Computer Science and Application 计算机科学与应用, 2020, 10(6), 1252-1258
Published Online June 2020 in Hans. http://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2020.106129

Hans汉斯

Research on Formal Verification Method of RISC-V Microcontroller

Yue Pan, Zijing Cheng

Spacestar Technology Co. Ltd., Beijing
Email: 798977621@qq.com

Received: May 28th, 2020; accepted: Jun. 11th, 2020; published: Jun. 18th, 2020

Abstract

Embedded microcontrollers are widely deployed within power IoT system. It's vital to guarantee functional correctness of the embedded microcontroller, which acts as core component in self-developed IoT ground communication node of transmission tower stability monitor by Spacestar. Since verifying the functional correctness and completeness of a processer differs from those of ASIC designs, a formal verification method for RISC-V based microcontroller is analyzed in this paper. This method incorporates the common merits of formal verification, based on open sourced tool chain. It is easily set up and highly automated, applicable as functional verification solution for low-cost microcontrollers.

Keywords

Microcontroller, RISC-V, Formal Verification, EDA

RISC-V微控制器形式化方法研究

潘　越，程子敬

航天恒星科技有限公司，北京
Email: 798977621@qq.com

收稿日期：2020年5月28日；录用日期：2020年6月11日；发布日期：2020年6月18日

摘　要

电力物联网系统内大量部署了嵌入式微控制器。在航天恒星自主研发的输配电杆塔稳定性监视系统的地面通信节点中，必须保证起核心作用的嵌入式微控制器的功能正确性。由于验证处理器功能正确和完备性与其它ASIC电路相比更为困难，本文分析了一种基于RISC-V指令集的微控制器形式化验证方法。该方

文章引用: 潘越, 程子敬. RISC-V 微控制器形式化方法研究[J]. 计算机科学与应用, 2020, 10(6): 1252-1258.
DOI: 10.12677/csa.2020.106129

案具有形式化验证的一般优点：基于开源的工具链，搭建简单，自动化程度高，适用于低成本微控制器的功能验证解决方案。

关键词

微控制器，RISC-V，形式化验证，电子设计自动化

1. 引言

顺应我国经济社会的发展要求，对公共电力基础设施适应高新技术产业提出了“智能化”的更高要求[1]。“智能电网”将电力资源高效率地输送、调配，保证供电质量和稳定[2] [3]。智能化设施将海量的传感器设备和通信节点布设到电力系统中，形成电力物联网系统[4] [5]，其运算能力的主要承载者来自于大规模部署的嵌入式控制器[6]。因此，智能化广泛在电力物联网的实现需要大量经过功能验证的微控制器，尤其是在涉及到电力等关键基础设施的情况下，需要严格地保证软硬件的功能正确性[7] [8]。

RISC-V 是一种新兴的开源指令集，具有简洁、紧凑、模块化、易于实现的特点，是近几年国内外处理器研发创业公司的首选[6] [9]。对处理器进行功能验证不同于其它 ASIC 的验证[10]，在于任何处理器的实现需要符合它依赖的指令集规则。在目前，很多实现将能够运行定点/浮点性能测试程序，或 Linux 操作系统作为其功能正确性的标志，但这类方法往往不能穷尽所有的测试用例，而高级语言的编译器也可能隐藏硬件中的一些设计错误。对处理器或其它复杂硬件最可靠的功能验证方式是形式化验证，通过数学证明检查硬件实现与一套预设规则集是否等价。形式化验证能够系统地、完备地发现设计中的错误，或证明设计中没有错误[11]。

在后续的章节，首先对适用于 RISC-V 控制器设计的形式化验证流程进行概述，然后分析了 RISCV-Formal，一个用于发现设计错误的开源工具链，最后将该工具链应用于一个具体的 RISC-V 嵌入式处理器实现。

2. 形式化验证流程

处理器形式化功能验证流程要解决的两个问题是：已知一个处理器电路的设计和其寄存器合法的初始状态，1) 处理器是否会运行到某些不可接受的状态，称为安全属性(Safety Property)；2) 一些可接受的的状态是否可以运行到，称为活跃属性(Liveness Property) [12]。证明形式分为有界检查和无界检查，有界检查仅考虑从初始状态开始经过预设的时钟数后是否存在属性违例；无界检查则推测经过任意时钟数后是否会存在属性违例，通常通过 k 步数学归纳法验证[11] [13]。

如图 1 所示，RISC-V 形式化验证分为三个主要步骤：1) 产生/获取对指令集的形式化规约；2) 在硬件设计过程的关键点中插入断言；3) 对 HDL 代码进行属性证明[14] [15]。

2.1. 形式化规约

在一开始，处理器指令集架构是以自然语言发布的规范文档，它规定了一个处理器要实现或兼容此指令集的设计需要能够处理哪些指令，指令的格式，以及运行指令产生的后果。指令集架构规范是与具体的机器无关的[1]。

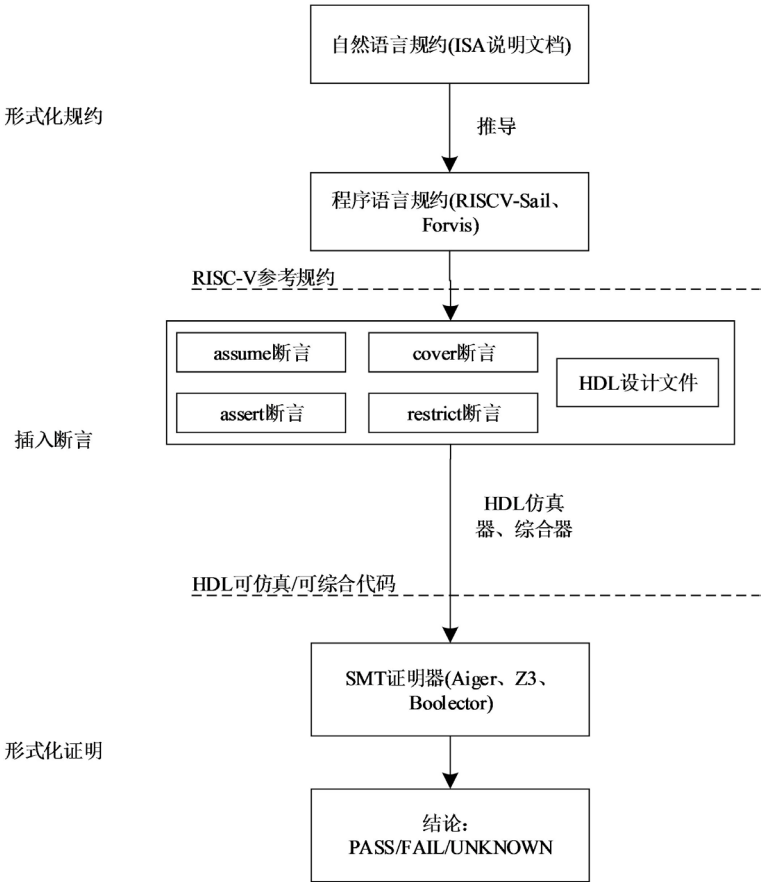

Figure 1. RISC-V formal verification workflow
图 1. RISC-V 形式化验证流程

自然语言描述的规范可能带有各种歧义或疏漏。为了避免阅读时产生歧义，进一步便于作为自动化证明系统的输入，程序语言规约用规范的方式定义和描述。程序化的规约也和机器实现无关[10]。

2.2. 插入断言

现代处理器通过硬件描述语言(HDL)设计。HDL 的硬件描述通过综合器输出为逻辑门构成的网表格式，或直接在仿真器上运行。

设计错误本质上就是对规约的违反，因此要在仿真时发现这些违例，可通过在代码中加入断言的方式表征处于安全状态时处理器状态所必须具备的特性。

YOSYS 综合器支持四种断言，如下表 1 所示：

Table 1. Four categories of assertions
表 1. 四种断言类型

类型	assume	assert	cover	restrict
用法	在求解逻辑表达式时赋值为真，设置其它断言的前提条件	在求解逻辑表达式时若为假，则判定为违例	在求解逻辑表达式时若为真，则判定为活跃	类似于 Assume，但不作为其它断言的前提，仅用于缩小解空间，加速运算

在编写代码时，断言结构一般被内嵌于一个独立的 always-case 结构中，用于捕获电路部分信号之间的关系。assume 断言和 assert 断言用来组成安全特性规则，出现 assert 语句内的表达式为假时，判断安

全特性违例；cover 断言用来构造活跃特性规则，如果 cover 语句内的表达式能够在有效的步数内取得一次真，则说明所监视的状态可达，因此 cover 语句只能在有界模型中测试。

2.3. 形式化证明

当前广泛用于硬件验证的软件是基于可满足性模块理论(Satisfiability Modulo Theories, SMT)的固定规模布尔变量证明器，典型的有 Z3、Yices 2、CVC4、boolector 等[10] [16]。SMT 证明器的输入是一组变量声明和一组断言，然后通过无量词的皮尔斯伯格自动机算法决策这组断言是否能够被满足[17]。表 2 是一个 SMT-LIB 2 格式的输入实例，它描述了 RISC-V 指令集中的 add 指令所必须满足的特性。

Table 2. SMT input example—instruction add
表 2. SMT 输入实例——add 指令

行为	从 rs1 和 rs2 寄存器中读取内容，进行定点加法后回存到 rd 指定的寄存器中
形式化描述	(declare-const(regf Array (_ BitVec 64))) ;定义寄存器文件 regf (declare-const rs1 (_ BitVec 5)) (declare-const rs2 (_ BitVec 5)) (declare-const rd (_ BitVec 5)) ;定义 5 比特寄存器索引 (declare-const sum (bvadd(select(regf, rs1), select(regf, rs2)))) ;sum 等于从 regf 的索引 rs1, rs2 读取定点数之和 (store(sum, regf, rd)) ;将 sum 存储到 regf 索引为 rd 处 (assert(not(= (select(regf, rd)), sum))) (check-sat)

在表 2 中，断言(assert(not(= (select(regf, rd)), sum)))宣布从 regf 的索引 rd 处读取的值一定等于 sum，在此将其取否，通过(chech-sat)命令求解是否存在使整个表达式为真的赋值；当证明器找到一个解时返回 sat，无解时返回 unsat。由于希望断言恒成立，则等价于其否断言无解。证明处理器实现的指令行为符合规约是否符合规约则转换为该指令对应断言的否是否无解，这是进行符合性验证的一般思路[11] [13]。

3. 形式化验证工具链

虽然在上述的三个步骤中，可选取的开源或闭源工具有很多，但是 RISCV-Formal 是一整套依托于开源工具的验证框架。RISCV-Formal 的主体部分是 HDL 综合器 YOSYS 的前端工具 SymbiYosys，这一工具可将 YOSYS 的综合输出网表转交给后端的 SMT 证明引擎。SymbiYosys 支持多种 SMT 证明器，包括有界模型和无界模型的工作模式。在有界模式下，SymbiYosys 每个周期把电路状态的快照翻译成命题逻辑，由证明器解析；在无界模式下，它仍然先进行一次完整的 k 步有界检查，如果正确，它再尝试进行 k 步归纳，证明第 k + 1 步的状态一定正确。

在 YOSY 断言分类下，assume、assert、cover、restrict 断言和 SMT 输入格式有一一对应的关系。

1) assume 和 restrict 断言对于证明器是同一种，它们都被转换成(declare-)赋值，这些确定的变量作为已知的皮尔斯伯格自动机输入序列，可以减少证明的时间；

2) cover 断言被原样地转化成 SMT 断言，当证明器找到一个解时，说明断言描述的状态是可达的；

3) assert 断言被转换成取否的 SMT 断言，如果证明器发现了一个解，则说明发现了一个不符合性问题，这个解的变量赋值情况被解析成波形文件报告给用户，告知这一非法状态是如何到达的。

RISCV-Formal 能够验证两种形式化符合性问题，它通过以时钟周期为单位调用证明器求解状态来解决这两种问题，它们调用证明器的时机和求解的问题如下：

对单条指令行为进行的端到端有界模型检查。在每个用户使能的指令测试中，处理器首先无约束地运行预设的时钟周期，在最后一个周期时，证明从提交单元交付的每个指令字的交付前/交付后状态符合规约。

对多条指令序列的行为进行一致性有界模型检查。在该模式下，检测器先在复位状态下运行 N 个周期，再在置位状态下运行 M 个周期；在 M 个周期内，检测器任意选择一个周期锁存当前状态，然后在最后一个周期比对前后的状态。这一机制可以发现：PC 一致性问题，即第 k 条指令注册的 PC 是否与第 k + 1 条指令读取的 PC 相同；寄存器一致性问题，即对同一寄存器的写入和读取的结果是否匹配；活跃性问题，即某条指令提交后，有后续的指令继续执行；因果关系，即对前一指令存在依赖关系的指令是否提前提交。

通过这些测试用例的实现将可以保证符合最基本的 RISCV-32I 规范。除去正确地实现合法指令及流水线外，该控制器必须能够发现未定义的非法指令，这意味着指令译码逻辑必须是完备的(在译码合法指令时总返回真，译码非法指令时总返回假)。截至到本文撰写时，RISCV-Formal 尚未支持 64I 和 32E 两种变体，考虑到微控制器基本是 32 位的，选择 32E 的实现需要临时补全 32 个寄存器的寄存器文件。

4. 应用实例

在图 2 的智能电力巡检系统中，提出了一种在丘陵、山地地形，气候恶劣，经常有大风暴雨的环境下，对电力传输杆塔进行远程智能监控的解决方案。在这一方案在被控地区的传输塔体上安装多种传感器(倾角传感器、位置传感器、结构应力传感器等)，这些不同制式的数据在地面通信节点汇总，进行转换处理后中继到卫星或附近的基站。通信节点需要用到一款微控制器作为融合传感器到卫星链路/基站的协议转换节点[1]。我们希望通过一个自主研发的 RISC-V 核心代替基于 ARM Cortex-M 的核心。

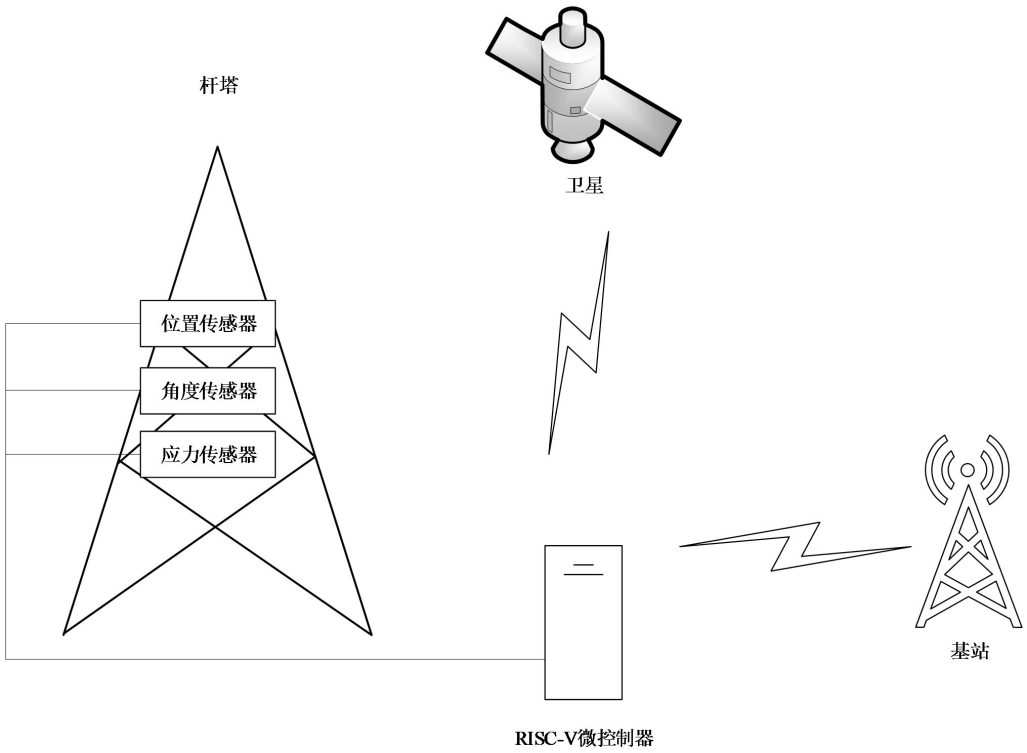

Figure 2. Ground sensor hub
图 2. 传感器地面节点

PICORV32 是一款小面积、低功耗，具有相当可配置性的 RISC-V 32 位实现。通过控制描述文件的宏定义，它可以配置为支持压缩、乘法扩展，并可以选择 32E 或 32I 中的一种基础整数值指令集。它还带有一个可选的中断控制器，以及 AXI-Lite 或 Wishbone 内存总线。

RISCV-Formal 为 PICORV32 准备了 76 个单指令测试用例，覆盖了全部整数指令，部分常用的控制与状态寄存器指令和它们的汇编别名。其它测试包括：单字只读内存模拟，用于检查内存系统到检查器的路径上是否存在问题，指令内存和数据内存均可以被测试；等价性测试，形式化地证明电路在移除 RISCV-Formal 逻辑前后功能等价；完备性测试保证非法指令不被提交。

为了测试 RISCV-Formal 的工作流程，在 PICORV32 的源文件中注入了一个常见的 bug。因为 RISC-V 的最小指令长度为 16 位，因此所有指令的地址应当至少是 16 的整数倍；在规约中，跳转并链接寄存器指令 JALR 会把一个符号扩展的立即数和一个链接寄存器相加，将结果的最低位直接清零后，作为新的取指地址写入 PC。有些实现忘记把最低位清零，或只是把写入 PC 的副本最低位清零。表 3 中进行的替换复现了这一 bug。

在替换后，对 JALR 指令的测试用例发现了这一错误，并生成了一个违例的反例，用户可以通过波形查看软件分析反例的波形(如图 3)。因为这一 bug，RISCV-Formal 检查器取回的下一条 PC 值与预期的不符，报告错误的指令的细节在最后一个周期从 rvfi_*端口中读取，例如编码和操作数可以分别从 rvfi_insn 和 rvfi_rs*_rdata 输出信号中获取，这可以为除错工作指明方向，提高排查效率。

Table 3. Injected JARL bug
表 3. 注入的 JARL 错误

注入前	current_pc = latched_store ? (latched_stalu ? alu_out_q : reg_out) & ~1 : reg_next_pc;
注入后	current_pc = latched_store ? (latched_stalu ? alu_out_q : reg_out) : reg_next_pc;

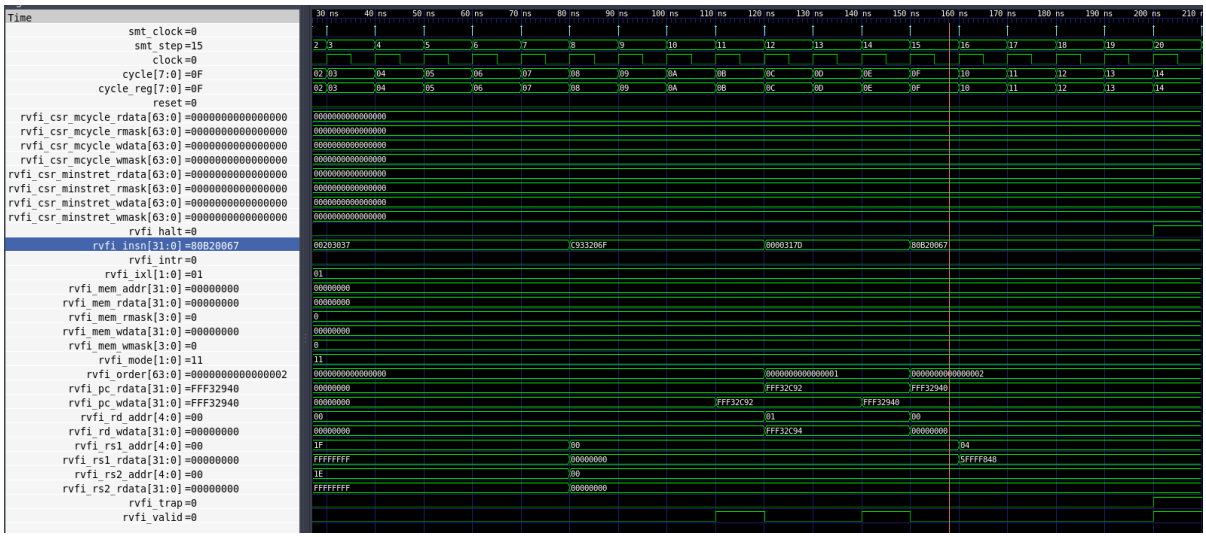

Figure 3. JALR counterexample waveform
图 3. JALR 反例波形图

5. 结论

在实际的测试中，RISCV-Formal 确实发现了一些被高层次软件屏蔽的处理器实现错误。

当然，这套工具仍然存在一些实用性的问题。例如目前，它没有实现从程序规约文件推导断言测试

用例的能力；这意味着从指令集规范到 sby 测试用例是由开发者编写的，这在很大程度上削弱了程序规约的使用价值；此外，RISC-V 规范的草案版本除整数指令集外正处于快速迭代中，一些模块的定义或设计逻辑发生了根本的变化(例如此前所有强制要求的计数寄存器被移除出基础指令集，而转为统一的控制与状态寄存器指令)，那么下一版本的定案规约将同目前的版本有很大差异，如何充分利用社区的劳动成果，把机器无关的程序规约转化成具体的、机器相关的 HDL 断言是一件很有价值的工作。

推广 RISCV-Formal 及类似形式化验证框架的意义在于，让形式化方法在小规模电路设计(不仅限于微控制器)团队中普及起来，让硬件验证成为一门严谨的数学，有助于提高数字产品设计研发的质量，提高 IC 设计行业的整体水平[8] [18]。

最后，硬件的形式化验证可以从根源上排除功能设计上的错误，不仅限于处理器的设计，还适用于其它种类数字电路，乃至软件系统的设计[9] [18]。目前，国内应用形式化验证的正式系统实现尚较为缺乏，这使得开源硬件运动缺少与涉及国计民生的关键应用结合的自信。推广形式化验证工作方法有助于进一步推进我国 IT/IC 事业的蓬勃发展，将开源硬件/软件的资源和成功转化到这些关键系统领域中。

参考文献

[1] 徐晓寅. 信息通信技术支撑泛在电力物联网建设[J]. 通信电源技术, 2019, 36(12): 186-187.

[2] 周小艳, 何为, 胡国辉. 基于 ZigBee 无线传感器网络的变电站人员定位的改进算法研究[J]. 电力系统保护与控制, 2013, 41(17): 56-62.

[3] 柏钰昇. 关于智能电网对智慧城市的支撑作用的分析[J]. 工业设计, 2017(9): 130-131.

[4] 余贻鑫. 智能电网实施的紧迫性和长期性[J]. 电力系统保护与控制, 2019, 47(17): 1-5.

[5] 李易. 泛在电力物联网在电力系统中应用的展望[J]. 科技创新与应用, 2019(22): 175-176.

[6] 雷思磊. RISC-V 架构的开源处理器及 SoC 研究综述[J]. 单片机与嵌入式系统应用, 2017, 17(2): 56-60+76.

[7] 刘益青, 高伟聪, 魏鹏, 等. 基于 MCU + DSP 多处理器构架的微机保护硬件平台设计[J]. 电力系统保护与控制, 2010, 38(10): 89-91+95.

[8] 李其高. 面向 IoT 终端设备的 RISC-V 微控制器设计与分析[J]. 单片机与嵌入式系统应用, 2018, 18(3): 64-66+69.

[9] 贾琳, 樊晓桠. 32 位 RISC 微处理器流水线设计[J]. 计算机工程与应用, 2005(14): 115-117.

[10] 郑飞君, 严晓浪, 葛海通, 等. 使用布尔可满足性的组合电路等价性验证算法[J]. 电子与信息学报, 2005(4): 651-654.

[11] Wintersteiger, C.M., Hamadi, Y. and De Moura, L. (2013) Efficiently Solving Quantified Bit-Vector Formulas. *Formal Methods in System Design*, **42**, 3-23. https://doi.org/10.1007/s10703-012-0156-2

[12] Kovásznai, G., Fröhlich, A. and Biere, A. (2016) Complexity of Fixed-Size Bit-Vector Logics. *Theory of Computing Systems*, **59**, 323-376. https://doi.org/10.1007/s00224-015-9653-1

[13] Olivo, O. and Emerson, E.A. (2011) A More Efficient BDD-Based QBF Solver. In: *International Conference on Principles and Practice of Constraint Programming*, Springer, Berlin, 675-690. https://doi.org/10.1007/978-3-642-23786-7_51

[14] 朱峰, 鲁征浩, 朱青. 形式化验证在处理器浮点运算单元中的应用[J]. 电子技术应用, 2017, 43(2): 29-32.

[15] 郭莹. 布尔可满足性问题研究综述[J]. 软件导刊, 2017, 16(5): 204-206.

[16] 王翀, 吕荫润, 陈力, 等. SMT 求解技术的发展及最新应用研究综述[J]. 计算机研究与发展, 2017, 54(7): 1405-1425.

[17] Bouton, T., De Oliveira, D.C.B., Déharbe, D., *et al*. (2009) veriT: An Open, Trustable and Efficient SMT-Solver. In: *International Conference on Automated Deduction*, Springer, Berlin, 151-156. https://doi.org/10.1007/978-3-642-02959-2_12

[18] 李东泽, 曹凯宁, 曲明, 等. 五级流水线 RISC-V 处理器软硬件协同仿真验证[J]. 吉林大学学报(信息科学版), 2017, 35(6): 612-616.