

存储器内建自测试的程序控制方法

刘蓬侠, 鲁建壮

(国防科学技术大学 计算机学院, 湖南 长沙 410073)

摘 要: 文中提出了一种利用处理器的指令系统编写特定程序, 通过程序运行来控制完成整个存储器内建自测试过程的方法. 基于此方法的设计已经成功应用于一款处理器中, 有效地提高了芯片的可测试性和应用系统的容错性.

关键词: 存储器; 内建自测试; 指令系统; 程序

中图分类号: TN432

文献标识码: A

文章编号: 1000-7180(2011)03-0182-04

A Program-Controlled Method for Memory BIST

LIU Peng-xia, LU Jian-zhuang

(School of Computer Science, National University of Defense Technology, Changsha 410073, China)

Abstract: In this article, we opened a kind of novel method. Differed from the current methods which use I/O pins of a processor, this method used the system instruction of a processor to design the special program, which controls the whole process of memory BIST. This method effectively enhanced the testability of the chip and the fault tolerance of an application system, and it has been used in a real chip successfully.

Key words: memory; BIST; instruction system; program

1 引言

在处理器芯片内部存储器是整个芯片的程序和数据吞吐中心, 它和芯片可靠性密切相关, 必须加强其可测试性设计. 由于存储器的物理故障类型及其故障模型都比一般数字逻辑要复杂和多样, 需要对存储器进行特殊的测试处理, 专门为其生成测试码. 工业实践表明, 在外部引脚有限和生产工艺集成度不断提高的前提下, 将存储器地址和数据生成逻辑、测试控制逻辑和测试结果检测逻辑都集成在芯片内的存储器 BIST 是能够较好平衡存储器测试的生产成本和测试成本, 提高测试效率的可测试性设计技术^[1].

目前, 有 2 种存储器 BIST 控制机制^[2]. 一种是使用专用的外部引脚, 作为外部测试台控制存储器 BIST 所需的信号接口; 另一种是在目标处理器设计符合 JTAG 标准时, 复用 JTAG 标准接口 TCK 作为存储器 BIST 的时钟信号接口. 其共同点是: 需要少量

的外部引脚; 控制命令由处理器芯片外部发出; 测试结果要发给处理器芯片外部作进一步处理等.

为进一步减少对芯片外部引脚的使用, 实现在线存储器 BIST, 文中提出了一种利用处理器的指令系统编写特定程序, 通过程序运行来控制完成整个存储器内建自测试过程的方法. 在线 BIST 的另一个优势在于, 随着应用需求的提高, 多个处理核或多个芯片的阵列式应用日益增多, 如果能够在线检测芯片故障, 通过冗余设计或降级处理, 可大大提高整个系统的设计可靠性, 降低系统故障的检测和排除开销.

2 程序控制方法

2.1 目标存储空间的确定

以往, 存储器 BIST 主要用于芯片筛选和板级自检等离线测试, 即芯片在测试时, 芯片及其所在的应用系统处于非工作状态, 在存储器 BIST 运行前

收稿日期: 2010-01-13; 修回日期: 2010-03-07

基金项目: 教育部“高性能微处理器技术”创新团队基金

后,无需保存和恢复芯片及其系统的状态.因此,存储器 BIST 目标存储空间的确立根本不是问题,理所应当涵盖整个片内存储空间.但是,如果要求在线测试,即在芯片及其所在的应用系统处于工作状态时,能够随时启动、运行存储器 BIST;存储器 BIST 结束后,芯片及其系统能够继续正常运行,就必须在存储器 BIST 运行前保存芯片及其系统的状态,在存储器 BIST 运行后恢复芯片及其系统的状态.此时,存储器 BIST 目标存储空间的确立就会引发如何保存和恢复芯片及其系统状态的问题.

文中提出的程序控制方法就是一种存储器 BIST 在线测试方法,即系统运行的任何时候,都可以调用对应的程序,启动存储器 BIST 并获取结果.为满足上述保存和恢复芯片及其系统状态的要求,提出三种解决方案:

(1) 存储器 BIST 目标存储空间包含整个片内存储空间,为此,设计实现透明的存储器 BIST 算法^[3],使得整个片内存储空间在被执行该算法前后的值保持一致;

(2) 存储器 BIST 目标存储空间包含整个片内存储空间,为此,设计实现面向存储器 BIST 的系统自举控制,使得在存储器 BIST 后,重新进行系统自举引导.

(3) 存储器 BIST 的目标存储空间不包含用于存储芯片自举程序的空间(通常是从 0 地址开始的一片空间),用于控制存储器 BIST 的程序代码存储在片外存储器,控制存储器 BIST 的程序代码内含对片内存储空间的保存和恢复代码.

第一种方案面临的问题是:透明的存储器 BIST 算法比作为其基础的 march 算法更复杂,硬件逻辑实现难度加大,面积开销增加.若目标芯片对面积开销不敏感,而且透明存储器 BIST 算法能够有 EDA 工具自动代码实现,第一种方法最直接.

第二种方案面临的问题是:对正常自举和存储器 BIST 后自举的区别控制机制设计有难度,不容易保证正确性.相比上述第一种方法,第二种方法实现难度更大.

实际采用了第三种方案,因为:①用于控制存储器 BIST 的程序代码存储在片外存储器,避免了控制存储器 BIST 的程序代码本身的恢复问题;②控制存储器 BIST 的程序代码内含对片内存储空间的保存和恢复代码,避免了片内存储空间的恢复问题;③存储器 BIST 的目标存储空间不包含用于存储芯片自举程序的空间,这一处理方式与上述①、②结合

在一起,避免了芯片重新自举等一系列恢复状态的引导过程;④能够使用通用的 march 算法,用 EDA 工具自动代码实现,面积开销也得到控制.

2.2 相关指令和控制机制

由于目前通用的存储器 BIST 的 march 算法的特殊性,整个存储器 BIST 过程应该是原子操作,即从启动到结束,不能被任何其它访存操作打断.因此,程序控制式的存储器 BIST 需要相应的阻塞机制,即启动存储器 BIST 的同时,阻止处理器访问片上存储器,不响应其对片上存储器的取指和数据访问操作,直至整个存储器 BIST 过程结束.为简化程序员的设计复杂度,整个控制过程采用普通指令实现,CPU 访问的阻止操作即 BIST 过程的原子性保护过程对程序员透明,具体解决方法如下.

2.2.1 利用写存储器指令启动 BIST

由芯片字长决定的处理器存储空间,一般都不会使用完而存在保留空间,因此,本方法增加一个 BIST 控制寄存器 BCR(BIST Control Register),为它分配一个保留空间地址. BCR 共 1 位,系统复位、BIST 成功完成或 BIST 测试失败时置为逻辑 0,用户只可以写入,读出总是为零.用写存储器指令向 BCR 写入逻辑 1 启动 BIST.

如果进行 BIST 的片内存储器是 RAM,则控制 BIST 的程序中,在用写存储器指令启动 BIST 前,需要用户程序显示的将其内容保存到片外存储器.如果进行 BIST 的片内存储器是 cache,则写存储器指令启动 BIST 后,需要利用 cache 原有的机制,启动进行一系列对用户透明的硬件操作:清空 cache 的内容,并将脏数据回写到外部存储器,而后向 BIST 逻辑发出启动信号.如果片内存储器是可配置的,既可以作为 RAM,又可以作为 cache,则上述 2 种操作都需要.

2.2.2 利用读存储器指令查询 BIST 结果

同样基于保留空间的存在,本方法增加一个 BIST 查询寄存器 BQR(BIST Query Register),也为它分配一个保留空间地址. BQR 至少 2 位,用于记录成功完成(高位)和测试失败(低位)信息,系统复位或 BCR 为逻辑 1 时置为逻辑 0,逻辑值“10”表示 BIST 成功结束,所用算法未检测到任何存储器故障;逻辑值“01”表示 BIST 失败结束,所用算法检测到了存储器故障;测试失败位的优先级高于成功完成位;逻辑值“11”保留.如果需要(如存储器修复)知道 BIST 失败的细节,例如存在故障的存储单元等,则可以对 BQR 进行扩展,或者另外增加新的

BIST 寄存器(也为它分配一个保留空间地址),用于存储这些信息.对于 BQR,用户只可以读出,不能写入.如果 BCR 为高,则禁止 BQR 读出.

因为整个存储器 BIST 过程应该是原子操作,所以用写存储器指令向 BCR 写入逻辑 1 启动 BIST 后,阻止 CPU 访问 BIST 正在测试的存储器以及相关的控制寄存器的逻辑自动生效,直至 BIST 完成,该部分逻辑的设计思想主要是:“不作为”,如果 CPU 访问到上述存储空间,对于读则暂时不返回数据,对于写则不返回完成信号.调用读存储器指令,读 BQR 查询 BIST 结果,也采用了类似策略:BCR 为高即 BIST 没有完成时,存储器控制部件不发送该寄存器内容给访存指令执行部件.如前所述这里的阻止和控制逻辑对用户透明,从而尽可能降低用户的设计复杂度和执行开销.

2.3 逻辑设计

首先,根据片内目标存储体的接口和选定的 march 算法,手工或由 EDA 工具(例如 Mentor Graphics 公司的 MBISTArchitech 工具)自动生成存储器地址生成逻辑、数据生成逻辑、测试控制逻辑和测试结果检测逻辑的 RTL 级代码^[4].在存储器地址生成逻辑中,要根据芯片自举程序的存储空间,确定生成的起止地址.

然后,将生成的代码置于合适的逻辑层次,和其它逻辑进行必要的互连.为了能够在线由程序控制存储器 BIST 并节省芯片外部引脚,连接片内系统时钟信号作为存储器 BIST 所有逻辑的输入时钟信号.

如果进行 BIST 的片内存储器是 cache 或可配

置成 cache,并假设所有的复位信号为低有效,假设存在 1 位的清 cache 启动标志寄存器 CCSR(Cache Clean Start Register)和 1 位的清 cache 完成标志寄存器 CCOR(Cache Clean Over Register),而且都是高有效,则:

(1) 为了能够在线由程序控制存储器 BIST 并节省芯片外部引脚,存储器 BIST 逻辑的输入复位信号为 $\text{Reset}_{\text{BIST}} = \text{Reset}_{\text{sys}} \cdot (\text{BCR} + \overline{\text{CCOR}})$.

(2) CCOR 的复位信号为 $\text{Reset}'_{\text{CCOR}} = \overline{\text{BCR}} \cdot \text{Reset}_{\text{CCOR}}$,使 BCR 由低变高的跳变可以复位 CCOR.

(3) 作为 CCSR 的数据输入,总的清 cache 信号为 $\text{Clean}'_{\text{cache}} = \text{BCR} \cdot \overline{\text{CCOR}} + \text{Clean}_{\text{user}}$,使 BCR 为高并且清 cache 完成标志为 0 时可以触发清 cache 操作, $\text{Clean}_{\text{user}}$ 表示用户显示启动的 cache clean 操作.

(4) 存储器 BIST 测试控制逻辑的输入启动信号为 $\text{Start}_{\text{BIST}} = \text{BCR} \cdot \text{CCOR}$.

(5) 存储器 BIST 测试结果检测逻辑输出的成功完成信号和测试失败信号连接至 BQR,它们的上升沿分别使 BQR 的 2 位置位,并且使 BCR 复位.

整个设计的逻辑框图如图 1 所示.

3 典型软件流程

基于上述硬件设计,用户用软件程序控制存储器 BIST 的典型软件流程如图 2 所示.

首先,用相关指令,将片内目标存储器中配置成 RAM 的部分内的用户有效数据,通过处理器自身的相关逻辑,搬移到用户自己指定的外存区域.然

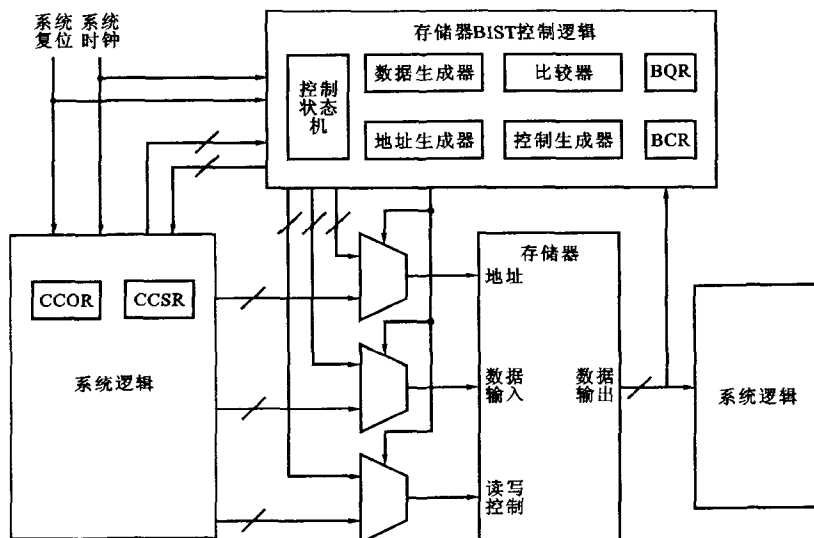


图 1 程序控制存储器 BIST 的逻辑设计框图

后,用写存储器指令写 BCR,启动存储器 BIST.再用读存储器指令读 BQR 查询结果.最后,根据 BIST 的完成情况,软件进行分支处理.如果存储器 BIST 成功完成,则马上转入正常程序运行;否则,无论对于处理器自身还是处理器所在的应用系统,都可以考虑使用中断方式进行后处理.

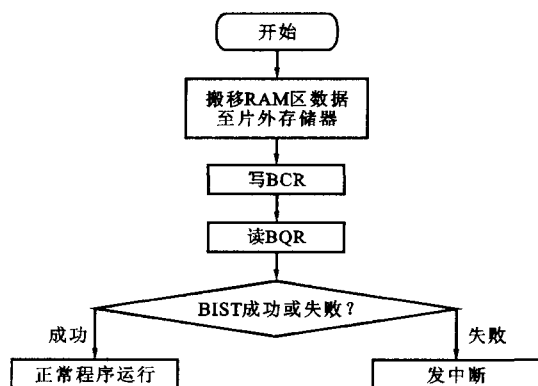


图2 程序控制存储器 BIST 的软件流程

4 结束语

文中程序控制存储器 BIST 的方法,具有以下优点:①整个处理所用到的指令具有通用性,易于扩展;②关键处理(如保持处理的原子性)对用户透明,易于使用;③用户可以根据需要,在应用系统运行的任何时候,通过调用特定的程序段,启动存储器 BIST 并查询结果,可用于冗余系统中的实时在线

检测;④在程序控制的存储器 BIST 工作过程中,处理器还可以访问片外存储器,进行与片上存储器不相关的一些处理工作,从而进一步提高了芯片特殊情况下的可用性.这种方法已经在某款 DSP 芯片上成功应用,有效地提高了芯片的可测试性和应用系统的容错性.

参考文献:

- [1] Michael L. Bushnell, Vishwani D. Agrawal. Essentials of electronic testing for digital, memory & mixed-signal VLSI circuits[M]. 北京:电子工业出版社,2005.
- [2] Agrawal V D, Kime C R, Saluja K K. A tutorial on built-in self-test, part1:principles[J]. Design & Test of Computers, IEEE, 1993,10(1):73-82.
- [3] Nicolaidis M. Transparent BIST for RAMs[C]// Proceed of the international test Conf. USA:IEEE, 1992: 598-607.
- [4] IEEE std 1364-2001. IEEE standard verilog hardware description language[S]. NY:Design Automation Standards Committee of the IEEE Computer Society, 2001.

作者简介:

刘蓬侠 男,(1971-),博士,副研究员.研究方向为高性能微处理器和高性能数字信号处理器的研究与设计,重点研究可测试性和可调试性.

鲁建壮 男,(1977-),博士,副研究员.研究方向为处理器体系结构、微体系结构及存储系统.

(上接第 181 页)

参考文献:

- [1] 王强,普杰信,刘伟. TCP 拥塞控制慢启动策略的研究[J]. 微电子学与计算机,2007,24(12):210-212.
- [2] 钟伯成,张杰. IP 网络中 TCP 拥塞控制协议的稳定性分析[J]. 微电子学与计算机,2005,22(12):177-178.
- [3] Dirceu Cavendish, Kazumi Kumazoe, Masato Tsuru. An adaptive TCP slow start for high speed networks [C]//Proceedings of First International Conference on Evolving Internet. Cannes: PFLDnet, 2009(8):15-20.
- [4] Kapoor R, Chan J J, Chen Y C. CapProbe: A simple and accurate capacity estimation technique[C]// Proceedings of Sigcomm 04, Portland, Oregon, 2004(9):67-78.
- [5] Floyd S. Limited Slow Start for TCP with Large Congestion Windows. RFC 3742 [S]. The Internet Society, 2004.
- [6] Sangtae Ha, Injong Rhee. Hybrid slow start for high-bandwidth and long-distance networks [C]// North

Carolina State: PFLDnet, 2007.

- [7] Andrea Baiocchi, Angelo P. Castellani, Francesco Vacirca. YeAH-TCP: Yet Another Highspeed TCP[C]. Italy: PFLDnet, 2007.
- [8] Makoto Nakamura, Mary Inaba, Kei Hiraki. Fast ethernet is sometimes faster than gigabit ethernet on LFN [C]. Tokyo: PDCS, 2003.
- [9] Kun Tan, Qian Zhang, Murari Sridharan. Compound TCP: a scalable and TCP-friendly congestion control for high-speed networks[C]. Beijing: PFLDnet, 2006.

作者简介:

王国栋 男,(1981-),博士研究生.研究方向为高速网络,传输协议.

任勇毛 男,(1981-),博士,助理研究员.研究方向为高速网络,传输协议.

李俊 男,(1968-),博士,研究员,博士生导师.研究方向为下一代互联网、高速网络.