

故障芯片重利用的三维存储器成品率提高方法

刘 军^{1,2}, 王堂亚^{1,2}, 任福继³

(1 合肥工业大学 计算机与信息学院, 安徽 合肥 230009;

2 情感计算与先进智能机器安徽省重点实验室, 安徽 合肥 230009;

3 日本德岛大学 先端技术科学教育部, 日本 德岛 7708502)

摘 要: 提出了一种对故障芯片进行重新利用以提高三维存储器成品率的方法. 该方法首先将存储块划分为多个存储子块, 然后对存储芯片进行绑定前的测试以获取存储芯片的故障分布信息, 从而获取存储芯片中有故障的存储子块数量, 并根据所提的芯片选择算法来选定合适的存储芯片进行绑定. 接着在三维存储器中选择一个无故障的存储块作为全局冗余, 利用全局冗余中的存储子块对三维存储器中有故障的存储子块进行修复. 实验结果表明, 所提方法有效的提高了三维存储器的成品率.

关键词: 三维存储器; 成品率; 故障芯片; 全局冗余; 存储块

中图分类号: TP391

文献标识码: A

文章编号: 1000—7180(2016)10—0007—06

Bad-Die Recycling Technique for Yield Enhancement of Three-Dimensional Memories

LIU Jun^{1,2}, WANG Tang-ya^{1,2}, REN Fu-ji³

(1 School of Computer and Information, Hefei University of Technology, Hefei 230009, China;

2 Anhui Province Key Laboratory of Affective Computing and Advanced Intelligent Machine,

Hefei 230009, China; 3 Department of Information Science & Intelligent Systems, Faculty of

Engineering, The University of Tokushima, Tokushima 7708502, Japan)

Abstract: In order to make the best of the discarded bad dies, a yield enhancement technique by bad-die recycling for three-dimensional memory is proposed. In the presented technique, each block is divided into several sub-blocks firstly. After that, every die must go through the process of pre-bond test to get the fault information of blocks in a die, and then counting the number of the faulty sub-blocks in a die. Finally a fault-free block is chosen as global redundancy, and the sub-blocks in the global redundancy are used to repair faulty sub-blocks. The experimental results show that the proposed methodology is effective for yield enhancement of the three-dimensional memory.

Key words: three-dimensional memories; yield; bad-die; global redundancy; block

1 引言

三维存储器是三维集成电路的一个重要应用. 与二维存储器相比, 三维存储器在堆叠过程中不可避免的会产生新的存储单元故障, 导致三维存储器的成品率不高. 为提高三维存储器的成品率, 提出了很多有效地三维存储器修复方法^[1].

存储器的修复主要是利用存储器中的冗余单元替换存储器中的故障单元^[2]. 但存储芯片中冗余单元的数量受到电路面积和成本的制约, 所以冗余单元的数量是有限的. 若存储芯片中故障单元的数量超过冗余单元的数量, 则在二维存储器中此存储芯片将不能被修复而被丢弃. 但在三维存储器中, 为了提高存储芯片的利用率, 提出了存储芯片冗余共享

的方案^[1]. 冗余共享方案使得每层存储芯片都可以使用其它层的冗余单元来修复自身的故障单元, 从而有效提高了芯片的利用率. 但即使采用此种冗余共享方案, 仍有大量的存储芯片因没有足够的冗余单元对故障单元进行修复而被丢弃, 成为了不可被利用的故障芯片. 而这些被丢弃的存储芯片中有的芯片仅有很少的存储块不能被修复, 而其余的都是无故障的可利用存储块, 从而造成了很大的浪费. 为了解决此问题, 文献[3]提出了一种针对被丢弃的故障芯片重利用(recycle)方法. 此方法通过 DTHS (Dual-TSV Hardwired Switch) 连接需要重新利用的存储芯片, 把两层存储芯片组成存储容量相当于一层存储芯片的绑定结构, 用故障芯片中的无故障存储块作为冗余, 对故障存储块进行修复, 实现对故障芯片的重新利用以提高三维存储器的成品率. 但是在该方法中, 每两层故障芯片经过绑定后只能形成存储容量相当于一个无故障存储芯片, 故障芯片的重新利用率最高只有 50%.

为进一步提高被丢弃故障芯片的重新利用率, 本文提出了一种新的故障芯片重利用的三维存储器成品率提高方法.

2 背景知识

2.1 三维存储器结构

基于 TSVs 连接的三维存储器, 是利用 TSVs 作为层与层之间连接的垂直总线, 实现二维存储器的垂直堆叠. 文献[4]中描述了一种用 TSVs 垂直连接的三维存储器结构和三维解码器结构. 图 1 给出了此三维存储器和三维解码器结构. 从图 1 中可以看到, 三维存储器阵列垂直贯穿多层. 每一层对应的地址解码器也垂直堆叠, 形成一个三维解码器. 与二维存储器结构相比, 三维存储器结构有效的降低了电路功耗、减小了访问延迟^[4].

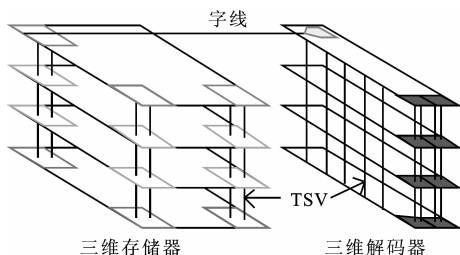


图 1 三维存储器结构图

2.2 三维存储器的堆叠方法

三维存储器层与层之间的堆叠主要有三种方

式: 晶元与晶元之间的堆叠(W2W)^[1]; 芯片与晶元之间的堆叠(D2W)^[5]; 芯片与芯片之间的堆叠(D2D)^[5]. W2W 就是选择两个晶元进行绑定(bond), 此种方法绑定的效率很高, 但绑定后芯片的成品率很低^[6]. D2W 就是选定一个芯片和晶元中的一个芯片进行绑定. D2D 就是选择两个单独的芯片进行绑定. 与 W2W 方式相比, 以 D2W 和 D2D 方式绑定的三维存储器成品率更高. 本文所提方法是基于 D2D 绑定方式的三维存储器.

2.3 存储器芯片的测试与修复

存储芯片在绑定前要进行绑定前测试与修复. 每个存储芯片含有若干个存储块, 每个存储块有一定数量的行列冗余. 对存储芯片进行绑定前的测试, 以获取芯片中每个存储块的故障信息. 根据存储块中故障信息, 分析出每个存储块需要的冗余数量, 然后根据自身冗余的数量, 判断利用自身的冗余能否修复这些故障单元^[7]. 根据绑定前测试与修复的情况, 把存储芯片分为好的芯片(无故障芯片)、自身可修复芯片、绑定后可修复芯片和不可修复芯片.

3 所提修复方法的完整流程

假设三维存储器有 L 层存储芯片, 存储芯片中的每个存储块划分为 M 个存储子块. 在本文所提方法中, 从三维存储器的 L 层存储芯片中选择一个无故障的存储块作为全局冗余, 利用冗余中的存储子块对有故障的存储子块进行修复, 其具体流程如图 2 所示.

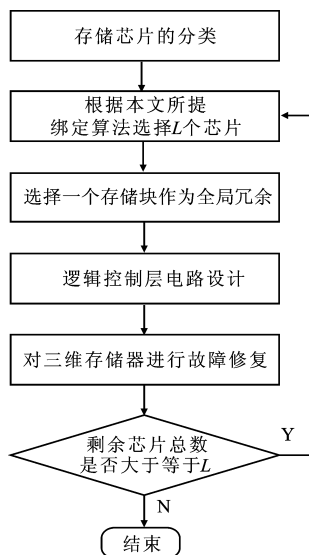


图 2 所提修复方法的流程图

图 2 中首先对所有的存储芯片进行绑定前的测

试以获取每个存储芯片的故障信息,并获取每个存储芯片中有故障的存储子块数量,进而根据每个存储芯片中的故障信息,对存储芯片进行分类.然后根据本文所提的芯片绑定策略,选择 L 个存储芯片.选定 L 个存储芯片后,接着根据每个存储芯片中故障的分布情况,选择一个无故障的存储块作为全局冗余.然后对三维存储器的逻辑控制层进行电路设计,以完成故障存储子块地址到全局冗余存储子块地址的映射.接着把逻辑控制层与所选定的 L 层存储芯片进行绑定,然后对三维存储器进行故障修复.最后判断剩余存储芯片的总数是否大于等于 L ,若大于等于 L ,则重复上述过程,当小于 L 时,结束上述过程.

3.1 存储芯片的分类

存储芯片绑定前都要进行绑定前的测试以获取每个存储芯片的故障信息.每个存储芯片中的存储块都划分为若干个存储子块,根据每个存储子块的故障信息,从而确定每个存储芯片中有故障的存储子块的数量.

每个独立的芯片里都有若干个存储块,假设一个芯片里面有四个存储块,对于一个被丢弃的故障芯片,可能所有的存储块都是有故障单元,也可能只有一个存储块或者部分存储块有故障单元.我们把所有存储块都有故障单元的芯片叫做全故障芯片;只有一个或者部分存储块有故障单元的芯片叫做部分故障芯片.对丢弃的故障芯片进行重新利用就要对芯片的故障单元进行修复,对故障单元进行修复需要冗余单元,我们需要在故障芯片组成的三维存储器里选择一个存储块作为三维存储器的全局冗余.对于全故障芯片,由于每个存储块中都有故障单元,因而不能从中选出一个可作为全局冗余的存储块.对于部分故障芯片,至少有一个存储块是无故障的,所以肯定能选择一个存储块作为全局冗余.对于故障芯片,在绑定前要根据故障分布的情况进行分类,进而保证堆叠后的三维存储器中有无故障的存储块作为全局冗余.

3.2 芯片选择算法

此算法是选取 L 个合适的存储芯片进行绑定.在选取的 L 个存储芯片中,首先,要保证至少有一个无故障的存储块可以作为三维存储器的全局冗余.在本文中,若存储芯片中所有存储块都有故障,则称此存储芯片为全故障芯片;若存储芯片中只有一个或者部分存储块有故障,则称此存储芯片为部分故障芯片.所以在所选的 L 层存储芯片中,至少要有一

层存储芯片是部分故障芯片.其次,在三维存储器中要保证 L 层存储芯片中有故障的存储子块的数量不能超过全局冗余中存储子块的数量.若有故障的存储子块数量多于全局冗余中存储子块的数量,将导致三维存储器不可被修复,从而整个三维存储器被丢弃.

因为在三维存储器中每个存储块均划分为 M 个存储子块,选定一个无故障存储块作为全局冗余,所以全局冗余中存储子块的数量即为 M .在选定的 L 层存储芯片中有故障的存储子块总数则要小于等于 M .

下面是本文所提的三维存储器绑定过程中芯片的选择算法.

芯片选择过程:

输入: $S = \{(D_1, F_1), (D_2, F_2), \dots, (D_n, F_n)\}$,

1. $m = M$, count = 0

2. for $i = 1$ to $L - 1$

3. $W = m - (L - i)F_{\min}$

4. 从 S 中随机选择一个芯片 D_j , 故障数为 $F_j < W$ ($j = 1$ to n) 作为第 i 层

5. $m = m - F_j$ // 更新

6. $S = S - F_j$ // 更新

7. $F_{\min} = S_{\min}$ // 更新

8. If $D_j = 1$ // 所选存储芯片是部分故障芯片

9. Count = 1

10. END for // 结束前 $L - 1$ 存储芯片的选择

11. $W = m - (L - i)F_{\min}$

12. If Count = 1 // 所选取的前 $L - 1$ 层存储芯片中有部分故障芯片

13. 从 S 中随机选择一个芯片 D_j , 故障数为 $F_j < W$ ($j = 1$ to n) 作为第 L 层

14. Else

15. Count = 0 // 所选取的前 $L - 1$ 层存储芯片中都是全故障芯片

16. 从 S 中选择一个芯片 D_j , 故障数为 $F_j < W$ ($j = 1$ to n) and $B_j = 1$ 作为第 L 层

17. END // L 层芯片选择结束

其中集合 $S = \{(D_1, F_1), (D_2, F_2), \dots, (D_n, F_n)\}$ 表示所有故障芯片和对应故障芯片中的故障存储子块数量及芯片的类型. D_j ($1 \leq j \leq n$) 表示第 j 块芯片,若 $D_j = 1$ 表示此存储芯片为部分故障芯片, $D_j = 0$ 表示此存储芯片为全故障芯片. F_j ($1 \leq j \leq n$) 表示芯片 D_j 含有故障存储子块的数量. $F_{\min}$ 表示集合 S 中 F_j 最小值, $F_{\min} = S_{\min} = \min(F_1, F_2, \dots, F_n)$. M 表示三维存储器全局冗余中存储子块数量.

在算法中首先对 m 和 count 进行赋值。 m 表示三维存储器中可用的全局冗余存储子块数量,初始化时 $m = M$ 。 count 表示所选存储芯片中是否已有部分故障芯片,若有, $\text{count} = 1$,若没有, $\text{count} = 0$,初始化时 $\text{count} = 0$ 。

然后从集合 S 中随机选择一个芯片 D_j 作为存储器的第 i ($1 \leq i \leq L-1$) 层,选定的存储芯片 D_j 其故障存储子块数 F_j 应满足 $F_j \leq W$, W 表示每次选定的存储芯片中所能含有故障存储子块数量的最大值, $W = m - (L - i)F_{\min}$,所选定的芯片 D_j 需要数量大小为 F_j 的冗余存储子块予以修复,所以余下 $L - i$ 层芯片可用的冗余总数为 $m - F_j$,更新 $m = m - F_j$ 。从集合 S 中删除 D_j ,更新 S ,并更新 $F_{\min} = S_{\min}$ 。然后判断 D_j ,若 $D_j = 1$,则 $\text{count} = 1$ 。执行 $L-1$ 次,结束 For 循环。至此三维存储器的前 $L-1$ 层芯片已选定,接着选定最后一层存储芯片。

此时先更新 W ,然后下一步判断 count ,若 $\text{count} = 1$,则表示在前 $L-1$ 层的存储芯片中,至少有一层存储芯片为部分故障芯片,则从 S 中随机选择一个存储芯片作为三维存储器的第 L 层存储芯片;若 $\text{count} = 0$,则表示在前面选定的 $L-1$ 层存储芯片均为全故障芯片,此时在 S 中选择的存储芯片必须为部分故障芯片,即所选存储芯片应满足 $D_j = 1$ 且 $F_j < W$ 。至此完成 L 层存储芯片的选择。

3.3 作为全局冗余存储块的选定

本文所提方法根据每一层芯片的故障信息来选定一个存储块作为全局冗余。首先,所选定的存储块必须是无故障的存储块;其次所选定的存储块是三维存储器中离逻辑控制层最近的存储芯片中的存储块。

首先,因为对故障单元的修复,是利用冗余替换需要修复的故障单元,所以作为全局冗余的存储单元必须是无故障的。因而要选择不故障的存储块作为三维存储器的全局冗余。

其次当对故障单元进行访问时,由于故障单元已被修复,经过地址解码器的地址映射,对冗余单元进行访问。访问信号从逻辑控制层经过层与层之间的 TSVs 到达要访问的位置,访问信号在层与层之间传输时会有延迟。因而要选定一个被访问时信号传输经过层数最少的存储块作为全局冗余,即离逻辑控制层最近的存储块作为全局冗余,以减少信号传输经过的层数,提高访问的速度,提高三维存储器的性能。

3.4 三维存储器逻辑控制层的设计

为了实现冗余单元对故障单元的修复,在三维存储器的逻辑控制层中要进行相应的电路设计。

本文提出的逻辑控制层的结构图如图 3 所示。图 3 中电路由内建自测试电路、内建冗余分析电路、保险丝宏模块、地址解码器、全局冗余分析电路和信号选择器(MUX)构成。

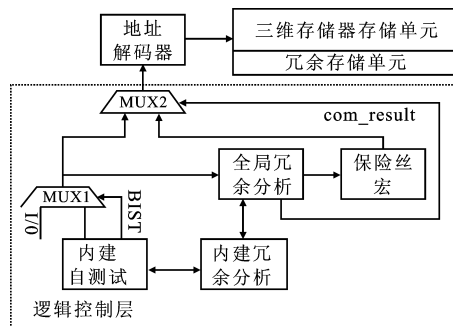


图3 本文逻辑层逻辑控制单元结构

内建自测试电路:内建自测试(BIST, Built-in Self-Test)电路测试存储单元是否有故障。

内建冗余分析电路:内建冗余分析(BIRA, Built-in Redundancy-Analysis)电路记录测试出的故障单元地址,并给故障单元分配行列冗余单元。

全局冗余分析电路:全局冗余分析(GRA, Global Redundancy-Analysis)电路根据内建冗余分析电路的记录信息和经过行列冗余修复后的存储单元故障信息,给有故障的存储子块分配全局冗余存储子块。

熔丝宏:熔丝宏(Fuse Macro)存储故障信息以完成故障单元到冗余单元的地址重映射。

逻辑控制层有两种工作模式:测试/修复模式和正常工作模式。在测试/修复模式下,BIST 电路给对三维存储器的存储单元进行测试,并把测试结果传输到 BIRA 电路中。BIRA 电路记录下故障单元的地址并给故障单元分配行列冗余,接着 BIRA 电路把所记录的信息传输到 GRA 电路中。GRA 电路记录下 BIRA 中的信息,用全局冗余中存储子块对有故障的存储子块进行修复。在正常工作模式下,BIST 处于不工作状态,访问地址被传输到 GRA 电路和 MUX2 中,将访问地址与 GRA 中故障单元地址进行比较,若访问地址不在 GRA 的记录中,GRA 中输出触发信号 com_result ,以控制访问地址直接

传输到地址解码器中,选中所要访问的存储单元。若访问地址在 GRA 的记录中,即所要访问的存储单元是故障单元,触发信号 com_result 控制 MUX2,熔丝宏中输出与故障地址相对应的映射地址,传输到地址解码器中,完成故障存储单元到冗余存储单元的地址重映射。

3.5 所提修复方法示例

本部分介绍了一个四层三维存储器的故障修复示例. 在这个四层三维存储器中,最底层是逻辑控制层,其余三层为存储芯片,每个存储芯片中有四个存储块,每个存储块划分为四个存储子块. 只要存储子块中有故障单元,那么存储子块即为故障存储子块,就要用一个冗余存储子块去修复故障存储子块,其具体情况如图 4 所示.

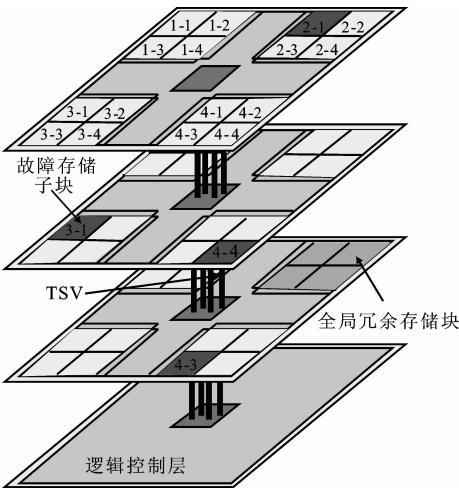


图 4 故障修复示例

从图 4 中可以看到,第一层存储芯片的 4-3 存储子块是故障存储子块,第二层存储芯片的 3-1 和 4-4 是故障存储子块,第三层存储芯片的 2-1 故障存储子块,第一层存储芯片的第二个存储块被选定为全局冗余存储块. 全局冗余存储块中有四个冗余存储子块,利用这四个冗余存储子块对四个不同位置故障存储子块进行修复,进而得到一个好的三维存储结构.

4 实验结果和分析

本部分对本文所提方法与文献[3]方法的实验结果进行了比较和分析,用 C 编程实现了本文所提方法和文献[3]方法.

4.1 故障分布模型

本文实验采用常用的 Polya-Eggenberger 分布模型对每个存储块进行故障单元的注入. Polya-Eggenberger 分布模型适用于集成电路成品率问题的建模^[8]. Polya-Eggenberger 分布函数有两种不同分布情况,第一种分布参数 $\lambda=8$,参数 $\alpha=2.382$. 第二种分布参数 $\lambda=8$,参数 $\alpha=0.6232$. 当分布函数参数 $\lambda=8, \alpha=2.382$ 时故障的注入分布具有聚簇性,参数 $\lambda=8, \alpha=0.6232$ 时故障的注入是均匀地.

本文实验基于这两种分布情况进行比较.

Polya-Eggenberger 分布函数的函数曲线如图 5 所示,横坐标表示每个存储块中注入的故障数,纵坐标表示注入的故障数相对应的概率.

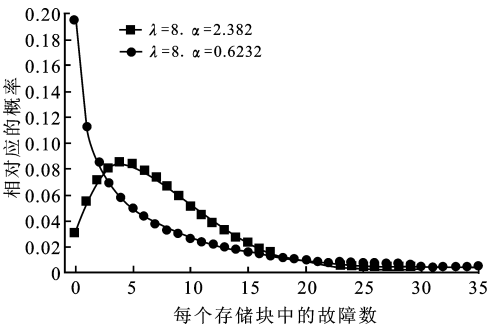


图 5 参数 $\lambda=8, \alpha=2.382$ 与参数 $\lambda=8, \alpha=0.6232$ 时 Polya-Eggenberger 分布函数的图像

4.2 本文方法的重新利用率

故障芯片的重新利用率是指可以被重新利用的故障芯片数量在所有故障芯片总数中的比例.

对故障芯片尚未进行重新利用时,假设三维存储器的成品率为 $1-P$,那么被丢弃的故障芯片所占比例为 P . 若故障芯片的重新利用率为 Q ,则存储器成品率提高的部分为 $P \times Q$,则整个三维存储器的成品率 $Z = 1 + P \times (Q - 1)$,在 P 不变的情况下 Z 随着 Q 的增大而增大. 三维存储器的成品率和故障芯片的重新利用率呈线性关系,因而本文通过对故障芯片重新利用率的提高以提高三维存储器的成品率.

存储芯片中存储块数量以及存储块中存储子块的数量对故障芯片重新利用率都有直接的影响,本文分别比较存储芯片在存储块数量为 4、8 和 16 时以及存储块中存储子块的数量分别为 16、64 和 256 时芯片的重新利用率. 在实验中,将每个存储块的大小设定为 $1\,024 \times 1\,024 \times 8$ 位.

图 6 描述了三维存储器在不同情况下故障芯片的重新利用率的不同. 图 6(a) 和 (b) 分别表示在故障分布模型两种不同参数下故障芯片的重新利用率情况,在图 6(a) 中参数 $\lambda=8, \alpha=2.382$,图 6(b) 中参数 $\lambda=8, \alpha=0.6232$. 在图 6(a) 和 (b) 中横坐标表示每个故障芯片中存储块的数量和每个存储块中存储子块的数量,纵坐标表示故障芯片的重新利用率. 从图 6 可以看出,芯片中的存储块数增多,存储器的修复率相应的降低. 当存储块中存储子块的数量越大时,故障芯片的重新利用率越高.

4.3 本文所提方法与已有方法的比较

图 7 是本文方法与文献[3]方法中故障芯片重

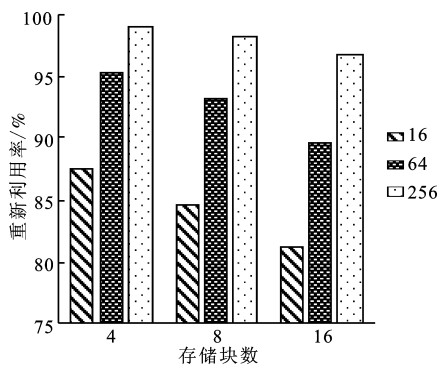
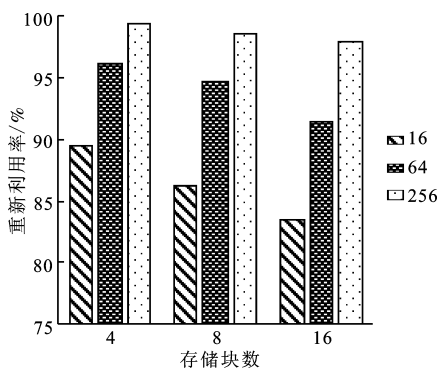
(a) $\lambda=8, \alpha=2.382$ (b) $\lambda=8, \alpha=0.6232$

图6 三维存储器在不同情况下故障芯片的重新利用率

利用率的比较. 如图7所示, 两种方法的比较是在存储芯片中存储块划分为64个存储子块的情况下, 分别比较芯片中存储块数为4、8和16时故障芯片重新利用率的情况.

从图6、图7可以看出文献[3]中故障芯片重新利用率最大时都小于50%, 本文提出的方法的重新利用率最小时为81.3%, 最大时为98.5%, 与文献[3]相比有了很大的提高, 有效的提高了三维存储器的成品率.

5 结束语

本文提出了对故障芯片重新利用以提高三维存储器成品率的方法. 此方法把故障芯片中的存储块划分为若干个存储子块, 然后根据芯片绑定前的测试, 以确定每个存储芯片中故障存储子块的数量. 根据本文所提的芯片选择算法选择合适的芯片进行绑定, 为了实现冗余存储子块对故障存储子块的修复, 对逻辑控制层进行电路设计, 然后把所选定的存储芯片与逻辑控制层堆叠成三维存储器. 然后在三维存储器中, 选定一个合适的存储块作为全局冗余, 通过全局冗余中的存储子块对故障存储子块进行修复, 实现对故障芯片的重新利用. 故障芯片的重新利

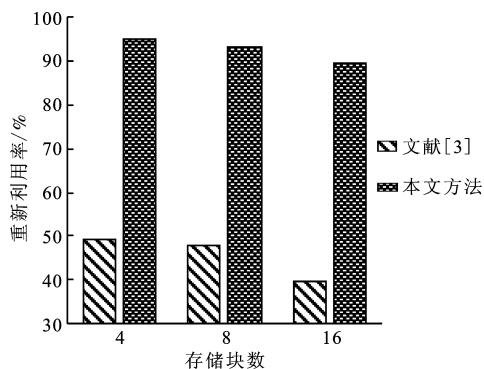
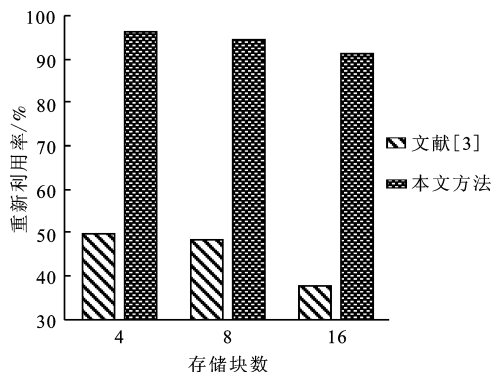
(a) $\lambda=8, \alpha=2.382$, 存储块划分数 $N=64$ (b) $\lambda=8, \alpha=0.6232$, 存储块划分数 $N=64$

图7 本文与文献[3]的故障芯片重新利用率的比较

用率得到了很大的提高, 有效的提高了三维存储器的成品率.

参考文献:

- [1] Jiang L, Ye R, Xu Q. Yield enhancement for 3D-stacked memory by redundancy sharing across dies[C] // Proceedings of the International Conference on Computer-Aided Design. China, Yiwu, IEEE Press, 2010: 230-234.
- [2] Chou Y F, Kwai D M, Shieh M D, et al. Reactivation of spares for off-chip memory repair after die stacking in a 3-D IC with TSVs[J]. Circuits and Systems I: Regular Papers, IEEE Transactions on, 2013, 60 (9): 2343-2351.
- [3] Chou Y F, Kwai D M, Wu C W. Yield enhancement by bad-die recycling and stacking with through-silicon vias[J]. Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, 2011, 19 (8): 1346-1356.
- [4] Puttaswamy K, Loh G H. 3D-integrated SRAM components for high-performance microprocessors [J]. Computers, IEEE Transactions on, 2009, 58(10): 1369-1381.

表 2 与参考文献中 SM4 实现的性能比较

设计	工艺/ μm	延时/ ns	面积	CBC 吞吐率/ (Mb/s)
文献[5]	0.18	10	78 525 μm^2	100
文献[6]	0.18	20	19.6 k 门	200
文献[7]	0.18	5.4	30 569 μm^2	185
文献[8]	0.18	5	32 k 门	801
本文	0.18	8.91	649 516 μm^2	1 790
文献[9]	0.13	5	22 k 门	800
本文	0.13	6.15	267 361 μm^2	2 600

6 结束语

根据本文中提出的针对 SM4 关键路径进行化简的方法,实现了 4 轮合 1 的 SM4 电路.综合结果表明 4 轮合 1 的 SM4 电路比本文的其他方案有更高的单位面积吞吐率,且 CBC 模式下的吞吐率高于已发表的同类设计.因此,这种对 SM4 关键路径进行化简的方法是有效的,并且对其他分组对称算法提高 CBC 模式下的吞吐率有参考价值.

参考文献:

[1] 国家密码管理局. GM/T 0002-2012. SM4 分组密码算法[S]. 北京:中国标准出版社,2012.

[2] 李大为,赵旭鑫,武萌. SMS4 密码算法的高速流水线实现[J]. 电子器件,2007,30(2):590-592.

[3] Shastry P V S, Somani N, Gadre A, et al. Rolled architecture based implementation of AES using T-Box [C]// 55th IEEE International Midwest Symposium on Circuits and Systems (MWSCAS). Boise, Idaho,

USA; IEEE, 2012; 626-630.

[4] 刘楠,陈迅,李军伟. 面向 CBC 模式的 AES 高速芯片设计与实现[J]. 电子技术应用,2014,40(1):64-67.

[5] 王晨光,乔树山,黑勇. 分组密码算法 SM4 的低复杂度实现[J]. 计算机工程,2013,39(7):177-180.

[6] 徐艳华,白雪飞,郭立. 适合 SMS4 算法硬件实现的 S 盒构造新方法[J]. 中国科学技术大学学报,2009,39(11):1164-1170.

[7] Shang Ming, Zhang Qinglong, Liu Zongbin, et al. An ultra-compact hardware implementation of SMS4[C]// 3rd IIAI International Conference on Advanced Applied Informatics (IIAI-AAI). Kitakyushu, Japan; IEEE Computer Society, 2014; 86-90.

[8] Bai Xuefei, Guo Li, Huang Lu, et al. A fast VLSI design of SMS4 cipher based on twisted BDD S-Box architecture[C]// 2009 International Conference on Networks Security, Wireless Communications and Trusted Computing (NSWCTC 2009). Wuhan, China; IEEE Computer Society, 2009; 345-348.

[9] Yan Weiwei, You Kaidi, Han Jun, et al. Low-cost reconfigurable VLSI implementation of the SMS4 and AES algorithms[C] // 2009 IEEE 8th International Conference on ASIC (ASICON). Changsha, China; IEEE, 2009; 135-138.

作者简介:

符天枢 男,(1989-),硕士研究生.研究方向为信息安全算法的数字大规模集成电路设计与实现.
E-mail:fts13@mails.tsinghua.edu.cn.

李树国 男,(1963-),教授,博士生导师.研究方向为信息安全算法及其大规模集成电路设计与实现.

(上接第 12 页)

[5] Lee J, Park K, Kang S. A Die-selection method using search-space conditions for yield enhancement in 3D memory[J]. ETRI Journal, 2011, 33(6): 904-913.

[6] Rab M T, Bawa A, Touba N. Implementing defect tolerance in 3D-ICs by exploiting degrees of freedom in assembly[C]//Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), 2012 IEEE International Symposium on. USA,Austin,IEEE, 2012; 178-181.

[7] Kang W, Lee C, Cho K, et al. A die selection and matching method with two stages for yield eehancement of 3-D memories[C]//Test Symposium (ATS), 2013 22nd Asian. Taiwan, Yilan, IEEE, 2013; 301-306.

[8] Huang R F, Li J F, Yeh J C, et al. A simulator for e-

valuating redundancy analysis algorithms of repairable embedded memories[C] // Memory Technology, Design and Testing, 2002. (MTDT 2002). Proceedings of the 2002 IEEE International Workshop on. France, Bendor,IEEE, 2002; 68-75.

作者简介:

刘 军 男,(1978-),博士,教授,硕士生导师.研究方向为嵌入式系统、计算机体系结构.

王堂亚(通讯作者) 男,(1990-),硕士研究生.研究方向为三维存储器故障单元容错技术. E-mail:993973922@qq.com

任福继 男,(1959-),留日博士,教授,博士生导师.研究方向为智能机器人、情感计算、自然语言处理、智能科学、大数据与社会计算、云与健康计算.