https://doi.org/10.1051/jnwpu/20244230506

寄存器传输级硬件设计信息流建模与安全验证

秦茂源[1], 侯佳滢[1], 李家乐[1], 唐时博[2], 邰瑜[2]

(1.西安工业大学 计算机科学与工程学院, 陕西 西安 710021;
2.西北工业大学 网络空间安全学院, 陕西 西安 710072)

摘 要:近年来,已有大量研究证明信息流分析能够有效地对设计安全属性与安全行为进行建模。然而,现有的门级抽象层次的信息流分析方法往往受制于算力和验证效力等因素难以应对大规模设计,而 RTL 抽象层次的信息流分析方法需借助类型系统等形式化语言对硬件设计进行重新描述。因此,提出了一种寄存器传输级硬件设计信息流建模与安全验证方法。该方法在寄存器传输级功能模型的基础上构建附加安全属性的信息流跟踪逻辑模型,从信息流角度建模设计安全行为和安全属性,并利用 EDA 测试验证工具,以无干扰为策略捕捉违反安全策略的有害信息流,检测硬件设计安全漏洞。以 Trust-Hub 硬件木马测试集为测试对象的实验结果表明:所提方法能够有效检测设计内潜藏的硬件木马。

关 键 词:硬件安全;信息流安全模型;信息流安全验证;安全漏洞检测
中图分类号:TP309 **文献标志码**:A **文章编号**:1000-2758(2024)03-0506-08

近年来的研究表明,计算机硬件可能因潜藏的硬件木马、侧信道等安全漏洞遭受到严重的安全威胁。例如,毛保磊等[1]证明了攻击者可利用 RSA 硬件密码算法核的时间侧信道以滑动窗口攻击方式提取密钥。传统的功能性验证由于缺乏安全相关的属性和模型,难以验证硬件设计的安全性[2]。信息流分析提供了一种形式化手段可对设计安全属性进行建模,能够克服功能性电路模型在安全属性描述和验证能力上的不足。

设计阶段的硬件信息流分析技术主要是在设计阶段建模数据流动,以仿真测试、形式化验证等手段检查设计内信息流动是否符合信息流安全策略,从而实现设计安全验证。设计阶段的硬件信息流分析技术主要包含信息流跟踪[2-4]和类型系统检查[5-6]2 个大类。信息流跟踪是在功能模型基础上构造附加的信息流跟踪电路,并通过捕捉不同安全域间的非法数据流动检查设计安全性。例如,Hu 等[7]提出的一种门级信息流跟踪方法(gate level information flow tracking,GLIFT)。该方法采用单比特标签标记数据的安全属性,考虑污染输入对输出的影响,构建标准门级网表的门级信息流跟踪逻辑模型。然而,门级信息流跟踪逻辑模型在建模隐式流时保守处理单变量翻转情况,可能捕捉到未实际存在的信息流,引发误报[8-9]。类型系统检查采用类型系统语言对设计规范和安全属性进行描述,进而结合静态验证、定理证明等手段检查设计行为是否符合安全策略。例如,证明可携带硬件(proof carrying hardware,PCH)框架提供了一种基于高阶逻辑的形式化硬件描述语言 Formal-HDL,用于翻译设计和定义信号的安全级别,进而结合定理证明器以人机交互方式开发设计安全行为的证明[10-11]。再如,SecVerilog 借助类型系统对 Verilog 语言语法和语义进行扩展,采用标记(annotations)方式向代码每个变量显式添加安全类型,进而结合编译器在编译阶段静态检查是否违反信息流安全策略[5]。然而,设计翻译和添加注解需要设计验证人员对形式化语言有深入了解,且往往伴随巨大的工作量,以及可能的人为错误。

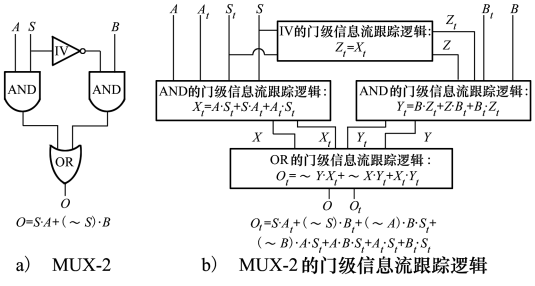
针对上述硬件信息流分析技术的问题,本文提出了一种寄存器传输级(register transfer level,RTL)

收稿日期:2023-05-27
基金项目:国家自然科学基金(U23B2041,62074131)、西安市碑林区科技计划(GX2136)与陕西省科技计划(2022JM-379)资助
作者简介:秦茂源(1985—),讲师
通信作者:邰瑜(1982—),助理研究员 e-mail:taiyu@nwpu.edu.cn

硬件设计信息流建模与安全验证方法。该方法在寄存器传输级设计功能性模型的基础上构建了一种可与 EDA 流程整合的寄存器传输级信息流跟踪逻辑模型(RTLIFT 逻辑模型),能够精确捕捉 RTL 设计内所有实际存在的信息流,继而可结合 EDA 测试验证工具验证安全属性,检测违反信息流安全策略的设计安全漏洞。

1　信息流跟踪概述

信息流跟踪为数据分配用以反映数据安全属性的污染标签,并按照一定的标签传播规则构造可传播安全属性的信息流跟踪(information flow tracking, IFT)逻辑模型。当数据在功能电路中参与运算时,数据的污染标签也随之在 IFT 逻辑模型中传播。因而,通过观测污染标签的流动即可捕捉违反安全策略的数据流动。为便于理解,本文以 GLIFT 为例具体说明信息流跟踪技术。



a)　MUX-2　　b)　MUX-2 的门级信息流跟踪逻辑

图 1　门级信息流跟踪

GLIFT 采用了一种精确的标签传播规则,考虑受污染输入对输出的影响。图 1a)所示为 1 个两输入选择器(MUX-2)。其中,A, B, S, O 分别表示 MUX-2 的 2 个输入、选择线和输出。其 GLIFT 逻辑可通过分步构造 2 个与门、1 个或门和 1 个非门的 GLIFT 逻辑模型而得出,如图 1b)所示。其中,A_t, B_t, S_t, O_t 分别表示 A, B, S, O 的污染标签。若选择线 S 为非污染(即 $S_t=0$),输入 A 和 B 的逻辑值不相同且只有 1 个污染时(即 $A_t=1$ 或 $B_t=1$),输出是否污染与 S 的取值相关。若选择线 S 为污染的(即 $S_t=1$),2 个输入的逻辑值不相同时,输出必然是污染的。然而,若选择线 S 为污染的,2 个输入相同为真且不污染时,GLIFT 逻辑会保守地标记输出为污染,产生误报(即,输入相同为真时,选择线逻辑值的改变并不会导致输出发生变化,输出应该是未污染的)。其根本原因在于,以分步构造的 MUX-2 的 GLIFT 逻辑会包含 1 个冗余最小项 ABS_t。当选择线 S 为污染且 2 个输入相同为真且未污染时,输出总是污染的。针对 GLIFT 逻辑的不足,通过提高模型的抽象层次,构建 RTL 抽象层次的 IFT 逻辑模型。

2　RTL 信息流安全验证方法

寄存器传输级硬件设计信息流建模与安全验证方法,如图 2 所示,包含 RTL 设计信息流建模和安全属性验证 2 个组成部分。其中,RTL 设计信息流建模是利用 IFT 技术将 RTL 设计建模为附加信息流跟踪逻辑的寄存器传输级信息流跟踪逻辑模型(RTLIFT 逻辑模型)。安全属性验证是在 RTLIFT 逻辑模型上以仿真测试、形式化验证等手段验证安全属性检测设计安全漏洞。

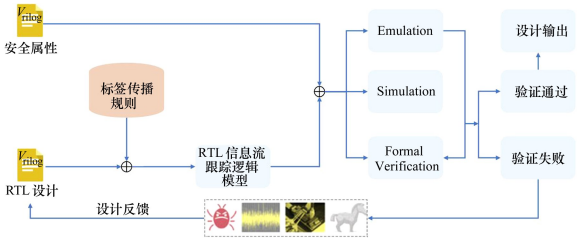


图 2　RTL 设计信息流安全验证方法

2.1　RTLIFT 逻辑模型

RTLIFT 逻辑模型以细粒度污染标签标记每个数据的比特位,用于表征该数据的安全级别(逻辑“1”和“0”分别表示污染的和非污染的安全属性)。当数据传播时,数据的标签也随之在 RTLIFT 逻辑中传播。标签传播过程中考虑受污染输入对输出的影响,当且仅当受污染的输入改变对输出造成影响时,输入的污染标签传递给输出,导致输出污染。为便于理解,本文分别以连续赋值语句和条件判断语句为例具体说明 RTLIFT 逻辑模型。

2.1.1　连续赋值语句的 RTLIFT 逻辑

连续赋值语句是 Verilog 数据流建模的基本语句,用于对线网 wire 进行赋值。其基本语法如公式(1)所示。其中,LHS_target 和 RHS_expression 分别为赋值语句的左侧表达式和右侧表达式。由于该语句中数据是显式流动的,因而污染的 RHS_expression 必然会将污染标签传递给 LHS_target,导致 LHS_target 的污染。因此,该语句的 RTLIFT 逻辑

可被表示为公式(2)所示的连续赋值语句。其中,RHS_expression_t 和 LHS_target_t 分别表示 LHS_target 和 RHS_expression 的污染标签。通常情况下,RHS_expression 主要由线网/寄存器类型的变量、多个变量的运算组成。因此,RHS_expression 的污染标签可通过构造门级信息流跟踪逻辑计算得出。以语句 assign $o[2:0]=x[2:0]\ \&\ y[2:0]$ 为例。在该语句中,$x[2:0]$ 和 $y[2:0]$ 按位与运算。因此,它的污染标签可通过构造 $x[0]\ \&\ y[0]$,$x[1]\ \&\ y[1]$ 和 $x[2]\ \&\ y[2]$ 的门级信息流跟踪逻辑计算得出,即 $x_t[2:0]\ \&\ y[2:0]+y_t[2:0]\ \&\ x[2:0]+x_t[2:0]\ \&\ y_t[2:0]$。其中,$x_t$ 和 y_t 分别表示 x 和 y 的污染标签。由此可得,语句 assign $o[2:0]=x[2:0]\ \&\ y[2:0]$ 的 RTLIFT 逻辑可表示为图 3 所示的信息流跟踪逻辑。

assign LHS_target = RHS_expression (1)

assign LHS_target_t = RHS_expression_t (2)

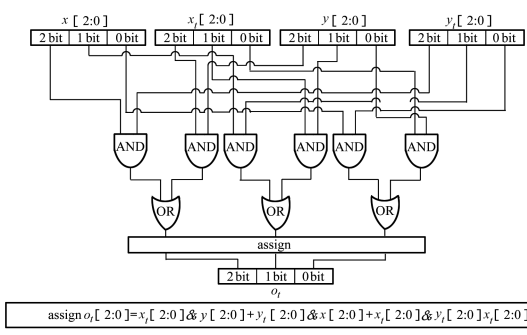


图 3 按位与逻辑为输入的连续赋值语句的 RTLIFT 逻辑

同理,可构造以位拼接、位选择、按位或、按位取反等操作为输入的连续赋值语句的 RTLIFT 逻辑,如表 1 所示。

表 1 部分连续赋值语句的 RTLIFT 逻辑

原始语句	RTLIFT
assign $o=\{x[2:0],y[2:0]\}$	assign $o_t=\{x_t[2:0],y_t[2:0]\}$
assign $o=y[1:0]$	assign $o_t=y_t[1:0]$
assign $o=\{3\{y[1]\}\}$	assign $o_t=\{3\{y_t[1]\}\}$
assign $o=y[2:0]+x[2:0]$	assign $o_t=x_t[2:0]\ \&\ \overline{y[2:0]}+y_t[2:0]\ \&\ \overline{x[2:0]}+x_t[2:0]\ \&\ y_t[2:0]$
assign $o=\overline{x[2:0]}$	assign $o_t=x_t[2:0]$

2.1.2 条件判断语句的 RTLIFT 逻辑

不同于连续赋值语句,条件判断语句受到显式流和隐式流两方面的影响。因而,在构造条件判断语句的 RTLIFT 逻辑时,需要考虑判断条件以及各条件分支的输入对输出的影响。一般情况下,当判断条件为污染时,判断条件的翻转必定改变执行路径,造成输出的污染。实际上,若各分支的输入逻辑值相同,判断条件的翻转并不会改变输出的逻辑值。因而,输出的污染标签仅与分支的输入污染标签相关。反之,若各分支的输入逻辑值不相同时,输出的污染标签与判断条件的污染标签以及分支的输入污染标签皆相关。此外,若判断条件中个别位为污染时,其翻转并不改变判断条件的真值状态。因而,输出的污染标签仅与分支的输入污染标签相关。考虑以上所述条件,以图 4a)所示的 if-else 条件判断语句为例构造 RTLIFT 逻辑。

表 2 为 if-else 语句的 RTLIFT 逻辑的部分逻辑真值表。其中,sel,x,y,out 分别为判断条件、分支 1 的输入、分支 2 的输入、输出,sel_t,x_t,y_t,o_t 分别为它们的污染标签。如表 2 第一行(sel 为 001,x 为 000,y 为 001,sel_t 为 001,x_t 为 000,y_t 为 000)。当 sel 的第 0 位为污染的且未污染的输入 x,y 的逻辑值不相同时,sel 最低位的翻转会将判断条件的真值从 true 变为 false,执行分支 2,导致输出发生变化。因而,输出的最低位是污染的。再如表 2 第二行,当 sel 的第一位为污染的且未污染的 x,y 的逻辑值不相同时,sel 的第一位翻转并不会改变整个判断条件的真值,不会导致输出发生变化,因而输出一定是未污染的。再如表 2 第三行,当 sel 的第一位和 x 的第 0 位为污染的且 x,y 相同时,sel 的第 1 位翻转并不会改变整个判断条件的真值,而 x 的第 0 位翻转会导致输出发生变化,因而输出一定是污染的。如表 2 第四行,当 sel 和 x 的第 0 位为污染的且 x 和 y 的逻辑值不相同时,sel 的第 0 位翻转会改变整个判断条件的真值,导致输出发生变化,因而输出一定是污染的。再如表 2 第五行,当 sel 和 x 的第 0 位为污染的且 x,y 的逻辑值相同时,sel 的第 0 位翻转会改变整个判断条件的真值,然而输入 x 和 y 的逻辑值相同

且 y 为未污染的,不会改变输出,因而输出一定是未污染的。由此可得,if-else 语句的 RTLIFT 逻辑可表示为图 4b)所示的信息流跟踪逻辑。

```
1: module if-else(sel,clk,x,y,out);
2: input clk;
3: input[2:0]sel,x,y;
4: output[2:0]out;
5: always@(posedge clk)
6: begin
7:     if(sel) begin
8:     out<=x;end /**********分支 1**********/
9:     else begin
10:    out<=y;end /**********分支 2**********/
11: end endmodule
```

a) 原始 if 语句

```
1: module if-elss-RTLIFT(sel,clk,x,y,out,sel_t,clk_t,x_t,y_t,out_t);
2: input clk,clk_t;
3: input[2:0]sel,x,y,sel_t,x_t,y_t;
4: output[2:0]out,out_t;
5: always@(posedge clk)
6: begin
7:     if(sel) begin
8:     out<=x;
9:     out_t<=x_t | ({3{(((~sel[0])&(~sel[1])&(sel_t[2])) |
10:                    ((~sel[0])&(~sel_t[1])&(sel[2])) |
11:                    ((sel_t[0])&(~sel[1])&(sel[2])) |
12:                    ((sel_t[0])&(sel_t[1])&(sel_t[2])))}}
13:                    &((x_t & y_t) | (x^y))); end
14:    else begin
15:    out<=y;
16:    out_t<=y_t | ({3{(((~sel[0])&(~sel[1])&(sel_t[2])) |
17:                    ((~sel[0])&(sel_t[1])&(~sel[2])) |
18:                    ((sel_t[0])&(~sel[1])&(~sel[2])) |
19:                    ((sel_t[0])&(sel_t[1])&(sel_t[2])))}}
20:                    &((x_t&y_t) | (x^y))); end
21: end endmodule
```

b) if 语句的信息流跟踪逻辑

图 4　if 语句 RTLIFT 逻辑

表 2　if-else 语句 RTLIFT 逻辑的部分逻辑真值表

序号	sel	x	y	sel_t	x_t	y_t	out_t
1	001	000	001	001	000	000	001
2	001	000	001	010	000	000	000
3	001	001	001	010	001	000	001
4	001	000	001	001	001	000	001
5	001	001	001	001	001	000	000

2.2　安全属性验证

手动建模 RTL 设计电路往往非常耗时,且伴随人为错误的风险。为此,设计了一个基于 Python 的自动化建模工具。该工具以 RTL 设计电路的 AST 文件作为输入直接输出 Verilog 编写的 RTLIFT 逻辑模型。由于 RTLIFT 逻辑模型完全以 Verilog 描述,因而可与 EDA 流程进行整合,利用 EDA 测试验证工具以仿真测试或形式化验证等手段验证设计电路的 RTLIFT 逻辑模型是否违反安全属性。验证采用无干扰的强安全策略,即污染的数据不能流向未污染的区域。例如,根据 RTL 设计的安全需求,从机密性或完整性角度对设计进行安全域划分,将保密的或者不可信的数据标记为污染的,而将非密的或可信的区域标记为未污染的,验证污染的标签流动是否违反了安全策略。若污染的标签流向了未污染的区域时,则表明设计中存在违反安全策略的安全漏洞;反之,则表明设计是安全可信的。

3　实验结果及分析

3.1　实验设计

本文以 if-else 条件语句和 Trust-Hub 的部分 AES 硬件木马测试集[12]为测试验证对象,采用 RTL 硬件设计信息流安全验证方法和门级信息流分析方法分别建模,利用形式化验证工具 Questa Formal 验证设计安全属性,以验证 RTL 硬件设计信息流安全验证方法的精确性和检测硬件木马的有效性。

3.2　RTLIFT 模型的精确性验证

图 4a)所示模块 if-else 实例了一个以 sel 为判断条件的 if-else 语句。该语句中,判断条件 sel 会隐式地流向输出 out。门级信息流模型会保守地处理该隐式流。仅考虑污染的 sel 对执行分支的影响,忽略了各分支输入逻辑值对输出的影响,直接将 sel 的污染标签传递给输出,导致输出 out 的所有位为污染的。相反,RTLIFT 逻辑模型充分考虑判断条件与分支输入逻辑值对输出的影响,仅污染发生改变的输出位。根据上述分析,分别将模块 if-else 建模为 RTLIFT 逻辑模型和门级信息流模型,并约束 sel,x,y 以及它们的标签 sel_t,x_t,y_t 为 3′b010,3′b010,3′b111,3′b010,3′b000,3′b000,以图 5a)~5c)所示的断言(P1,P2,P3 和 P4)验证门级信息流模型和 RTLIFT 逻辑模型输出 out 所有位的标签是否污染。

图 5b)和图 5d)分别给出断言的验证结果。P1-P3 的验证结果表明:当 sel[1]为污染时,门级信息流模型会保守地将输出 out 的所有位标记为污染的,即 out_t=3′b111。实际上,虽然 sel[1]的变化改变了判断条件的真值(True 变为 False),但输出 out

[1]并未发生变化,仍保持逻辑“1”。因此,out 的污染标签应该为 3′b101,产生了误报。P4 的验证结果表明:RTLIFT 逻辑模型仅污染了 out 的第 0 位和第二位,因而在捕捉隐式流方面精确于门级信息流模型。

```
property P1;
@(posedge clk) disable iff(rst)
out_0_t=1'b1;
endproperty
assert property (P1);

property P2;
@(posedge clk) disable iff(rst)
out_1_t=1'b1;
endproperty
assert property (P2);

property P3;
@(posedge clk) disable iff(rst)
out_2_t=1'b1;
endproperty
assert property (P3);
```

a) 门级信息流模型的断言

```
--------------------
Property Summary          Count
--------------------
Assumed                       0
Proven                        3
Inconclusive                  0
Fired                         0
--------------------
Total                         3
--------------------
```

b) 门级信息流模型的验证结果

```
property P4;
@(posedge clk) disable iff(rst)
out_t==3'b101;
endproperty
assert property (P4);
```

c) RTLIFT 逻辑模型的断言

```
--------------------
Property Summary          Count
--------------------
Assumed                       0
Proven                        1
Inconclusive                  0
Fired                         0
--------------------
Total                         1
--------------------
```

d) RTLIFT 逻辑模型的验证结果

图 5 RTLIFT 逻辑模型的精确性验证

3.3 硬件木马检测

3.3.1 AES-T400 的硬件木马检测

AES-T400 包含了一个泄露密钥的硬件木马。当输入明文 state 为 128 位的全“1”时,硬件木马被激活并通过输出端口 Antena 泄露密钥 key。在该设计中,密钥 key 是安全关键的,不应流向非密的输出端口 Antena。为检测该木马,分别将 AES-T400 建模为 RTLIFT 逻辑模型和门级信息流模型,标记 key 为污染,以图 6a)所示的断言 P1 验证污染的 key 是否流向未污染的 Antena。

```
property P1;
@(posedge clk) disable iff(rst)
(Antena_f_t==1) |=>(Antena_t==1'b0);
endproperty
assert property (P1);
```

a) 验证 AES-T400 硬件木马的断言

```
------------------------
Property Summary          Count
------------------------
Assumed                       0
Proven                        0
Inconclusive                  0
Fired                         1
  Fired with Warning          1
------------------------
Total                         1
------------------------
Peak Memory 1.21 GB, Cumulative CPU Time 5 s, Elapsed Time 14 s
```

b) AES-T400 RTLIFT 逻辑模型的验证结果

```
------------------------
Property Summary          Count
------------------------
Assumed                       0
Proven                        0
Inconclusive                  0
Fired                         1
  Fired with Warning          1
------------------------
Total                         1
------------------------
Peak Memory 4.22 GB, Cumulative CPU Time 16 s, Elapsed Time 22 s
```

c) AES-T400 门级信息流模型的验证结果

图 6 AES-T400 门级信息流模型与 RTLIFT 逻辑模型的验证断言与验证结果

图 6b)和图 6c)为断言 P1 在 RTLIFT 逻辑模型和门级信息流模型的验证结果。验证结果表明,污染的 key 皆流向了非密的输出端口 Antena。因而,AES-T400 设计中必然存在一个可将密钥泄露至端口 Antena 的硬件木马。

3.3.2　AES-T800 的硬件木马检测

AES-T800 包含了一个泄露密钥的硬件木马。与 AES-T400 相比,其木马逻辑的触发条件为一个特定的输入明文序列(128′h3243f6a8_885a308d_313198a2_e0370734,128′h00112233_44556677_8899aabb_ccddeeff,128′h0,128′h1)。当检测到该明文序列时,该木马被激活并调制密钥从端口 Capacitance 以 CDMA 信号方式进行泄露。在该设计中,密钥 key 是安全关键的,不应流向非密的端口 Capacitance。为检测该木马,分别将 AES-T800 建模为 RTLIFT 逻辑模型和门级信息流模型,并标记 key 为污染,以图 7a)所示的断言 P3 验证污染的 key 是否流向未污染的 Capacitance。

```
property P3;
@(posedge clk) disable iff(rst)
(Capacitance_checkpoint_t==0) |=>##2;
(Capacitance_t==0);
endproperty
assert property (P3);
```

a) 验证 AES-T800 硬件木马的断言

```
------------------------
Property Summary                Count
------------------------
Assumed                         0
Proven                          0
Inconclusive                    0
Fired                           1
  Fired with Warning            1
------------------------
Total                           1
------------------------
Peak Memory 1.21 GB, Cumulartive CPU Time 5 s, Elapsed Time 14 s
```

b) AES-T800 RTLIFT 逻辑模型的验证结果

```
------------------------
Property Summary                Count
------------------------
Assumed                         0
Proven                          0
Inconclusive                    0
Fired                           1
  Fired with Warning            1
------------------------
Total                           1
------------------------
Peak Memory 4.22 GB, Cumulartive CPU Time 16 s, Elapsed Time 25 s
```

c) AES-T800 门级信息流模型的验证结果

图 7　AES-T800 门级信息流模型与 RTLIFT 逻辑模型的验证断言与验证结果

图 7b)~7c)为断言 P3 在 RTLIFT 逻辑模型和门级信息流模型的验证结果。验证结果表明,端口 Capacitance 为污染的,说明安全关键的 key 泄露至非密输出端口 Capacitance。因而,AES-T800 设计中存在一个硬件木马可将密钥泄露至端口 Capacitance。

3.3.3　AES-T1200 的硬件木马检测

AES-T1200 包含了一个泄露密钥的硬件木马。与 AES-T800 相比,其木马触发逻辑采用内部触发方式。当经过多次加密后,该木马被触发且泄露密钥至端口 Capacitance。为检测该木马,将 AES-T1200 建模为 RTLIFT 逻辑模型和门级信息流模型,并以图 8a)所示的断言 P1 验证污染的 key 是否流向未污染的 Capacitance。

```
property P1;
@(posedge clk) disable iff(rst)
(Capacitance_checkpoint_t==0) |=>##2;
(Capacitance_t==0);
endproperty
assert property (P1);
```

a) 验证 AES-T1200 硬件木马的断言

```
------------------------
Property Summary                Count
------------------------
Assumed                         0
Proven                          0
Inconclusive                    0
Fired                           1
  Fired with Warning            1
------------------------
Total                           1
------------------------
Peak Memory 1.21 GB, Cumulative CPU Time 5 s, Elapsed Time 22 s
```

b) AES-T1200 RTLIFT 逻辑模型的验证结果

```
------------------------
Property Summary                Count
------------------------
Assumed                         0
Proven                          0
Inconclusive                    0
Fired                           1
  Fired with Warning            1
------------------------
Total                           1
------------------------
Peak Memory 4.22 GB, Cumulative CPU Time 16 s, Elapsed Time 25 s
```

c) AES-T1200 门级信息流模型的验证结果

图 8　AES-T1200 门级信息流模型与 RTLIFT 逻辑模型的验证断言与验证结果

图 8b)~8c)为断言 P1 在 RTLIFT 逻辑模型和门级信息流模型的验证结果。端口 Capacitance 为污染的,说明安全关键的 key 泄露至非密输出端口 Capacitance。因而,AES-T1200 设计中存在一个硬件木马可将密钥泄露至端口 Capacitance。

3.3.4 硬件木马检测实验分析

表 3 罗列了采用 GLIFT 和 RTLIFT 对 AES-T400、AES-T800、AES-T1200 的检测结果。结果表明:采用 GLIFT 和 RTLIFT 都可检测设计中硬件木马。分析原因在于,污染标签的传播路径中不存在污染的信号作为判断条件的情况。

表 3 基于 GLIFT 和 RTLIFT 的硬件木马检测结果对比

验证对象	GLIFT	RTLIFT	验证时间/s		内存消耗/GB	
			GLIFT	RTLIFT	GLIFT	RTLIFT
AES-T400	可检测	可检测	25	14	4.22	1.21
AES-T800	可检测	可检测	25	14	4.21	1.21
AES-T1200	可检测	可检测	25	22	4.23	1.21

因此,采用 2 种方法所获得检测结果都为精确的。但是,验证过程中,采用 GLIFT 所占用的内存资源和所消耗的验证时间远高于 RTLIFT。因而,相比 GLIFT, RTLIFT 更适用于大规模设计的安全验证。

4 结 论

本文提出了一种寄存器传输级信息流安全验证方法。该方法提供了一个完全以 Verilog 描述的 RTL 抽象层次信息流跟踪逻辑模型,使得能够从信息流角度建模 RTL 设计的安全行为和安全属性,继而可结合 EDA 测试验证工具验证 RTL 设计的安全属性,精确捕捉设计内所有实际存在的显式流和隐式流,定性检测违反信息流安全策略的潜在安全漏洞。实现结果表明,本文所提出的寄存器传输级信息流安全验证方法与门级信息流分析方法相比能够更为精确地捕捉隐式流,消除单变量翻转所产生的误报,快速且准确地验证设计安全性,检测设计内潜藏的硬件木马。

参考文献:

[1] 毛保磊, 慕德俊, 胡伟, 等. RSA 时间信道滑动窗口攻击方法及量化分析[J]. 西安电子科技大学学报, 2017, 44(5): 115-120
MAO Baolei, MU Dejun, HU Wei, et al. Quantitative analysis of sliding windows attack for the RSA timing channel[J]. Journal of Xidian University, 2017, 44(5): 115-120 (in Chinese)

[2] HU W, WU L, TAI Y, et al. A unified formal model for proving security and reliability properties[C]//2020 IEEE 29th Asian Test Symposium, 2020: 1-6

[3] HU W, MAO B, OBERG J, et al. Detecting hardware trojans with gate-level information-flow tracking[J]. Computer, 2016, 49(8): 44-52

[4] ZHANG Q, HE J, ZHAO Y, et al. A formal framework for gate-level information leakage using z3[C]//2020 Asian Hardware Oriented Security and Trust Symposium, 2020: 1-6

[5] ZHANG D, WANG Y, SUH G E, et al. A hardware design language for timing-sensitive information-flow security[J]. ACM Sigplan Notices, 2015, 50(4): 503-516

[6] BIDMESHKI M M, MAKRIS Y. VeriCoq: a Verilog-to-Coq converter for proof-carrying hardware automation[C]//2015 IEEE International Symposium on Circuits and Systems, 2015: 29-32

[7] HU W, ARDESHIRICHAM A, KASTNER R. Hardware information flow tracking[J]. ACM Computing Surveys, 2021, 54(4): 1-39

[8] ARDESHIRICHAM A, HU W, KASTNER R. Clepsydra: modeling timing flows in hardware designs[C]//2017 IEEE/ACM

International Conference on Computer-Aided Design, 2017: 147-154
[9] ARDESHIRICHAM A, HU W, MARXEN J, et al. Register transfer level information flow tracking for provably secure hardware design[C]//Design, Automation & Test in Europe Conference & Exhibition, 2017: 1691-1696
[10] GUO X, DUTTA R G, HE J, et al. QIF-Verilog: quantitative information-flow based hardware description languages for pre-silicon security assessment[C]//2019 IEEE International Symposium on Hardware Oriented Security and Trust, 2019: 91-100
[11] JIN Y, MAKRIS Y. Proof carrying-based information flow tracking for data secrecy protection and hardware trust[C]//2012 IEEE 30th VLSI Test Symposium, 2012: 252-257
[12] SHAKYA B, HE T, SALMANI H, et al. Benchmarking of hardware trojans and maliciously affected circuits[J]. Journal of Hardware and Systems Security, 2017(1): 85-102

Register transfer level hardware design information flow modeling and security verification method

QIN Maoyuan[1], HOU Jiaying[1], LI Jiale[1], TANG Shibo[2], TAI Yu[2]
(1.School of Computer Science and Engineering, Xi'an Technological University, Xi'an 710021, China;
2.School of Cybersecurity, Northwestern Polytechnical University, Xi'an 710072, China)

Abstract: Information flow analysis can effectively model the security behavior and security properties of hardware design. However, the existing gate level information flow analysis methods cannot deal with large-scale designs due to computing power and verification effectiveness, and the register transfer level (RTL) information flow analysis methods require formal languages to rewrite hardware designs. This paper proposes a RTL hardware design information flow modeling and security verification method. Based on the RTL functional model, this method develops an information flow tracking logical model to model security behavior and security properties of RTL hardware designs from the perspective of information flow. This method can be integrated into EDA flows and uses EDA testing and verification tools to capture security property violations and detect security vulnerabilities based on non-interference security policy. The results on experiments with Trust-Hub hardware Trojan benchmarks show that the proposed method can effectively detect hardware Trojans.

Keywords: hardware design; information flow security model; information flow security verification; security vulnerability detection

引用格式: *秦茂源, 侯佳滢, 李家乐, 等. 寄存器传输级硬件设计信息流建模与安全验证[J]. 西北工业大学学报, 2024, 42(3): 506-513*
QIN Maoyuan, HOU Jiaying, LI Jiale, et al. Register transfer level hardware design information flow modeling and security verification method[J]. Journal of Northwestern Polytechnical University, 2024, 42(3): 506-513 (in Chinese)

© 2024 Journal of Northwestern Polytechnical University.

This is an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.