

基于差异和模块重构技术的动态可重构设计

王开宇, 夏书峰, 刘东旭

(大连理工大学 电子信息工程学院, 辽宁 大连 116024)

摘 要: 传统的动态重构方案分为基于差异的部分重构和基于模块的动态重构, 两种方案存在各自的应用范围. 提出了一种将两种方案的优势结合的设计理念, 既提高了重构速度又为系统的升级提供了可扩展平台. 具体设计了一个由 4 种不同的超前进位加法器构成的动态可重构系统, 通过对系统的布局仿真, 对得到的数据进行分析得到了重构速度和系统的资源利用率.

关键词: 动态重构; 基于差异部分重构; 基于模块部分重构; 超前进位加法器

中图分类号: TP32

文献标识码: A

文章编号: 1000-7180(2010)11-0051-04

A Dynamic Reconfiguration Method Based on Difference and Model Structure

WANG Kai-yu, XIA Shu-feng, LIU Dong-xu

(Department of Electronic Engineering, Dalian University of Technology, Dalian 116024, China)

Abstract: The traditional dynamic reconfiguration schemes are divided into the Difference-based method and the Model-based method, both of which are limited into specific situation. This paper introduces a combination of the above two methods and successes in speeding up the reconfiguration process and allowing for extendibility. A series of CLAs (Carry Look-ahead Adder) based on this new method were designed to test and verify the speed and resource utilization.

Key words: dynamic reconfiguration; difference-based design; model-based design; CLA design

1 引言

动态重构的核心思想就是在系统的当前工作状态下, 微控制器通过加载到存储器中的配置数据流对 FPGA 进行重新的整体或部分配置来动态地改变其内部的电路结构, 但并不切断 FPGA 的运行状态, 从而不需要增减硬件, 利用软件程序, 通过软硬件结合的方式来达到设计要求.

传统的动态重构方式有两种: 基于差异 (Difference-based) 的部分动态重构和基于模块 (Model-based) 的部分动态重构. 前者是通过比较重构前后电路差别, 并根据差别生成相应的比特流配置相应的重构模块. 而如果设计中可重构的逻辑块很大、功能很复杂, 则需要使用基于模块的部分可重构. 近年来混合设计理念在系统设计的改善方面显露出巨大

的优势^[1], 本设计提出了一种基于差异和模块的混合设计方案 (Difference&Model-based) 将上述两种重构方式结合在一个系统设计当中, 充分发挥各自的优点. 通过一个验证性实验得出一些在构建复杂系统时重要的结论.

2 系统概述

实验采用的 FPGA 是 Xilinx 公司生产的支持部分可重构的 Virtex-II Pro xc2vp30 FF896-7 FPGA 开发平台. 动态重构系统性能的好坏在很大程度上取决于系统的构架, 为了使 CPU 和动态重构区域之间的通信时间尽可能短, 使 CPU 更快地配置动态文件, 实验中将动态重构区域布置在 CPU 附近, 同时选用了 Vertex-II Pro 提供的处理器硬核作为重构系统的 CPU. 一个典型的最小系统结构如图 1(a) 所

示,其结构拓扑图如图 1(b)所示。

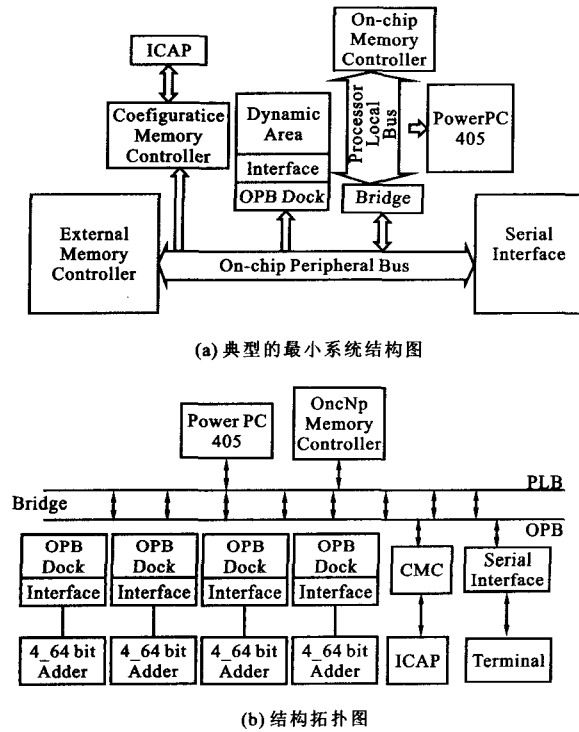


图 1 一个典型的动态可重构系统原理图

对上述完整的动态重构系统可按区域划分为静态区域和动态区域。静态区域是指在整个系统运行当中物理结构和功能不发生变化的区域,在本系统中包括处理器硬核、内部数据存储器 and 配置动态文件存储控制器 CMC。动态区域是指在整个系统运行当中该区域的所有器件在时间上共享该区域并能够动态地改变物理结构和功能的区域,该区域也是在动态配置文件的指示下进行重构的,在本系统中包括动态重构区、内部通信端口 OPB Dock 和系统内部 OPB 总线交换端。其中 OPB 可作为协处理器^[2]。此外,采用 PLB(the Processor Local Bus)和 OPB(On-Chip Peripheral Bus)两种总线结构作为系统内部通信总线,并通过总线宏 BusMacro 来调节系统内各模块之间的协调工作使之满足通信协议,同时存放动态配置文件的 ICAP(the Internal Configuration Access Port)的控制器 CMC 也连接到 OPB 上。

3 动态重构系统的实现

为了验证本系统的可行性,设计了一个在同一动态区域中分别进行四种超前进位加法器的重构系统,一方面来验证动态重构的速度,另一方面来比较不同进位和位数的加法器的资源利用率和速度。四种加法器分别为由 4 位超前加法器组成的 8 位加法

器、由 8 位超前加法器组成的 8 位加法器、由 4 位超前加法器组成的 64 位加法器和一个由 8 位超前加法器组成的 64 位加法器。

3.1 基于差异和模块重构的混合设计方案

本设计中采用的混合设计方案首先利用基于差异(Difference-based)的部分动态重构思想^[3],使用 Xilinx 公司提供的 Jbits 3.0 library 进行库文件的建立^[4]。类似于数字通信中的时分复用技术,将重构电路在时间上进行分割。采用一种新颖的重构设计方案,通过创建一个历史元件库文件(library)来记录重构区域的使用信息,这样当再次对某一区域进行重构时便会通过比较重构数据的先验信息来删除冗余信息进而以更快的速度来进行重构。

当各个重构电路的时间关系建立好后,再利用基于模块(Model-based)的部分动态重构思想^[5]将各个模块在空间上进行分割^[6]。这种空间划分并不是实际意义上的划分,只是对重构区域进行某种程度上的优化,系统在每次进行动态重构时,会根据现有空间的大小来对分割好的区域进行合理布局,并且各模块之间已经模块与接口(OPB Dock)之间可以按照相应的协议相互通信。

混合方案的设计流程依然遵循正常的步骤。对重构系统的设计采用先差异后模块的部分重构方案既能够将重构前后的有效信息充分利用、抑制冗余,加快重构速度,又能够在每次重构系统时安装模块约束进行合理布局,节省空间和资源。

3.2 硬件设计部分

为了设计的通用性,可以通过 ISE 8.2i 内部提供的 IP 核构造超前进位加法器,而不用 HDL 语言设计。在进一步的系统构建中,只需用改变重构区域内的 IP 核便可实现不同功能的系统,比如数字系统中滤波器、编码器等。另一种方法还可以通过较高抽象级别的 VHDL 程序实现比如用 64 位加法指令 PADDD 等。采用 VHDL 语言对动态重构区电路进行描述,作为验证性系统,这里只是注重原理和实现方法。

通过 XPS 8.2i 进行重构区域的电路设计后,其原理图如图 2 所示。其中 A1,A2,A3,A4 分别代表 4 位超前进位的 8 位加法器、8 位超前进位的 8 位加法器、4 位超前进位的 64 位加法器和 8 位超前进位的 64 位加法器,矩形实心模块即为前面提到的系统内部 OPB 总线交换端。每个模块所占区域的大小代表了其内部电路的复杂程度。图 2(a)代表了以一种最简洁的动态区域分配模型,各重构模块简单地连接到 OPB Dock,彼此之间没有信息流通。由于本实

验设计的系统在动态重构过程中会参照上次重构的电路结构而进行本次的电路,为此还可能出现图 2 (b)所示的动态区域分配模型. A3 和 A1 之间, A4 和 A2 之间以及 A3, A4 之间存在信息流通,并通过 A1 和 A2 连接到 OPB Dock. 这里所指的信息流通是指在动态重构过程当中相互参照从而实现重构.

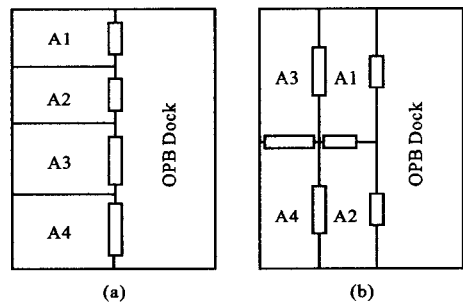


图 2 动态重构区域的资源分配模型

通过 PlanAhead 进行系统的布局布线时,利用总线宏 BusMacro 对动态区域的进行区域约束,使其满足对地址的正确编码和译码. 这样 PlanAhead 在布局

布线时始终能够自动满足约束条件,防止重构过程因为 I/O 地址错误而失败. 系统布局布线后的区域约束如图 3 所示. 配置后的系统资源利用率如表 1 所示.

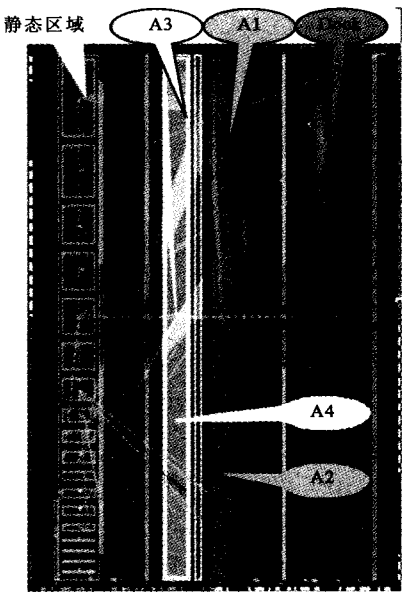


图 3 系统布局布线后的区域约束图

表 1 系统资源利用率

Number of	4_8 adder	8_8 adder	4_64 adder	8_64 adder
External IOBs	26 out of 556 4%	26 out of 556 4%	194 out of 556 34%	194 out of 556 34%
LOCed IOBs	14 out of 26 53%	0 out of 26 0%	14 out of 194 7%	0 out of 194 0%
SLICES	11 out of 13 696 1%	14 out of 13 696 1%	81 out of 13 696 1%	109 out of 13 696 1%
PIN DELAY	1.354ns	1.676ns	4.029ns	3.159ns

3.3 软件设计部分

系统搭建后,按照任务要求用 C 语言完成对主程序的编程.其核心部分为实现动态重构的语句:

```
XIo_Out32(XPAR_OPB_DCR_DOCK_0_DCR_BASEADDR,0x00000000); // disable
XHwIcap_CF2Icap(&MyIcap, "4-8adder.bit");
XIo_Out32(XPAR_OPB_DCR_DOCK_0_DCR_BASEADDR,0x00000001); // enable
```

该语句实现了对外设输入的八位二进制数进行加法.其中 XIo_Out32 为总线宏控制函数, 0x00000000/0x00000001 中最低位为总线宏使能信号,控制动态配置文件的导入. XHwIcap_CF2Icap 为动态配置文件调用函数,通过 Xilinx 公司提供的 ICAP(Internal Configuration Access Port)硬核将配置文件调入动态区域实现动态重构.其他加法器的实现与此类似,这里不再赘述.

4 结果仿真

4.1 动态重构的构建时延

通过不同时刻调用不同的加法器进行动态重构

的平均时延图如图 4 所示系统,从左侧 Y 轴可以看出系统对于 8 位加法器的重构要比 64 位加法器的重构所耗费的时间短,但并不是简单的倍数关系,而且 4 位超前进位的 64 位加法器比 8 位超前进位的加法器实现还要慢一些,上述两点是由于 FPGA 在根据配置文件进行动态重构时,需要改变相应的 Slices、IOBs 和 OPB 总线接口,显然多位数的加法器在重构时耗费的时间会长一些,但对于位数的加法器,重构时间并没有因为超前进位的位数多而延长.当然,上述讨论是在指定四次重构独立发生的情况下重构的延时.若四次重构的发生是相关的,也就是说有一定的先后顺序,那么重构发生的时间就会有很大的不同.这是应为动态重构的基本思想就是动态局部地改变当前电路的结构从而实现不同的要求,若首先运行的是超前 4 位的 8 位加法器,之后要求系统重构为超前 4 位的 64 位加法器或是超前 8 位的 8 位加法器,那么系统便会在原有的硬件结果基础上根据其与重构文件的异同来部分的修改硬件电路,这就会节省一部分时间,由于上述类似情况会有很多种可能的延时,并且

篇幅所限,未能在此列出.

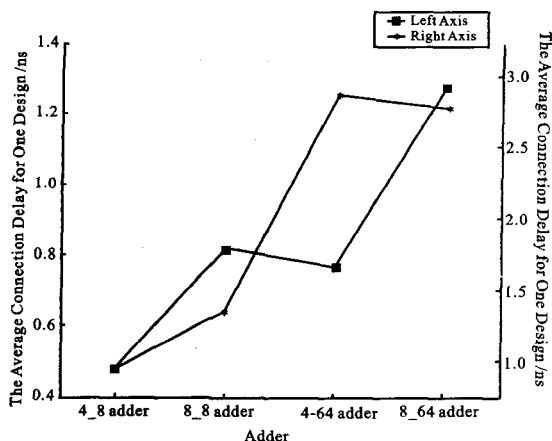


图4 重构系统构建时延图

注意到图4右侧的Y轴列出了在10个最坏网格情况下系统的平均时延情况.首先通过与左侧Y轴对比可以看出,相同条件下,8位超前进位64位加法器比正常情况下延时时间要长得多,而其他三种加法器的相对时延依次减小,对于4位超前进位的8位加法器而言,相对延时非常小.这说明重构系统越复杂,系统的稳定性就越差,在受到相同的内外界因素干扰下延时就越长.因此,在利用动态重构的框架来构架一个复杂系统时,要充分考虑到不稳定因素的存在并加以控制.

4.2 重构系统的执行速度

下面分析一下在重构区域内构建的四种加法器的执行速度.通过ChipScope对重构系统进行在线仿真得到不同加法器的工作时序,绘出图5(假设各个重构独立发生).

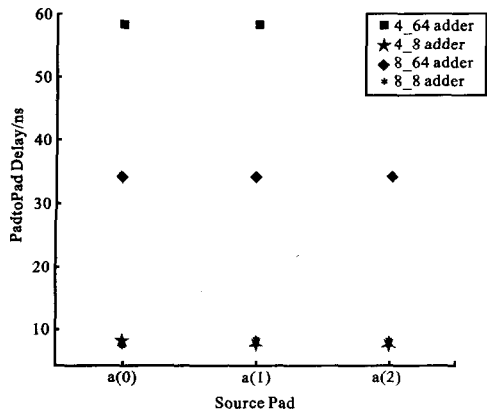


图5 重构系统的执行时间

对于重构8位加法器而言,采用4位超前进位和采用8位超前进位执行的速度非常相近,并且远

远快于64位加法器.但对于64位加法器而言,采用8位超前进位要比采用4位超前进位快很多,虽然在此仅是以加法器为例,但这个实际例子指导我们在构建一个复杂系统时,一方面要考虑到硬件系统自身的复杂度,另一方面还要考虑到算法本身的效率,只有软硬件有机地结合才能充分发挥系统的功能达到预期的效果.

5 结束语

提出了一种全新的设计理念,将传统的动态可重构设计方案进行优势结合,既能够加快系统动态重构的速度和资源利用率,又能应对大规模系统设计对开发周期的要求.通过对四种超前进位加法器重构系统的设计验证了该方案的可行性和优越性.将该方案应用于嵌入式系统的开发当中能够缩短设计周期和降低设计成本,同时对于系统的向上兼容性也有较大的改善.

参考文献:

- [1] Stephen Douglas Craven. Structured approach to dynamic computing application development[C]// Bradley Department of Electrical and Computer Engineering. Blacksburg, Virginia, 2008.
- [2] Enrique Cantó, Francesc Fons. Reconfigurable OPB Co-processors for a microblaze self-reconfigurable SOC mapped on spartan-3 FPGAs[D]. Spain/Catalonia, Electrical and Automatic Engineering University Rovira i Virgili Av, 2006.
- [3] Emi Eto. Difference-based partial reconfiguration[EB/OL]. [2009-11-10]. <http://www.xilinx.com/support/documentation/application-notes/xapp290.pdf>.
- [4] Miguel L Silva. Support for partial run-time reconfiguration[J]. Journal of Systems Architecture, 2006(52): 709-726.
- [5] Ali Behboodian. Model-based design[EB/OL]. [2009-11-15]. <http://www.xilinx.com/publications/magazines/dsp-02/xc-pdf/p52-55-2-dsp-mbd.pdf>.
- [6] 许骏. 基于模块的动态可重构系统设计[J]. 计算机工程与设计, 2008, 29(6): 249-251.

作者简介:

王开宇 男, (1973-), 博士研究生, 讲师. 研究方向为数模混合集成电路设计.