

引用格式:杨丽婷,王琴,倪昊,等. pFlash 故障测试算法[J]. 微电子学与计算机, 2019, 36(9): 55-60. [YANG L T, WANG Q, NI H, et al. Fault test algorithm based on pFlash[J]. Microelectronics & Computer, 2019, 36(9): 55-60.]

pFlash 故障测试算法

杨丽婷¹, 王 琴¹, 倪 昊², 赵子鉴²

(1 上海交通大学 电子信息与电气工程学院, 上海 200240; 2 中芯国际集成电路制造有限公司, 上海 201203)

摘 要: Flash 存储器快速发展得益于便携式设备的进步, 而 pFlash 存储器因编程电压低、功耗小并可以有效抑制带-带隧道效应而被广泛应用. 由于可靠性和成本是衡量所有存储器设备性能的两个关键指标, 本文提出了一种高效的 pFlash 故障测试算法来减小存储器的测试成本并保证其故障覆盖率. 通过改变写入的测试向量以及写入的方式来准确定位 pFlash 存储器的故障. 测试效率较于 March-like 算法提高了 33%, 且故障覆盖率仍然为 100%. 此算法用于开发 pFlash 内建自测试电路(BIST, Built-in Self Test)可以适当减少其硬件开销, 用于测试机台可以减少测试时间.

关键词: pFlash 故障检测算法; 棋盘格式向量; March-like 算法; Flash 故障模型

中图分类号: TN459

文献标识码: A

文章编号: 1000-7180(2019)09-0055-06

Fault test algorithm based on pFlash

YANG Li-ting¹, WANG Qin¹, NI Hao², ZHAO Zi-jian²

(1 College of Electronic and Electrical Engineering, Shanghai Jiaotong University, Shanghai 200240, China;

2 Semiconductor Manufacturing International Corporation, Shanghai 201203, China)

Abstract: Flash memory rapidly development benefits from the progress of portable devices, while pFlash has been widely used due to the low programming voltage, low power consumption and the ability to effectively suppress the band-to-band tunneling effect. Since reliability and cost are two key indicators to measure the performance of all memory and guarantee its fault coverage. The algorithm can locate pFlash fault by changing the test vectors and write mode. Compared with March-like algorithm, the test efficiency is improved by 33% and the fault coverage is still 100%. This algorithm is used to develop pFlash Built-in Self Test circuit (BIST), which can reduce its hardware overhead and reduce test time for the test machine.

Key words: pFlash fault detection algorithm; checkerboard format vector; march-like algorithm; flash fault model

1 引言

随着电子产业的不断发展和功能复杂度的不断提高, 市场对于存储器内存容量的需求越来越大^[1]. 工艺上特征尺寸在革新中不断减小, 从而导致存储单元之间的相互影响不断变大, 那些由于工艺上的缺陷而造成故障出现的概率也变大^[2-3]. 因此, 保证存储器的可靠性以及尽可能降低测试成本是至关重要的. 由于 Flash 存储器的存储单元中浮栅这种特

殊结构的存在, 使其除了可能会出现典型的存储器故障, 还会出现 Flash 特有的故障^[4-6]. 因此, Flash 需要用特定的算法来检测故障. 文献[7]中, 针对 DC-P (Disturbance-programming), DC-E (Disturbance-Erase) 和 DD (drain disturbance) 这三种故障模型, 假设了三种情况, 提出了三种算法. 它们是分别对于 nFlash 只发生三种故障中的一种, 同时发生两种故障, 同时发生三种故障的这三种情况提出了不同的算法, 三种算法整体故障覆盖率并不能达到

100%,且测试算法冗长.文献[8]中,同样是针对DC-P,DC-E和DD三种故障模型提出了算法.该算法巧妙地利用这三种故障模型与耦合故障的共通点,将这三种故障与耦合故障对应起来,从而可以将干扰故障当成耦合故障来激活并检测.此算法步骤较少,从而减少了测试时间,降低了测试成本,但该算法没有覆盖到RD故障.针对基于字的Flash的故障,文献[9]提出March-like算法.但该算法没有考虑与存储单元处在同一字线和同一位线的周围其他存储单元对其造成的累积干扰的影响.本文针对以上测试算法的不足做出系列改进,使得测试算法的长度缩短,测试效率提高.

2 pFlash 故障模型

Flash存储器是基于浮栅结构而存在的,通过给浮栅上注入电子来写0以及将浮栅上电子抽走来写1.在工艺上,浮栅与衬底和浮栅与控制栅之间均是由绝缘体 SiO_2 薄膜来隔离.生成薄膜的厚度很大程度上决定了电子穿过薄膜所需要的电压值大小,而在芯片制造中由于很多因素造成生成 SiO_2 的厚度有时会出现不均匀或者存在偏差的情况.而在衬底与浮栅之间或者浮栅与控制栅之间 SiO_2 薄膜偏薄情况下,某些单元容易发生热电子击穿或隧穿效应,因此干扰也是不可避免的^[4].

图1是pFlash存储单元的结构图.编程时,单元漏端的电子通过热电子注入,从漏端进入到浮栅;擦除时,单元浮栅上的电子通过隧穿效应,从浮栅上进入源端,实现擦除电子的效果.NOR类型pFlash存储器的故障模型包括以下几种:

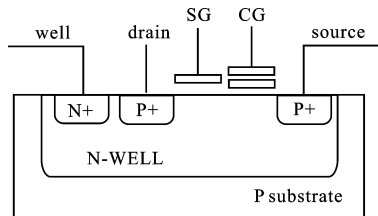


图1 pFlash的存储单元

(1)栅擦干扰(Gate Erase Disturbance, GED),GED是一个单元在编程时,使得另一个跟该单元处于同一条字线上的存储数据为0的单元被擦除.

(2)栅写干扰(Gate Program Disturbances, GPD),GPD是一个单元在编程时,使得另一个跟该单元处于同一条字线上的没有写0的单元被写为0.

(3)漏擦干扰(Drain Erase Disturbance,

DED),DED是一个单元在编程时,使得另一个跟该单元处于同一条位线上的没有写0的单元被写为0.

(4)漏写干扰(Drain Program Disturbance, DPD),DPD是一个单元在编程时,使得另一个跟该单元处于同一条位线上的存储数据为0的单元被擦除.

(5)读干扰(Read Disturbance, RD,也叫Soft-program),在pFlash中RD模型和n型Flash的RD导致的结果是相反的.在pFlash中,RD故障是在读取数据时,被选中的单元从0状态转变为1状态的情况.

(6)过擦(Over Erase, OE)也是一种Flash存储器特有的故障模型.存在过擦故障的存储单元无法正常编程,表现为状态保持为1.

3 March-like pFlash 测试算法

在文献[9]中,March-like算法既测试了Flash特有的干扰故障类型,也覆盖了典型的存储器故障类型.因为目前Flash存储器大多数是面向字的存储器,所以我们这里只研究面向字的算法.假设存储器位宽为16,具体算法实现如下^[8,10]:

$(f); \downarrow (r1111, w0000, r0000); (r0000);$

$(f); \uparrow (r1111, w0000, r0000); (r0000);$

$(f); (w0011); (r0011); (f); (w1100); (r1100);$

$(f); (w0101); (r0101); (f); (w1010); (r1010);$

March-like算法是用来测试n型Flash存储器的故障.在n型Flash中RD故障表现为状态为1的存储单元转化为状态为0的单元.在p型Flash中RD故障表现为状态为0的单元转化为状态为1的单元.原算法中没有两次读入1单元的元素,所以可能会无法检测到RD故障.但是原算法的前两行元素放在pFlash的测试流程中,却恰好可以检测pFlash的RD故障.两次读0,可检测到读0时发生RD故障的存储单元.

同一条位线上的存储单元相互之间的影响是相同的,因为它们都处于相同的电压环境下,所以受到干扰的几率相同.同一条字线上的存储单元也是如此.

图2为 4×4 bit pFlash存储单元.每一个交点为一个存储单元,BL0,BL1,BL2,BL3是四条位线,WL0,WL1,WL2,WL3是四条字线.

存储单元受到与其在同一字线或者位线上的周围其他存储单元编程的影响,从而导致其发生干扰

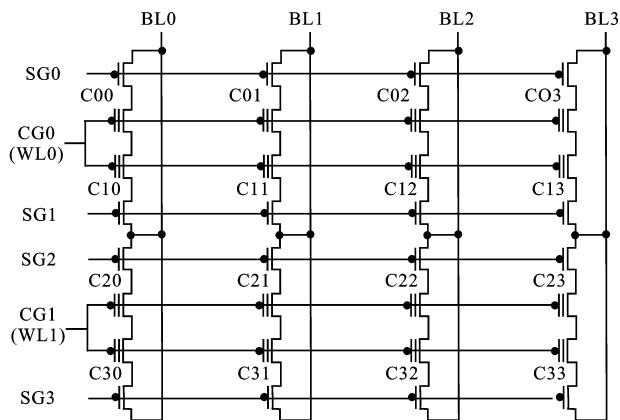


图2 4×4 pFlash 存储单元

故障. 干扰存在两种情况:一种是由于某个存储单元存在极容易受到干扰,导致只要其同一条字线或者同一条位线上有单元编程,它就会发生干扰故障,如图2中4×4bit存储单元阵列中,假设C22就是这一类容易发生干扰故障的存储单元,那么位线BL1或者字线WL1上的其他存储单元编程时,都可能会对C22造成干扰;另一种是由于一些存储单元同时受到周围与其在同一字线和位线的其他多个存储单元干扰效果的叠加,从而编程单元发生状态翻转.

最后导致其数据状态发生翻转,如图2中分别对C12,C21,C23和C32编程,使得C22受到干扰,干扰累积导致C22可能发生状态翻转.以上两种情况均假设C22的数据为1.

根据以上分析,假设将16'h5555和16'hAAAA这两个测试向量交替写入存储器中,即偶数地址写入16'h5555,奇数地址写入16'hAAAA或者偶数地址写入16'hAAAA,奇数地址写入16'h5555,此时这种类似棋盘格的数据分布方式,充分考虑到存储数据为1的单元受到同一字线和同一位线上与其相邻的每一个存储单元对其可能产生的干扰,可以高效激活那些受到周围存储单元编程积累干扰引起的故障.

因此,本文提出棋盘格 March-like 算法,此算法不仅可以使 pFlash 故障覆盖率达到 100%,也缩短了算法的测试长度,降低了其复杂程度.算法具体内容如下(此处假设 pFlash 存储器数据的位宽为 16):

(f); $\downarrow$ (r1111,w0000,r0000);(r0000);
 (f); $\uparrow$ (r1111,w0000,r0000);(r0000);
 (f);(w5555,wAAAA);(r5555,rAAAA);
 (f);(wAAAA,w5555);(rAAAA,r5555);

式中,元素(w5555,wAAAA)是指偶数地址写入向

量 16'h5555,奇数地址写入向量 16'hAAAA;元素(wAAAA,w5555)是指偶数地址写入向量 16'hAAAA,奇数地址写入向量 16'h5555.

4 验证仿真

首先来验证棋盘格 March-like 算法可以覆盖到 SA0,SA1 和 RD 以及 OE 故障.本文验证结果是联合 Flash 的行为级 verilog 模型进行算法仿真的.具体模拟步骤如下:

(1)擦除整个 Flash;(2)随机选择一个地址写入向量 36'hf_ffaf_ffff,这是在模拟 SAF0 故障;(3)读取存储数据来检测 SAF0 故障;(4)随机选择一个地址写入 36'h0_0000_0010 用于模拟故障 SAT1 或者故障 OE;(5)读 0;(6)擦除整个 Flash;(7)随机选择一个地址写入向量 36'h0_0010_0000,这一步是在模拟 RD 故障;(8)读取数据并且遍历所有地址,以检测 RD 故障.其中步骤(3)、(4)、(5)是循环执行的,直到遍历所有地址.步骤(1)、(3)、(4)、(5)、(8)是在模拟算法元素 {f;(r1111,w0000,r0000);(r0000)}.以上模拟过程可以验证算法对 SAF0,SAF1 和 OE 以及 RD 故障的检测.

图3是上述模拟故障仿真所生成的检测 SAF0 故障的波形图,图中信号 Num 表示的数字与上述模拟故障和检测故障的步骤一一对应,Num=1,表示第一步,以此类推.DOUT 表示的是从 Flash 读取的数据.图中显示在 ADDR=16'h0008 时,第 20 位和第 22 位(从 0 位开始计数)发生 SAF0 故障.第三步是在执行算法第二个元素的 r1111.检测到地址 16'h0008 处存在 SAF0 故障.如仿真波形图 3 中 Num=3 时,DOUT=36'hf_ffaf_ffff 说明在地址为 16'h0008 的存储单元检测到 SAF0 故障.

图4是上述模拟故障仿真所生成的检测 SAF1 或者 OE 故障的波形图.Num=4 模拟了故障 SAT1 或者故障 OE 的场景.图中显示在 ADDR=16'h0010 时,在第 0 位和第 4 位发生 SAT1 或者 OE 故障.波形图中显示在执行算法第一行的第二个元素,即 Num=5 时,在地址 16'h0010 处检测到 SAF1 或者 OE 故障.如仿真波形图 4 中 Num=5 时,DOUT=36'h0_0000_0010 说明在地址为 16'h0010 的存储单元检测到 SAF1 或者 OE 故障.

图5是上述模拟故障仿真所生成的检测 RD 或者 DPD 故障的波形图,第七步模拟了故障 RD 或者 DPD 的场景.执行算法的第一行的第三个元素时,检测到地址 16'h000c 处检测到故障.从波形图中可

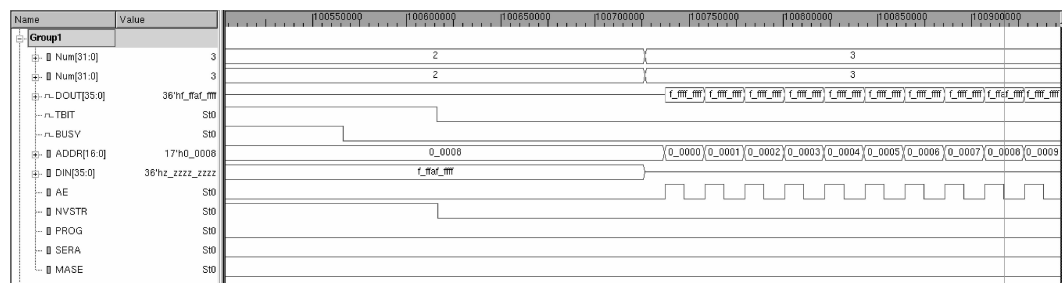


图 3 检测 SAF0 故障波形图

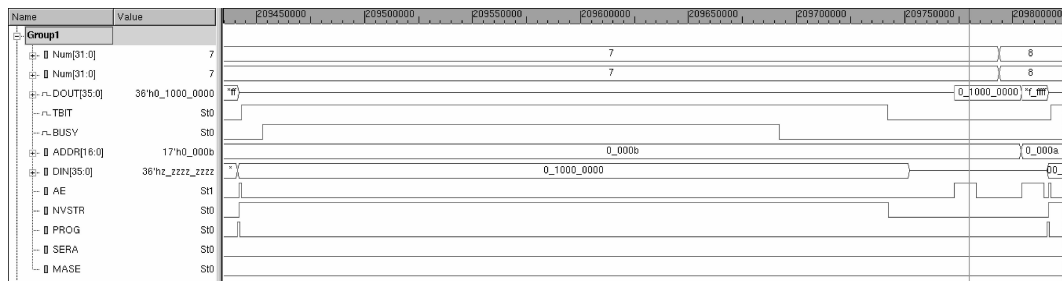


图 4 检测 SAT1 或者 OE 故障波形图

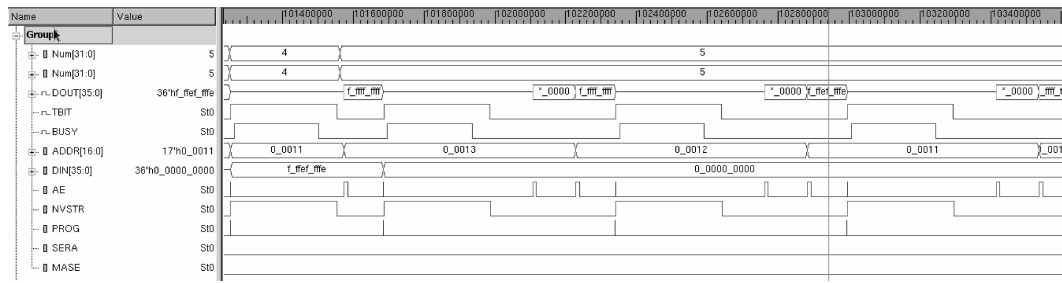


图 5 检测 RD 故障或 DPD 故障波形图

以看出当 ADDR=16'h000c 时,DOUT = 36'h0_0010_0000 说明在地址 16'h000c 处的存储单元检

在文献[9]中,有定理证明,满足以下两个条件的测试算法可检测所有地址解码器故障(AF 故障):

- 条件 1:有元素 $\uparrow(rx, \cdots, wx)$;
 - 条件 2:有元素 $\downarrow(rx, \cdots, wx)$;
- 下面模拟算法中以下元素: f ; $(w5555, wAAAA)$; $(r5555, rAAAA)$;

以上三个元素可以检测到 GED, GDP 和 DED 三种故障. 再加上元素 $\{f; (wAAAA, w5555); (rAAAA, r5555)\}$ 可以确保故障检测可以覆盖到每个存储单元.

首先先来验证 GED 故障,具体验证过程如下:
(1)擦除整个 Flash;(2)读取存储器中每个存储单元的内容,确认此时每个存储单元都没有发生故障;(3)对偶数地址写入测试向量 0101,对奇数地址写入测试向量 1010;(4)再次擦除整个 Flash,为

测到 RD 故障,该地址的存储单元中第 28 位(从 0 位开始计数)发生 RD 或者 DPD 故障.

了模拟 GED 故障做准备;(5)在模拟 GED 故障,用系统函数产生一个随机数作为地址,如果为偶数地址,则写入向量 1101,如果为奇数地址,则写入向量 1011(向量 1101 表示该地址对应的存储单元的第 3 位发生 GED 故障,向量 1011 表示改地址对应的存储单元的第 1 位发生 GED 故障,从 0 开始计数);(6)读取每个存储单元. 其中步骤(1)、(3)、(6)是在模拟棋盘格 March-like 测试算法中的 $\{f; (w0101, w1010); (r0101, w1010)\}$ 元素.

图 6 中是模拟验证过程的第五步所生成的波形图,图中显示系统函数生成了 16'h0008 这个地址,在这个地址的存储单元中写入向量 36'h5_5d55_5555. 图 7 中是在验证过程的第六步所生成波形图,检测出地址为 16'h0008 的存储单元发生 GED 故障.

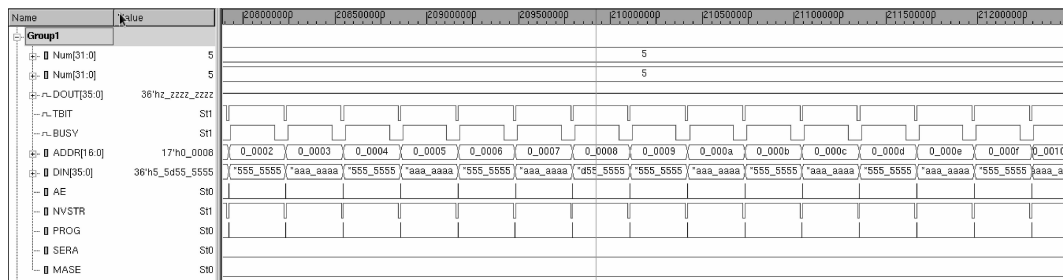


图 6 检测 GED 故障波形图(1)

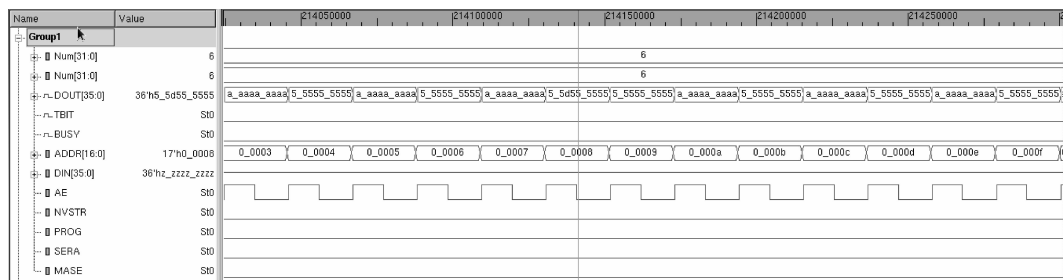


图 7 检测 GED 故障波形图(2)

接着验证 DED 故障,具体验证步骤如下:

(1)擦除整个 Flash;(2)读取存储器中每个存储单元的内容,确认此时每个存储单元都没有发生故障;(3)对偶数地址写入测试向量 0101,对奇数地址写入测试向量 1010;(4)再次擦除整个 Flash,为了模拟 DED 故障做准备;(5)在模拟 DED 故障,用系统函数产生一个随机数作为地址,如果为偶数地址,则写入向量 0001,如果为奇数地址,则写入向量 1000(向量 0001 表示该地址对应的存储单元的第 2 位发生 DED 故障,向量 1000 表示该地址对应的存储单元的第 1 位发生 DED 故障);(6)读取每个存储单元. 步骤(1)、(3)、(6)共同完成棋盘格 March-like 故障测试算法中的 $\{f;(\omega 0101, \omega 1010);(r 0101, \omega 1010);\}$ 这一元素的模拟.

图 8 中是模拟验证过程的第五步所生成的波形图,系统函数生成了 16'h0008 这个地址,在这个地址的存储单元中写入向量 36'h5_5155_5555. 图 9 中是在验证过程的第六步所生成的波形图,检测出地址为 16'h0008 的存储单元发生 DED 故障.

最后,由于 GPD 故障表现形式和 DED 故障表现形式一样,因此此处不再重复验证.

根据以上仿真验证结果,从而得到故障覆盖率的统计如表 1.

表 1 新算法的故障覆盖率统计

故障类型	G	G	D	D	S	S	C
	E	P	E	P	R	O	A
	D	D	D	D	D	E	F
覆盖率	100%	100%	100%	100%	100%	100%	100%

表 2 是一个 55 nm 的 pFlash 的三个 IP 的测试数据,检测新算法效率提高的百分比. 其中包括三个型号的 Flash IP,整片擦除所用的时间大约为 10 ms,存储量不同,整片擦除所用的时间会有一些偏差,但是均小于 10 ms 的,而且擦除时间的数量级比编程所用的时间小两个数量级,对整体测试时间的影响不大,所以此处用 10 ms 来表示这三个 IP 的擦除时间. 每个地址写 0 需要消耗的时间为 30 μ s,读取每个地址的存储数据需要消耗的时间为 30 ns. March-like pFlash 的测试长度是 $4(F) + 4n(P) + 8n(R)$, March-like 算法的测试长度是 $6(F) + 6n(P) + 10n(R)$. F 表示一次整片擦除操作, P 和 R 分别表示一次编程和一次读取操作, n 表示 Flash 的字数. 从而可得到表中数据.

表 2 三个 pFlash IP 测试时间对比

IP 型号	March-like 算法	新算法	效率提升
32KX32	5 829.60 ms	3 887.68 ms	33.31%
64KX32	11 599.20 ms	7 735.36 ms	33.31%
78KX128	14 123.40 ms	9 418.72 ms	33.31%

根据统计的三个 pFlash IP 测试数据,可以得到测试效率提高 33.31%. 但是不同的存储大小的 FlashIP,效率提升的效果有微小偏差,所以该算法效率基本提升了 33%.

5 结束语

通过对已有的 March-like 算法的调整和改进,本文提出棋盘格 March-Like 故障测试算法. 它减少

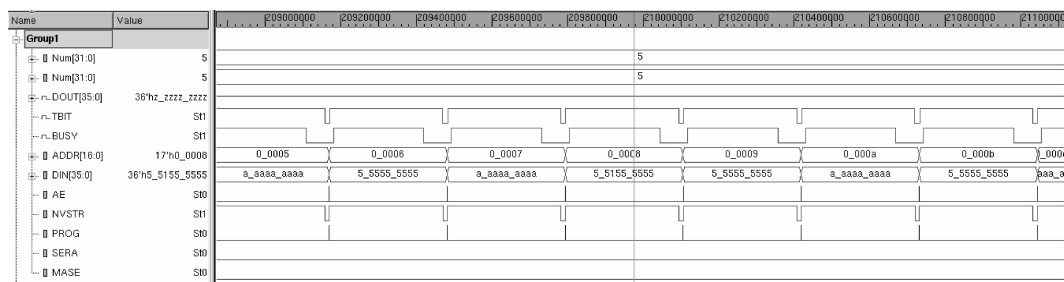


图 8 检测 DED 故障波形图(1)

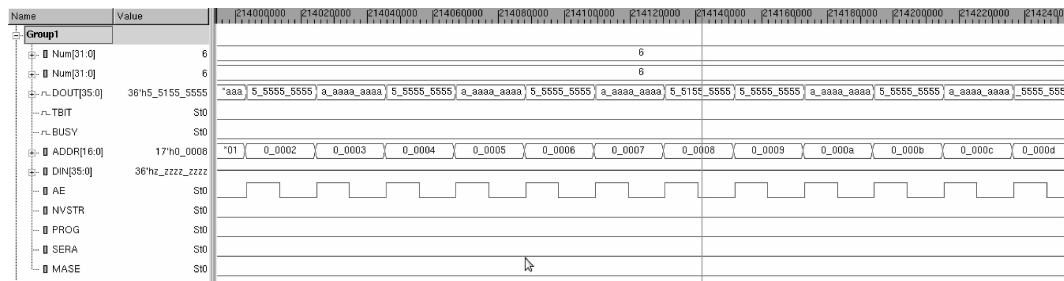


图 9 检测 DED 故障波形图(2)

了测试步骤,降低了测试的复杂程度,从而减少了测试时间,测试效率提升大约 33%。棋盘格 March-like 算法的长度相对于 March-like 测试算法的长度减少了 $2(F)+2n(P)+2n(R)$ 。测试步骤的减少可以使得应用该算法的 Flash BIST 测试电路在 testchip 上所占的面积减少,从而可以降低因为可测性设计而带来的成本,也可以减少应用该算法对 pFlash 进行测试的测试机台的使用时间,降低测试成本。此外,该算法还可以用于检测 SRAM 等易失性存储器中出现的常见故障。该算法在保证故障覆盖率的同时,还有效提高了测试效率。

参考文献:

[1] LAWTON G. Improved flash memory grows in popularity [J]. Computer, 2006,39(1):16-18.

[2] LARCHER L, PAVAN P, MAURELLI A. Flash memories for SOC:An overview on system constraints and technology[C]//Proc. IEEE Int. Workshop System-on-Chip Real-Time Appl., Banff, AB, Canada, 2005: 73-77.

[3] 鉴海防,王占和,李印增,等. SoC 嵌入式 Flash 存储器的内建自测试设计[J]. 微电子与计算机, 2005, 22(4):87-91.

[4] 潘中良,系统芯片 SOC 的设计与测试[M]. 北京:科学出版社,2009:250-251.

[5] IEEE. IEEE 1005 standard definitions and characterization of floating gate semiconductor arrays [C]//IEEE Standards Department, Piscataway, 1999.

[6] PIERPAOLO NICOSIA, FULVIO NAVA. Test strategies on non volatile memories electrical wafer sort on NAND, nor flash and phase change memories[C]// 22nd IEEE Non-Volatile Semiconductor Memory Workshop, Monterey, CA, USA,2007: 11-18.

[7] MOHAMMAD M, SALUJA K, YAP A. Testing flash memories[C]// 13th international Conference on VLSI Design. Calcutta, India,2000: 406-411.

[8] MOHAMMAD G MOHAMMAD, KEWAL K SALUJA. Flash memory disturbances: modeling and test [C]//Proceedings 19th IEEE VLSI Test Symposium. VTS 2001. Marina Del Rey, CA, USA ,2001.

[9] JEN-CHIEH YEH, KUO-LIANG CHENG, YUNG-FA CHOU, et al. Flash memory built-in self-test using March-like algorithms [J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2007(26): 1101-1113.

[10] VAN DE GOOR A J. Testing semiconductor memories: theory and practice [M]. Chichester, UK: John Wiley& Sons, Inc,1991.

作者简介:

杨丽婷 女,(1992-),硕士研究生.研究方向为系统级集成电路设计. E-mail:ylt_819@sjtu.edu.cn.

王 琴 女,(1975-),博士,副研究员.研究方向为系统级集成电路设计、SOC 设计等.

倪 昊 男,(1975-),博士.研究方向为 NVM 架构设计.

赵子鉴 男,(1986-),博士,工程师.研究方向为数字集成电路设计.