

3D SRAM 中 TSV 开路测试算法研究与实现*

赵振宇, 邓全, 李鹏, 蒋剑锋, 曲连华, 唐皓月
(国防科技大学计算机学院, 湖南长沙 410073)

摘要:基于三维集成电路技术实现的三维静态随机存储器,其电路中使用了大量的过硅通孔。目前过硅通孔制造工艺尚未成熟,使得过硅通孔容易出现开路或短路故障,从而给三维静态随机存储器的测试带来新的挑战。现有的过硅通孔专用测试方式虽然能够探测出过硅通孔的故障,但需要特定的测试电路来实现,这就增加了额外的面积开销,同时加大了电路设计的复杂度。因此,提出一种使用测试算法来探测过硅通孔开路故障的方法。在不增加额外面积开销的情况下,通过内建自测试电路解决三维静态随机存储器中过硅通孔的开路故障检测问题。结果显示,该过硅通孔测试算法功能正确,能够准确探测到过硅通孔的开路故障,并快速定位过硅通孔的开路位置。

关键词:三维集成电路;过硅通孔;开路故障;测试;算法

中图分类号:TN47 **文献标志码:**A **文章编号:**1001-2486(2016)05-007-07

Research and implementation of TSV open test algorithm in 3D SRAM

ZHAO Zhenyu, DENG Quan, LI Peng, JIANG Jianfeng, QU Lianhua, TANG Haoyue
(College of Computer, National University of Defense Technology, Changsha 410073, China)

Abstract: In 3D-IC (three dimensional-integrated circuit) technology based 3D SRAM(three-dimensional static random access memory), a large number of TSVs (through silicon vias) have been implemented in circuits. The manufacturing process of TSV is not mature, which makes TSVs more prone to open defects and brings new challenges to the 3D SRAM test. The existing method of TSV test can find out where the faults are, but it needs extra specific circuit to implement, which increases both the area overhead and its design complexity. In consideration of what was discussed above, a new idea to detect the open defects of TSV based on a test algorithm was proposed. It proved to be an effective method to detect the open defects of TSV in 3D SRAM using BIST(built-in self test) without any extra overhead. Results show that the proposed method has no functional problem with the algorithm and it can realize the purpose of TSV open fault detection.

Key words: three dimensional-integrated circuit; through silicon via; open defects; test; algorithm

基于三维集成电路(Three Dimensional-Integrated Circuit, 3D-IC)技术的三维静态随机存储器(Three-Dimensional Static Random Access Memory, 3D SRAM)采用多层器件结构组成,通过内在的垂直互连介质过硅通孔(Through Silicon Via, TSV)完成不同层器件之间的通信^[1]。3D SRAM 拥有 2D SRAM 所不具有的很多优势,例如可以缩短管芯(die)与管芯(die)之间的互连线长度,提高输入/输出(Input/Output, IO)带宽,从而减小互连线延迟,提升性能,并降低功耗^[2]。3D SRAM 还可以进行异质集成,以此缩小设计面积和制造成本。

3D SRAM 中有超过数以万计的 TSV,由于 TSV 的制造工艺尚不成熟,铜柱内约有 42% 的概

率产生气泡或空隙^[3],从而导致 TSV 产生开路故障;又由于 TSV 之间寄生耦合电容的影响,TSV 存在开路故障时,浮空导线与相邻导线之间的寄生耦合电容对浮空端的电压影响很大。在 TSV 尺寸继续减小的情况下,TSV 之间的间距也越来越小,这种耦合效应也越明显,不仅影响自身,而且还可能影响周围的电路,最终导致 SRAM 失效。相关研究表明,规模为 10^4 个 TSV 的芯片中出现 TSV 失效的概率为 63.214%^[4],严重影响 3D 芯片的良品率。因此,如何有效地对 TSV 进行测试是 3D-IC 中急需解决的问题。近年来学术界和工业界对片上 TSV 测试结构的研究从未中断过。

文献[5]提出了一种用于 TSV 故障诊断的内建自测试(Built-In Self Test, BIST)结构,并将

* 收稿日期:2015-05-30

基金项目:国家自然科学基金资助项目(61176030,61272139)

作者简介:赵振宇(1973—),男,辽宁朝阳人,教授,博士,硕士生导师,E-mail:zyzhao@nudt.edu.cn

TSV 建模为 2D 矩阵的形式。尽管这种测试结构拥有面积开销小的优势,但一次只能测试 TSV 矩阵中的一行,当 TSV 数量较多时,将耗费很长的测试时间。而另一种 BIST 结构^[6]使用基于 KAF (Kth-Aggressor-Fault) 可编程模型产生有效测试序列,用于探测 TSV 之间的串扰故障,这种结构可以提供更精准的 TSV 测试结果。然而,由于每个 TSV 都需要单独的发送器和接收器,这将需要很大的面积开销。输出阈值分析测试结构^[7]通过使用环形振荡器来探测 TSV 的故障。这种结构能有效处理由 TSV 故障引起的信号延时问题,并且具有面积开销小的优势,但能探测的 TSV 故障类型有限。

针对 3D SRAM 中字线和位线 TSV 的全开路故障,提出了一种有效的片上 TSV 测试方法。将 TSV 测试机制与 Memory BIST 结构相结合,使用测试算法来探测 TSV 开路故障。由于共用 Memory BIST,本文所提测试方法可以有效缩短测试时间和面积开销。结果显示,该 TSV 测试算法功能正确,能够准确探测到 TSV 的开路故障,并快速定位 TSV 的开路位置。

1 TSV 开路故障行为

1.1 TSV 开路故障耦合模型

目前,对 TSV 的耦合电学建模研究已经相当深入,相比传统的使用固定故障、桥接故障来分析 TSV 的故障,MAF (Maximum Aggressor Fault) 故障模型^[8]则被广泛地应用于探测 TSV 的耦合故障。文献^[9]结合 TSV 的生产工艺建立了 TSV 的串扰 Spice 电路模型。文献^[10]全面考虑了 TSV 的寄生分量建立了 TSV 耦合电学模型。为了更好地实现对 3D SRAM 中字线 TSV 和位线 TSV 的全开路故障进行模拟以及分析 TSV 开路故障对存储器的影响,从 TSV 之间的耦合效应入手,基于 TSV 开路故障电学模型和 TSV 耦合电学模型,建立了 TSV 开路故障耦合效应电学模型,如图 1 所示。该模型既考虑了 TSV 的开路故障,同时又考虑了 TSV 之间的耦合效应,模型中的参数均可由文献^[11]中的公式计算得到。

1.2 TSV 故障定位

3D 堆叠存储器可以通过多种方式来实现,一种有效的方式是用 TSV 做垂直总线,将多个 SRAM 层连起来,接到处理器上^[12];另外一种方式是将存储阵列分很多小块,并在垂直方向上进行堆叠,用 TSV 代替原有互连线充当字线和位

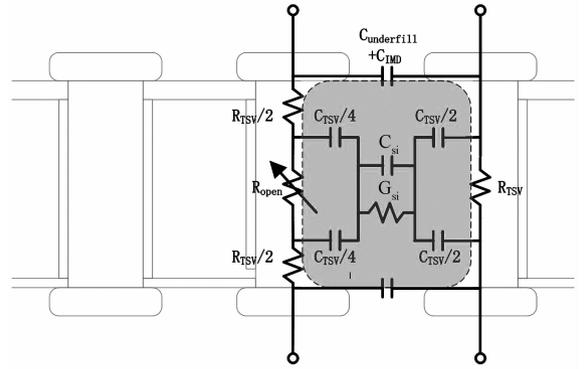


图 1 TSV 开路故障耦合效应电学模型

Fig. 1 TSV open defect coupling model

线,并通过 TSV 连接到底层的外围电路^[13];本文依据第二种堆叠方式建立了图 2 所示的 3D SRAM 电路模型。

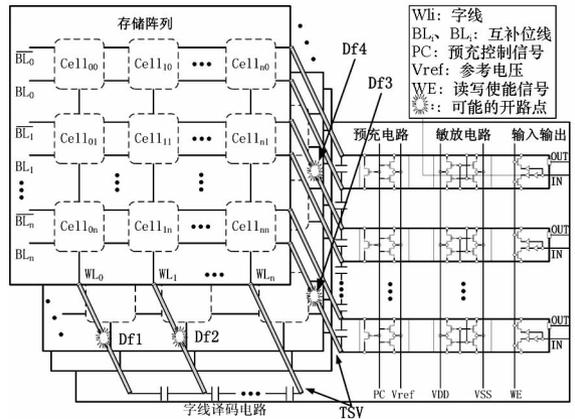


图 2 3D SRAM 电路模型

Fig. 2 3D SRAM model

分析电路结构可知,TSV 的开路故障既可以发生在字线上,也可以发生在位线上,且均位于 TSV 互连线上。根据电路结构对称性特点以及 TSV 开路故障的随机性,可以将 TSV 开路故障的植入位置定位在四个地方 (Df1 ~ Df4)。由于存储单元以阵列形式布放,字线和位线分别位于不同的边,TSV 也是按顺序布放,相对位置都是对称的,所以这四处开路故障位置足以表征整个 3D SRAM 中的所有 TSV 开路故障情况。

1.3 TSV 开路故障模拟

将 TSV 开路故障逐一注入图 2 故障位置中,采用图 1 所示 TSV 开路故障耦合效应电学模型,并用 Hspice 进行模拟。模拟环境设置为:65 nm 工艺,1.2 V 电源电压,常温 (25°C),典型 corner (TT)。故障位置、操作类型、最小开路电阻值以及最大失效概率阻值整理如表 1 所示,TSV 开路引起的 SRAM 失效在考虑工艺偏差情况下的最

大失效率用 F_{mc} 来表示。

表 1 TSV 开路导致 3D SRAM 失效的最小电阻值

Tab.1 TSV open-circuit affect the 3D SRAM failure and their minimum values

开路位置	模式	操作	$Df_{i-nom}/$ kΩ	$Df_i/$ kΩ	$F_{mc}/$ %
Df1	访问相邻字线	写	136.1	1051	92
Df2	访问相邻字线	写	7.8	18.6	100
Df3	BL	1w0	24.5	32.5	100
	开路	r0	318	20 000	62
Df4	\overline{BL}_1	0w1	26.4	38.4	100
	开路	r1	11.9	18.9	100

表 1 中,定义 Df_{i-nom} 为在标称条件下导致 SRAM 失效的 TSV 最小开路电阻值;定义最大失效率阻值 Df_i 为在考虑工艺偏差时,导致 SRAM 最大概率(100%)出现功能故障的 TSV 开路最小电阻值。

对应的曲线图如图 3 所示,图中横坐标为 TSV 开路电阻的相对最小阻值(Df_{i-rel}),由式(1)获得;纵坐标为 SRAM 失效率,最大值为 100%。

$$Df_{i-rel} = \frac{Df_i}{Df_{i-nom}} \quad (1)$$

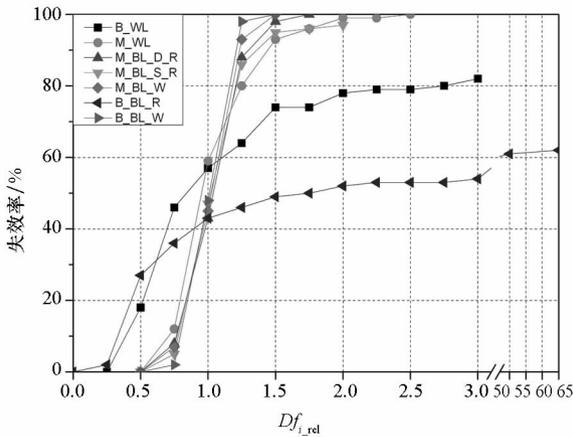


图 3 3D SRAM 失效率随 TSV 开路电阻变化

Fig.3 3D SRAM failure probability change under TSV open-circuit resistance

通过图 3 中曲线数据可得出如下结论:
在考虑工艺偏差的情况下:

1) 当注入故障位置的开路电阻阻值等于该故障位置在标称条件下最小可探测开路电阻值时,引起的存储器失效率大约在 40% ~ 60% 之间(即 $Df_{i-rel} = 1$)。换句话说,在标称条件下,TSV 存在开路故障时会导致存储器产生故障行为的最

小可探测值,在考虑工艺偏差对电路影响的情况下,失效率不到 60%。

2) 随着 TSV 开路电阻值的增加,存储器的失效率也随之升高。从图 3 中还可以看到,当 Df_{i-rel} 处在 0.5 ~ 1.5 之间时,TSV 的开路电阻只有小幅增加,但由此引起的存储器失效率却呈直线上升,最高达到 100%。这说明当考虑工艺偏差对电路的影响时,TSV 的开路故障对存储器功能故障的影响更严重。

3) 对于 Df1 处开路故障位置,最小开路电阻值是标称条件下 TSV 开路电阻值的 3.0 倍(B_WL),即 $Df_{i-rel} = 3.0$;相应地,对于 Df2 处开路故障位置,最小开路电阻值是标称条件下 TSV 开路电阻值的 2.5 倍(M_WL),即 $Df_{i-rel} = 2.5$;对于 Df3 处开路故障位置,最小开路电阻值是标称条件下 TSV 开路电阻值的 1.5 倍(B_BL_W),即 $Df_{i-rel} = 1.5$;对于 Df4 处开路故障位置,最小开路电阻值是标称条件下 TSV 开路电阻值的 2.0 倍(M_BL_S_R),即 $Df_{i-rel} = 2.0$ 。

除此之外,本小节还分别对字线和位线的操作类型以及开路位置进行广泛的模拟研究,模拟结果如表 2 所示。

表 2 3D SRAM 中 TSV 开路故障行为

Tab.2 3D SRAM TSV open-circuit fault behavior

开路位置	模式	操作	TSV 开路故障行为
字线	访问开路字线	读/写	不可访问
	访问相邻字线	读 写	访问多个单元 可改变浮空单元
	BL 和 \overline{BL} 同时开路	读 写	读出错误值 不可访问
位线	BL 开路	读	固定输出“1”
		写	不可以正确写
	\overline{BL} 开路	读 写	固定输出“0” 不可以正确写

字线 TSV 存在开路故障时,存储器所对应的故障行为主要有三类,即“不可访问”“访问多个单元”以及“可以改变浮空单元里的值”。位线 TSV 存在开路故障时,存储器所对应的故障行为有四类,分两种情况:当两根位线同时开路时,故障行为有“读出错误值”和“不可访问”两类,当其中某一根位线开路时,故障行为有“不可以正确

写”和“固定输出 0 或 1”两类。

2 故障原语及 TSV 开路测试算法

2.1 故障原语定义

功能故障检测的两个基本组成部分为:读写操作序列以及相应单元的状态变化。因此,3D SRAM 中每一种 TSV 开路故障的单一单元故障原语可采用 S, F, R 的组合 $\langle S/F/R \rangle$ 来表示存储器的故障行为;多单元故障用 $\langle Sa; Sv/F/R \rangle$ 来表示,其中 $Sa(Sv)$ 代表施扰单元(受扰单元)。 S 是敏化操作序列,描述触发存储单元故障行为的操作或状态, $S \in \{0, 1, 0w0, 1w1, 0w1, 1w0, r0, r1\}$; F 是故障行为,用于描述被检测有故障单元的行为或状态, $F \in \{0, 1, \uparrow, \downarrow\}$,“0”和“1”表示“0”状态和“1”状态, $\uparrow(\downarrow)$ 表示上升(下降)跳变;此外,当“ S ”为读操作时,还应该加上一个敏化操作序列的“读取结果” R (Result), R 描述存储单元读操作的输出结果, $R \in \{0, 1, -\}$ 。当“ S ”不为读操作时, R 为“-”。

2.2 故障行为与 March 元素

不可访问的故障行为,存储单元的逻辑控制已经失效,无法进行读写操作,故全 0 或全 1 的操作可以检测出此类故障。针对不可访问故障的最简化 March 元素为: $\Downarrow(w0)\Downarrow(r0)$ 或 $\Downarrow(w1)\Downarrow(r1)$, 其中“ \Downarrow ”为地址增序或降序。

访问多个单元故障行为,普通的测试算法就可以检测到。针对访问多个单元故障的最简化 March 元素为: $\Uparrow(r0, w1)$ 或 $\Downarrow(r1, w0)$, 其中“ \Uparrow ”为地址增序,“ \Downarrow ”为地址降序。

可改变浮空单元故障行为,需要敏化存储器单元 a, v 状态为: $(0w1, 0)$, 或 $(1w0, 1)$ 。先初始化存储器单元为 0 或 1,之后按照降序顺序进行 $w1$ 或 $w0$ 操作,即可敏化状态 $(0w1, 0)$ 或 $(1w0, 1)$, 在 $w1$ 或 $w0$ 操作之前添加 $r0$ 或 $r1$ 操作,可以读出存储单元的数据,从而判断是否发生故障。针对可改变浮空单元故障最简化的 March 元素为: $\Downarrow(w0)\Downarrow(r0, w1)$ 或 $\Downarrow(w1)\Downarrow(r1, w0)$ 。

读出错误值故障行为,需要敏化存储单元 a, v 的状态为 $(r0, 0)$ 或 $(r1, 1)$ 。先将存储单元初始化为 0 或 1,然后按照降序顺序进行 $r0$ 或 $r1$ 操作,即可敏化存储器状态为 $(r0, 0)$ 或 $(r1, 1)$ 。针对读出错误故障最简化的 March 元素为 $\Downarrow(w0)\Downarrow(r0)$ 或 $\Downarrow(w1)\Downarrow(r1)$ 。

固定输出 0 或 1 故障行为,必须初始化所有存储单元数据为 1 或 0,然后进行 $r1$ 或 $r0$ 操作。

如果读出数据为 1 或 0,证明该单元没有故障;如果读出数据为 0 或 1,则证明该单元存在故障。针对固定输出 0 或 1 故障最简化的 March 元素为 $\Downarrow(w1)\Downarrow(r1)$ 或 $\Downarrow(w0)\Downarrow(r0)$ 。

不可以正确写故障行为,需先敏化存储单元的状态为 1 或 0,再进行 $w0$ 或 $w1$ 操作,之后进行 $r0$ 或 $r1$ 操作,如果读出数据为 0 或 1,证明该单元没有故障;如果读出数据为 1 或 0,则证明该单元存在故障。针对不可正确写故障的最简化的 March 元素为 $\Downarrow(w0)\Uparrow(w0, r0)$ 或 $\Downarrow(w1)\Uparrow(w1, r1)$ 。

因此,根据前述 3D SRAM 中每一种 TSV 开路故障行为的故障原语定义,可以得到 3D SRAM 中 TSV 开路故障的故障原语及其 March 元素,结果如表 3 所示。

表 3 TSV 开路故障的故障原语及其 March 元素
Tab.3 TSV open-circuit fault primitives and March elements

TSV 开路故障行为	故障原语	March 元素
不可访问	—	$\Downarrow(w0)\Downarrow(r0)$ $\Downarrow(w1)\Downarrow(r1)$
访问多个单元	—	$\Uparrow(r0, w1)\Downarrow(r1, w0)$
可改变浮空单元	$\langle 0w1; 0/\uparrow/- \rangle$ $\langle 1w0; 1/\downarrow/- \rangle$	$\Downarrow(w0)\Downarrow(r0, w1)$ $\Downarrow(w1)\Downarrow(r1, w0)$
读出错误值	$\langle r1; x/\downarrow/0 \rangle$ $\langle r0; x/\uparrow/1 \rangle$	$\Downarrow(w0)\Downarrow(r0)$ $\Downarrow(w1)\Downarrow(r1)$
固定输出“1”	$\langle r0/0/1 \rangle$	$\Downarrow(w1)\Downarrow(r1)$
固定输出“0”	$\langle r1/1/0 \rangle$	$\Downarrow(w0)\Downarrow(r0)$
不可以正确写	$\langle 0; 0w0/\uparrow/- \rangle$ $\langle 1; 1w1/\downarrow/- \rangle$	$\Downarrow(w0)\Uparrow(w0, r0)$ $\Downarrow(w1)\Uparrow(w1, r1)$

2.3 TSV 开路测试算法

表 3 中所提的这些基本故障类型的 March 元素可以为本节 TSV 开路测试算法的研究提供基础,通过对所有测试序列进行算法合并与优化,从而得到符合设计要求的测试算法。经合并优化之后的算法如式(2)所示:

$$\textcircled{1} \quad \textcircled{2}\textcircled{3} \quad \textcircled{4}\textcircled{5}\textcircled{6} \quad \textcircled{7}\textcircled{8} \quad \textcircled{9}$$

$$\Downarrow(w1)\Uparrow(w1, r1)\Downarrow(r1, w0, r0)\Downarrow(r0, w1)\Uparrow(r1) \quad (2)$$

M0 M1 M2 M3 M4

此算法共九个操作步骤,分为五个状态,分别为 $\textcircled{1} \sim \textcircled{9}$ 九个操作步骤和 M0 ~ M4 五个状态:

M0: 存储单元初始化为状态“1”;

M1:按地址增序对所有存储单元进行 $w1$ 和 $r1$ 的操作;

M2:按地址降序对所有存储单元进行 $r1$, $w0$ 和 $r0$ 的操作;

M3:按地址降序对所有存储单元进行 $r0$ 和 $w1$ 的操作;

M4:按地址增序对所有存储单元进行 $r1$ 操作。

式(2)检测故障的原理总结如表4所示。

表4 TSV 开路故障及其 March 元素检测原理

Tab.4 TSV open-circuit fault and the testing theory of

March elements

TSV 开路故障行为	故障原语	March 元素
不可访问	—	M1
访问多个单元	—	M3
可改变浮空单元	$\langle 0w1;0/\uparrow/- \rangle$ $\langle 1w0;1/\downarrow/- \rangle$	M2,1/M2,2;M3
读出错误值	$\langle r1;x/\downarrow/0 \rangle$ $\langle r0;x/\uparrow/1 \rangle$	M2
固定输出“1”	$\langle r0/0/1 \rangle$	M3,2;M4
固定输出“0”	$\langle r1/1/0 \rangle$	
不可以正确写	$\langle 0;0w0/\uparrow/- \rangle$ $\langle 1;1w1/\downarrow/- \rangle$	M0,M1

表4中 March 元素列, M_i 表示测试算法中对第 i 个 March 元素, $M_{i,j}$ 表示测试算法中第 i 个 March 元素的第 j 个操作,如 $M2,2$ 表示第 2 个 March 元素中的第 2 步操作“ $w0$ ”。

检测原理为第 1 个 March 元素 $\uparrow(w1)$ 用于检测不可以正确写故障;第 2 个 March 元素 $\uparrow(w1,r1)$ 用于检测不可访问故障和不可以正确写故障;第 3 个 March 元素 $\downarrow(r1,w0,r0)$ 用于检测读出错误值故障,第 3 个 March 元素 $\downarrow(r1,w0,r0)$ 的第一个元素 $r1$ 和第二个元素 $w0$ 用于检测可改变浮空单元故障;第 4 个 March 元素 $\downarrow(r0,w1)$ 用于检测访问多个单元故障和可改变浮空单元故障,第 4 个 March 元素 $\downarrow(r0,w1)$ 的第二个元素 $w1$ 用于检测固定输出 0 或 1 故障;第 5 个 March 元素 $\uparrow(r1)$ 用于检测固定输出 0 或 1 故障。

3 TSV 开路测试算法实现与验证

前面已经从理论上推导出了 TSV 开路故障的测试算法,本节将根据以上研究成果实现具体的 TSV 开路测试算法,并验证提出的 TSV 测试算

法的正确性和有效性。

3.1 算法代码实现

TSV 开路测试算法共五个状态,本文采用 Verilog 语言实现 TSV 测试算法有限状态机的设计,算法的伪代码描述如算法 1 所示。

算法 1 TSV 开路测试算法有限状态机

Alg.1 FSM of TSV open test algorithm

1. For $I=1$ to n ; Do // 地址增序
2. Write 0 to cell I // 往 cell I 当中写 0
3. Continue
4. For $I=1$ to n ; Do // 地址增序
5. Write 1 to cell I // 往 cell I 当中写 1
6. Read cell I verify 1 // 读 cell I 当中的 1
7. Continue
8. For $I=n$ to 1; Do // 地址降序
9. Read cell I verify 1 // 读 cell I 当中的 1
10. Write 0 to cell I // 往 cell I 当中写 0
11. Read cell I verify 0 // 读 cell I 当中的 0
12. Continue
13. For $I=n$ to 1; Do // 地址降序
14. Read cell I verify 0 // 读 cell I 当中的 0
15. Write 1 to cell I // 往 cell I 当中写 1
16. Continue
17. For $I=1$ to n ; Do // 地址增序
18. Read cell I verify 1 // 读 cell I 当中的 1
19. Continue
20. End

完成整个 TSV 测试算法需要 $9 \times N$ 个循环,因此所有数据完成读写操作所需的总时间为 $9 \times N \times D$,其中 N 为地址深度, D 为数据位宽。

3.2 算法电路实现

本文所提测试算法的一大优势就是可以采用 2D 的测试方法来完成 TSV 开路故障的测试,所以该算法的测试电路依然可以采用典型的 BIST 测试电路,如图 4 所示。

系统工作流程如图 5 所示。

当进入 BIST 模式时,外部控制信号 $BIST_start$ 有效,系统进入测试模式,多路选择器 MUX 将选择从 FSM 控制器中产生的数据作为系统输入,存储器输出的数据被送到输出响应比较器,与 BIST 控制器产生的理想数据进行比对,进而判断存储器是否存在故障,如果存在故障,输出响应器的故障标志位有效,并向冗余修复模块发送故障地址和故障类型,便于后续修复工作。最后,当 FSM 状态控制器根据算法步骤完成所有测试步骤后, $BIST_done$ 信号有效,表示存储器完成了自测试。

当 $BIST_start$ 无效时,系统进入正常的功能

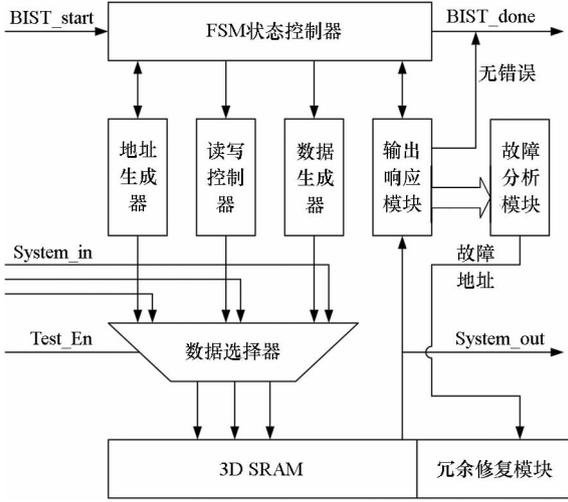


图 4 BIST 测试电路

Fig. 4 BIST test circuit

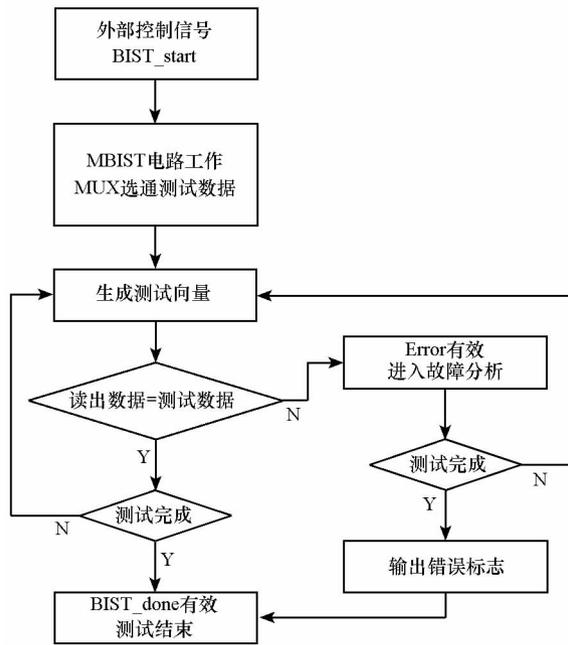


图 5 系统工作流程

Fig. 5 Systematic work flow

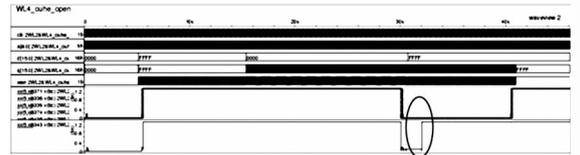
模式,多路选择器 MUX 将选择从 System_in 输入的数据作为系统输入, System_in 和 System_out 有效,如果有错误单元,系统将按照重新分配的新地

址进行数据存储,并由 System_out 输出。

3.3 算法验证

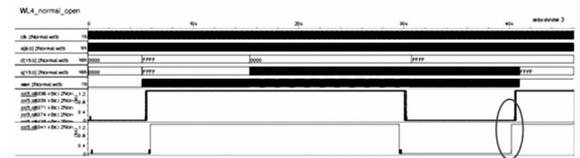
本文采用 Verilog 和 Nanosim 仿真工具,对 3D SRAM 进行测试。仿真中加入 TSV 开路故障模型,通过观察字线开路和位线开路的 TSV 开路测试算法中的测试情况,分析了 TSV 开路测试算法的有效性及正确性。

字线 TSV 存在开路故障的模拟结果如图 6 所示(模拟的是 WL2 和 WL4)。



(a) 注入故障

(a) With fault injection



(b) 未注入故障

(b) Without fault injection

图 6 字线 TSV 注入开路故障模拟结果

Fig. 6 Result of open-circuit fault injection on word line

图 6 中,椭圆为 TSV 测试算法中的某一步写操作。从图 6(b)中椭圆可以看出, WL2 和 WL4 上所接单元都在期望的时间点上写入了正确的数据,而图 6(a)椭圆中的波形明显不同步,这是由于 WL4 所接的 TSV 产生了开路故障,在 WL4 期望的访问时间点上并没有成功写入数据,且在访问 WL2 的时候,由于耦合效应,与 WL2 的写操作同步了,所以在 WL2 写操作的同时, WL4 也进行了相同的写操作。

类似地,可以得到位线的模拟验证结果。首先往任意一根与图 2 中 BL 相连的位线 TSV 中注入开路故障,模拟结果如图 7 所示。

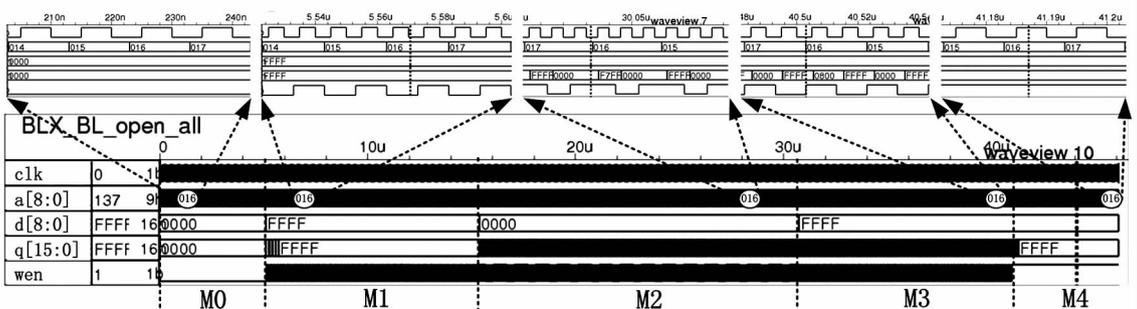


图 7 位线 TSV 注入开路故障模拟结果

Fig. 7 Result of open-circuit fault injection on bit line

从图7可以看出,以016地址(与BL相连地址的其中之一)为例,当与BL相连的TSV存在开路故障时,M1状态读出了“1”,这是正确的结果;M2状态的第一个读操作读出了“0”,这是错误的结果,第二个读操作读出了“1”,这是正确的结果;M3状态读操作读出了“1”,这是错误的结果;M4状态读出了“1”,这也是正确的结果。这样便探测出了位线TSV的开路故障。由于 \overline{BL} 和BL是一对互补位线,因此与 \overline{BL} 相连时TSV存在开路故障时的仿真结果和与BL相连时TSV存在开路故障的仿真结果是一样的。

对于普通字线和普通位线的模拟结果可以采用类似的方法得到,整理结果如表5所示。

根据TSV测试算法的读写操作顺序,对于每个地址输出响应器所期望的输出结果应该是“11001”这样一个序列。通过仔细观察可以发现,当位线TSV存在开路故障时,与故障TSV相连的地址的输出都是“10011”这样一个序列;而当普通位线BL存在开路故障时,与故障位线BL相连的地址的输出都是“00000”这样一个序列;当普通位线 \overline{BL} 存在开路故障时,与故障位线 \overline{BL} 相连的地址的输出都是“11111”这样一个序列。

表5 开路故障类型统计

Tab.5 Classification of open-circuit fault

期望 Q值	$\Downarrow(w0)$	$\Uparrow(w1,r1)$	$\Downarrow(r1,w0,r0)$	$\Downarrow(r0,w1)$	$\Uparrow(r1)$	故障 类型	
情况1	初始 化0	1	1	0	0	1	BL
情况2		0	0	0	0	0	\overline{BL}
情况3		1	1	1	1	1	BL_TSV

因此可以依据某个故障地址的输出来判断是什么类型的故障。当输出结果为情况1时,认为是普通BL存在开路故障;当输出结果为情况2时,认为是普通 \overline{BL} 存在开路故障;当输出结果为情况3时,认为是位线TSV存在开路故障。

4 结论

针对绑定后3D SRAM中TSV的全开路故障展开全面分析,提出一种使用测试算法来探测TSV开路故障的方法。该方法将TSV测试机制与Memory BIST结构相结合,可以在不使用TSV专用测试电路且不增加额外面积开销的情况下解决3D SRAM中TSV的开路测试。模拟结果表明,该TSV测试算法功能正确,能够准确探测到

TSV的开路故障,并快速定位TSV的开路位置,实现了探测TSV开路故障的目的。

参考文献(References)

- [1] Koyanagi M, Fukushima T, Tanaka T. Three-dimensional integration technology and integrated systems[C]//Proceedings of Asia South Pacific Design Automation Conference, Asp-Dac 2009, Yokohama, Japan, 2009: 409-415.
- [2] Khan N, Rao V S, Lim S, et al. Development of 3-D silicon module with TSV for system in packaging[J]. IEEE Transactions on Electronic Components and Packaging Technologies, 2010, 33(1): 3-9.
- [3] Gong S Q, Liu W, Tan J B, et al. Foundry TSV integration and manufacturing challenges[C]//Proceedings of IEEE International Interconnect Technology Conference / Advanced Metallization Conference, 2014: 385-388.
- [4] Swinnen B, Ruythooren W, De Moor P, et al. 3D integration by Cu-Cu thermo-compression bonding of extremely thinned bulk-Si die containing 10 μ m pitch through-Si vias[C]//Proceedings of Electron Devices Meeting, 2006: 1-4.
- [5] Huang Y J, Li J F, Chou C W. Post-bond test techniques for TSVs with crosstalk faults in 3D ICs[C]//Proceedings of International Symposium on VLSI Design, Automation, and Test (VLSI-DAT), 2012: 1-4.
- [6] Pasca V, Anghel L, Benabdenbi M. Configurable Thru-Silicon-Via interconnect Built-In Self-Test and diagnosis[C]//Proceedings of Test Workshop, 2011: 1-6.
- [7] Lin Y H, Huang S Y, Tsai K H, et al. Parametric delay test of post-bond through-silicon vias in 3-D ICs via variable output thresholding analysis[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2013, 32(5): 737-747.
- [8] Cuvillo M, Dey S, Bai X, et al. Fault modeling and simulation for crosstalk in system-on-chip interconnects[C]//Proceedings of IEEE/ACM International Conference on Computer-aided Design, 1999: 297-303.
- [9] Yoon K, Kim G, Lee W, et al. Modeling and analysis of coupling between TSVs, metal and RDL interconnects in TSV-based 3D IC with silicon interposer[C]//Proceedings of the 11th Electronics Packaging Technology Conference. Los Alamitos: IEEE Computer Society Press, 2009: 702-706.
- [10] Kim J, Cho J, Kim J. TSV modeling and noise coupling in 3D IC[C]//Proceedings of Electronic System-Integration Technology Conference, 2010: 1-6.
- [11] 贺翔.面向三维集成的硅通孔互连信号完整性与电气建模研究[D].南京:南京航空航天大学,2012.
HE Xiang. Signal integrity analysis and electrically modeling of through silicon via connection in 3D integration[D]. Nanjing: Nanjing University of Aeronautics and Astronautics, 2012. (in Chinese)
- [12] Woo D, Seong N, Lewis D, et al. An optimized 3D-stacked memory architecture by exploiting excessive, high-density TSV bandwidth[C]//Proceedings of IEEE 16th International Symposium on High Performance Computer Architecture (HPCA), 2010: 1-12.
- [13] Loh G H. 3D-stacked memory architectures for multi-core processors[C]//Proceedings of International Symposium on Computer Architecture. IEEE, 2008: 453-464.