(12) 按照专利合作条约所公布的国际申请

(19) 世界知识产权组织
国 际 局

(43) 国际公布日
2016 年 12 月 22 日 (22.12.2016)

WIPO | PCT

(10) 国际公布号
WO 2016/202157 A2

(51) 国际专利分类号: 无分类

(21) 国际申请号: PCT/CN2016/083560

(22) 国际申请日: 2016 年 5 月 26 日 (26.05.2016)

(25) 申请语言: 中文

(26) 公布语言: 中文

(30) 优先权:
201510334812.4 2015 年 6 月 16 日 (16.06.2015) CN

(71) 申请人: 深圳市中兴微电子技术有限公司 (ZTE MICROELECTRONICS TECHNOLOGY CO.,LTD.) [CN/CN]; 中国广东省深圳市盐田区大梅沙 1 号厂房, Guangdong 518085 (CN)。

(72) 发明人: 张丰举 (ZHANG, Fengju); 中国广东省深圳市盐田区大梅沙 1 号厂房, Guangdong 518085 (CN)。

(74) 代理人: 北京派特恩知识产权代理有限公司 (CHINA PAT INTELLECTUAL PROPERTY OFFICE); 中国北京市海淀区海淀南路 21 号中关村知识产权大厦 B 座 2 层, Beijing 100080 (CN)。

(81) 指定国 (除另有指明，要求每一种可提供的国家保护): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW。

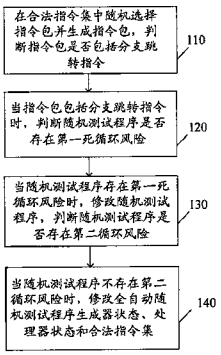
(84) 指定国 (除另有指明，要求每一种可提供的地区保护): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), 欧亚 (AM, AZ, BY, KG, KZ, RU, TJ, TM), 欧洲 (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG)。

本国际公布:

— 不包括国际检索报告，在收到该报告后将重新公布(细则 48.2(g))。

(54) Title: RANDOM TESTING PROGRAM GENERATION METHOD AND DEVICE, APPARATUS, AND STORAGE MEDIUM

(54) 发明名称 : 一种随机测试程序生成方法及装置、设备、存储介质



110 Randomly selecting instruction packet from valid instruction set and generating instruction packet, and determining whether instruction packet includes any branch and jump instructions

120 If instruction packet includes branch and jump instruction, determining whether random testing program has first endless loop risk

130 If random testing program has first endless loop risk, modifying random testing program and determining whether random testing program has second loop risk

140 If random testing program does not have second loop risk, modifying status of automated random testing program generator, status of processor, and valid instruction set

图 1

(57) Abstract: Disclosed is a random testing program generation method, comprising: randomly selecting an instruction packet from a valid instruction set and generating the instruction packet, and determining whether the instruction packet includes any branch and jump instructions; if the instruction packet includes a branch and jump instruction, determining whether the random testing program has a first endless loop risk; if the random testing program has a first endless loop risk, modifying the random testing program and determining whether the random testing program has a second loop risk; if the random testing program does not have a second loop risk, modifying the status of the automated random testing program generator (ARTPG), the status of the processor, and the valid instruction set. Also disclosed are a random testing program generation device, apparatus, and storage medium.

(57) 摘要: 本发明公开了一种随机测试程序生成方法，包括：在合法指令集中随机选择指令包并生成指令包，判断所述指令包是否包括分支跳转指令；当所述指令包包括所述分支跳转指令时，判断所述随机测试程序是否存在第一死循环风险；当所述随机测试程序存在所述第一死循环风险时，修改所述随机测试程序，判断所述随机测试程序是否存在第二循环风险；当所述随机测试程序不存在第二循环风险时，修改全自动随机测试程序生成器(ARTPG)状态、处理器状态和所述合法指令集。本发明还同时公开了一种随机测试程序生成装置、设备、存储介质。

WO 2016/202157 A2

一种随机测试程序生成方法及装置、设备、存储介质

技术领域

本发明涉及处理器设计领域，尤其涉及一种随机测试程序生成方法及装置、设备、存储介质。

背景技术

现有技术中，在进行处理器设计时，对处理器的功能验证是一个重点和难点，如何能够完整的覆盖处理器的功能及保证处理器验证的质量成为处理器设计的关键。目前，纯粹靠验证人员手动编写处理器验证所需要的程序和数据容易漏掉对某些处理器功能点的验证。因此，如果采用高效生成随机汇编测试程序或者随机机器测试程序的方式可以改善或解决上述技术问题。

现有技术中，随机测试程序生成方法需要采用不受控的全随机，导致生成的随机测试程序合法性存在问题，同时随机测试程序的目的性差且难以保证处理器的功能覆盖率；此外，现有技术中的随机测试程序生成方法需要编写程序段或者编写复杂的随机程序约束模版，导致生成的随机测试程序随机性差，并难以做到对人工设计程序的有效补充。现有技术中其它的随机测试程序生成方法对随机测试程序中某点或某方面进行了改善，例如，改善浮点随机数的合法性，改善权重对随机指令的控制，改善对分支跳转指令的控制，及改善流水线干扰。

发明内容

有鉴于此，本发明实施例期望提供一种随机测试程序生成方法及装置、设备、存储介质，不仅能全自动的生成处理器验证所需的随机测试程序，

而且能有效的控制随机测试程序的合法性和功能覆盖率，从而能提升处理器验证的效率。

为达到上述目的，本发明实施例的技术方案是这样实现的:

第一方面，本发明实施例提供了一种随机测试程序生成方法，包括:

在合法指令集中随机选择指令包并生成所述指令包，判断所述指令包是否包括分支跳转指令;

当所述指令包包括所述分支跳转指令时，判断所述随机测试程序是否存在第一死循环风险;

当所述随机测试程序存在所述第一死循环风险时，修改所述随机测试程序，判断所述随机测试程序是否存在第二循环风险;

当所述随机测试程序不存在第二循环风险时，修改全自动随机测试程序生成器 ARTPG 状态、处理器状态和所述合法指令集。

在本发明的其他实施例中，所述修改所述随机测试程序包括:

当分支跳转地址超过程序空间界限时，修改所述分支跳转地址使其不超过所述程序空间界限;

当分支跳转判定条件在声明到使用之间没有修改操作时，增加修改指令;

当所述修改操作的收敛方向与所述分支跳转判定条件不一致时，修改所述修改操作的收敛方向使其与所述分支跳转判定条件一致;

当所述指令包包括函数调用指令或软件中断指令时，添加现场保存及恢复的指令。

在本发明的其他实施例中，所述修改所述 ARTPG 状态包括:

根据功能覆盖率反馈信息调整随机权重。

在本发明的其他实施例中，所述修改所述合法指令集包括:

将修改后的输出寄存器添加到对应的合法输入操作数;

将所述合法输入操作数对应的指令添加到所述合法指令集;

当所述指令包包括所述函数调用指令或所述软件中断指令时，将当前返回地址压至堆栈，在所述函数调用指令或所述软件中断指令前添加所述现场保存的指令。

在本发明的其他实施例中，在所述修改所述随机测试程序之后及在所述判断所述随机测试程序是否存在第二循环风险之前，还包括：执行指令局部仿真或者执行指令全局仿真。

在本发明的其他实施例中，所述修改 ARTPG 状态、处理器状态和所述合法指令集之后，还包括:

当所述随机测试程序生成的第一发射指令数等于随机选择的第二发射指令数且所述随机测试程序满足终止条件时，输出所述随机测试程序、随机测试程序执行结果和覆盖率分析报告。

第二方面，本发明实施例还提供了一种随机测试程序生成装置，包括:

第一判断模块，配置为在合法指令集中随机选择指令包并生成所述指令包，判断所述指令包是否包括分支跳转指令;

第二判断模块，配置为当所述指令包包括所述分支跳转指令时，判断所述随机测试程序是否存在第一死循环风险;

第三判断模块，配置为当所述随机测试程序存在所述第一死循环风险时，修改所述随机测试程序，判断所述随机测试程序是否存在第二循环风险。

修改模块，配置为当所述随机测试程序不存在第二循环风险时，修改全自动随机测试程序生成器 ARTPG 状态、处理器状态和所述合法指令集。

在本发明的其他实施例中，所述第三判断模块，配置为修改所述随机测试程序包括:

当分支跳转地址超过程序空间界限时，修改所述分支跳转地址使其不

超过所述程序空间界限;

当分支跳转判定条件在声明到使用之间没有修改操作时，增加修改指令;

当所述修改操作的收敛方向与所述分支跳转判定条件不一致时，修改所述修改操作的收敛方向使其与所述分支跳转判定条件一致;

当所述指令包包括函数调用指令或软件中断指令时，添加现场保存及恢复的指令。

在本发明的其他实施例中，所述修改模块，配置为修改所述 ARTPG 状态包括:

根据功能覆盖率反馈信息调整随机权重。

在本发明的其他实施例中，所述修改模块，配置为修改所述合法指令集包括:

将修改后的输出寄存器添加到对应的合法输入操作数;

将所述合法输入操作数对应的指令添加到所述合法指令集;

当所述指令包包括所述函数调用指令或所述软件中断指令时，将当前返回地址压至堆栈，在所述函数调用指令或所述软件中断指令前添加所述现场保存的指令。

在本发明的其他实施例中，所述第三判断模块，还配置为在所述修改所述随机测试程序之后及在所述判断所述随机测试程序是否存在第二循环风险之前，执行指令局部仿真或者执行指令全局仿真。

在本发明的其他实施例中，还包括:

输出模块，配置为当所述随机测试程序生成的第一发射指令数等于随机选择的第二发射指令数且所述随机测试程序满足终止条件时，输出所述随机测试程序、随机测试程序执行结果和覆盖率分析报告。

第三方面，本发明实施例提供一种计算机存储介质，所述计算机存储

介质中存储有计算机可执行指令，该计算机可执行指令用于执行本发明第一方面实施例提供的随机测试程序生成方法。

第四方面，本发明实施例提供一种随机测试程序生成设备，所述设备包括:

存储介质，配置为存储计算机可执行指令;

处理器，配置为执行存储在所述存储介质上的计算机可执行指令，所述计算机可执行指令包括:

在合法指令集中随机选择指令包并生成所述指令包，判断所述指令包是否包括分支跳转指令;

当所述指令包包括所述分支跳转指令时，判断所述随机测试程序是否存在第一死循环风险;

当所述随机测试程序存在所述第一死循环风险时，修改所述随机测试程序，判断所述随机测试程序是否存在第二循环风险;

当所述随机测试程序不存在第二循环风险时，修改全自动随机测试程序生成器 ARTPG 状态、处理器状态和所述合法指令集。

本发明实施例所提供的随机测试程序生成方法及装置、设备、存储介质，由随机测试程序生成装置在合法指令集中随机选择指令包并生成所述指令包，判断所述指令包是否包括分支跳转指令；当所述指令包包括所述分支跳转指令时，所述随机测试程序生成装置判断所述随机测试程序是否存在第一死循环风险；当所述随机测试程序存在所述第一死循环风险时，所述随机测试程序生成装置修改所述随机测试程序，判断所述随机测试程序是否存在第二循环风险；当所述随机测试程序不存在第二循环风险时，所述随机测试程序生成装置修改全自动随机测试程序生成器（ARTPG，Automatic Random Test Program Generator）状态、处理器状态和所述合法指令集。由于本发明实施例可以动态更新合法指令集和处理器状态，也使得

随机测试程序的生成与合法性检测所述随机测试程序能够同时进行；这样，不但能全自动的生成处理器验证所需的随机测试程序，而且能有效的控制随机测试程序的合法性和功能覆盖率，进而能提升处理器验证的效率。

附图说明

图 1 为本发明实施例 1 提供的随机测试程序生成方法的实现流程示意图；

图 2 为本发明实施例 1 提供的寄存器文件数据结构示意图；

图 3 为本发明实施例 1 提供的指令生成器数据结构示意图；

图 4 为本发明实施例 1 提供的功能单元列表数据结构示意图；

图 5 为本发明实施例 1 提供的指令列表数据结构示意图；

图 6 为本发明实施例 1 提供的操作数链表数据结构示意图；

图 7 为本发明实施例 1 提供的合法指令集树状链表数据结构示意图；

图 8 为本发明实施例 2 至 6 提供的随机测试程序生成方法的实现流程示意图；

图 9 为本发明实施例 7 提供的随机测试程序生成装置的组成结构示意图。

具体实施方式

本发明实施例中，由随机测试程序生成装置在合法指令集中随机选择指令包并生成指令包，判断所述指令包是否包括分支跳转指令；当所述指令包包括所述分支跳转指令时，所述随机测试程序生成装置判断所述随机测试程序是否存在第一死循环风险；当所述随机测试程序存在所述第一死循环风险时，所述随机测试程序生成装置修改所述随机测试程序，判断所述随机测试程序是否存在第二循环风险；当所述随机测试程序不存在第二循环风险时，所述随机测试程序生成装置修改 ARTPG 状态、处理器状态和

所述合法指令集。

下面结合附图及具体实施例对本发明再做进一步详细的说明。

实施例 1

图 1 为本发明实施例 1 提供的随机测试程序生成方法的实现流程示意图，如图 1 所示，所述方法包括：步骤 110：在合法指令集中随机选择指令包并生成指令包，判断指令包是否包括分支跳转指令。

这里，所述在合法指令集中随机选择包并生成指令包的过程包括：随机测试程序生成装置生成不大于预设的处理器最大指令数的随机数；根据随机数从合法指令集树状链表中选择随机的指令，并随机选择指令所需的随机操作数。

步骤 120：当指令包包括分支跳转指令时，判断随机测试程序是否存在第一死循环风险。

步骤 120 中，当指令包中包括分支跳转指令时，首先追朔到分支跳转指令条件变量的声明，这里，寄存器被加载（Load）。之后，随机测试程序生成装置结合分支跳转指令，判断随机测试程序是否存在第一死循环风险。判断准则包括：

前向跳转不存在风险，这里，所述前向跳转为跳转到程序未执行的部分；后向跳转存在风险，这里，所述后向跳转为跳转到程序已执行的部分；

后向跳转时，从条件变量声明到分支跳转指令间，条件变量被修改，修改操作的收敛方向与所述分支跳转的判定条件一致。

这里，所述死循环风险为程序无限次的后向跳转。

步骤 130：当随机测试程序存在第一死循环风险时，修改随机测试程序，判断随机测试程序是否存在第二循环风险。

在步骤 130 中，所述修改所述随机测试程序包括：

当分支跳转地址超过程序空间界限时，修改所述分支跳转地址使其不

超过所述程序空间界限;

当分支跳转判定条件在声明到使用之间没有修改操作时，增加修改指令;

当所述修改操作的收敛方向与所述分支跳转判定条件不一致时，修改所述修改操作的收敛方向，使其与所述分支跳转判定条件一致;

当所述指令包包括函数调用指令或软件中断指令时，添加现场保存及恢复的指令。

这里，所述当所述指令包包括函数调用指令或软件中断指令时，添加现场保存及恢复的指令包括:

当所述指令包包括函数调用指令或软件中断指令时，直接在该指令前面添加现场保存的指令或指令包。

在步骤 130 中，所述修改所述随机测试程序还包括:

当所述指令包包括函数返回指令或中断返回指令时，直接在该指令后面添加现场恢复的指令或指令包。

在步骤 130 中，在所述修改所述随机测试程序之后及在所述判断所述随机测试程序是否存在第二循环风险之前，所述方法还包括：执行指令局部仿真或者执行指令全局仿真。

这里，指令仿真分为指令局部仿真和指令全局仿真，指令局部仿真为从变量声明到分支跳转之间的指令进行仿真，对所遇未定操作数赋予随机值后，仿真到分支跳转指令，跟踪跳转方向，指定次数内可以前向跳转则风险解决。指令全局仿真为从头开始仿真，主要用于当生成随机测试程序时开放直接跳转指令时使用。

这里，所述第二死循环风险为程序地址跳转回同一地址的死循环风险。

步骤 140: 当随机测试程序不存在第二循环风险时，修改 ARTPG 状态、处理器状态和合法指令集。

在步骤 140 中，所述修改 ARTPG 状态包括：根据生成的指令或指令包的大小，修改已生成程序的代码大小；根据生成的指令或指令包需要执行的周期数，修改已生成程序需要执行的周期数，所述周期数为预估，循环时根据前面的检测和仿真结果修改；根据指令生成器和处理器状态的各个项被使用次数计算和修改覆盖率；根据功能覆盖率反馈信息调整随机权重。

在步骤 140 中，所述修改处理器状态包括：根据所选的随机指令所指向的函数计算指令执行结果；根据执行结果修改处理器状态。

在步骤 140 中，所述修改所述合法指令集包括：

将修改后的输出寄存器添加到对应的合法输入操作数；

将所述合法输入操作数对应的指令添加到所述合法指令集；

当所述指令包包括所述函数调用指令或所述软件中断指令时，将当前返回地址压至堆栈，在所述函数调用指令或所述软件中断指令前添加所述现场保存的指令。

在步骤 140 之后，即在所述修改 ARTPG 状态、处理器状态和所述合法指令集之后，所述方法还包括：

当所述随机测试程序生成的第一发射指令数等于随机选择的第二发射指令数且所述随机测试程序满足终止条件时，输出所述随机测试程序、随机测试程序执行结果和覆盖率分析报告。

在步骤 110 之前，所述方法还包括：

步骤 101：初始化处理器初始状态。

在步骤 101 中，随机测试程序生成装置根据验证人员提供的处理器状态描述初始化 ARTPG 中指令仿真器对应的处理器初始状态。

这里，所述处理器初始状态是随机测试程序中的一种数据结构，与另一种数据结构，即处理器状态类似，但是二者主要有两个区别：第一，所述处理器初始状态并没有当前值和当前周期指令或指令包执行结束后的值

这两种状态；第二，所述处理器初始状态不需要初始化所有的寄存器文件和状态寄存器，其中，未指定的内容可默认为处理器的复位值。

具体地，指令级精准的指令仿真器是一种处理器状态。处理器状态包括处理器对应的所有寄存器文件、状态寄存器的当前值、当前周期指令或指令包执行结束后的值，指令级精准的指令仿真器添加部分流水线寄存器的值。图 2 为寄存器文件数据结构，状态寄存器和寄存器文件基本相同，不同的是状态寄存器的进入列表（Entry List）为结构体数组，而寄存器文件中的 Entry List 为结构体。

步骤 102：初始化合法指令集。

这里，随机测试程序生成装置根据 ARTPG 中指令仿真器对应的处理器状态，初始化合法指令集；初始化合法指令集与后面的修改合法指令集的标准一致。如图 7 所示，合法指令集为一个动态树状链表，在每一次选择和生成随机指令后可以动态修改合法指令集树状链表。需要说明的是，采用其它的数据结构类型同样适用。

步骤 103：初始化 ARTPG 状态。

在步骤 103 中，随机测试程序生成装置使用随机程序约束初始化 ARTPG 状态。

这里，所述随机程序约束包括两个约束条件：

第一个约束条件为终止条件，终止条件包括描述随机测试程序生成的终止条件；随机测试程序生成的终止条件设置为：程序大小达到一定标准；程序执行周期数达到一定数值；及测试程序的功能覆盖率达到一定标准。其中，功能覆盖率可以分级描述，所有级别的功能覆盖率满足标准时终止，程序大小和程序执行周期数比覆盖率的设置具有更高优先级。当功能覆盖率作为终止条件时，程序大小和程序执行周期数被设置为 0。

第二个约束条件为合法性检测条件，合法性检测条件为描述死循环的

判定标准。

步骤 104：判断生成的随机测试程序是否满足终止条件。

这里，所述判定标准包括：程序执行周期数是否满足终止条件；程序大小是否满足终止条件；及功能覆盖率是否满足终止条件。

步骤 105：当所述随机测试程序不满足终止条件时，随机选择所述随机测试程序的第二发射指令数。随机测试程序生成装置随机选择当前周期需要并行发射的第二发射指令数。

步骤 106：判断所述随机测试程序生成的第一发射指令数是否等于随机选择的第二发射指令数。

步骤 107：当所述随机测试程序生成的第一发射指令数等于随机选择的第二发射指令数时，选择功能单元。

这里，如图 4 所示，功能单元属于覆盖率收集，即反馈控制随机权重的数据结构，覆盖率收集的数据结构属于指令生成器状态。如图 3 所示，指令生成器状态包含当前的代码大小（Code Size）、周期数（Cycle Count）、覆盖率状态（Coverage State）。指令生成器状态还包括与覆盖率状态相关的对于处理器硬件结构和指令集架构（ISA，Instruction Set Architecture）覆盖率收集的数据结构。其中，如图 5 和 6 所示，覆盖率收集的数据结构包括功能单元列表、指令列表和操作数链表。指令列表中的函数指针指向指令功能对应的处理函数，用来支持指令仿真。

需要说明的是，步骤 110 至步骤 140 中需要步骤 101-107 中选择的第一发射指令数和功能单元，以便根据第一发射指令数和功能单元选择指令包。至此，随机测试程序生成的过程就完成了。

实施例 2

图 8 为本发明实施例 2 提供的随机测试程序生成方法的实现流程示意图，如图 8 所示，所述方法包括：

步骤 201：初始化处理器初始状态。

步骤 202：初始化合法指令集。

步骤 203：初始化 ARTPG 状态。

步骤 204：判断随机测试程序是否满足终止条件。

步骤 204 判断为是后，执行步骤 218。

步骤 218：输出随机测试程序、随机测试程序执行结果和覆盖率分析报告。

步骤 219：释放空间并退出。

实施例 3

图 8 为本发明实施例 3 提供的随机测试程序生成方法的实现流程示意图，如图 8 所示，所述方法包括：

步骤 201：初始化处理器初始状态。

步骤 202：初始化合法指令集。

步骤 203：初始化 ARTPG 状态。

步骤 204：判断随机测试程序是否满足终止条件。

步骤 205：随机选择所述随机测试程序的第二发射指令数。

步骤 206：判断随机测试程序生成的第一发射指令数是否等于随机选择的第二发射指令数。

步骤 206 判断为是后，执行步骤 204。

步骤 204：判断随机测试程序是否满足终止条件。

步骤 218：输出随机测试程序、随机测试程序执行结果和覆盖率分析报告。

步骤 219：释放空间并退出。

本实施例是当随机测试程序生成的第一发射指令数等于随机选择的第二发射指令数且满足终止条件时提供的随机测试程序生成方法。

实施例 4

图 8 为本发明实施例 4 提供的随机测试程序生成方法的实现流程示意图，如图 8 所示，所述方法包括:

步骤 201: 初始化处理器初始状态。

步骤 202: 初始化合法指令集。

步骤 203: 初始化 ARTPG 状态。

步骤 204: 判断随机测试程序是否满足终止条件。

步骤 205: 随机选择所述随机测试程序的第二发射指令数。

步骤 206: 判断随机测试程序生成的第一发射指令数是否等于随机选择的第二发射指令数。

步骤 207: 选择功能单元。

步骤 208: 在合法指令集中选择指令包。

步骤 209: 判断指令包是否包括分支跳转指令。

步骤 209 判断为否后，执行步骤 214。

步骤 214: 选择普通指令操作数。

步骤 215: 修改 ARTPG 状态。

步骤 216: 修改处理器状态。

步骤 217: 修改合法指令集。

步骤 206: 判断随机测试程序生成的第一发射指令数是否等于随机选择的第二发射指令数。

步骤 206 判断为是后，执行步骤 204。

步骤 204: 判断随机测试程序是否满足终止条件。

步骤 204 判断为是后，执行步骤 218。

步骤 218: 输出随机测试程序、随机测试程序执行结果和覆盖率分析报告。

步骤 219：释放空间并退出。

本实施例是当随机测试程序的指令包不包括分支跳转指令时提供的随机测试程序生成方法。

实施例 5

图 8 为本发明实施例 5 提供的随机测试程序生成方法的实现流程示意图，如图 8 所示，所述方法包括：

步骤 201：初始化处理器初始状态。

步骤 202：初始化合法指令集。

步骤 203：初始化 ARTPG 状态。

步骤 204：判断随机测试程序是否满足终止条件。

步骤 205：随机选择所述随机测试程序的第二发射指令数。

步骤 206：判断随机测试程序生成的第一发射指令数是否等于随机选择的第二发射指令数。

步骤 207：选择功能单元。

步骤 208：在合法指令集中选择指令包。

步骤 209：判断指令包是否包括分支跳转指令。

步骤 210：选择分支跳转指令操作数。

步骤 211：判断随机测试程序是否存在第一死循环风险。

步骤 212：修改随机测试程序。

步骤 213：判断随机测试程序是否存在第二循环风险。

步骤 213 判断为是后，执行步骤 208。

步骤 208：在合法指令集中选择指令包。

步骤 209：判断指令包是否包括分支跳转指令。

步骤 209 判断为否后，执行步骤 214。

步骤 214：选择普通指令操作数。

步骤 215：修改 ARTPG 状态。

步骤 216：修改处理器状态。

步骤 217：修改合法指令集。

步骤 206：判断随机测试程序生成的第一发射指令数是否等于随机选择的第二发射指令数。

步骤 206 判断为是后，执行步骤 204。

步骤 204：判断随机测试程序是否满足终止条件。

步骤 204 判断为是后，执行步骤 218。

步骤 218：输出随机测试程序、随机测试程序执行结果和覆盖率分析报告。

步骤 219：释放空间并退出。

本实施例是当随机测试程序存在第一死循环风险和第二循环风险时提供的随机测试程序生成方法。

实施例 6

图 8 为本发明实施例 6 提供的随机测试程序生成方法的实现流程示意图，如图 8 所示，所述方法包括：

步骤 201：初始化处理器初始状态。

步骤 202：初始化合法指令集。

步骤 203：初始化 ARTPG 状态。

步骤 204：判断随机测试程序是否满足终止条件。

步骤 205：随机选择所述随机测试程序的第二发射指令数。

步骤 206：判断随机测试程序生成的第一发射指令数是否等于随机选择的第二发射指令数。

步骤 207：选择功能单元。

步骤 208：在合法指令集中选择指令包。

步骤 209: 判断指令包是否包括分支跳转指令。

步骤 210: 选择分支跳转指令操作数。

步骤 211: 判断随机测试程序是否存在第一死循环风险。

步骤 211 判断为否后，执行步骤 215。

步骤 215: 修改 ARTPG 状态。

步骤 216: 修改处理器状态。

步骤 217: 修改合法指令集。

步骤 206: 判断随机测试程序生成的第一发射指令数是否等于随机选择的第二发射指令数。

步骤 206 判断为是后，执行步骤 204。

步骤 204: 判断随机测试程序是否满足终止条件。

步骤 204 判断为是后，执行步骤 218。

步骤 218: 输出随机测试程序、随机测试程序执行结果和覆盖率分析报告。

步骤 219: 释放空间并退出。

本实施例是当随机测试程序存在第一死循环风险时提供的随机测试程序生成方法。

实施例 7

图 9 为本发明实施例 7 提供的随机测试程序生成装置的组成结构示意图，如图 9 所示，所述装置包括:

第一判断模块 310，配置为在合法指令集中随机选择指令包并生成所述指令包，判断所述指令包是否包括分支跳转指令;

第二判断模块 320，配置为当所述指令包包括所述分支跳转指令时，判断所述随机测试程序是否存在第一死循环风险;

第三判断模块 330，配置为当所述随机测试程序存在所述第一死循环风

险时，修改所述随机测试程序，判断所述随机测试程序是否存在第二循环风险。

修改模块 340，配置为当所述随机测试程序不存在第二循环风险时，修改 ARTPG 状态、处理器状态和所述合法指令集。

具体地，所述第三判断模块 330，配置为修改所述随机测试程序包括：

当分支跳转地址超过程序空间界限时，修改所述分支跳转地址使其不超过所述程序空间界限；

当分支跳转判定条件在声明到使用之间没有修改操作时，增加修改指令；

当所述修改操作的收敛方向与所述分支跳转判定条件不一致时，修改所述修改操作的收敛方向使其与所述分支跳转判定条件一致；

当所述指令包包括函数调用指令或软件中断指令时，添加现场保存及恢复的指令。

具体地，所述修改模块 340，配置为修改所述 ARTPG 状态包括：

根据功能覆盖率反馈信息调整随机权重。具体地，所述修改模块 340，配置为修改所述合法指令集包括：

将修改后的输出寄存器添加到对应的合法输入操作数；

将所述合法输入操作数对应的指令添加到所述合法指令集；

当所述指令包包括所述函数调用指令或所述软件中断指令时，将当前返回地址压至堆栈，在所述函数调用指令或所述软件中断指令前添加所述现场保存的指令。

进一步地，所述第三判断模块 330，还配置为在所述修改所述随机测试程序之后及在所述判断所述随机测试程序是否存在第二循环风险之前，执行指令局部仿真或者执行指令全局仿真。

进一步地，所述装置还包括：

输出模块 350，配置为当所述随机测试程序生成的第一发射指令数等于随机选择的第二发射指令数且所述随机测试程序满足终止条件时，输出所述随机测试程序、随机测试程序执行结果和覆盖率分析报告。

在实际应用中，所述第一判断模块 310、第二判断模块 320、第三判断模块 330、修改模块 340 和输出模块 350 均可由任何编程语言基于任何软件或硬件平台编程实现。同时，所述随机测试程序生成方法可以直接或者经简单修改后生成针对任何目标处理器的随机测试程序。

在实际应用中，所述第一判断模块 310、第二判断模块 320、第三判断模块 330、修改模块 340 和输出模块 350 均可由位于任意计算机设备中的中央处理器（CPU，Central Processing Unit）、数字信号处理器（DSP，Digital Signal Processor）、微处理器（MPU）、或可编程逻辑阵列（FPGA，Field Programmable Gate Array）实现。

需要说明的是，本发明实施例中，如果以软件功能模块的形式实现上述的随机测试程序生成方法，并作为独立的产品销售或使用时，也可以存储在一个计算机可读取存储介质中。基于这样的理解，本发明实施例的技术方案本质上或者说对现有技术做出贡献的部分可以以软件产品的形式体现出来，该计算机软件产品存储在一个存储介质中，包括若干指令用以使得一台计算机设备（可以是个人计算机、服务器、或者网络设备等）执行本发明各个实施例所述方法的全部或部分。而前述的存储介质包括：U 盘、移动硬盘、只读存储器（ROM，Read Only Memory）、磁碟或者光盘等各种可以存储程序代码的介质。这样，本发明实施例不限制于任何特定的硬件和软件结合。

相应地，本发明实施例再提供一种计算机存储介质，所述计算机存储介质中存储有计算机可执行指令，该计算机可执行指令用于执行本发明实施例中随机测试程序生成方法。

相应地，本发明实施例再提供一种随机测试程序生成设备（计算机设备），所述设备包括：

存储介质，配置为存储计算机可执行指令；

处理器，配置为执行存储在所述存储介质上的计算机可执行指令，所述计算机可执行指令包括：在合法指令集中随机选择指令包并生成所述指令包，判断所述指令包是否包括分支跳转指令；当所述指令包包括所述分支跳转指令时，判断所述随机测试程序是否存在第一死循环风险；当所述随机测试程序存在所述第一死循环风险时，修改所述随机测试程序，判断所述随机测试程序是否存在第二循环风险；当所述随机测试程序不存在第二循环风险时，修改全自动随机测试程序生成器 ARTPG 状态、处理器状态和所述合法指令集。

应理解，说明书通篇中提到的“一个实施例”或“一实施例”意味着与实施例有关的特定特征、结构或特性包括在本发明的至少一个实施例中。因此，在整个说明书各处出现的“在一个实施例中”或“在一实施例中”未必一定指相同的实施例。此外，这些特定的特征、结构或特性可以任意适合的方式结合在一个或多个实施例中。应理解，在本发明的各种实施例中，上述各过程的序号的大小并不意味着执行顺序的先后，各过程的执行顺序应以其功能和内在逻辑确定，而不应对本发明实施例的实施过程构成任何限定。上述本发明实施例序号仅仅为了描述，不代表实施例的优劣。

需要说明的是，在本文中，术语“包括”、“包含”或者其任何其他变体意在涵盖非排他性的包含，从而使得包括一系列要素的过程、方法、物品或者装置不仅包括那些要素，而且还包括没有明确列出的其他要素，或者是还包括为这种过程、方法、物品或者装置所固有的要素。在没有更多限制的情况下，由语句“包括一个……”限定的要素，并不排除在包括该要素的过程、方法、物品或者装置中还存在另外的相同要素。

在本申请所提供的几个实施例中，应该理解到，所揭露的设备和方法，可以通过其它的方式实现。以上所描述的设备实施例仅仅是示意性的，例如，所述单元的划分，仅仅为一种逻辑功能划分，实际实现时可以有另外的划分方式，如：多个单元或组件可以结合，或可以集成到另一个系统，或一些特征可以忽略，或不执行。另外，所显示或讨论的各组成部分相互之间的耦合、或直接耦合、或通信连接可以是通过一些接口，设备或单元的间接耦合或通信连接，可以是电性的、机械的或其它形式的。

上述作为分离部件说明的单元可以是、或也可以不是物理上分开的，作为单元显示的部件可以是、或也可以不是物理单元；既可以位于一个地方，也可以分布到多个网络单元上；可以根据实际的需要选择其中的部分或全部单元来实现本实施例方案的目的。

另外，在本发明各实施例中的各功能单元可以全部集成在一个处理单元中，也可以是各单元分别单独作为一个单元，也可以两个或两个以上单元集成在一个单元中；上述集成的单元既可以采用硬件的形式实现，也可以采用硬件加软件功能单元的形式实现。

本领域普通技术人员可以理解：实现上述方法实施例的全部或部分步骤可以通过程序指令相关的硬件来完成，前述的程序可以存储于计算机可读取存储介质中，该程序在执行时，执行包括上述方法实施例的步骤；而前述的存储介质包括：移动存储设备、只读存储器（Read Only Memory，ROM）、磁碟或者光盘等各种可以存储程序代码的介质。

或者，本发明上述集成的单元如果以软件功能模块的形式实现并作为独立的产品销售或使用时，也可以存储在一个计算机可读取存储介质中。基于这样的理解，本发明实施例的技术方案本质上或者说对现有技术做出贡献的部分可以以软件产品的形式体现出来，该计算机软件产品存储在一个存储介质中，包括若干指令用以使得一台计算机设备（可以是个人计算

机、服务器、或者网络设备等）执行本发明各个实施例所述方法的全部或部分。而前述的存储介质包括：移动存储设备、ROM、磁碟或者光盘等各种可以存储程序代码的介质。

以上所述，仅为本发明的具体实施方式，但本发明的保护范围并不局限于此，任何熟悉本技术领域的技术人员在本发明揭露的技术范围内，可轻易想到变化或替换，都应涵盖在本发明的保护范围之内。因此，本发明的保护范围应以所述权利要求的保护范围为准。

工业实用性

本发明实施例提供的技术方案中，由于本发明实施例可以动态更新合法指令集和处理器状态，也使得随机测试程序的生成与合法性检测所述随机测试程序能够同时进行；这样，不但能全自动的生成处理器验证所需的随机测试程序，而且能有效的控制随机测试程序的合法性和功能覆盖率，进而能提升处理器验证的效率。

权利要求书

1、一种随机测试程序生成方法，所述方法包括:

在合法指令集中随机选择指令包并生成所述指令包，判断所述指令包是否包括分支跳转指令;

当所述指令包包括所述分支跳转指令时，判断所述随机测试程序是否存在第一死循环风险;

当所述随机测试程序存在所述第一死循环风险时，修改所述随机测试程序，判断所述随机测试程序是否存在第二循环风险;

当所述随机测试程序不存在第二循环风险时，修改全自动随机测试程序生成器 ARTPG 状态、处理器状态和所述合法指令集。

2、根据权利要求 1 所述的方法，其中，所述修改所述随机测试程序包括:

当分支跳转地址超过程序空间界限时，修改所述分支跳转地址使其不超过所述程序空间界限;

当分支跳转判定条件在声明到使用之间没有修改操作时，增加修改指令;

当所述修改操作的收敛方向与所述分支跳转判定条件不一致时，修改所述修改操作的收敛方向使其与所述分支跳转判定条件一致;

当所述指令包包括函数调用指令或软件中断指令时，添加现场保存及恢复的指令。

3、根据权利要求 1 所述的方法，其中，所述修改所述 ARTPG 状态包括:

根据功能覆盖率反馈信息调整随机权重。

4、根据权利要求 1 所述的方法，其中，所述修改所述合法指令集包括:

将修改后的输出寄存器添加到对应的合法输入操作数;

将所述合法输入操作数对应的指令添加到所述合法指令集;

当所述指令包包括所述函数调用指令或所述软件中断指令时，将当前返回地址压至堆栈，在所述函数调用指令或所述软件中断指令前添加所述现场保存的指令。

5、根据权利要求 1 至 4 任一项所述的方法，其中，在所述修改所述随机测试程序之后及在所述判断所述随机测试程序是否存在第二循环风险之前，所述方法还包括：执行指令局部仿真或者执行指令全局仿真。

6、根据权利要求 5 所述的方法，其中，所述修改 ARTPG 状态、处理器状态和所述合法指令集之后，所述方法还包括:

当所述随机测试程序生成的第一发射指令数等于随机选择的第二发射指令数且所述随机测试程序满足终止条件时，输出所述随机测试程序、随机测试程序执行结果和覆盖率分析报告。

7、一种随机测试程序生成装置，所述装置包括:

第一判断模块，配置为在合法指令集中随机选择指令包并生成所述指令包，判断所述指令包是否包括分支跳转指令;

第二判断模块，配置为当所述指令包包括所述分支跳转指令时，判断所述随机测试程序是否存在第一死循环风险;

第三判断模块，配置为当所述随机测试程序存在所述第一死循环风险时，修改所述随机测试程序，判断所述随机测试程序是否存在第二循环风险。

修改模块，配置为当所述随机测试程序不存在第二循环风险时，修改全自动随机测试程序生成器 ARTPG 状态、处理器状态和所述合法指令集。

8、根据权利要求 7 所述的装置，其中，所述第三判断模块，配置为

修改所述随机测试程序包括:

当分支跳转地址超过程序空间界限时，修改所述分支跳转地址使其不超过所述程序空间界限;

当分支跳转判定条件在声明到使用之间没有修改操作时，增加修改指令;

当所述修改操作的收敛方向与所述分支跳转判定条件不一致时，修改所述修改操作的收敛方向使其与所述分支跳转判定条件一致;

当所述指令包包括函数调用指令或软件中断指令时，添加现场保存及恢复的指令。

9、根据权利要求 7 所述的装置，其中，所述修改模块，配置为修改所述 ARTPG 状态包括:

根据功能覆盖率反馈信息调整随机权重。

10、根据权利要求 7 所述的装置，其中，所述修改模块，配置为修改所述合法指令集包括:

将修改后的输出寄存器添加到对应的合法输入操作数;

将所述合法输入操作数对应的指令添加到所述合法指令集;

当所述指令包包括所述函数调用指令或所述软件中断指令时，将当前返回地址压至堆栈，在所述函数调用指令或所述软件中断指令前添加所述现场保存的指令。

11、根据权利要求 7 至 10 任一项所述的装置，其中，所述第三判断模块，还配置为在所述修改所述随机测试程序之后及在所述判断所述随机测试程序是否存在第二循环风险之前，执行指令局部仿真或者执行指令全局仿真。

12、根据权利要求 11 所述的装置，其中，所述装置还包括:

输出模块，配置为当所述随机测试程序生成的第一发射指令数等于

随机选择的第二发射指令数且所述随机测试程序满足终止条件时，输出所述随机测试程序、随机测试程序执行结果和覆盖率分析报告。

13、一种计算机存储介质，所述计算机存储介质中存储有计算机可执行指令，该计算机可执行指令用于执行权利要求 1 至 6 任一项所述的随机测试程序生成方法。

14、一种随机测试程序生成设备，所述设备包括:

存储介质，配置为存储计算机可执行指令;

处理器，配置为执行存储在所述存储介质上的计算机可执行指令，所述计算机可执行指令包括:

在合法指令集中随机选择指令包并生成所述指令包，判断所述指令包是否包括分支跳转指令;

当所述指令包包括所述分支跳转指令时，判断所述随机测试程序是否存在第一死循环风险;

当所述随机测试程序存在所述第一死循环风险时，修改所述随机测试程序，判断所述随机测试程序是否存在第二循环风险;

当所述随机测试程序不存在第二循环风险时，修改全自动随机测试程序生成器 ARTPG 状态、处理器状态和所述合法指令集。

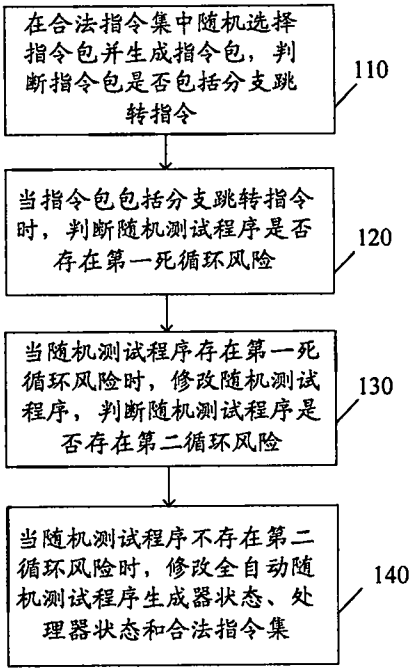

在合法指令集中随机选择指令包并生成指令包，判断指令包是否包括分支跳转指令
110
当指令包包括分支跳转指令时，判断随机测试程序是否存在第一死循环风险
120
当随机测试程序存在第一死循环风险时，修改随机测试程序，判断随机测试程序是否存在第二循环风险
130
当随机测试程序不存在第二循环风险时，修改全自动随机测试程序生成器状态、处理器状态和合法指令集
140

图 1

功能单元0 (FU0)	功能单元名	权重	次数	指令列表指针
功能单元1 (FU1)	功能单元名	权重	次数	指令列表指针
功能单元2 (FU2)	功能单元名	权重	次数	指令列表指针
功能单元3 (FU3)	功能单元名	权重	次数	指令列表指针
……	……	……	……	指令列表指针
功能单元n (Fun)	功能单元名	权重	次数	指令列表指针

图 4

指令0(Inst0)	指令名	函数指针	权重	次数	操作数列表指针
指令1(Inst1)	指令名	函数指针	权重	次数	操作数列表指针
指令2(Inst2)	指令名	函数指针	权重	次数	操作数列表指针
指令3(Inst3)	指令名	函数指针	权重	次数	操作数列表指针
……	……	函数指针	……	……	操作数列表指针
指令n(Instn)	指令名	函数指针	权重	次数	操作数列表指针

图 5

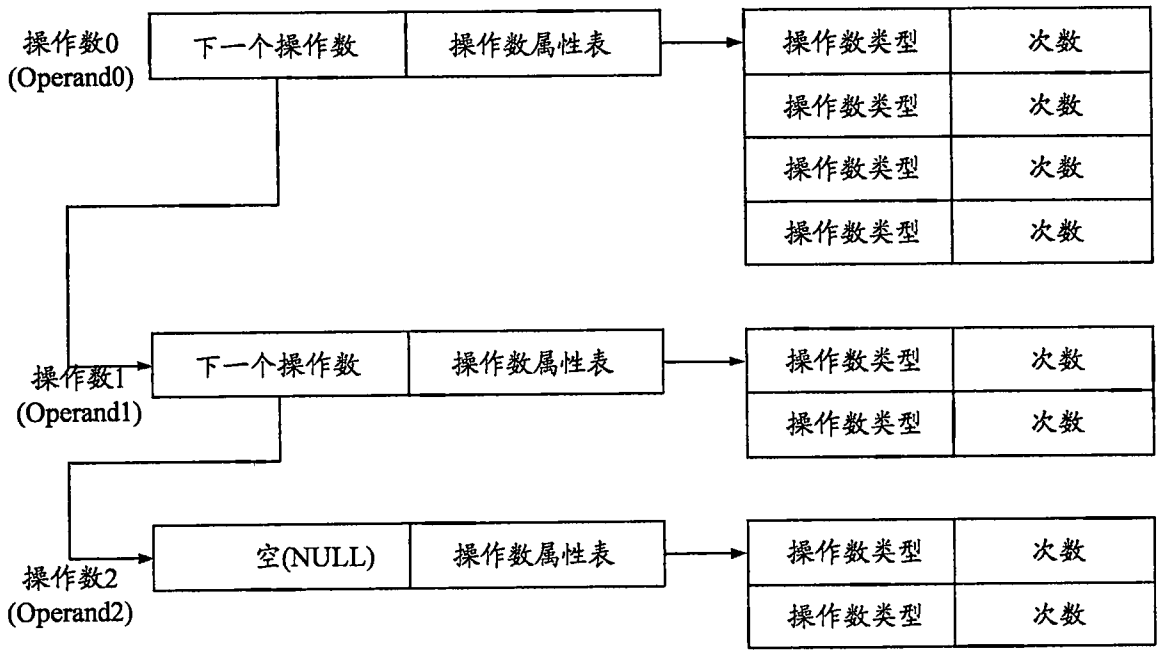

操作数0
(Operand0)
下一个操作数
操作数属性表
操作数类型
次数
操作数类型
次数
操作数类型
次数
操作数类型
次数
操作数1
(Operand1)
下一个操作数
操作数属性表
操作数类型
次数
操作数类型
次数
操作数2
(Operand2)
空(NULL)
操作数属性表
操作数类型
次数
操作数类型
次数

图 6

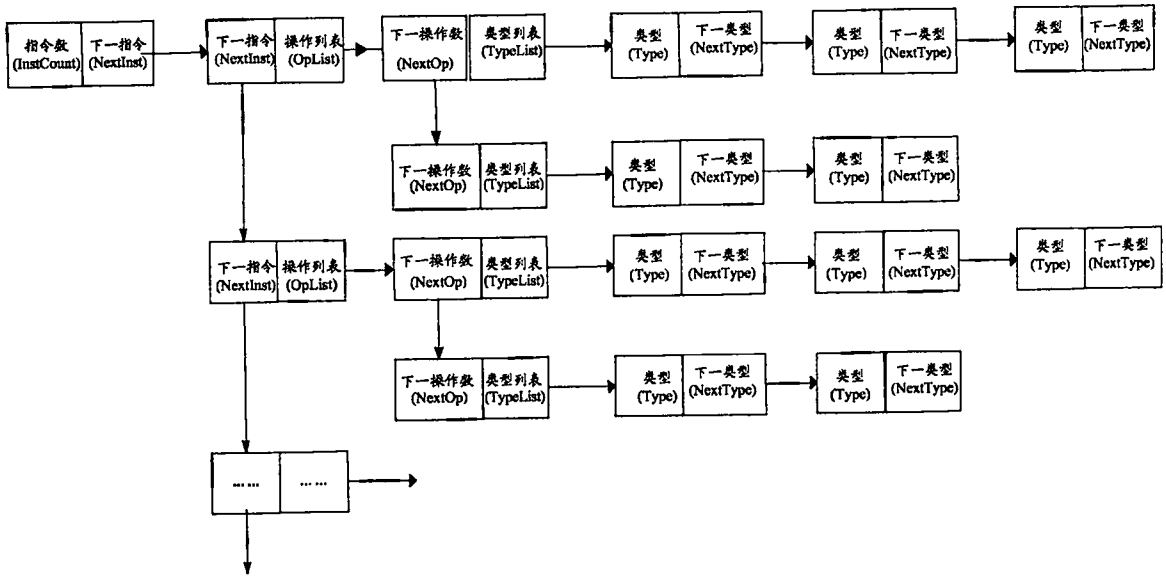

指令数
(InstCount)
下一指令
(NextInst)
下一指令
(NextInst)
操作列表
(OpList)
下一操作数
(NextOp)
类型列表
(TypeList)
类型
(Type)
下一类型
(NextType)
类型
(Type)
下一类型
(NextType)
类型
(Type)
下一类型
(NextType)
下一操作数
(NextOp)
类型列表
(TypeList)
类型
(Type)
下一类型
(NextType)
类型
(Type)
下一类型
(NextType)
下一指令
(NextInst)
操作列表
(OpList)
下一操作数
(NextOp)
类型列表
(TypeList)
类型
(Type)
下一类型
(NextType)
类型
(Type)
下一类型
(NextType)
类型
(Type)
下一类型
(NextType)
下一操作数
(NextOp)
类型列表
(TypeList)
类型
(Type)
下一类型
(NextType)
类型
(Type)
下一类型
(NextType)
…… ……

图 7

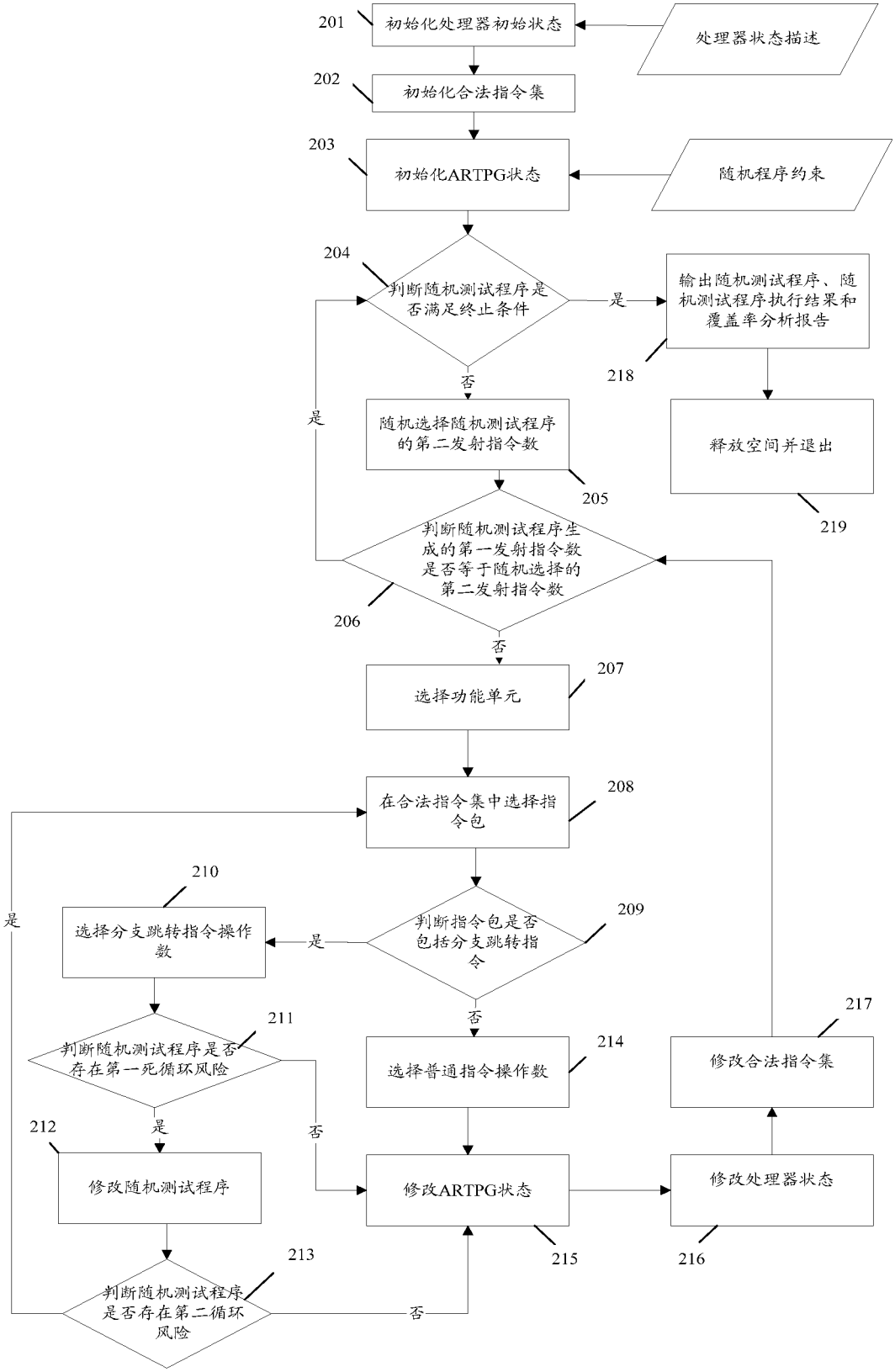

201
初始化处理器初始状态
处理器状态描述
202
初始化合法指令集
203
初始化ARTPG状态
随机程序约束
204
判断随机测试程序是否满足终止条件
是
输出随机测试程序、随机测试程序执行结果和覆盖率分析报告
218
否
是
随机选择随机测试程序的第二发射指令数
205
释放空间并退出
219
判断随机测试程序生成的第一发射指令数是否等于随机选择的第二发射指令数
206
否
207
选择功能单元
208
在合法指令集中选择指令包
210
是
选择分支跳转指令操作数
是
判断指令包是否包括分支跳转指令
209
否
211
判断随机测试程序是否存在第一死循环风险
是
否
212
修改随机测试程序
214
选择普通指令操作数
217
修改合法指令集
修改ARTPG状态
修改处理器状态
215
216
213
判断随机测试程序是否存在第二循环风险
否

图 8

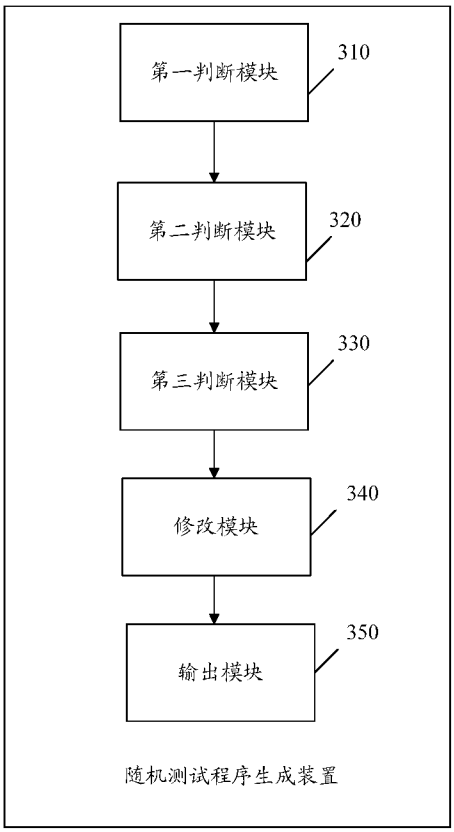

第一判断模块
310
第二判断模块
320
第三判断模块
330
修改模块
340
输出模块
350
随机测试程序生成装置

图 9