基于哈希表的高效存储器内建自修复方法

郭旭峰,于 芳,刘忠立
(中国科学院微电子研究所,北京 100029)

摘 要: 现有存储器内建自修复方法要么遍历式地址比较效率低,要么并行地址比较功耗高,都不适用于大故障数存储器.对此,本文提出一种高效的存储器内建自修复方法,该方法对占故障主体的单元故障地址以哈希表形式进行存储,以利用哈希表的快速搜索特性提升地址比较效率.本文方法修复后的存储器在1个时钟周期内即可完成地址比较,修复后存储器性能不受任何影响,与目前广泛采用的基于CAM的方法处于同一水平,但功耗方面却具有明显优势.计算机模拟实验表明,对于512×512×8bits的存储器在同等冗余开销的情况下本文方法修复率相对于ESP方法平均提高了32.25%.

关键词: 内建自修复;哈希表;内建冗余分析;内建自测试
中图分类号: TP333 **文献标识码:** A **文章编号:** 0372-2112 (2013)07-1371-07
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2013.07.020

An Efficient Memory Built-in Self-Repair Method Based on Hash Table

GUO Xu-feng, YU Fang, LIU Zhong-li
(*Institute of Microelectronics of Chinese Academy of Sciences, Beijing* 100029, *China*)

Abstract: Existing built-in self-repair (BISR) methods are either inefficient using traversal address compare or high power consumption using parallel address compare. Neither is suitable for memories with large fault quantity. In order to solve this issue, an efficient memory BISR method based on hash table is proposed. In the proposed method cell fault addresses which contribute the main part of memory faults were stored in the form of a hash table. With quick search feature of hash table the proposed method improves address compare efficiency radically. Under work mode address compare process can be finished within 1 clock cycle, and this will ensure repaired memory against additional performance penalty. The proposed method achieves the same address compare efficiency to CAM based method with much lower power consumption. Experimental results show that for 512×512×8 bits memory the proposed method achieves 32.25% repair rate increment on average to ESP method with the same redundancy overhead.

Key words: built-in self-repair; hash table; built-in redundancy analysis; built-in self-test

1 引言

随着集成电路工艺尺寸不断减小,存储器的规模越来越大,密度越来越高,存储器故障数量也不可避免的随之增加,严重影响了存储器成品率.为了保证存储器的成品率,存储器内建自修复(BISR,Built-In Self-Repair)技术得到了快速发展,凭借其低成本,高修复率和灵活的在线修复特性,BISR技术目前已成为提高存储器成品率必不可少的重要技术手段.

存储器BISR技术的核心思想是采用预置的冗余存储单元替代检测到的故障存储单元,从而达到修复目的.近二十年来,研究者们对存储器BISR技术进行了广泛的研究,提出了众多的存储器BISR方法.较早期的文献提出了基于冗余列的修复方法[1],其修复能力很有限.在此基础上发展出将冗余行/列分段的修复方法[2,3],细化了冗余资源的颗粒度,在一定程度上提升了修复率.部分研究提出了行列二维冗余修复方法[4,5],使得修复率和冗余资源利用率都得到了明显提高,但是其冗余分析算法复杂,硬件实现占用面积较大.有些方法采用联合不同类型冗余资源的策略来提高修复率[6].还有一类修复方法采用冗余字作为冗余资源[7],此类方法冗余资源颗粒度小,对单元故障具有较高的修复率,同时冗余分析简单易实现,其不足之处是不能实现行/列故障的修复.

收稿日期:2012-05-04;修回日期:2013-02-27;责任编辑:孙瑶
基金项目:"核高基"国家重大科技专项(No.Y1GZ212002)

现有存储器 BISR 方法修复后的存储器工作过程中需要将访问地址与故障地址列表中的故障地址依次比较,以判断访问地址是否为故障地址.这种遍历式地址比较效率低,故障数较多时这种不足尤其明显.对此有文献提出了基于地址分割的自修复方法来降低地址比较次数[8],但仍然无法达到较理想的效果.目前实用的修复方法多采用内容可寻址存储器(CAM, Content Addressable Memory)保存故障地址[9],利用 CAM 并行地址比较来保证工作效率,但故障数较多时 CAM 并行比较的功耗较高,限制了这种方法在低功耗领域的应用.

针对以上情况,本文提出一种高效的存储器 BISR 方法,该方法采用字冗余策略,冗余资源分成若干组,其中一组用于单元故障修复,其余每组既可用于行故障修复也可用于列故障修复,这使得冗余分析变得十分简单;与现有方法不同,本文方法单元故障地址基于哈希表形式存储,以利用哈希表的快速搜索特性提升地址比较效率.本文方法保留了字冗余策略修复率高,冗余分析简单的优点,增加了对行/列故障的修复能力,同时与传统方法相比地址比较效率得到大幅提升.

2 基于哈希表的内建自修复方法

采用字冗余策略的完整存储器 BISR 电路包括内建自测试(BIST, Built-In Self-Test)模块和内建冗余分析(BIRA, Built-In Redundancy Analysis)模块.BIST 模块的功能是对存储器进行测试并向 BIRA 模块输出故障地址.BIRA 模块的功能是判断访问地址是否为故障地址,并为故障地址分配冗余资源,进行故障地址重映射,从而完成存储器的自修复.图 1 是本文提出的存储器 BISR 方法的总体结构框图.虚线框标注的 BIRA 模块是本文方法的核心部分,包括哈希函数逻辑,单元故障哈希表,行/列故障列表,地址比较逻辑,多组冗余存储及数据选通逻辑.本文方法的核心思想是:单元故障地址基于哈希表形式存储,工作过程中根据访问地址直接计算出比较对象在哈希表中的存储地址,并将比较对象读出送入地址比较逻辑,同时送入地址比较逻辑的还有各个行/列故障地址和访问地址,经比较如果访问地址无故障,则各冗余组被置为无效状态,同时选通主存储区数据通道;如果访问地址为故障地址,则使相应的冗余组有效,同时选通该冗余组的数据通道,从而完成对该故障地址的修复.下文分节详细介绍冗余资源的组织,故障地址的存储,哈希函数设计以及故障地址重映射机制.

2.1 冗余资源的组织

本文方法采用细颗粒度的字冗余策略.冗余资源分成若干组,设存储器地址线宽为 $w = r + c$,其中 r 为行地址线宽,c 为列地址线宽,存储器大小为 $N = 2^w$.令 $m = \max\{r, c\}$,则取每个冗余组的大小为 $S = 2^m$.按照此原则设定冗余组大小可使每个冗余组既可用于行故障修复也可用于列故障修复.

2.2 故障地址的存储

存储器 BISR 技术需要对 BIST 模块检测出的故障地址进行存储,以便工作过程中访问地址与之进行比较.对于占故障比例较小的行/列故障,本文方法采用常规方式将其存储在行/列故障列表中.如图 1 所示,列表中每个存储项包括使能标记,行/列标记和行/列故障地址.其中行/列标记为 1 时表示存储的是行故障地址,行/列标记为 0 时表示存储的是列故障地址.对于占故障主体的单元故障,本文提出以哈希表的形式进行存储.哈希表存储的特征在于数据在表中的存储地址与数据内容之间存在特定的映射关系,称之为哈希函

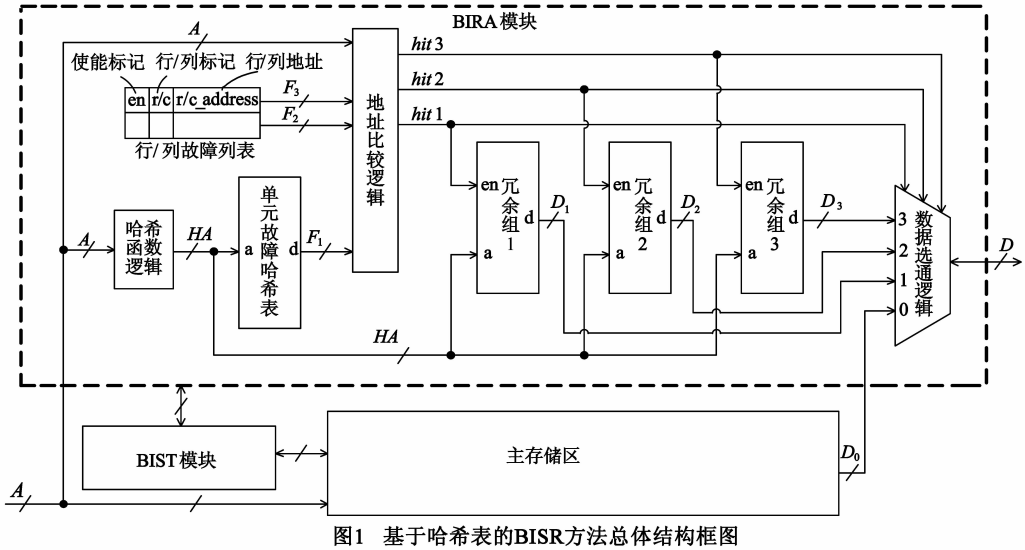

图1 基于哈希表的BISR方法总体结构框图

数,表示为 $h = H(k)$,其中 k 为数据内容,h 称为 k 的哈希地址,即数据 k 在哈希表中的存储地址.哈希表存储可以快速定位搜索对象在表中的位置,无需对列表进行耗时的遍历.以哈希表形式存储单元故障地址,经修复的存储器可根据访问地址的哈希地址,快速从哈希表中读取比较对象并送入地址比较逻辑.在一个时钟周期内即可完成地址比较,从根本上避免了传统方法中耗时的遍历式地址比较,大大提高了 BIRA 模块的工作效率,且该特性不受故障数多少的影响.

2.3　哈希函数设计

哈希函数是哈希表存储的关键,为了满足运算快速,结构简单易实现的要求,本文方法哈希函数逻辑以下列哈希函数为基础:

$$HA = H(A) = A_r \hat{} A_c$$

函数中 A 表示主存储区中的存储单元地址,A_r 和 A_c 分别为 A 的行地址和列地址部分,HA 为 A 的哈希地址.符号 ^ 为按位异或运算符,操作数 A_r 和 A_c 低位对齐,长度较短的操作数高位用"0"补齐.由 2.1 节所述可知 HA 的位宽为 $m = \max\{r, c\}$,故取哈希表的大小为 $N_h = 2^m$.基本哈希函数只需 m 个并行的异或门即可实现,结构简单,运算快速.图 2 所示为基本哈希函数逻辑的示意图,图中 A_c 为较短的操作数,高位补"0"后与 A_r 进行按位异或运算.

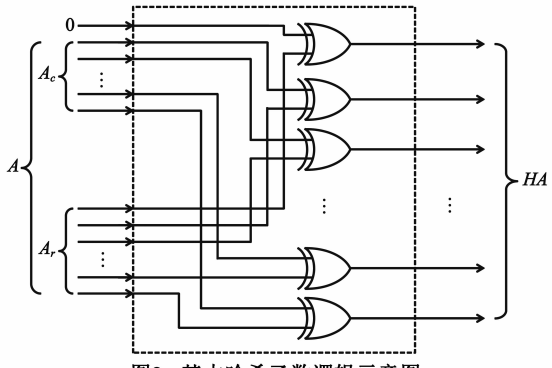

图2　基本哈希函数逻辑示意图

由上述内容可知哈希表存储的实质是将较大的主存储区地址空间映射到较小的哈希地址空间,必然存在某两个存储地址拥有相同的哈希地址,如果这两个存储地址都是故障地址,那么哈希表无法用一个哈希地址同时保存这两个故障地址,这样的情况称之为哈希冲突(下文简称冲突),在发生冲突的情况下存储器是不可修复的.

为了降低冲突概率保证本文方法的修复率,本文方法采用多哈希函数的设计,依据是某个故障分布在多个哈希函数下均发生冲突的概率要远低于单哈希函数下的冲突概率.修复过程中只需要选通其中一个无冲突的哈希函数作为最终的哈希函数就可以实现存储器的修复.

多哈希函数的设计以上述基本哈希函数为基础,将位数较长的操作数进行循环移位即可得到一个新的哈希函数,根据移动位数的不同就可以得到多个不同的哈希函数.图 3 所示为多哈希函数的示例,图(a)表示基本哈希函数,图(b)和图(c)分别表示循环左移 1 位和循环左移 2 位形成的新哈希函数.如图 3 所示,多哈希函数不需要对基本哈希函数电路结构做任何更改,只需适当地改变输入的连接关系.

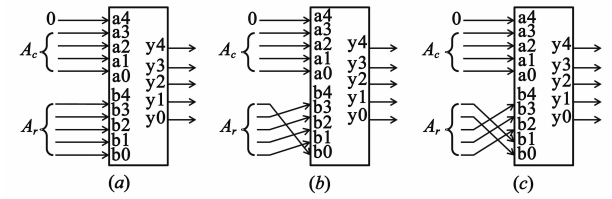

图3　多哈希函数设计示意图

上述多个哈希函数的共同特征是:对于主存储区同一行的存储单元,它们的行地址相同,而列地址不同,经按位异或运算得出的哈希地址必然是不等的,即无冲突的;同理,同一列存储单元的哈希地址也是无冲突的,这将为故障地址重映射机制的设计提供便利.

2.4　故障地址重映射

当某个访问地址经比较确定为故障地址后,需要将该故障地址重映射至冗余存储区.由 2.1 节和 2.3 节可知每个冗余组的大小与哈希表的大小相等,即 $S = N_h = 2^m$,且同一行/列存储单元的哈希地址无冲突,在此基础上令每个冗余组共享哈希地址作为重映射地址,如图 1 中所示.这样的地址重映射机制一方面免去了保存重映射地址所需的映射表,减小了电路面积;另一方面在地址比较的过程中同时获得了重映射地址,省去了传统方法读取映射表获取重映射地址的过程,明显提高了 BIRA 模块的运行速度.

3　工作流程

BISR 电路有两种工作模式:测试/修复模式和正常工作模式,下面分节详细介绍本文方法的测试/修复流程和正常工作流程,并用实例加以说明.

3.1　测试/修复流程

①存储器上电后首先启动 BIST 模块对存储器进行测试,将检测到的故障地址暂存在 BIST 模块中;

②将行/列故障地址依次存入行/列故障列表,如果行/列故障数超出列表大小则 BISR 电路给出不可修复信号;

③选通第一个哈希函数;

④将哈希表初始化为空(初始数据的哈希地址与

该数据实际所在地址不等);

⑤将故障地址依次存入哈希表,如果存入过程中无冲突发生,则测试/修复完成,存储器进入正常工作模式;如果存入过程中发生冲突,则选通下一个哈希函数并循环执行④⑤两步;

⑥如果所有哈希函数下均发生冲突,该存储器不可修复,BISR 电路给出不可修复信号.

3.2 正常工作流程

①如图 1 所示,访问地址 A 经哈希函数逻辑计算出哈希地址 HA,读取哈希表中地址 HA 处的数据 F_1;

②访问地址 A 和 F_1 以及各行/列故障地址并行送入地址比较逻辑,比较结果是与冗余组一一对应的多个 *hit* 信号,作为各冗余组的使能信号和数据选通信号;

③如果全部 *hit* 信号均为 0,说明访问地址 A 无故障,则各冗余组被置为无效状态,同时选通主存储区与 IO 间的数据通道;

④如果有某个 *hit* 信号为 1,说明访问地址 A 为故障地址,则与该 *hit* 信号对应的冗余组被置为有效状态,HA 作为故障地址 A 在冗余组的重映射地址,同时选通该有效冗余组与 IO 间的数据通道,故障地址得以修复.

3.3 BISR 工作流程实例

为了进一步明确本文方法的工作流程,本节用一个简单的实例进行说明.图 4(a)为一个 8×4 的存储器示意图,其地址表示为 $C_1C_0R_2R_1R_0$,其中 $R_2R_1R_0$ 表示行地址,C_1C_0 表示列地址.该存储器存在行故障 f_r = 101,列故障 f_c = 01,以及单元故障 f_{s1} = 00001,f_{s2} = 11010.

测试/修复流程如下:

①存储器上电后首先启动 BIST 模块进行测试,将检测到的故障地址 f_r,f_c,f_{s1}和 f_{s2}暂存在 BIST 模块中;

②将行/列故障地址 f_r,f_c 存入行/列故障地址列表中,结果如图 4(b)所示,行/列标记显示第一项保存的是行故障 f_r,第二项保存的是列故障 f_c;

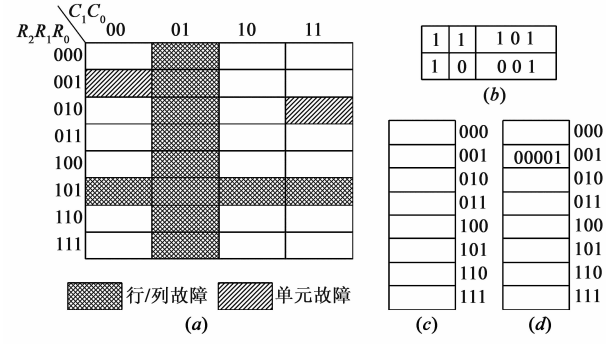

图4 示例存储器故障分布及故障地址保存示意图

③选通哈希函数 $HA = H_1(A) = R_2R_1R_0\hat{}\ 0C_1C_0$,将哈希表初始化为空,如图 4($c$)所示;

④计算故障地址 f_{s1}的哈希地址 h_1 = 001 ^ 000 = 001,经检查哈希表的 001 地址为空,将 f_{s1}存入哈希表地址 001 处,如图 4(d)所示;

⑤计算故障地址 f_{s2}的哈希地址 h_2 = 010 ^ 011 = 001,经检查哈希表 001 地址已经存入故障地址 f_{s1},说明在哈希函数 $HA = H_1(A)$下发生冲突;

⑥选通哈希函数 $HA = H_2(A) = R_1R_0R_2\hat{}\ 0C_1C_0$,将哈希表初始化为空,如图 5($a$)所示;

⑦计算故障地址 f_{s1}的哈希地址 h_1 = 010 ^ 000 = 010,经检查哈希表 010 地址为空,将 f_{s1}存入哈希表地址 010 处,如图 5(b)所示;

⑧计算故障地址 f_{s2}的哈希地址 h_2 = 100 ^ 011 = 111,经检查哈希表的 111 地址为空,将 f_{s2}存入哈希表地址 111 处,如图 5(c)所示;

⑨单元故障存储完毕,无冲突发生,选通 $HA = H_2(A)$作为最终哈希函数,测试/修复模式完毕,存储器进入正常工作模式.

正常工作模式下,A 为访问地址,R,C 分别为 A 的行地址与列地址,假设访问地址依次为 10100,11010,01101.工作流程分别如下:

当 A = 10100 时:

①计算 A = 10100 的哈希地址 h = 001 ^ 010 = 011,读取哈希表地址 011 处的数据 F_1,如图 5(c)所示哈希表地址 011 处为空,根据前述哈希表初始化的原则可知 $H_2(F_1)\neq H_2(A)$,那么必然有 $F_1\neq A$,又有 $R\neq f_r$,$C\neq f_c$,故图 1 中所有 *hit* 信号均为 0;

②比较结果表明地址 A 无故障,则图 1 中各冗余组被置为无效状态,选通主存储区与 IO 间的数据通道.

当 A = 11010 时:

①计算 A = 11010 的哈希地址 h = 100 ^ 011 = 111,读取哈希表地址 111 处的数据 F_1,如图 5(c)所示,F_1 = 11010,即 $F_1 = A$,又有 $R\neq f_r$,$C\neq f_c$,故图 1 中 $hit1 = 1$,其余 *hit* 信号为 0;

②$hit1 = 1$ 说明地址 A 为单元故障,则图 1 中的冗

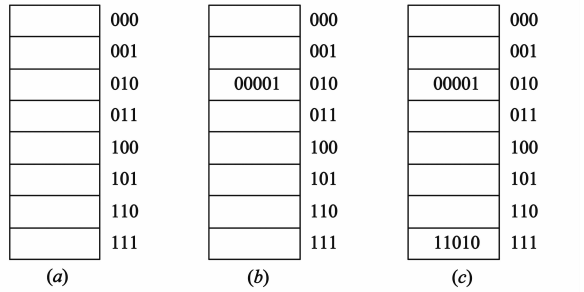

图5 单元故障地址在哈希表中的保存示意图

余组1被信号 $hit1$ 置为有效状态,哈希地址 $h=111$ 作为 A 在冗余组 1 的重映射地址,同时选通冗余组 1 与 IO 间的数据通道,故障地址 A 得到修复.

当 $A=01101$ 时:

①计算 $A=01101$ 的哈希地址 $h=011\hat{}\ 001=010$,读取哈希表地址 010 处的数据 F_1,如图 5(c)所示,$F_1=00001$,显然 $F_1\neq A$,又有 $R=f_r$,$C=f_c$,本文方法中位于故障行和故障列交点的存储单元优先归为行故障,故图 1 中 $hit3=1$,其余 hit 信号为 0;

②$hit3=1$ 说明地址 A 为行故障,则图 1 中的冗余组 3 被信号 $hit3$ 置为有效状态,哈希地址 $h=010$ 作为 A 在冗余组 3 的重映射地址,同时选通冗余组 3 与 IO 间的数据通道,故障地址 A 得到修复.

4　实验结果与分析

为了评估本文方法的修复率,冗余资源开销和工作效率,我们进行了计算机模拟实验.由于单个存储器的故障数近似于泊松分布,而故障的物理分布近似于均匀分布[10,11],实验中我们以泊松分布模型生成故障数,并以均匀分布模型向存储器随机注入故障地址.如图 6 所示,针对不同规模的存储器泊松分布的参数将有所不同.

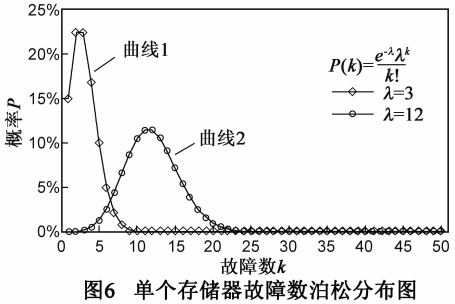

图6　单个存储器故障数泊松分布图

我们将本文方法与二维冗余修复方法中优秀的 ESP(Essential Spare Pivoting)方法进行了比较[12].ESP 方法的思想是:当某行或某列上的单元故障数达到一定的阈值时,则相应的使用行或列冗余进行修复,ESP 方法的修复率接近二维冗余方法的最优解,且在二维冗余修复方法中其面积开销控制的较为理想.

表 1 列出的是第一组实验结果,实验选用的存储器规模为 $N=512\times512\times8$bit,故障数分布如图 6 中曲线 1 所示(平均故障数 $\lambda=3$,最大故障数为 50),注入的故障中行故障占 10%,列故障占 10%,单元故障占 80%,每种冗余配置情况下存储器样本数均为 50000.表中(N_{sr},N_{sc})与 R_1 分别表示 ESP 方法冗余行,列数量和修复率,N_s 与 R_2 分别表示本文方法的冗余组数量和修复率.实验结果显示在同等冗余开销的情况下,本文方法修复率相对于 ESP 方法修复率获得了平均 32.52% 的提升.尤其在冗余量较小的情况下修复率提升较为明显,在某些情况下获得了最高 63.29% 的提升.

表 1　本文方法与 ESP 方法实验结果比较

($N=512\times512\times8$bit,$\lambda=3$)

ESP		本文方法		$\Delta R=R_2-R_1$	$N_s/(N_{sr}+N_{sc})$
(N_{sr},N_{sc})	R_1	N_s	R_2		
(0,1)	14.10%	1	51.70%	+37.60%	100%
(1,0)	14.15%			+37.55%	100%
(1,1)	39.44%	2	96.57%	+57.13%	100%
(0,2)	33.28%			+63.29%	100%
(1,2)	63.29%	3	99.99%	+36.70%	100%
(2,1)	63.14%			+36.85%	100%
(1,3)	80.94%	4	100.00%	+19.06%	100%
(2,2)	81.04%			+19.96%	100%
(2,3)	91.52%	5	100.00%	+8.48%	100%
(3,2)	91.40%			+8.60%	100%
平均值				+32.52%	100%

表 2 列出的是第二组实验结果,与第一组实验不同的是:实验选用的存储器规模为 $N=1024\times1024\times32$bit,故障数分布如图 6 中曲线 2 所示(平均故障数 $\lambda=12$,最大故障数为 50),意在检验本文方法对大故障数存储器的修复能力.实验结果显示在修复率略高于 ESP 方法(平均 5.34%)的情况下,本文方法的冗余开销平均仅为 ESP 方法的 26.9%.在保证 100.00% 修复率的情况下本文方法冗余开销仅为 ESP 方法的 25%.

表 2　本文方法与 ESP 方法实验结果比较

($N=1024\times1024\times32$bit,$\lambda=12$)

ESP		本文方法		$\Delta R=R_2-R_1$	$N_s/(N_{sr}+N_{sc})$
(N_{sr},N_{sc})	R_1	N_s	R_2		
(5,5)	35.79%	3	64.84%	+29.05%	33.3%
(6,6)	59.32%			+5.52%	25.0%
(7,7)	78.86%	4	92.59%	+13.73%	28.6%
(8,8)	90.86%			+1.73%	25.0%
(9,9)	96.76%	5	99.47%	+2.71%	27.8%
(10,10)	99.04%			+0.43%	25.0%
(11,11)	99.77%	6	99.99%	+0.22%	27.3%
(12,12)	99.95%			+0.04%	25.0%
(13,13)	99.99%	7	100.00%	+0.01%	26.9%
(14,14)	100.00%			+0.00%	25.0%
平均值				+5.34%	26.9%

存储器 BISR 方法的地址比较效率决定了修复后存储器的性能.我们通过实验将本文方法与现有两种旨

在提升工作效率的 BISR 方法进行了对比,对比项目包括不同工作模式下地址比较所需的时钟周期数及地址比较次数(并行比较视为多次比较),实验存储器的大小为 $N = 1024 \times 128 \times 8$bit,分别注入固定的故障数 10,20,50 进行对比实验.表 3 的实验结果表明:在测试/修复模式下,本文方法的工作效率(时钟周期数)明显优于基于地址分割的方法,略低于基于 CAM 的方法.在正常工作模式下,本文方法地址比较仅需 1 个时钟周期,明显优于基于地址分割的方法,与目前广泛采用的基于 CAM 的方法处于同一水平,但地址比较次数远低于基于 CAM 的方法,这使得本文方法在功耗方面相对基于 CAM 的方法具有明显优势,这主要得益于单元故障的哈希表存储形式.

表 3 本文方法与现有方法地址比较时钟周期数及比较次数对比

故障数	工作模式	本文提出的 BISR 方法		基于地址分割的 BISR 方法		基于 CAM 的 BISR 方法	
		时钟周期数	比较次数	时钟周期数	比较次数	时钟周期数	比较次数
$k = 10$	测试/修复模式	9	9	36	36	9	45
	正常工作模式	1	3	7	7	1	10
$k = 20$	测试/修复模式	23	23	146	146	19	190
	正常工作模式	1	5	15	15	1	20
$k = 50$	测试/修复模式	114	114	855	855	49	1275
	正常工作模式	1	11	35	35	1	50

最后对本文方法中核心的 BIRA 模块(含哈希表及冗余存储区)面积占用进行估算.由于存储单元是存储器各部分的主体,我们以 BIRA 内部存储单元面积近似代表 BIRA 模块面积,以存储器内部全部存储单元面积近似代表存储器面积.对于 $N = 1024 \times 1024 \times 32$bit 的存储器,采用 7 个冗余组则有:冗余存储大小 $S_{\text{total}} = 7 \times 1024 \times 32$bit,哈希表大小 $N_h = 1024 \times 20$bit(存储单元的大小等于存储器地址线宽),因此:

$$\text{BIRA 模块面积比例} \approx \frac{S_{\text{total}} + N_h}{S_{\text{total}} + N_h + N} \times 100\% \approx 0.74\%$$

5 结论

现有存储器 BISR 方法要么遍历式的地址比较效率低,要么并行地址比较功耗高,都不能很好的适用于故障数较多的存储器.对此,本文提出一种新型的高效存储器 BISR 方法,与现有方法不同,本文方法对占故障主体的单元故障以哈希表形式进行存储,以利用哈希表的快速搜索特性提升地址比较效率.计算机模拟实验显示与传统的二维冗余方法 ESP 相比,本文方法在修复率,冗余开销方面都有明显优势.另外还通过对比实验证明了本文方法的高效性,在正常工作模式下,本文方法在 1 个时钟周期内即可完成地址比较,保证了修复后存储器性能不受任何影响,与目前广泛采用的基于 CAM 的方法处于同一水平,但在功耗方面相对基于 CAM 的方法却具有明显优势.经估算对于 $1024 \times 1024 \times 32$bit 的存储器本文方法中核心的 BIRA 模块(含哈希表及冗余存储区)面积仅占存储器面积约 0.74%.本文方法修复率高,冗余资源开销小,工作高效,具有较高的实用价值.

参考文献

[1] I Kim, Y Zorian, G Komoriya, H Pham, FP Higgins, JL Lweandowski. Built in self repair for embedded high density SRAM [A]. Proceedings of International Test Conference (ITC) [C]. USA: IEEE, 1998. 1112 – 1119.

[2] 陈则王,苏建华,王友仁.嵌入式 SRAM 测试算法及其诊断实现[J].计算机辅助设计与图形学学报,2010,22(5):865 – 870.
Chen Zewang, Su Jianhua, Wang Youren. Test algorithm and diagnosis implementation for embedded SRAM [J]. Journal of Computer-Aided Design & Computer Graphics, 2010, 22(5): 865 – 870. (in Chinese)

[3] 付祥,王达,李华伟,胡瑜,李晓维.一种嵌入式存储器的内建自修复机制[A].第四届中国测试学术会议论文集[C].河北秦皇岛,2006. 15 – 19.

[4] SK Lu, YC Tsai, CH Hsu, KH Wang, CW Wu. Efficient built-in redundancy analysis for embedded memories with 2-D redundancy [J]. IEEE Transaction on VLSI Systems, 2006, 14(1): 34 – 42.

[5] TW Tseng, JF Li, DM Chang. A built-in redundancy-analysis scheme for RAMs with 2D redundancy using 1D local bitmap [A]. Proceedings of Conference Design Automation and Test [C]. Europe (DATE): Munich, 2006. 53 – 58.

[6] 苏建华,陈则王,王友仁,姚睿.嵌入式 SRAM 的一种高可靠性内建冗余分析策略研究[J].宇航学报,2010,31(11):2597 – 2603.
Su Jianhua, Chen Zewang, Wang Youren, Yao Rui. Higher reliability built-in redundancy analysis strategy for embedded SRAM [J]. Journal of Astronautics, 2010, 31(11): 2597 – 2603. (in Chinese)

[7] V Schober, S Paul, O Picot. Memory built-in self-repair using redundant words [A]. Proceedings of International Test Conference (ITC) [C]. Baltimore, 2001. 995 – 1001.

[8] 俞洋,李嘉铭,乔立岩.基于地址分割的嵌入式存储器内建自修复方法[J].电子学报,2010,38(2A):169 – 173.
Yu Yang, Li Jiaming, Qiao Liyan. Memory built-in self-repair method based on address partitioning strategy [J]. Acta Electronica Sinica, 2010, 38(2A): 169 – 173. (in Chinese)

[9] 谢远江,王达,胡瑜,李晓维.利用内容可寻址技术的存储

器 BISR 方法[J].计算机辅助设计与图形学报,2009,21(4):467 – 473.
Xie Yuanjiang, Wang Da, Hu Yu, Li Xiaowei. Memory BISR based on content addressable memory[J]. Journal of Computer-Aided Design & Computer Graphics, 2009, 21(4): 467 – 473. (in Chinese)

[10] 赵天绪,郝跃.集成电路局部缺陷模型及其相关的功能成品率分析[J].微电子学,2001,31(2):138 – 142.
Zhao Tianxu, Hao Yue. A local defect model of IC's and analysis of its related functional yield[J]. Microelectronics, 2001, 31(2): 138 – 142. (in Chinese)

[11] 郝跃,朱春翔,刘志镜.集成电路硅片上缺陷空间分布的分形表征[J].电子学报,1996,24(8):10 – 14.
Hao Yue, Zhu Chunxiang, Liu Zhijing. The fractal description of the defect spatial distributions on the wafer of integrated circuits[J]. Acta Electronica Sinica, 1996, 24(8): 10 – 14. (in Chinese)

[12] C T Huang, C F Wu, J F Li, C W Wu. Built-in redundancy analysis for memory yield improvement [J]. IEEE Transaction on Reliability, 2003, 52(4): 386 – 399.

作者简介

郭旭峰　男,1983 年 1 月生于黑龙江省哈尔滨市,2006 年于北京航空航天大学获得工学学士学位,目前为中国科学院微电子研究所在读博士研究生,主要研究方向为基于 LUT 结构的 FPGA 综合工具开发以及存储器内建自修复技术.
E-mail: phxwings@yahoo.com.cn

于　芳　女,生于 1960 年,中国科学院微电子研究所研究员、博士生导师,主要研究方向为基于 SOI 工艺抗辐射集成电路研究与设计,基于 LUT 结构的 FPGA 芯片及配套 EDA 工具开发等.
E-mail: yufang@ime.ac.cn