

临界路径跟踪算法中自屏蔽和多路敏化现象的研究

侯明星¹, 何 亮¹, 黄斌科¹, 张喜俊²

(1 西安交通大学 电子与信息工程学院, 陕西 西安 710049; 2 中国电子科技集团公司 第四十一研究所, 山东 青岛 266555)

摘 要: 目前临界路径跟踪算法中确定扇出源临界性时, 未同时考虑扇出源的自屏蔽和多路敏化现象, 进而导致处理结果出现近似问题, 对此研究了自屏蔽和多路敏化现象对扇出源临界性的影响, 提高了算法的准确性. 通过使用 18 值符号仿真, 可以同时检测固定故障和跳变时延故障, 提高了算法的有效性. 使用 C++ 编程语言对改进后的临界路径跟踪算法进行实现, 并应用于 ISCAS'85 标准电路进行故障模拟, 实验结果证明了该算法的准确性和高效性.

关键词: 临界路径跟踪算法; 自屏蔽; 多路敏化; C++

中图分类号: TP31

文献标识码: A

文章编号: 1000-7180(2016)11-0099-05

Study on Self-Masking and Multiple Path Sensitization of Critical Path Tracing Algorithm

HOU Ming-xing¹, HE Liang¹, HUANG Bin-ke¹, ZHANG Xi-jun²

(1 School of Electronics and Information Engineering, Xi'an Jiaotong University, Xi'an 710049, China;

2 The 41st Institute of CETC, Qingdao 266555, China)

Abstract: Current critical path tracing algorithm will lead approximation of results in fault diagnosis, since self-masking and multiple path sensitization in determining criticality of fan-out stem are not considered. The present paper studies the effects of self-masking and multiple path sensitization on determining criticality of fan-out stem, and the accuracy of the algorithm is improved. Based on an 18-valued algebra in symbolic simulation, stuck-at faults and transition faults can be detected simultaneously, and the efficiency of the algorithm is improved. Critical path tracing algorithm, considering self-masking and multiple path sensitization in determining the criticality of fan-out stem, is implemented with the C++ language. The improved algorithm is applied to ISCAS85 standard circuits for fault diagnosis, and experimental results prove the accuracy and efficiency of the algorithm.

Key words: critical path tracing algorithm; self-masking; multiple path sensitization; C++

1 引言

目前常用的数字电路故障模拟算法主要有平行故障模拟法、演绎故障模拟法、同时故障模拟法^[1]和临界路径跟踪法^[2]. 其中前三种算法针对插入的每个故障, 均需进行一次模拟计算, 且计算过程中无法立即确定当前测试向量能否检测该故障, 从而导致大量无效计算. 而临界路径跟踪算法针对给定测试向量, 只进行一次无故障仿真后, 即可由初级输出(PO)向初级输入(PD)跟踪每条临界引线并确定当前测试向量的所有可测故障, 故该算法的效率较其他算法更

高. 然而该算法是一种近似算法, 如何提高该算法的故障模拟精确性, 并提高其效率具有重要价值.

1984 年 M. Abramovici 首次提出了临界路径跟踪算法^[2], 最初该算法不考虑扇出源的临界性, 且只能用于检测固定故障, 即固定 0 故障(SA0)和固定 1 故障(SA1). 1993 年 P. Girard 提出了使用标签传递的方法来确定扇出源临界性^[3], 同时基于 6 值符号仿真, 将临界路径跟踪算法应用到跳变时延故障的检测, 即下降时延故障(SF0)和上升时延故障(SR1), 该方法利用符号值表示逻辑值的跳变, 避免了传统方法在时序仿真中需要考虑时延大小的问

题,但其未考虑 X 值和 Z 值,且忽略了扇出源的自屏蔽现象. 2005 年 Lei Wu 等在判定扇出源的临界性上给出了一系列规则^[4],同时也考虑了 X 值,然而规则的应用不适于多输入逻辑门.

本文同时考虑了自屏蔽和多路敏化对扇出源临界性的影响,并将电路元件扩展为多输入逻辑门,且使用包括 X 和 Z 在内的 18 值真值表进行符号仿真,使得该方法可以同时检测固定故障和跳变时延故障. 文中研究了扇出源的自屏蔽和多路敏化现象,并进行了算法的实现和标准电路的故障模拟仿真.

2 算法理论及应用

2.1 算法理论

临界路径跟踪算法中有如下定义^[2].

定义 1 如果测试向量 t 能检测 l 引线上的 f 故障,则引线 l 在 t 下是临界的.

定义 2 如果改变一个门的输入值即可改变其输出值,则该输入是敏感的.

定义 3 如果一个门至少存在一个敏感输入,则该门为敏感门,否则为非敏感门.

根据上述定义,临界路径跟踪算法的基本处理过程:对于每个测试向量,首先执行无故障符号仿真来决定电路中每条引线的符号值,然后从 PO 开始逆向回溯,根据敏感门的敏感输入跟踪所有临界引线,直到遇到 PI 时停止. 最终每条临界引线都可对应当前测试向量的可测故障.

与其他算法的逻辑仿真不同,本文算法使用符号仿真^[5],规定 $C0, C1, CX, CZ$ 表示稳定信号的符号值, $F0, R1$ 表示具有相反初值和终值的符号值,且基于 18 值真值表进行无故障符号仿真,使得本文算法可以同时检测固定故障和跳变时延故障.

临界路径跟踪算法的核心思想:如果一个门的输出临界,则其敏感输入及该输入的前驱也临界. 然而当回溯过程遇到扇出源时,不能由任意一条扇出分支的临界性就直接决定扇出源的临界性. 因此,要确定扇出源的临界性,需首先判断其是否发生自屏蔽或多路敏化现象.

具体来说,当扇出源的故障效应沿着两条以上路径进行传递时,若以相反的极性汇聚到一个敏感门后,故障效应的传递被阻止,则扇出源发生自屏蔽;当扇出源的故障效应沿着两条以上路径进行传递时,若汇聚到一个非敏感门后,故障效应仍可继续传递,则扇出源发生多路敏化.

本文算法使用标签传递的方法^[3]判定扇出源

是否发生自屏蔽或多路敏化,一个标签定义为 (G_i, N_i) ,其中, G_i 为产生标签的门, N_i 为 G_i 的输入序号,标签可通过相关输入传递到对应前驱,且 $0 \leq i < k, k$ 为 G_i 的输入个数. 门 G_j 中包含的所有标签组成该门的标签集,记为 $\text{Lab}(G_j)$.

接下来具体讨论自屏蔽和多路敏化现象的分析过程.

2.1.1 自屏蔽

当一个汇聚门 RG 敏感时,将与其相关且涉及自屏蔽的扇出源标记为待分析, RG 产生与所有输入相关的标签,标签通过各自输入传递到前驱后,进一步沿着各敏感门的敏感输入向着 PI 方向传递,当到达待分析扇出源时,如果该扇出源包含汇聚门相关输入的所有标签,则扇出源发生自屏蔽.

自屏蔽对扇出源临界性的影响:如果扇出源的所有扇出分支都不临界,则扇出源不临界;如果任一扇出分支临界且扇出源发生自屏蔽,则扇出源不临界.

以图 1 所示电路为例,对自屏蔽的判定过程以及自屏蔽对扇出源临界性的影响进行说明. 电路图中敏感输入以黑点标记,各引线符号值是在给定测试向量为 $(F0, C0, C1)$ 时,基于 18 值真值表进行无故障符号仿真得到的.

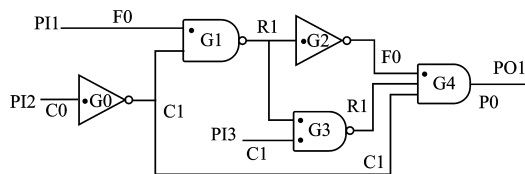


图 1 自屏蔽实例

回溯跟踪过程开始于 $PO1$,并设定 $PO1$ 临界,具体处理过程如下:

(1)处理 $G4$: $G4$ 临界且敏感,将汇聚门为 $G4$ 且涉及自屏蔽的扇出源 $G0$ 和 $G1$ 标记为待分析, $G4$ 产生的 3 个标签 $(G4, 0)$, $(G4, 1)$ 和 $(G4, 2)$ 分别传递到 $G2, G3$ 和 $G0$;

(2)处理 $G2$: $G2$ 临界且敏感,因此 $\text{Lab}(G2) = \{(G4, 0)\}$ 传递到 $G1$;

(3)处理 $G3$: $G3$ 敏感,因此 $\text{Lab}(G3) = \{(G4, 1)\}$ 传递到 $PI3$ 和 $G1$;

(4)处理 $G1$: $G1$ 是待分析扇出源且存在临界的扇出分支, $G4$ 为其汇聚门, $\text{Lab}(G1) = \{(G4, 0), (G4, 1)\}$,该标签集并包含 $G4$ 产生的所有相关输入标签,因此 $G1$ 发生自屏蔽, $G1$ 不临界, $G1$ 敏感, $\text{Lab}(G1)$ 传递到 $PI1$;

(5)处理 $G0$: $G0$ 的所有扇出分支都不临界,故

G0 不临界。

(6)跟踪到达初级输入 PI1, PI2, PI3, 算法停止。

依照上述算法过程,即可确定数字电路中任意扇出源涉及自屏蔽现象时的临界性。

2.1.2 多路敏化

当一个汇聚门 RG 为非敏感门时,将与其相关且涉及多路敏化的扇出源标记为待分析, RG 产生与控制输入^[1]相关的标签,标签通过各自输入传递到前驱门后,进一步沿着敏感门的敏感输入向着 PI 方向传递,当到达待分析扇出源时,如果该扇出源包含汇聚门相关输入的所有标签,则扇出源发生多路敏化。

多路敏化对扇出源临界性的影响,可分为汇聚门临界和不临界两种情况,即简单多路敏化和复杂多路敏化。

对于发生多路敏化的扇出源,如果汇聚门临界,扇出源也临界;如果汇聚门不临界,扇出源也不临界,并将汇聚门标签集合并到扇出源标签集,再去除由汇聚门产生的标签后,扇出源标签集继续向前传递。

以图 2 所示电路为例,对简单多路敏化的判定过程以及简单多路敏化对扇出源临界性的影响进行说明。电路图中敏感输入以黑点标记,各引线符号值是在给定测试向量为(C1, R1, C0)时,基于 18 值真值表进行无故障符号仿真得到的。

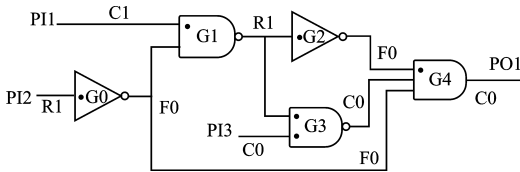


图 2 简单多路敏化实例

回溯跟踪过程开始于 PO1,并设定 PO1 临界,具体处理过程如下:

(1)处理 G4:G4 为非敏感门,将汇聚门为 G4 且涉及多路敏化的扇出源 G0 标记为待分析,G4 产生的 3 个标签(G4, 0),(G4, 1)和(G4, 2)分别传递到 G2,G3 和 G0;

(2)处理 G2:G2 敏感,因此 $Lab(G2) = \{(G4, 0)\}$ 传递到 G1;

(3)处理 G3:G3 敏感,因此 $Lab(G3) = \{(G4, 1)\}$ 传递到 PI3;

(4)处理 G1:G1 敏感,因此 $Lab(G1) = \{(G4, 0)\}$ 传递到 G0;

(5)处理 G0:G0 为待分析的扇出源,其汇聚门为 G4, $Lab(G0) = \{(G4, 0), (G4, 2)\}$,

该标签集并没有包含 G4 相关输入的所有标签,故 G0 不发生多路敏化,G0 不临界;

(6)跟踪到达初级输入 PI1, PI2, PI3, 算法停止。

以图 3 所示电路为例,对复杂多路敏化的判定过程以及复杂多路敏化对扇出源临界性的影响进行说明。电路图中敏感输入以黑点标记,各引线符号值是在给定测试向量为(C1, F0, C1, C1)时,基于 18 值真值表进行无故障符号仿真得到的。

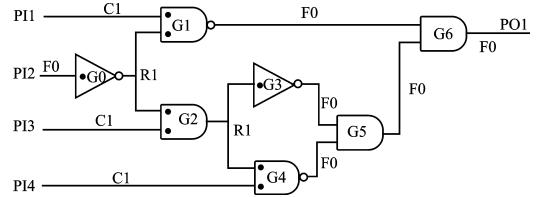


图 3 复杂多路敏化实例

回溯跟踪过程开始于 PO1,并设定 PO1 临界,具体处理过程如下:

(1)处理 G6:G6 为非敏感门,将汇聚门为 G6 且涉及多路敏化的扇出源 G0 标记为待分析,G6 产生的 2 个标签(G6, 0)和(G6, 1)分别传递到 G1 和 G5;

(2)处理 G5:G5 为非敏感门,将汇聚门为 G5 且涉及多路敏化的扇出源 G2 标记为待分析,G5 产生的 2 个标签(G5, 0)和(G5, 1)并分别传递到 G4 和 G3;

(3)处理 G4:G4 敏感,因此 $Lab(G4) = \{(G5, 1)\}$ 传递到 G2 和 PI4;

(4)处理 G3:G3 敏感,因此 $Lab(G3) = \{(G5, 0)\}$ 传递到 G2;

(5)处理 G2:G2 为待分析扇出源,其汇聚门为 G5, $Lab(G2) = \{(G5, 0), (G5, 1)\}$,故 G2 发生多路敏化,但 G5 不临界,G2 也不临界,将 $Lab(G5)$ 合并到 $Lab(G2)$,再去除 G5 产生的标签后得到 $Lab(G2) = \{(G6, 1)\}$,G2 敏感, $Lab(G2)$ 传递到 G0 和 PI3;

(6)处理 G1:G1 敏感,因此 $Lab(G1) = \{(G6, 0)\}$ 传递到 G0 和 PI1;

(7)处理 G0:G0 为待分析的扇出源,其汇聚门为 G6, $Lab(G0) = \{(G6, 0), (G6, 1)\}$,故 G0 发生多路敏化,因 G6 临界,G0 也临界;

(8)跟踪到达初级输入 PI1, PI2, PI3, PI4, 算法停止。

依照上述算法过程,即可确定数字电路中任意扇出源涉及多路敏化现象时的临界性。

2.2 算法应用

本文改进的临界路径跟踪算法可确定电路中所有引线的临界性,最终根据每条临界引线的符号值即可确定当前测试向量的可测故障类型,规则如下:

- (1)如果引线临界且符号值为 C0 或 P0,则可检测该引线上的 SA1 故障;
- (2)如果引线临界且符号值为 C1 或 P1,则可检测该引线上的 SA0 故障;
- (3)如果引线临界且符号值为 F0,则可检测该引线上的 SA1 故障和 SF0 故障;
- (4)如果引线临界且符号值为 R1,则可检测该引线上的 SA0 故障和 SR1 故障.

3 程序实现

改进后的临界路径跟踪算法的程序实现过程为:(1) 输入使用 Verilog 语言描述的待测电路网表文件,进行电路建模和结构分析,输出电路的汇聚回路(包括扇出源、对应汇聚门以及二者之间的内部引线);(2) 输入测试向量集和电路汇聚回路,使用临界路径跟踪算法进行故障模拟,最后输出测试向量集的故障覆盖率.

本文算法程序使用 C++ 编程实现,图 4 和图 5 所示分别为算法程序中主函数 CPT()和函数 Sensitive_Path()的伪代码.

```
CPTO{
    PO临界;
    while (待处理队列不为空) {
        门G出队列;
        if (G为待分析扇出源) {
            if (G涉及自屏蔽且任一扇出分支临界) {
                Check_Masking(G);
                if (G发生自屏蔽)
                    G不临界
            }
            if (G涉及多路敏化) {
                Check_Multiple_Sens(G);
                if (G发生多路敏化) {
                    if (RG临界)
                        G临界;
                    else
                        Lab(G)=Lab(G)∪Lab(RG);
                        Lab(G)中取出RG产生的标签;
                }
            }
        }
        if (G临界)
            将门G的可测故障放于故障列表中;
            Sensitive_Path ();
    }
}
```

图 4 主函数 CPT()的伪代码

图 4 中主函数 CPT()的功能是:当处理待分析扇出源 G 时,函数 Check_Masking(G)确定 G 是否包含其敏感汇聚门 RG 相关输入的所有标签,如果是,则 G 发生自屏蔽,类似的,函数 Check_Multiple_Sens(G)确定 G 是否包含其非敏感汇聚门 RG 相关输入的所有标签,如果是,则 G 发生多路敏化,继

```
Sensitive_PathO{
    if (门G不是PI){
        if (G是敏感门){
            if (如果G是汇聚门) {
                相关且涉及自屏蔽的扇出源标记为待分析;
                for (对G的第i个输入)
                    i的前驱为FG,Lab(FG) = (G,i);
            }
            if (G临界) {
                for (对G的第i个输入) {
                    if (i是控制输入) {
                        i临界
                        i的前驱FG临界且Lab(FG)=Lab(G);
                    }
                    未被处理过的i前驱放入待处理队列;
                }
            }
        }
    }
    if (G是非敏感门) {
        if (如果G是汇聚门) {
            相关且涉及多路敏化的扇出源标记为待分析;
            for (对G的第i个输入) {
                if (i是控制输入)
                    i的前驱为FG,Lab(FG)=(G,i);
                未被处理过的i前驱放入待处理队列;
            }
        }
    }
}
```

图 5 函数 Sensitive_Path()的伪代码

而可进一步确定该扇出源的临界性,并将可测故障放于故障列表中;当处理非扇出源时,直接执行函数 Sensitive_Path().

图 5 中函数 Sensitive_Path()的功能是:当处理敏感的汇聚门 RG 时,将涉及自屏蔽的扇出源标记为待分析,RG 产生与输入相关的标签并传递到对应前驱;当处理非敏感的汇聚门 RG 时,将涉及多路敏化的扇出源标记为待分析,RG 产生与控制输入相关的标签并传递到对应前驱;当处理其他情况时,直接确定非扇出源的临界性且将标签集沿着敏感门的敏感输入进行传递.

4 实验结果及分析

使用改进后的临界路径跟踪算法对文中三个实例电路进行故障模拟,程序运行结果如表 1 所示,将该结果与理论分析结果进行对比,二者相一致,证明了算法程序实现的正确性.以图 1 电路为例,应用本文算法对扇出源 G0 和 G1 的自屏蔽现象进行分析,最终得到电路中只有 G2 和 G4 临界,进一步可得到测试向量的可测故障.

表 1 三个实例电路的故障模拟结果

电路	测试向量	可测故障
图 1	(F0, C0, C1)	G4:SA1 故障 G2:SA1 故障和 SF0 故障
图 2	(C1, R1, C0)	G4:SA1 故障
图 3	(C1, F0, C1, C1)	G6:SA1 故障和 SF0 故障 G0:SA0 故障和 SR1 故障

其次,分别使用本文算法程序和另一故障仿真工具 LIFTING [6]对 ISCAS'85 标准电路进行故障

模拟,并对二者的故障覆盖率和CPU耗时进行对比,结果如表2所示,其中故障仿真工具LIFTING使用平行故障模拟法,且r(SAF)和r(STF)分别表

表2 不同算法对ISCAS'85标准电路的故障模拟结果对比

电路	逻辑门个数	测试向量个数	LIFTING		本文算法		
			CPU耗时/s	r(SAF)/%	CPU耗时/s	r(SAF)/%	r(STF)/%
C17	6	9	0.063	68.00	0.015	68.00	56.00
C432	160	76	10.545	96.48	3.073	97.05	93.41
C499	202	83	35.004	96.05	4.511	97.10	95.46
C880	383	97	49.984	96.33	4.801	96.70	94.24
C1355	546	128	279.29	95.76	56.96	95.63	90.94

由表2可知,对于同一电路,使用相同的测试向量集,本文算法的固定故障覆盖率接近于故障仿真工具LIFTING,CPU耗时却大幅降低,且随电路复杂度的增加,检测效率的提升更加显著.另外,本文算法还可同时检测跳变时延故障,证明了本文算法的精确性和高效性.

5 结束语

本文同时考虑了自屏蔽和多路敏化现象对扇出源临界性的影响,并使用18值真值表进行符号仿真,本文算法可以同时检测固定故障和跳变时延故障,从而实现了临界路径跟踪算法的改进和完善.与传统的故障仿真方法相比,改进后的临界路径跟踪算法在故障模拟精度和诊断效率方面都有了显著提升,证明了本文算法的精确性和高效性.

参考文献:

[1] 杨士元. 数字系统的故障模拟与可靠性设计[M]. 北京:清华大学出版社,2000.

[2] Abramovici M, Menon P R, Miller D T. Critical proceedings of the 20th path tracing - An alternative to fault simulation [C]// IEEE Design Automation Conference, Piscataway NJ,USA;ACM, 1983; 214-220.

[3] Girard P, Landrault C, Pravossoudovitch S. A reconvergent fanout analysis for the CPT algorithm used in

示固定故障和跳变时延故障的故障覆盖率. 程序运行环境为Windows平台的Visual Studio 2010,计算机配置:内存为4 GB,CPU为AMD Phenom x4.

delay-fault diagnosis [C]// proceedings of the date European Test Conference, Rotterdam, Netherlands; IEEE, 1993; 83-88.

[4] Wu L, Walker D M H. A fast algorithm for critical path tracing in VLSI digital circuits [C]// IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems. Monterey, CA, USA; IEEE, 2005; 178-186.

[5] Bosio A, Girard P, Pravossoudovich S, et al. SoC symbolic simulation: a case study on delay fault testing [J]. IEEE Workshop on Design Diagnostic of System, 2008,16(5):966-975.

[6] Bosio A, Di Natale G. LIFTING: a Flexible Open-so simulator [C]// proceedings of the 20087 17th Asian Test Symposium, Washington,DC,USA;ACM,2008; 35-40.

作者简介:

侯明星 女,(1990-),硕士. 研究方向为数字电路建模和故障模拟.

何亮 男,(1975-),博士,讲师. 研究方向为数据挖掘与软件工程.

黄斌科(通讯作者) 男,(1974-),博士,副教授. 研究方向为微波毫米波器件、数字电路故障诊断.
E-mail: bkhuang@mail. xjtu. edu. cn.

张喜俊 男,(1980-),硕士,高级工程师. 研究方向为测试测量仪器软件.

(上接第98页)

[8] 肖毅华,张浩锋,胡德安,等. 基于改进PIB搜索法的SPH方法在高速冲击模拟中的应用[J]. 振动与冲击, 2015,34(23):88-92.

[9] 杨秀峰,刘谋斌. 光滑粒子动力学SPH方法应力不稳定性的一种改进方案[J]. 物理学报, 2012, 61(22): 255-262.

[10] 张中兴,李冬梅,李涛,等. 一种新的有效模拟不可压缩流体的SPH方法[J]. 计算机应用研究, 2015, 32(3):946-949.

[11] 李会珍. 采用SPH方法的拉格朗日方式流体运动模拟[D]. 合肥:安徽大学,2013.

作者简介:

梁志剑 男,(1978-),博士,副教授,硕士生导师. 研究方向为仿真与可视化、计算机技术.
E-mail: zhijianliang@163. com.

周文娟 女,(1991-),硕士研究生. 研究方向为仿真与可视化.