线的形式大批量地对 FPGA 进行自动测试和诊断，以适应未来大量 FPGA 芯片的可靠性需求。

参考文献

[1] 刘春玲, 栾良龙. FPGA 的发展概况与发展趋势[J]. 经济技术协作信息, 2019(34): 8-9.

[2] 华经产业研究院. 2020-2025 年中国 FPGA 行业投资战略规划市场调研报告[EB/OL]. Available in http://www.hjbaogao.com.cn/bg/2019-11-28/505614.html, 2020.

[3] 张颖, 毛志明, 陈鑫. 基于静态随机存取存储器型 FPGA 的测试技术发展[J]. 电子与封装, 2021, 21(01): 37-47.

[4] 杨超. 基于 J750 的 Virtex-Ⅱ 型 FPGA 测试[D]. 成都: 电子科技大学, 2018.

[5] Stroud C, Konala S, Chen P, et al. Built-in self-test of logic blocks in FPGAs (Finally, a free lunch: BIST without overhead!)[C]//Proceedings of 14th VLSI Test Symposium. IEEE, 1996: 387-392.

[6] Huang W K, Meyer F J, Chen X T, et al. Testing configurable LUT-based FPGA's[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 1998, 6(2): 276-283.

[7] Abramovici M, Stroud C, Hamilton C, et al. Using roving STARs for on-line testing and diagnosis of FPGAs in fault-tolerant applications[C]//International Test Conference 1999. Proceedings (IEEE Cat. No.99CH37034). IEEE, 1999: 973-982.

[8] Abramovici M, Stroud C E. BIST-based test and diagnosis of FPGA logic blocks[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2001, 9(1): 159-172.

[9] 杨嵩. Virtex 型 FPGA 的测试理论和方法研究[D]. 成都: 电子科技大学, 2011.

[10] Renovell M, Figueras J, Zorian Y. Test of RAM-based FPGA: methodology and application to the interconnect[C]//Proceedings. 15th IEEE VLSI Test Symposium (Cat. No.97TB100125). IEEE, 1997: 230-237.

[11] Renovell M, Portal J M, Figueras J, *et al*. Testing the interconnect of RAM-based FPGAs[J]. IEEE Design & Test of Computers, 1998,15(1): 45-50.

[12] Liu P, Xu N, Hui F. An Automatic Method for Testing of FPGA Routing Resource[C]//2018 2nd IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC). IEEE, 2018: 918-923.

[13] Banik S, Roy S, Sen B. Test configuration generation for different FPGA architectures for application independent testing[C]//2019 32nd International Conference on VLSI Design and 2019 18th International Conference on Embedded Systems (VLSID). IEEE, 2019: 395-400.

[14] 周涛. SRAM 型 FPGA 的测试技术研究[D]. 成都: 电子科技大学, 2006.

[15] 罗俊杰. FPGA 故障检测方法研究及软件实现[D]. 合肥: 中国科学院大学, 2015.

[16] Stroud C, Wijesuriya S, Hamilton C, et al. Built-in self-test of FPGA

interconnect[C]//Proceedings International Test Conference 1998 (IEEE Cat. No.98CH36270). IEEE, 1998: 404-411.

[17] Sun X L, Xu J, Chan B, et al. Novel technique for built-in self-test of FPGA interconnects[C]//Proceedings International Test Conference 2000 (IEEE Cat. No.00CH37159). IEEE, 2000: 795-803.

[18] Stroud C, Lashinsky M, Nall J, et al. On-line BIST and diagnosis of FPGA interconnect using roving STARs[C]//Proceedings Seventh International On-Line Testing Workshop. IEEE, 2001: 27-33.

[19] Smith J, Xia T, Stroud C. An Automated BIST Architecture for Testing and Diagnosing FPGA Interconnect Faults[J]. Journal of Electronic Testing, 2006,22(3): 239–253.

[20] 杨会平. 基于 BIST 的 SRAM 型 FPGA 测试技术研究[D]. 成都: 电子科技大学,2012.

[21] Tahoori M B. Application-dependent testing of FPGA interconnects[C]//Proceedings 18th IEEE Symposium on Defect and Fault Tolerance in VLSI Systems. IEEE, 2003: 409-416.

[22] 电子科技大学. 一种应用相关型 FPGA 自动化测试配置方法, 中国, 发明专利, CN201610895371.X[P]. 2017-03-29.

[23] Banik S, Roy S. Application dependent testing of FPGA interconnect using satisfiability modulo theory[C]//2018 3rd International Conference for Convergence in Technology (I2CT). IEEE, 2018: 1-5.

[24] Tahoori M B, McCluskey E J, Renovell M, et al. A multi-configuration strategy for an application dependent testing of FPGAs[C]//22nd IEEE VLSI Test Symposium. IEEE, 2004: 154-159.

[25] Tahoori M B. Application-Dependent Testing of FPGAs[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2006, 14(9): 1024-1033.

[26] Almurib H A F, Kumar T N, Lombardi F. A Single-Configuration Method for Application-Dependent Testing of SRAM-based FPGA Interconnects[C]//2011 Asian Test Symposium. IEEE, 2011: 444-450.

[27] Kumar T N, Lombardi F. A Novel Heuristic Method for Application-Dependent Testing of a SRAM-Based FPGA Interconnect[J]. IEEE Transactions on Computers, 2013, 62(1): 163-172.

[28] Cilardo A. New Techniques and Tools for Application-Dependent Testing of FPGA-Based Components[J]. IEEE Transactions on Industrial Informatics, 2015, 11(1): 94-103.

[29] Banik S, Roy S, Sen B. Application-Dependent Testing of FPGA Interconnect Network[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2019, 27(10): 2296-2304.

[30] 罗毅. 动态可重构 FPGA 的电路测试技术研究[D]. 成都: 电子科技大学, 2009.

[31] 张娜. 基于 FPGA 的故障检测与定位的容错机制的研究[D]. 北京: 北京化工大学, 2011.

[32] Doumar A, Ito H. Detecting, diagnosing, and tolerating faults in SRAM-based field programmable gate arrays: a survey[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2003, 11(3): 386-405.

[33] Tahoori M B. Diagnosis of open defects in FPGA interconnect[C]// 2002 IEEE International Conference on Field-Programmable Technology. IEEE, 2002: 328-331.

[34] Tahoori M B. High Resolution Application Specific Fault Diagnosis of FPGAs[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2011, 19(10): 1775-1786.

[35] Nandha K T, Almurib H A F, Chin-Ee N. Fine grain faults diagnosis of FPGA interconnect[J]. Microprocessors & Microsystems, 2013, 37(1): 33-40.

[36] Almurib H A F, Nandha K T, Lombardi F. Scalable Application-Dependent Diagnosis of Interconnects of SRAM-Based FPGAs[J]. IEEE Transactions on Computers, 2014, 63(6): 1540-1550.

[37] Yu Y L, Xu J, Huang W K, et al. Minimizing the number of programming steps for diagnosis of interconnect faults in FPGAs[C]// Proceedings Eighth Asian Test Symposium (ATS'99). IEEE, 1999: 357-362.

[38] Das D, Touba N A. A low cost approach for detecting, locating, and avoiding interconnect faults in FPGA-based reconfigurable systems[C]//Proceedings Twelfth International Conference on VLSI Design. (Cat. No.PR00013). IEEE, 1999: 266-269.

[39] Stroud C, Nall J, Lashinsky M, et al. BIST-based diagnosis of FPGA interconnect[C]//Proceedings. International Test Conference. IEEE, 2002: 618-627.

[40] Nirmalraj T, Radhakrishnan S, Pandiyan S. Automatic diagnosis of single fault in interconnect testing of SRAM-based FPGA[J]. IET Computers & Digital Techniques, 2021, 15(5): 362-371.

[41] 项传银, 阮爱武, 李文等. 基于故障映射的 FPGA 互连资源故障测试与定位[J]. 仪器仪表学报, 2011, 32(09): 2010-2015.

[42] 马珂洁, 包杰, 周学功等. 基于局部重配置的 FPGA 互连测试诊断[J]. 计算机工程, 2011, 37(5): 249-252.

[43] Mojoli G A, Salvi D, Sami M G., et al. KITE: a behavioural approach to fault-tolerance in FPGA-based systems[C]//Proceedings. 1996 IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems. IEEE, 1996: 327-334.

[44] D'Angelo S, Metra C, Pastore S, et al. Fault-tolerant voting mechanism and recovery scheme for TMR FPGA-based systems[C]//Proceedings 1998 IEEE International Symposium on

Defect and Fault Tolerance in VLSI Systems (Cat. No.98EX223). IEEE, 1998: 233-240.

[45] Kretzschmar U, Astarloa A, Lázaro J, et al. Robustness of different TMR granularities in shared wishbone architectures on SRAM FPGA[C]//2012 International Conference on Reconfigurable Computing and FPGAs. IEEE, 2012: 1-6.

[46] Goncalves M V M, Villa P R C, Neto H C C, et al. A TMR Strategy with Enhanced Dependability Features Based on a Partial Reconfiguration Flow[C]//2015 IEEE Computer Society Annual Symposium on VLSI. IEEE, 2015: 161-166.

[47] Mahmoud D G., Alkady I G, Amer H H, et al. Fault secure FPGA-based TMR voter[C]//2018 7th Mediterranean Conference on Embedded Computing (MECO). IEEE, 2018: 1-4.

[48] Stott E, Sedcole P, Cheung P Y K. Fault tolerant methods for reliability in FPGAs[C]//2008 International Conference on Field Programmable Logic and Applications. IEEE, 2008: 415-420.

[49] Hatori F, Sakurai T, Nogami K, et al. Introducing redundancy in field programmable gate arrays[C]//Proceedings of IEEE Custom Integrated Circuits Conference - CICC '93. IEEE, 1993: 7.1.1-7.1.4.

[50] Kelly J L, Ivey P A. Defect tolerant SRAM based FPGAs[C]//Proceedings 1994 IEEE International Conference on Computer Design: VLSI in Computers and Processors. IEEE, 1994: 479-482.

[51] Lach J, Mangione-Smith W H, Potkonjak M. Low overhead fault-tolerant FPGA systems[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 1998, 6(2): 212-221.

[52] Lach J, Mangione-Smith W H, Potkonjak M. Enhanced FPGA reliability through efficient run-time fault reconfiguration[J]. IEEE Transactions on Reliability, 2000, 49(3): 296-304.

[53] Emmert J M, Bhatia D K. A fault tolerant technique for FPGAs[J]. Journal of Electronic Testing, 2000, 16(6): 591-606.

[54] Huang J, Tahoori M B, Lombardi F. Fault tolerance of switch blocks and switch block arrays in FPGA[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2005, 13(7): 794-807.

[55] Campregher N, Cheung P Y K, Constantinides G A, et al. Reconfiguration and Fine-Grained Redundancy for Fault Tolerance in FPGAs[C]//2006 International Conference on Field Programmable Logic and Applications. IEEE, 2006: 1-6.

[56] Moura de L, Bjørner N. Z3-a tutorial [EB/OL]. Available in http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.225.8231, 2014.

致 谢

我想硕士毕业论文应该是我学生生涯的最后一份作业了，这份作业来之不易，能够接触到这份作业得益于许多人对我的倾囊相助，因此在与这份作业相处起来也十分重视，能够圆满地完成这份作业，交出一份满意的答卷，也算是给我的人生安上了明亮而温馨的路灯。

张颖老师，一个心系学生的老师，用多么美好的词语去夸赞、去点缀都不为过分。我承认我是属于脑子较笨的那一类人，能够支持我爬到现在的只能是背负了太多，不得不往前冲，但你知道，有时候努力是没有用的，尤其是科研。那时刚进实验室的我完全可以用“驴”去形容，而是是蠢的那种，面对张老师的科研任务安排，我好像总是找不到出路，使尽全身力气一通乱打，却好似打在豆腐上，完不成任务怎么办？科研没进展怎么办？上个星期好像汇报的也是这个，这个星期的组会继续说？不幸而又幸运，突如其来的疫情打乱了整个世界的脚步，被病毒困扰的人流离失所甚至失去生命，而我现在还呼吸着新鲜的空气也是要感谢国家对疫情的快速反应与重视，这里向疫情期间的医务人员以及工作人员致以崇高的敬意与感谢。疫情阻挡了一切社会活动，也就意味着开不了学，“山高皇帝远”，在家里学习，摸鱼是少不了的，也就让我暂时远离了科研的烦恼，那段时间的头发好像也更加茂盛了些。在家里躲了好几个月的科研，开学就不得不面对张老师了，也许张老师看我对自己下不了狠手，便顺手推了一把，安排我写篇综述，也好在这篇综述，让我开启了科研的大门。师傅领进门，修行看个人，上道了之后自己也就野蛮生长了，但每当有科研迷茫期的时候，找张老师就准没错，这不，这份作业就是最好的证明。

陈鑫老师，也是为数不多的模范榜样，别人嘴里都说着当老师很轻松，但在我眼里好像不是这么回事，实验室每周雷打不动的组会是为了督促我们的科研进度，日复一日劳累地奔波于研究生和本科生之间而导致身体的正常温度都偏低，就这样每次见到我们还都是一幅标志性的笑脸与轻声细语的问候，那时就感觉科研一天的疲劳也不过如此。大事小事在陈老师眼里都是重要的事，亲力亲为是陈老师的处事态度，疲劳的身影后面看到的总是为学生们兜底的严谨与细心，仔细想想，当我的同门，陈老师的学生找到惬意而又高薪的工作时，也许就是为陈老师打的最响亮的 CALL 了吧。

这里也要感谢同门姚嘉祺、陈凯、马丽萍、单永欣和曹建鹏对我研究生生活的帮助和鼓励，感谢已经毕业的李森、施聿哲、葛明慧、刘小雨、金铮斐、高翔、张骁煜等师兄师姐们的热心指导，感谢华屹峰、杨济中、李源翔、姚娇艳、刘涛、王雷、张智维、白雨鑫、申志浩、兰宗易、谢玉东等师弟师妹们的陪伴。

最后要感谢我的父母与长辈给予我穷尽一生也无法偿还的爱，这是我前行的动力，是我永攀高峰的决心。

感谢所有在百忙之中抽出宝贵时间来审阅论文的各位专家评委老师！

在学期间的研究成果及发表的学术论文

攻读硕士学位期间发表论文情况

1. 张颖, **毛志明**, 陈鑫. 基于静态随机存取存储器型 FPGA 的测试技术发展[J].电子与封装,2021,21(01):37-47.
2. **毛志明**, 张颖, 姚嘉祺, 华屹峰, 杨济中, 陈鑫. FPGA 互连测试中的反馈桥接故障覆盖问题[J].计算机测量与控制,2021.（录用）
3. Jiaqi Yao, Ying Zhang, **Zhiming Mao**, Sen Li, Minghui Ge, Xin Chen. On-line Detection and Localization of DoS Attacks in NoC[C]// 2020 IEEE 9th Joint International Information Technology and Artificial Intelligence Conference, 2020:173-178.

攻读硕士学位期间申请的专利

1. 申请国家发明专利 1 项：一种 SRAM 型 FPGA 故障在线容错方法，排名第 1，申请号：202110830748.4；
2. 申请国家发明专利 1 项：基于机器学习的片上网络硬件木马检测平台，排名第 3，申请号：202111466605.6；

攻读硕士学位期间参加科研项目情况

1. 国家自然科学基金, 61701228，众核片上网络芯片的硬件木马在线检测关键技术研究；
2. 横向课题，PCIe 转 PCI/X 桥芯片 ASIC 设计。