# SRAM 型 FPGA 的可重构容错结构研究*

张程程 班 恬
(南京理工大学 南京 210094)

**摘 要**：针对 SRAM 型 FPGA，提出了一种基于动态可重构技术的容错设计方法，根据瞬态错误概率的高低来动态控制系统的冗余程度。在错误率低的时候，系统采用双备份比较(DWC)，具有较低的面积开销和功耗；在错误率高的时候，系统切换到三模冗余(TMR)排除单个错误的影响。采用基于代理逻辑(Proxy LUT)和早期获取部分可重构(EAPR)的设计方法，以 ISCAS'85 benchmark 电路中的大型代表电路为验证模块，叙述了动态可重构的容错结构的实现过程，并重点验证了动态可重构容错设计方法和其它静态容错方法相比，在面积和功耗上的优势，结果表明动态可重构容错结构相比混合容错结构而言，其面积开销和功率消耗较小。

**关键词**：容错；动态部分重构；面积开销；功耗
**中图分类号**：TP332.1 **文献标识码**：A **国家标准学科分类代码**：510.80

# Research on reconfigurable fault-tolerant architecture of SRAM-based FPGA

Zhang Chengcheng Ban Tian
(Nanjing University of Science and Technology, Nanjing 210094, China)

**Abstract**: This paper proposes a fault-tolerant design method for SRAM-based FPGA by using dynamic reconfiguration technology. This method adjusts the degree of redundancy of the system depending on the various soft error rate. When the error rate is low, the system adopts duplication with compare (DWC) which has lower area overhead and power consumption. If the soft error rate is high, the system switches to the triple modular redundancy (TMR) to eliminate the effects of a single error. By taking the representative circuits in ISCAS'85 benchmark as redundant modules, this paper explains the implementation of fault-tolerant structure of dynamic reconfiguration by using Proxy LUT and EAPR (early-access partial reconfiguration) technology. Finally, the paper compares the simulation results with the state-of-the-art static fault tolerant technique and thus validates the advantages of the proposed method in the aspects of area and power consumption, its area overhead and power consumption is small.

**Keywords**: fault-tolerant; partial dynamic reconfiguration; area overhead; power consumption

## 1 引 言

SRAM 型 FPGA 因其逻辑集成度高、使用方便、开发成本低，并且能够被重新编程，不但广泛应用于民用和工业领域，而且逐步在空间应用中采用。而太空中的辐射环境中存在各种各样的高能粒子，极易使 SRAM 型 FPGA 产生单粒子翻转(single event upset, SEU)，导致系统运行异常，因此必须应用容错技术来保证数字系统的可靠性。

冗余是为提高可靠性普遍采用的方法。三横冗余(triple modular redundancy，TMR)作为硬件冗余技术的代表方法而被普遍采用[1]。但此类静态的容错设计方法代价太高，因此研究人员一直致力于探索更加经济的容错设计方法[2]。德国和法国的研究人员针对 TMR 功耗过大的问题提出了一种新型混合容错结构[3]，如图 1 所示。

图 1 的结构采用了 3 个完全相同的组合逻辑模块(CL1、CL2、CL3)，在任意时刻它们当中只有两个并行运行，而第 3 个处于待机状态。输入数据分配器(Demux)和输出数据选择器(Mux)用来选择两个运行的 CL，因此在正常工作时有 3 种可能的配置(CL1、CL2 运行，CL1、CL3 运行，CL2、CL3 运行)。在无故障运行时，系统保持当前的配置不变，一旦出错，就会转向另一种配置。特殊的输入寄存器(Reg_in)能够存储每一个输入直到比较器证明其输出是

收稿日期：2016-03
* 基金项目：国家自然科学基金(61401205)资助项目

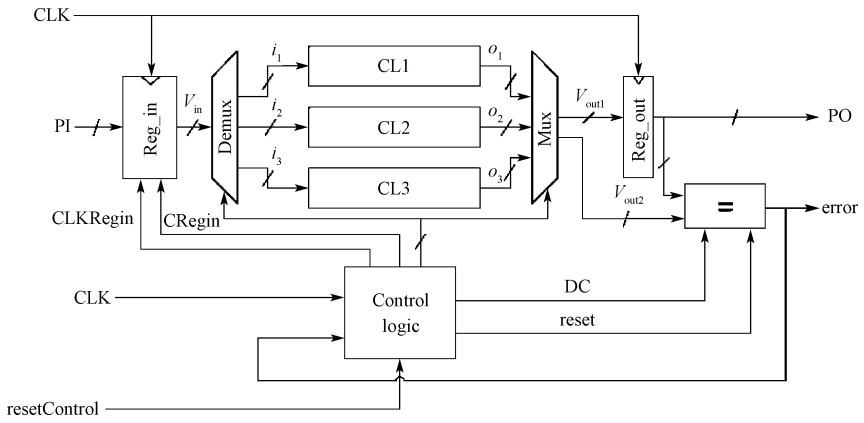


图 1　混合容错结构[3]

正确的，一旦出现错误，这个存储的输入就会被释放以便进行重新计算。

由于这种新型混合容错结构在任意时刻只有两个运行，所以相比传统 TMR 节省了功耗；但是系统本身除了 3 个冗余模块外还包括一些控制模块，所以面积开销相比 TMR 稍大。而采用 FPGA 的动态部分可重构技术，可以减少冗余的资源消耗。

## 2　动态部分可重构容错系统设计

### 2.1　动态部分可重构技术

FPGA 具有现场编程和可重复编程的灵活性，不需要通过重新制作改进设计。动态部分可重构[4]具有更好的灵活性，允许通过载入部分配置文件，通常是部分比特流文件来对运行的 FPGA 设计进行修改。经过完整的比特流文件配置 FPGA 之后，可以下载部分比特流文件以修改 FPGA 的可重构区域，这个过程并不影响未重构部分的运行[5]。

### 2.2　冗余容错技术

冗余是指将需要进行容错的逻辑模块进行备份，即同时运行多个相同的逻辑模块，对备份的所有逻辑模块的结果进行比较。如果有两个备份模块，只需附加一个比较器就能检测出电路是否出错，称为双备份比较(duplication with compare，DWC)。若有 3 个及以上的逻辑模块，将各个模块的输出结果通过一个多数表决器，从而输出正确的结果。

一般情况下，备份的模块越多，容错性能越好，但是硬件资源消耗也越大。TMR 是最常用的一种冗余容错手段，然而许多时候 TMR 的硬件资源消耗也仍然很高，而且还使功耗大增[6]。为了减小 TMR 的资源消耗，本文构建了一种利用动态部分可重构技术的冗余容错系统。

### 2.3　容错系统方案分析

DWC 的资源消耗要小于 TMR。但是 DWC 只能检测错误，不能输出正确的结果。而 TMR 在不出错的情况下，相当于是无用的。

本文针对 SRAM 型 FPGA，提出了一种基于动态可重构技术的容错设计方法，根据瞬态错误概率的高低来动态控制系统的冗余程度。在错误率低的时候，系统采用 DWC，具有较低的面积开销和功耗；在错误率高的时候，系统切换到 TMR 进行容错。本文选择在错误率高的时候进行重构，而不是一出错就重构，这是因为重构也是有代价的，频繁地重构会消耗过多的资源，如面积开销和功耗增大、速度降低、可靠性下降[7]。而我们的目标是用最小的代价来保证系统的可靠性，并且在某些特定应用中，几次偶然的错误出现并不会导致系统失效，所以选择在达到一定错误率之后重构。关于错误率的高低判断，本文采用在输出端加一个计数器的方式实现。当计数器在一定时钟周期内检测到一定数量的错误时，即认为达到了高的错误率，此时重构控制信号 care_out 由低电平变为高电平，系统开始重构为 TMR。

因此本文用 DWC 来检测电路是否出错，当达到一定错误率后，将 DWC 变成 TMR，以得到正确的结果。当正确的结果得出后，再次将 TMR 变回 DWC，变换过程如图 2 所示。

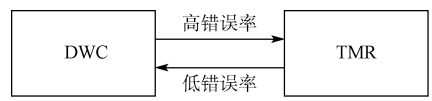


图 2　模式转换

这样，系统在正常工作时为 DWC，达到一定错误率后才变为 TMR，这就大大降低了冗余造成的资源消耗。而要完成从 DWC 到 TMR，再从 TMR 回到 DWC 的变换，就需要用到 FPGA 的动态部分可重构技术[8]。

## 3　动态部分可重构容错系统实现

本文采用基于 EAPR 的设计流程[9]。EAPR 设计方法是 Xilinx 最新提出的，它适用于 Virtex4 及 Virtex5 系列的

芯片[10],本文采用的是 Virtex5 系列的芯片 XC5VLX20T-1FF323,设计流程如图 3 所示[11]。

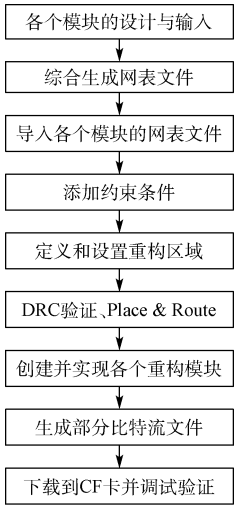


图 3　EAPR 设计流程

1)设计输入与综合

系统包括静态模块、可重构模块和顶层模块。首先进行模块划分[12],动态部分可重构设计中,对系统进行模块划分是相当重要的,需要根据自己的设计需求,划分静态模块和可重构模块,顶层模块中是静态模块和可重构模块的例化。本系统模块划分如图 4 所示。

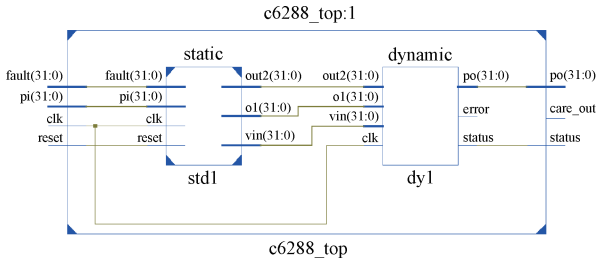


图 4　顶层模块

静态模块中是两个相同的冗余模块和一个控制模块,如图 5 所示。

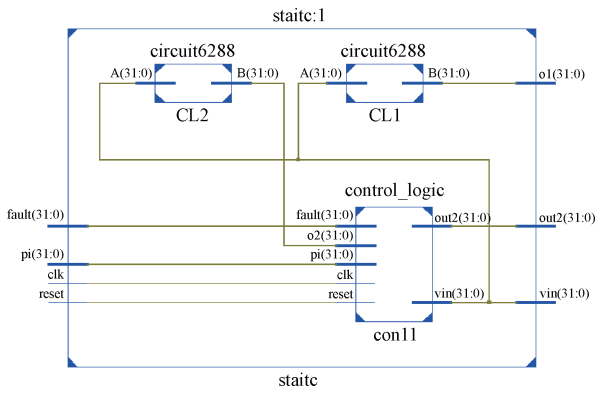


图 5　静态模块

可重构模块有两种实现。可重构模块的第一种实现如图 6 所示,此时可重构模块中是一个比较器,用来对静态模块中的两个冗余模块的结果进行比较,以检测是否有错误。

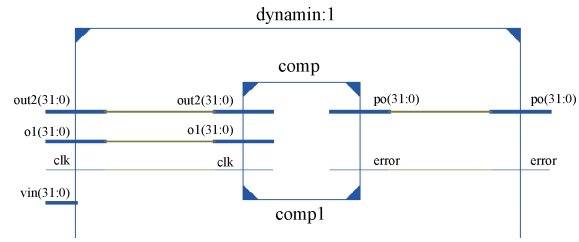


图 6　可重构模块的第 1 种实现

可重构模块的第二种实现如图 7 所示,此时可重构模块中是两个子模块,一个与静态模块中的两个子模块一样,为第 3 个冗余模块,另一个是多数表决器,对 3 个冗余模块的结果进行表决,输出有两个以上相同的结果。

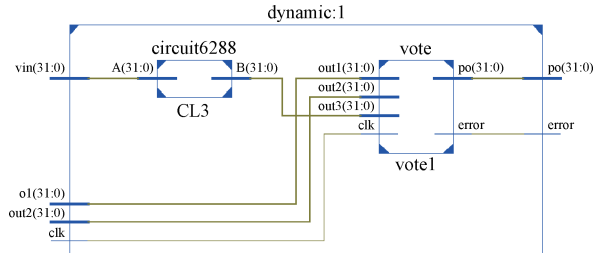


图 7　可重构模块的第 2 种实现

模块划分完成之后,就可以开始用 HDL 语言编写各个模块并进行综合[13]。需要注意的是,顶层模块中的可重构模块要写成黑盒子,静态模块也可以写成黑盒子,但这不是必须的。黑盒子的具体实现要另建一个工程去写。本文中有两种实现,所以要另建两个工程去编写可重构模块的两种具体实现。静态模块和所有的可重构模块在综合时都不添加 I/O 端口,顶层模块在综合时要添加 I/O 端口。完成这一步骤将产生所有模块的.ngc 网表文件。

本设计中没有采用基于总线宏的通信机制,而是采用基于 Proxy LUT 的通信设计。Proxy LUT 是 Xilinx ISE 集成开发平台版本升级以后提供的 LUT1(一种输入只有 1 位的查找表)自动插入手段,与总线宏一样,也是为了确保静态模块与动态模块之间的通信信号具有固定的通信路径和路由资源,在重构前后为静态模块屏蔽动态模块产生的干扰信号。使用 PlanAhead 设计时,一旦定义和划分好重构区域,PlanAhead 工具会自动为可重构模块中的每个端口插入一个代理的 LUT1 查找表。这样,数据或信号在从动态重构模块向静态模块传送的过程中,就会经由 LUT1 的固定路径发送出去,同样动态模块信号的接收也是类似原理。

2)在 PlanAhead 中进行重构实现

采用 PlanAhead 软件创建一个局部重构工程,导入之

前生成的顶层模块的网表文件。接下来划分动态模块的区域，划分好区域后，可将相应的动态模块的网表文件添加进相应动态区域。注意这一步，可以多添加一些空白的动态模块实现，然后让其作为首个实现。这样做可以减小初始配置文件的大小，也就减少了初始配置时间。然后运行设计规则检查(DRC)，添加策略，创建并实现第一个配置和第二个配置，最后生成比特流文件。

## 4　仿真及结果分析

### 4.1　仿真实现

动态可重构容错结构的仿真分两部分完成，首先验证 DWC 的情况下，电路是否能正确地报错并输出重构信号 care_out。将顶层模块中调用的动态模块设置为动态模块的第 1 个实现。然后调用 Modelsim 对顶层模块进行仿真，结果如图 8 所示，从图中可以看到，当在一定时钟周期内检测到一定数量的错误时，重构控制信号 care_out 由低电平变为高电平有效，说明电路可以正确地报错并输出重构信号。下面再来验证 TMR 时，电路能否在有一个冗余模块出错的情况下得到正确的结果。将顶层模块中调用的动态模块设置为动态模块的第 2 个实现，给第 2 个冗余模块注入故障(out2 出错)，其仿真输出如图 9 所示，从图中可以看出，即使有一个模块出错，TMR 电路仍能输出正确结果。

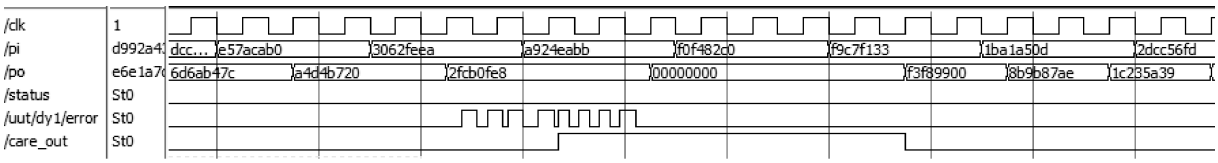


图 8　DWC 输出

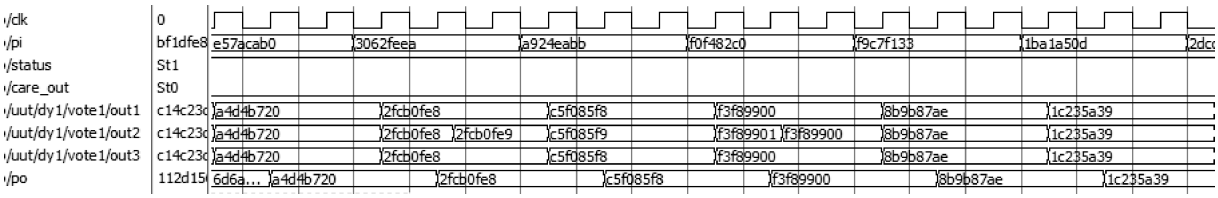


图 9　TMR 输出

### 4.2　结果比较及分析

本文以 ISCAS'85 benchmark 电路中的标准电路为冗余模块，采用 c432、c499、c880、c1355、c1908、c3540、c6288 这 7 个电路实现了混合容错结构(HFT)和动态部分可重构容错结构(DPR)。由于 c2670、c5315 和 c7552 的 I/O 端口较多，超出了本文所选芯片的可用 I/O 端口数目，因此本文对这 3 种电路没有进行仿真验证。表 1 列出了混合容错结构和动态部分可重构结构的面积比较结果，表 2 列出了功耗比较结果。

**表 1　面积比较结果**

| 硬件资源 | c432 | | c499 | | c880 | | c1355 | | c1908 | | c3540 | | c6288 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | BFT[3] | DPR | BFT[3] | DPR | BFT[3] | DPR | BFT[3] | DPR | BFT[3] | DPR | BFT[3] | DPR | BFT[3] | DPR |
| Slice Registers | 84 | 92 | 118 | 113 | 150 | 135 | 118 | 113 | 98 | 107 | 132 | 121 | 116 | 113 |
| Slice LUTs | 448 | 162 | 367 | 270 | 456 | 296 | 349 | 270 | 417 | 237 | 995 | 482 | 1 909 | 730 |
| occupied Slices | 213 | 99 | 149 | 176 | 215 | 164 | 149 | 175 | 130 | 122 | 432 | 209 | 865 | 357 |
| LUT Flip Flop pairs use | 485 | 186 | 401 | 317 | 508 | 336 | 351 | 317 | 437 | 278 | 1 036 | 520 | 1 955 | 779 |

**表 2　功耗比较结果　(mW)**

| | BFT[3] | DPR |
|---|---|---|
| c432 | 20 | 18 |
| c499 | 71 | 57 |
| c880 | 33 | 31 |
| c1355 | 66 | 57 |
| c1908 | 52 | 48 |
| c3540 | 52 | 46 |
| c6288 | 125 | 83 |

表 2 中的动态功耗平均值是在输入矢量分别取 20 组，每组 150 个输入数据的情况下所测得的。另外，上述功耗比较都是在时钟频率为 200 MHz(即时钟周期为 5 ns)的情况下得到的。

从上面的比较可以看出，在实现相同的电路功能，同时具有相同输入输出的情况下，动态部分可重构容错结构比混合容错结构消耗更少的硬件资源，这是因为混合容错结构虽然在任意时刻只有两个在运行，但是系统本身包括

3 个冗余模块,也就是说这 2 个冗余模块一直存在,所以面积开销大;而动态可重构容错结构只有在出错的时候系统才重构成 TMR,正常工作时只有两个模块,也就是说在系统不出错的时候(这也是绝大部分时候),系统就节省出了一个模块的硬件资源。这一个模块的硬件资源并不是要空在那里,等着给第 3 个模块使用。而是可以将进行容错的模块之后的模块写进去,从而能够提高芯片的利用率。这样做并不会影响整个系统功能的实现,因为一旦进行容错的模块出错,其之后的模块也必须暂停等待。与其让其等待,降低芯片时间上的利用率,还不如将其纳入第 3 个模块的区域中去,所以面积开销要小一些。

从功耗比较结果来看,动态可重构容错结构比混合容错结构稍显优势,根据动态功耗的计算公式:$P = C \cdot V^2 \cdot f$,在相同的时钟频率下,输出翻转频率决定了动态功率的消耗,这显然与错误出现的频率和次数有关。在出错的情况下,混合容错结构的输出翻转频率一定大于动态可重构容错结构。这是因为混合容错结构有一个重新计算的过程,一旦出错,就重新计算来对输出进行纠正,相对动态可重构容错结构就多了一次翻转,因为动态可重构容错结构是在达到一定错误率的时候才重构的,这样错误个数增多的时候,混合容错结构的输出翻转就更多,表现在功耗上就消耗更多的功耗。

从容错性能上来看,一旦出错,混合容错结构就转向下一种配置结构,容易造成错误积累,最终导致容错结构失效;而动态可重构容错结构在达到一定错误率的时候就重构成 TMR,也就是加载第二种实现的部分比特流文件,重回 DWC 时就加载第一种实现的部分比特流文件,这样只要不是器件本身出现损坏,也就是硬错误,系统中的软错误都能在加载部分比特流文件的时候消除,不会造成错误积累,所以从容错性能上来说,动态可重构容错结构还是比混合容错结构要好。

综上所述,本文从面积开销、功率消耗和容错性能方面研究论证了动态可重构容错结构相比静态容错结构的优势。

## 5 结 论

动态部分可重构技术能够大大提高芯片上的资源利用率,减少硬件资源消耗,同时功率消耗也比较小,因此成为现在国内外 FPGA 研究的热点[7],这也是本文主要研究的内容。本文提出的动态部分可重构容错设计方法和其它静态容错方法相比,面积开销和功耗有所下降。本文采用的是外部动态部分可重构,这种方式的重构控制器由外部处理器或上位机系统实现。今后可在可重构算法上进行深入研究,采用动态部分自重构来实现[14],这种方式的重构控制器一般由 FPGA 中的嵌入式微处理器实现[15],这将作为下一步的研究方向。

## 参考文献


[1] BALASUBRAMANIAN P, MASKELL D L. A distributed minority and majority voting based redundancy scheme[J]. Microelectronics Reliability, 2015,55(9-10):1373-1378.

[2] BAN T, NAVINER L. Progressive module redundancy for fault-tolerant designs in nanoelectronics [J]. Microelectronics Reliability, 2011, 51(9-11):1489-1492.

[3] TRAN D A, VIRAZEL A, BOSIO A, et al. A new hybrid fault-tolerant architecture for digital CMOS circuits and systems [J]. Journal of Electronic Testing, 2014, 30(4):401-413.

[4] XILINX. Partial reconfiguration user guide [S]. UG702 (v14.5) April 26, 2013.

[5] XILINX. Plan ahead software tutorial, overview of the partial reconfiguration flow[S]. UG743 (v13.2) July,2011.

[6] TONFAT J, LIMA KASTENSMIDT F, RECH P, et al. Analyzing the effectiveness of a frame-level redundancy scrubbing technique for SRAM-based FPGAs[J]. IEEE Transactions on Nuclear Science, 2015,62(6):3080-3087.

[7] 费亚男. 基于动态可重构技术的 FPGA 中 SEU 故障容错方法研究[D]. 哈尔滨:哈尔滨工业大学, 2013.

[8] 贾超群. 基于动态重构技术的 FPGA 电路容错性能评估系统设计[D]. 西安:西安电子科技大学, 2013.

[9] 宋保强,王友仁,张砦. 片上可重构阵列容错方法研究[J]. 电子测量与仪器学报,2015,29(6):830-836.

[10] 肖艺,鲁华祥,陈刚,陈旭. 基于仿生学的多层自适应容错重构阵列研究[J]. 仪器仪表学报,2016,37(2):437-445.

[11] 郑晓云,陶淑苹,冯汝鹏,等. SRAM 型 FPGA 抗单粒子翻转技术研究[J]. 电子测量技术,2015,38(1):59-63.

[12] 潘鸿飞. 基于直接比较测量值的惯性冗余系统容错技术研究[J]. 国外电子测量技术,2013,32(10):8-11.

[13] 丰坤,李跃华,张金林. 一种基于局部可重构技术的 DMR 设计[J]. 舰船电子对抗,2015,38(6):33-38.

[14] 李渊. 基于 EAPR 的通用总线自重构系统设计[J]. 电子测量技术,2015,38(2):16-19.

[15] 孙艳梅,姚睿,郭庆新. 基于动态局部重构的自重构系统设计[J]. 电子测量技术,2015,38(7):11-14.


## 作者简介


**张程程**,1989 年出生,硕士研究生,主要研究方向为数字电路的可靠性分析。

Email:zhchch0321@163.com

**班恬**,1985 年出生,博士,硕士生导师,主要研究方向为面向可靠性的数字设计和数字信号处理。