

基于 FLASH-March 算法的 FLASH 缺陷检测系统

郭进杰, 王 瑜, 李 婷, 霍宗亮

(中国科学院 微电子所, 北京 100029)

摘 要: 介绍了 FLASH 缺陷机理并提出了面向 8 bit 和 16 bit 的 FLASH-March 算法, 在此基础上设计并实现一种新的 FLASH 缺陷检测系统. 该系统以 FPGA 为硬件基础, 以 Microblaze 软核为 CPU 搭建系统主体架构, 实现了对 FLASH 缺陷的检测. 相比于传统方法, 本系统有实现简单, 成本低, 便于相关算法研究的特点. 本设计最终在实际中得到应用, 验证了设计的有效性和可靠性.

关键词: FLASH; 缺陷; FLASH-March 算法; FPGA

中图分类号: TN402

文献标识码: A

文章编号: 1000-7180(2014)10-0081-05

A Flash Fault Detection System Based on FLASH-March Algorithm

GUO Jin-jie, WANG Yu, LI Ting, HUO Zong-liang

(Institute of Microelectronics, Chinese Academy of Science, Beijing 100029, China)

Abstract: We introduced the mechanism of FLASH fault, and FLASH-March algorithm for 8 bit-oriented and 16 bit-oriented is also given, thus we propose a flash fault detection system. The system is based on FPGA and Microblaze, and achieved the detection of flash fault. Also, compared with Built-In Self-Test, the system has the characteristics of simple, low cost and easy to do research on related algorithms. Finally, the system is used in practice and the effectiveness of our system is verified.

Key words: FLASH; Defect; FLASH-March Algorithm; FPGA

1 引言

随着科技的快速发展, 电子产品对 FLASH 芯片的需求日益增长. 但在 FLASH 芯片工艺设计和样品生产的过程中可能产生缺陷, 因此样品流片后需要进行测试, 以检测芯片是否有缺陷, 保证设计和生产过程的正确性.

常用方法是通过 BIST(Built in Self Test)来验证芯片可靠性^[1-2]; 通过在芯片内集成硬件电路来产生测试向量, 进而实现测试算法, 这会带来如下问题:

首先, BIST 硬件电路确定后不可更改, 如果对算法进行改进需要重新设计硬件电路. 其次, 在工艺设计同时进行 BIST 电路设计需要耗费额外资源保证电路的正确性.

基于此, 本文提出一种不同于 BIST 的 FLASH

芯片测试系统: 以 FPGA 为硬件基础, 通过搭载 LINUX 系统和软件来产生测试向量, 验证算法可靠性以及芯片的正确性, 实现对 FLASH 的检测和故障定位.

相比于传统的 BIST 验证和测试方法, 本系统的优势主要体现在以下几方面:

(1) 不涉及 BIST 硬件电路, 能够灵活地对算法进行改进, 方便进行相关算法的研究. (2) 采用 FPGA 对 FLASH 芯片操作. 基于 FPGA 的可编程特点, 适用于多种存储芯片. (3) 用软件的形式来实现算法, 因此能够实现多种算法, 对芯片进行全面细致检测.

2 FLASH 闪存芯片缺陷模型

FLASH 主要有 NOR 型和 NAND 型两种, 具体又可分为 SLC(Single Layer Cell)和 MLC(Multi

Level Cell), SLC 存储形态只有 0 和 1 两种, 单个存储单元只能存储单 bit 数据; MLC 单个存储单元能存储 2 bit 甚至多 bit 数据. 本文以 NOR 型 MLC 存储芯片为测试目标, 芯片型号为 Numonyx 公司的 JS28F256P30T95^[3-4].

FLASH 典型的存储结构采用浮栅型结构, 通过改变 MOS 管的阈值电压实现对数据的存储. 如图 1 所示.

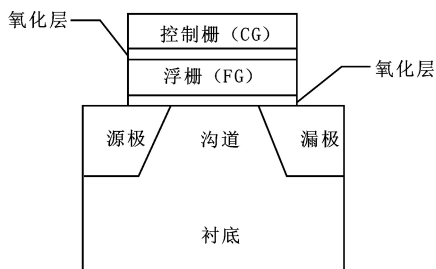


图 1 FLASH 存储单元结构

当 CG(Control Gate) 加正高压(编程)时, 电荷穿过栅氧化层被束缚在浮栅内, 使阈值电压变大. 这样当加栅电压进行读操作时, 沟道电流很小, 对应于存储状态 0. 当浮栅内电荷被释放(擦除)时, 使阈值电压减小. 相应的读操作有沟道电流, 对应存储状态 1.

实际应用中以多个存储单元组成的存储阵列为基础结构, 如图 2 所示.

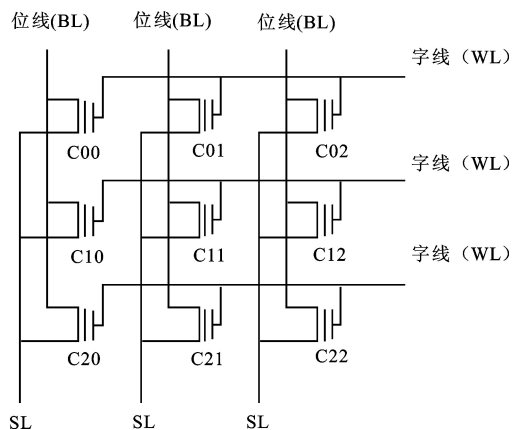


图 2 NOR FLASH 存储阵列结构

同一行的存储单元栅极相连, 称为字线 (Word Line), 同一列的存储单元漏极相连, 称为位线 (Bit Line), 同一列的存储单元源级相连, 称为 SL. 当需要对某个存储单元操作时, 在相应的字线和位线上加载电压即可选中该存储单元, 这样操作方式可能会造成以下问题^[5-6].

字线编程扰动 (word-line program disturbance): 当选定某一存储单元进行编程时, 会引起字

线方向未编程的存储单元编程. 这是由于当某个存储单元编程时, 字线方向栅极加高电压, 若某个存储单元的栅氧化层有缺陷, 可能导致该单元编程. 由于是栅极电压引起, 也可被称为 gate program disturbance (GPD).

字线擦除扰动 (word-line erase disturbance): 当选定某个存储单元编程时, 会引起字线方向某个已编程单元擦除. 这是由于栅极加高电压时, 如果某个存储单元在栅极和浮栅之间的氧化层有缺陷, 浮栅中的电子可能穿过氧化层泄露, 造成该单元被擦除, 也可称为 gate erase disturbance (GED).

位线编程扰动 (bit-line program disturbance): 当某存储单元编程时, 需要在漏极加高电压, 整个位线的漏极处于高电平. 若位线方向的某个存储单元衬底掺杂不均匀, 可能会形成很深的耗尽区, 从而导致电子在该区域中受电场加速进入浮栅, 进行误编程操作. 由于是漏极电压引起, 也可被称为 drain program disturbance (DPD).

位线擦除扰动 (bit-line erase disturbance): 当某存储单元编程时, 需要在位线方向加高电压. 若某个存储单元栅氧化层有缺陷, 浮栅中的电子可能会穿过栅氧化层进入漏极, 进行误擦除操作, 也可被称为 drain erase disturbance (DED).

过擦除扰动 (over-erase disturbance): 当对某个 Block 进行擦除时, 由于擦除同时进行, 因此擦除慢的单元可能导致擦除快的单元出现过擦除. 过擦除单元浮栅电荷减少, 可能使该单元编程时无法使阈值电压升高到需要电平, 这样该单元无法编程. 同时读取该单元时由于阈值电压减小, 漏电流一直存在, 导致错误读取.

读扰动 (read-disturbance): 对某个单元进行读操作时, 需要在字线方向和位线方向加载电压. 由于读操作电压比变成电压略低, 在多次读取之后可能导致该单元浮栅捕获电荷, 形成误写.

除了上述故障外, 传统的故障同样可能在 FLASH 中产生, 如 stuck-at 故障 (SAF), 即某个存储单元保持 0 或 1 不变, 还有诸如 transition 故障 (TF), state coupling 故障 (CFst) 等^[7].

3 Flash 闪存芯片缺陷验证算法

目前面向 FLASH 缺陷验证的算法主要有以下几种^[2-3]:

FLASH-March 算法, March-FT 算法, Cocktail March 算法. 表 1 所示为几种算法对故障的覆盖率

以及复杂度比较.

表 1 FLASH 测试算法特性

算法	FLASH MARCH	MARCH-FT	Cockmil-March
复杂度—故障 类型	2F+2NP+4N R	2F+2NP+ 6NR	2F+2NP+ 6NR+(2F+2NP +2NR) ^{log(m)}
故障覆 盖率			
WPD	100%	100%	100%
WEN	100%	100%	100%
BPD	100%	100%	100%
BED	100%	100%	100%
RD	100%	100%	100%
OED	100%	100%	100%
SAF	100%	100%	100%
TF	100%	100%	100%
SOF	50%	100%	100%
AF	100%	100%	100%
CF _{st}	750%	100%	100%

算法的复杂度表征算法运行时间,覆盖率表征是否能检测到该故障. 两者综合考虑,本文选择 March-FT 算法. 实际中如果有需求,可以在本系统采用多种算法,只需要对软件代码修正即可.

March-FT 算法简单表述如下^[2]:

{(f); ↑(r1, p0, r0); ↓ ↑(r0); (f); ↓(r1, p0, r0); ↓ ↑(r0)}

其中 f 表示擦除操作, r1 表示读操作,且读出数据应该为 1, p0 表示编程操作, r0 表示读操作,且读出数据应该为 0, ↑ 表示地址递增, ↓ 表示地址递减, ↓ ↑ 表示地址递增或者递减.

下面对算法进行阐述,以扰动故障为例.

(1)对芯片进行擦除操作,为激发故障做准备.

(2)从首地址开始,地址依次递增,进行(r1, p0, r0)操作. r1 表示对当前存储单元读操作,若读出数据为 0,说明有读扰动或者编程扰动. p0 进行编程操作,激励其余位线和字线方向的编程和擦除扰动. r0 进行读操作,若读出数据为 1,说明有过擦除扰动. 这样对整个芯片操作完成之后,理论上应该全部为 0.

(3)对整体进行读操作,若输出为 1,说明有擦除扰动或者过擦除扰动.

(4)将地址递减,重复上述操作. 这样对芯片操作完成后,可以检测所有的扰动故障.

上述为单 bit 输出的 March-FT 算法. 由于目前 FLASH 芯片多为 8 bit 或者 16 bit 输出,在此基础上,本文提出了对 8 bit 和 16 bit 输出的 March-FT 算法,如下:

8 bit 表征如下:

f; ↑(rFF, p00, r00); ↓ ↑(r00);
f; ↓(rFF, p00, r00); ↓ ↑(r00);
f; ↓ ↑(p0F); ↓ ↑(r0F); f; ↓ ↑(pF0); ↓ ↑(rF0);
f; ↓ ↑(p33); ↓ ↑(r33); f; ↓ ↑(pCC); ↓ ↑(rCC);
f; ↓ ↑(p55); ↓ ↑(r55); f; ↓ ↑(pAA); ↓ ↑(rAA);
16 bit 表征如下:
f; ↑(rFFFF, p0000, r0000); ↓ ↑(r0000);
f; ↓(rFFFF, p0000, r0000); ↓ ↑(r0000);
f; ↓ ↑(p00FF); ↓ ↑(r00FF); f; ↓ ↑(pFF00); ↓ ↑(rFF00);
f; ↓ ↑(p0F0F); ↓ ↑(r0F0F); f; ↓ ↑(pF0F0); ↓ ↑(rF0F0);
f; ↓ ↑(p3333); ↓ ↑(r3333); f; ↓ ↑(pCCC); ↓ ↑(rCCCC);
f; ↓ ↑(p5555); ↓ ↑(r5555); f; ↓ ↑(pAAAA); ↓ ↑(rAAAA);

其中 8 bit 的 March-FT 算法流程图表述见图 3.

4 FLASH 闪存验证系统的设计与实现

在本设计中,假设单个存储单元只出现一种缺陷,不存在缺陷重叠的现象. 在实际中若某单元出现缺陷重叠的现象,需要对单独对该单元进行分析.

FLASH 闪存芯片验证算法是基于对芯片的编程、擦除和读操作实现的,在硬件方面需要能够进行上述操作,最终采用的系统架构如图 4 所示.

系统工作如下:(1)进行算法验证时,在 CPU 上运行算法代码,使 CPU 给出相应指令. FLASH 控制器通过总线接受命令和数据,然后控制芯片进行相应操作,当有数据返回时,再次通过总线把数据传输到 CPU 进行验证,给出验证结果. 最终通过 UART 或者 USB 输出结果.(2)进行演示时,将数据通过 USB 传输到缓存,在 CPU 调度下 FLASH 控制器将数据写入闪存芯片. 随后 CPU 通过 FLASH 控制器读出数据,通过总线将数据传输到 DVI,完成演示功能.

4.1 FLASH 闪存控制器的实现

闪存控制器用来实现对 FLASH 的读、编程和擦除操作^[6-7],是算法实现的基础,在 FPGA 中以状态机的形式实现. 对应的状态有 IDEAL、READ、WRITE、ERASE、UNLOCK、READID 几种,自定义的状态转换如图 5 所示.

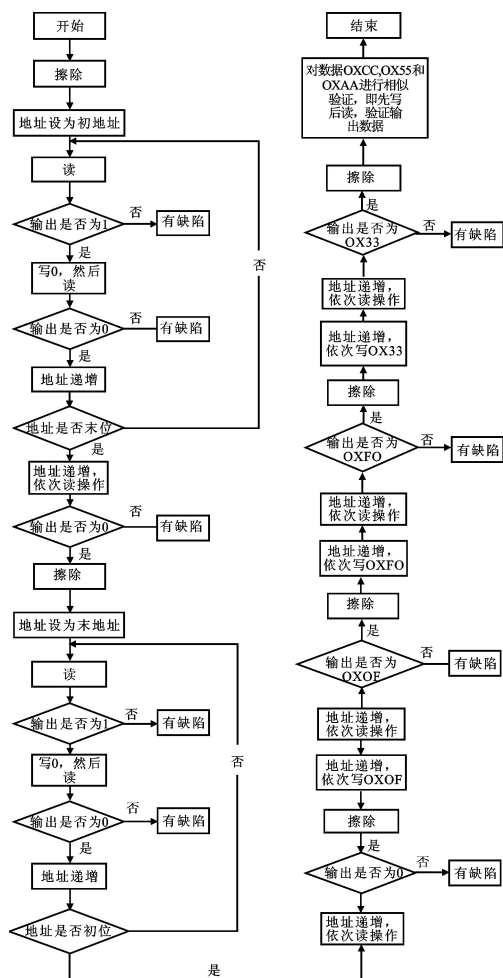


图3 8 bit 的 March-FT 算法流程图

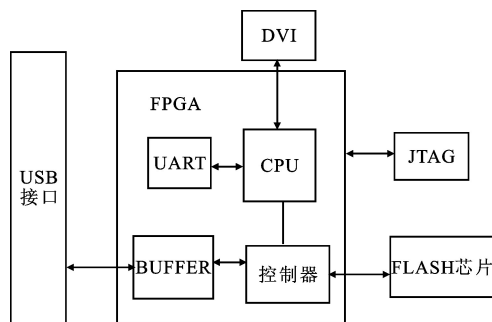


图4 测试系统架构

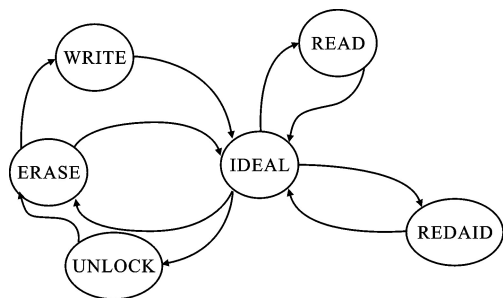


图5 FLASH 控制器状态转换图

初始态为 IDEAL, 由于 NOR FLASH 的特性, 当进行编程操作时, 首先对芯片 UNLOCK, 然后进行块擦除, 这样才能成功编程. 每次上电后进行一次 UNLOCK 即可, 但是每次编程前都需要进行擦除, 然后才能编程, 否则编程操作失败. READID 输出芯片的 ID 信息, 验证操作正确性.

需要注意的是由于需要将 FLASH 控制器挂载在总线上, 因此在 VERILOG 代码中不能直接定义 INOUT 端口, 而应该以三态门的形式定义, 这样最终生成的 INOUT 才能够正常实现输入输出功能, 完成与闪存芯片的数据交换.

4.2 FLASH 控制器的挂载

控制器最终封装为自定义 IP 的形式, 采用图 6 所示的方法将其挂载在 AXI 总线上.

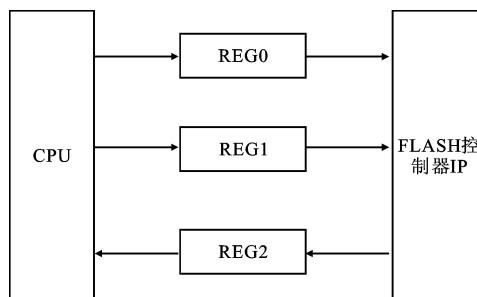


图6 FLASH 控制器的挂载

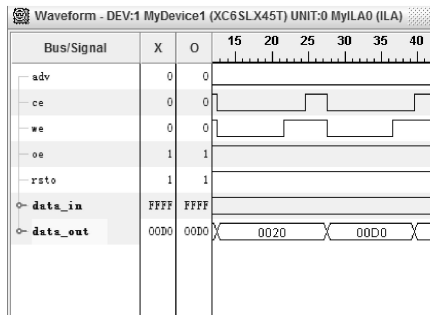
通过寄存器实现 FLASH 控制器的挂载, 其中 REG0、REG1 作为输入寄存器, 位宽 32 bit, 负责将指令、地址和数据传输到 FLASH 控制器; REG2 作为输出寄存器, 同样为 32 bit, 将读出的数据传输到 CPU. CPU 仅面向寄存器进行读写, 方便 CPU 对这一模块调用. FLASH 控制器从 REG0 和 REG1 读取数据作为输入, 将输出数据传输给 REG2, 实现 CPU 与 FLASH 控制器通信.

5 实验结果分析

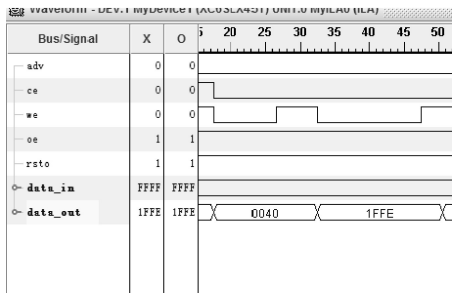
采用 Xilinx 的 Chipscope 工具对 FPGA 结果测试, 能够抓取 FPGA 管脚的信号值并进行显示, 从而真实反应 FPGA 运行状况.

图 7 所示为 FPGA 实现对 FLASH 芯片的擦除、读取和编程操作.

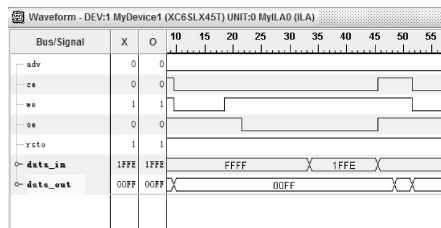
图 7(a)所示为控制器将 0X20 和 0XD0 命令依次按照时序要求输出到 FLASH 芯片, 完成擦除操作. 图 7(b)所示为控制器将命令 0X40 和数据 1FFE 按照时序要求输出到 FLASH 芯片, 完成编程操作. 图 7(c)所示为 FLASH 芯片接受命令 00FF, 然后给出输出结果 1FFE, 完成读操作. 可以看到, 本系统



(a) 擦除操作



(b) 编程操作



(c) 读操作

图7 FPGA 实现的操作

成功完成对 FLASH 的编程、擦除和读操作。

对单个 BLOCK 的测试结果表明系统成功实现 March-FT 算法,完成对 FLASH 的检测,实现预期功能。

表 2 所示为本系统测试结果与 BIST 对比。

表 2 测试结果分析

检测方法	本测试系统	BIST
是否检测能够检测到缺陷	能	能
对缺陷的覆盖率	对缺陷覆盖率为 100%,和采用的算法有关	对缺陷覆盖率为 100%,和采用的算法有关
测试速度	测试速度慢, 128 kB 测试时间为 12.2 s	测试速度快
系统成本	主体为 FPGA, 成本较低	占用硬件资源, 成本较高
灵活性	软件实现, 有很高的灵活性	硬件实现, 确定后不可更改

从表中可以看到,本系统能够实现缺陷检测功能. 与 BIST 相比,本系统在灵活性和成本方面更占

优势,缺点是测试速度较慢,因此在大规模芯片测试方面不适用,适于测试算法研究以及样品检测等。

6 结束语

本文提出了一套完整的 FLASH 芯片测试系统,该系统基于 March-FT 算法. 与 BIST 相比,本设计的优点如下:(1)能够实现不同的算法,适用于多种芯片.(2)实现简单,成本低廉. 不足之处主要是测试速度较慢,可以考虑从 FLASH 控制器实现方面进行改进. 最后对试验结果进行分析,验证了该系统的可行性以及实用性。

参考文献:

- [1] Shibaji Banerjee, Dipanwita Roy Chowdhury. Built-in self-test for flash memory embedded in SoC[C]//Proceedings of the Third IEEE International Workshop on Electronic Design, Test and Applications. Kuala Lumpur: IEEE, 2005: 6-384.
- [2] Yeh Jenchieh, Wu Chifeng. Flash memory built-In self-test using march-like algorithms[C]//Proceedings of the First IEEE International Workshop on Electronic Design, Test and Applications. Christchurch: IEEE, 2002: 137-141.
- [3] Stefano Di Carlo, Michele Fabiang, Roberto Piazza, et al. Exploring modeling and testing of NAND flash memories [C]//Design & Test Symposium (EWDTS), 2010 East-West. St. Petersburg: IEEE, 2010: 47-50.
- [4] Yeh Jenchieh, Cheng Kuoliang. Flash memory testing and built-in self-diagnosis with march-like test algorithms[J]. IEEE Transactions on computer-aided design of integrated circuits and systems, 2007, 26(6): 1101-1113.
- [5] Mohammad Gh Mohammad, Kewal K Saluja. Flash memory disturbances; modeling and test * [C]//VLSI Test Symposium, 19th IEEE Proceedings on. VTS 2001. Marina Del Rey, CA:IEEE, 2001: 218-224.
- [6] 田鹏,聂泽东,张正平,等. 一种兼容 AHB 总线的 Nor Flash 控制器 IP 设计[J]. 微电子学与计算机, 2013, 30(6): 88-91.
- [7] 梁宏明,胡东伟,陈杰. SOC 中 NAND Flash 控制器的一种新型结构和实现方法研究[J]. 微电子学与计机, 2009, 26(6): 89-92.

作者简介:

郭进杰 男,(1988-),硕士研究生.研究方向为数字 IC 设计与 FPGA 验证。