

DSP 随机测试程序自动生成技术

罗汉青, 梁利平, 叶甜春

(中国科学院 微电子研究所, 北京 100029)

摘 要: 提出了一种基于指令集结构, 并可与硬件设计和测试环境相契合的自动随机测试生成方法 ARTG(automatic random test program generation). 它包含用于构造指令的指令树、动态生成策略、分支跳转指令生成算法、流水线冲突预测算法和一套自动结果检查机制. 整个测试程序生成过程具有很高的自动化, 验证遇到 bug 时能够减少调试时间迅速找出 bug 方位. ARTG 已应用于自主研发的 IME—Diamond 数字信号处理器. 实验结果表明, 该方法产生的测试代码灵活高效, 加速了验证过程, 并且复杂度低.

关键词: 功能验证; 随机测试生成; 指令树; 动态策略; 结果检查

中图分类号: TP302

文献标识码: A

文章编号: 1000—7180(2013)11—0154—05

Automatic Random Test Program Generation for DSP

LUO Han-qing, LIANG Li-ping, YE Tian-chun

(Institute of Microelectronics of Chinese Academy of Sciences, Beijing 100029, China)

Abstract: In this paper, we present an instruction set structure based automatic random test program generation (ARTG), which can cooperate with hardware design and verification environment perfectly. It provides an instruction tree used for composing instructions, a kind of dynamic strategy, an algorithm for generation of branch and jump instructions, an algorithm for predicting pipeline hazard, and a result checking mechanism. ARTG also helps to high automation of verification progress, and locate the bugs rapidly. ARTG has been applied on IME—Diamond DSP. The experiment result shows that test program generated by ARTG are flexible and efficient with low complexity.

Key words: functional verification; random test generation; instruction tree; dynamic strategy; result checking

1 引言

随着微处理器复杂度和集成度越来越高, 功能验证遇到的挑战也日渐增多^[1]. 微处理器的功能验证已被公认为是硬件设计中的瓶颈, 其方法大致可分为两类: 形式化验证和基于模拟的验证. 形式化验证的方法近年来取得很大进展, 但是它适合于相对较小模块的验证, 对于微处理器验证作用有限. 因此, 基于模拟的验证方法仍然是微处理器功能验证的主要技术.

随机测试生成技术是指令级模拟验证方法的核心. 国内外学术界围绕随机测试生成技术都有许多讨论. 文献[2]提出了一套测试代码生成方法, 其中

有一个跳转管理模块专门管理跳转的生成; 文献[3]提出了 IBM 第三代随机程序生成器 Genesys-Pro 它包含测试模板语言、建模框架和生成引擎三部分, 验证人员可以非常灵活进行配置并产生高效测试代码; 文献[4]提出了一种可配置的随机测试生成方法, 验证人员可以根据参数分类配置, 该方法也提出了跳转指令生成算法以及解决 TLB 页表限制的算法; 文献[5]给出了一种验证程序生成方法, 该方法给根据指令的语法格式以及约束要求, 建立一套指令生成模型, 从而通过程序模板和状态转移路径产生随机指令序列; 文献[6]提出的测试程序自动生成技术, 是通过体系结构描述语言对目标处理器体系结构建模, 再使用约束求解技术来产生满足需求的

测试程序.

分析以上方法的优点和局限性,本文提出一种基于 IME-Diamond DSP 架构的随机指令生成序列生成方法.首先根据指令集格式特点建立指令树,然后通过具体参数的约束随机化产生随机指令序列;在此基础上,结合动态策略给出分支跳转指令生成算法;最后针对 IME-Diamond DSP 流水线特点提出了流水线冲突预测算法以及一套加速调试过程的自动结果检查机制.

2 总体设计思想

IME-Diamond DSP 采用自主 VLIW DSP 指令集和通用 CPU 指令集相结合的指令集结构.本文使用指令级验证方法,使用大量测试程序模拟验证.由于验证范围是整个芯片内核,这又要求测试程序的覆盖面大,不能局限于某些模块.因此从测试生成的角度看,生成测试程序要在必要的约束下尽量灵活.图 1 所示是本文整体验证设计框图.

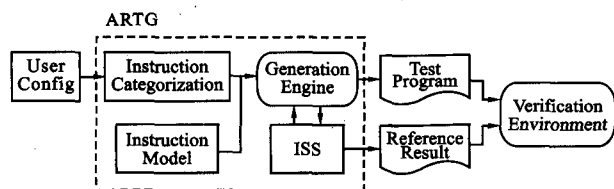


图 1 总体设计框图

由用户配置 ARTG,ARTG 将配置读入指令分类及选择部分(Instruction Categorization).指令模型(Instruction Model)包括指令树和参数约束,为了满足混合指令集的要求,这部分进行了一些不规则的设置.测试生成引擎(Generation Engine)根据指令分类和指令模型来产生指令所需参数,参数的最终产生需要参考模型(指令集仿真器 ISS)的反馈信息. ISS 是动态生成策略不可或缺的一部分.该策略是指测试生成引擎在产生指令参数时需要将该指令参数先送入 ISS 模拟,根据 ISS 模拟后提供的信息来判断是否采用本次产生的参数.由于 IME-Diamond 流水线在遇到相关时不做处理,本文在 ARTG 中添加了流水线冲突预测算法,该算法可以根据之前指令参数来预测会引起冲突的指令参数,从而避免冲突的发生.除此之外,与测试环境相结合,本文还提出了一套自动结果检查机制.该机制根据测试环境的需要来,由 ARTG 在产生测试程序的同时,也产生相配套的参考结果.该参考结果文件中包含开启测试环境自动检查结果标识,从而加速了定位 bug 的进程.

3 ARTG 设计实现

3.1 单指令构造

随机测试生成产生的每条指令都需要严格符合指令的语法结构.因此,在随机测试生成引擎中有一套完备的指令构造机制.各条指令所支持的各种语法格式都以指令模型的形式预先定义在测试生成引擎中.整个指令模型是一个树状结构.如图 2 所示,从根节点出发,按照语法格式将指令分成 ALU、LD/ST 等若干大类.进入其中某一格式大类之后,再根据源寄存器和目的寄存器数量、种类的不同,有无立即数,以及寻址方式的不同,进入不同的格式子类.当生成一条指令时,只需从根节点出发,选择一条路径达到其中任何一个叶节点即可.

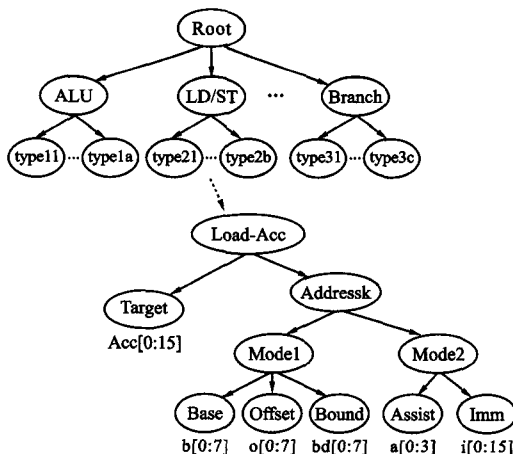


图 2 指令树

图 2 中给出了从存储器读数据载入累加器的格式指令 Load-Acc 的构造子树.首先看到 Load-Acc 格式子类属于 LD/ST 大类.然后进入该子类中,该格式包括:①一个类型是累加器的目的寄存器,该类寄存器的总数目是 16,参数可选范围是[0:15];②地址寄存器,该类寄存器根据不同类型的寻址方式使用不同的寄存器组.如图 2 给出的叶节点所示,当寻址方式是模式 1 时,地址寄存器使用的是基地址、偏移地址寄存器和边界寄存器,这 3 类寄存器各有 8 个,参数范围均是[0:7];当寻址方式是模式 2 时,使用辅助地址寄存器,该类可选寄存器有 4 个,参数范围是[0:3],另一个参数是立即数,其取值范围是[0:15].当所有参数选择完毕后即构成了一条完整的指令.

3.2 测试生成动态策略

在测试程序生成的过程中,某些指令可能会频繁引起异常发生,例如 load-word 地址非字对齐,而

在一般情况下不希望发生这种情况. 这就需要在测试生成过程中, 采用动态生成策略来避免诸如此类情况发生. 该策略的核心思想是在测试生成产生指令的同时, 将目前已生成的指令使用指令集模拟器模拟, 根据模拟结果来取舍当前生成的指令. 本文采用策略的流程如图 3 所示.

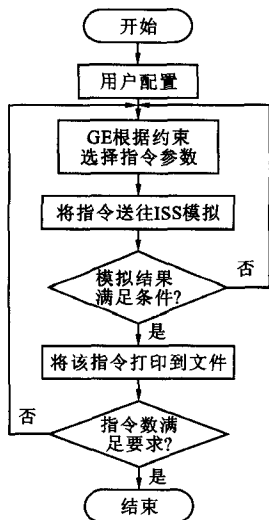


图 3 测试程序生成流程

在整个随机测试程序生成的过程中, 每当测试生成根据配置约束产生一条或一组指令后, 即刻将这些指令传递给指令集模拟器处理. 随后指令集模拟器将处理器的实时信息反馈回给测试生成引擎, 测试生成引擎根据这些反馈信息来判断当前产生的指令是否符合生成约束. 如果符合, 那么将当前指令送往打印文件; 如果不符合, 则舍弃当前指令, 测试生成引擎重新选择指令参数. 如此循环这个过程, 直到产生的指令数满足用户配置为止.

3.3 分支跳转指令生成算法

分支跳转指令的生成不同于其他指令, 它的操作的对象是 pc 值, 如果不慎重处理很容易生成带有死循环或者跳出指令地址范围的测试程序. 因此, 分支跳转指令的生成直接关系到整个测试程序的有效性. 本文的实现使用了两种不同的算法. 第一种本文称之为内嵌式算法, 第二种称为分段式算法.

3.3.1 内嵌式算法

内嵌式算法的思想是, 在指令流的生成过程中直接插入偏移量较小的分支跳转指令. 如果偏移量为正, 那么测试生成直接在分支指令之后继续添加生成一组指令, 一直到达目的 pc 值即可. 如果偏移量为负, 则需要在生成分支跳转指令之前追加生成一组指令, 用来确保该分支跳转指令不会使程序陷入死循环. 该算法实现步骤如下:

步骤 1: 如是当前需要生成一条分支跳转指令 (称之为 target), 那么先按照约束配置随机选择 pc 偏移量参数. 如果参数为正, 转移至步骤 5, 如果为负, 转移至步骤 2;

步骤 2: 根据约束配置选择一个寄存器充当计数寄存器 (称该寄存器为 r1), 选择另一个寄存器充当比较寄存器 (称该寄存器为 r2);

步骤 3: 从模拟器的反馈信息中获取 r1 当前的值, 设该值大小为 n, 并将该值增加一个固定量 g, 赋给 r2, 即 r2 的值将是 (n+g);

步骤 4: 在当前 pc 值处生成一组指令, 该组指令将执行如下内容: r1 的值自加 1; 将 r2 赋值为 (n+g); 判断 r1 和 r2 的值是否相等, 如果相等, pc 跳转到 (pc+8) 处, 即 target 的下一条指令处;

步骤 5: 根据步骤 1 中选择的参数生成指令 target;

步骤 6: 根据约束随机选择一条延时槽指令.

使用内嵌式生成算法主要基于如下两个原因:

①在很多应用程序中, 分支跳转指令所做的 pc 值的改变很小, 只是在某一局部范围内频繁跳转, 内嵌式算法可以产生大量随机程序验证该类情况; ②该算法不涉及复杂的 pc 运算, 实现起来非常简便. 正文内容.

3.3.2 分段式算法

分段式算法的核心思想是, 将随机生成指令部分与分支跳转指令目的地址处的代码分离开产生. 该算法产生的测试程序清晰的分为两段, 一段是随机指令流, 另一段是分支跳转指令的服务程序. 每生成一条分支跳转指令, 会相应的生成一个服务程序片段, 即每一段服务程序都是新产生的代码. 由于一些跳转指令的参数是绝对地址, 因此新产生的服务程序的起始地址都必须确定的. 每当生成一条分支跳转指令时, 随机测试生成引擎需要首先计算出当前服务程序的准确起始地址. 对于一个 32 位的指令编码, 分支指令偏移的编码通常是 16 位, 跳转范围有限. 因此, 一条分支跳转指令的服务程序不能距离太远, 这就限制了随机测试程序的总的指令数. 为了解决这一问题, 分段式算法核心思想的另一部分就在于, 按照指令的地址大小顺序, 随机指令流和服务程序代码交替产生. 每产生一段随机指令流, 即在临近的地址上安置服务程序. 新的随机指令流将在这两段代码地址之后产生. 如图 4 所示, 每一段 text 和一段特定的 service 相对应, 此每两段代码组成一个完整的段, 段与段之间相互独立. 本文采用

每段代码不超过 30000 行的方式,以确保每条分支指令的转移范围不会超过 16 位所表示的范围极限.该算法实现步骤如下:

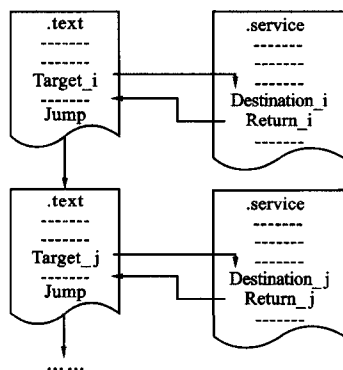


图4 分段式算法示意图

步骤 1: 获取当前所产生的指令总数, 计算当前 .text 段相应 .service 段的起始地址 base;

步骤 2: 获取当前 .service 段已有的代码量, 计算出即将产生程序的偏移 offset;

步骤 3: 使用 base 和 offset 计算出的地址, 在 .text 段产生当前所需生成目标指令的地址参数;

步骤 4: 在当前 .text 段生成目标指令;

步骤 5: 在 (base + offset) 地址处产生服务段代码。

该算法产生的好处有: ①可以有效避免内嵌式算法带来的代码冗余, 目标指令之前没有任何多余的代码; ②跳转范围较大, 且整个测试程序流紧凑, 不会造成大量代码浪费情况; ③调试方便, 由于随机程序流和服务程序段落分明, 当遇到与分支跳转指令有关的设计错误容易查找出错的位置。

3.4 指令冲突预测算法

IME-Diamond DSP 支持多指令同时发射, 一次可最多发射 8 条指令, 8 级流水. 为避免流水线上资源的读写冲突, 随机测试生成需要指令冲突预测算法来检测流水线上的各种指令和资源冲突情况, 从而产生满足处理器各种约束的有效指令序列. 概括说来, 该算法主要做两件事: ①分析同一发射组内的资源使用情况, 这包括功能部件以及各个寄存器的使用情况; ②分析流水线上寄存器的使用情况, 避免三种相关情况出现, 即 WAW、WAR 和 RAW. 由于各模块之间不同旁路的存在, 使得指令对寄存器的读写使用变得不规则, 给上述两种分析带来困难和麻烦. 而如果分析的粒度不够细, 又会使随机测试生成的约束过于严苛, 导致一些旁路无法验证到. 例如, load 类指令模块有一条旁路送到 mul 模块, 但

没有到 alu 模块的旁路. 那么, load 指令的目标寄存器对于 mul 类指令使用寄存器没有限制, 但是对于 alu 类指令使用相同的寄存器就会引起相关. 这种情况下, 算法需要能够甄别出这两种情况的差异, 否则将会漏掉 load 模块连接到 mul 模块的测试情况. 整个算法的关键部分在于寄存器资源使用情况标记, 将已经使用过的寄存器以及它的 bypass 信息、读或写、流水级信息记录在一个寄存器资源列表中, 当产生一条并行指令需要使用寄存器时, 可在此资源列表中查找会造成冲突的寄存器. 在具体的算法中, 需要维护一个寄存器使用链表以及一个寄存器使用标记数组. 寄存器使用链表 rscList 中的每个元素包含四个变量 {reg, pc, stage, bypass}, 它们分别代表寄存器编号、pc 值、流水级和 bypass 信息. 数组中的元素数与寄存器数相同, 每个元素包含一个变量, 表示该寄存器是否被标记. 算法如下:

输入: 所需要生成的指令列表 insList

输出: 一组并行发射的指令序列 insGroup

Instruction-generation(insList)

维护一个寄存器资源使用列表 rsclist

维护一个寄存器使用标记列表 reglabel

While FUnum! = 0

随机选择一条指令, 该指令所使用寄存器的信息写入 rscreq(pc, stage, bypass)

reglabel = Register-Analysis(rscreq)

在 reglabel 中随机选择未标记的寄存器作为指令的操作数 ops

使用 ops 生成单条指令 ins

将 ins 添加到 insGroup

将 (regi, pc, stage, bypass), 其中 regi ∈ ops, 添加到 rscList

-- FUnum;

Endwhile

返回 insGroup

Register-Analysis(rscreq)

For 每个 r ∈ Registerfile

For 每个 i ∈ rsclist

Diff1 = i.stage - rscreq.stage

Diff2 = (rscreq.pc - i.pc) / 4

If ((i.reg == r) && (Diff2 ≤ Diff1) && (i.bypass != rscreq.bypass))

在 reglabel 中标记 r

Endif

Endfor

Endfor
返回 reglabel

3.5 结果检查机制

将随机测试程序输入 DUV 后,需要有参考结果与 DUV 输出的结果相比较,以此判断 DUV 运行是否正确无误.因此验证方法中需要有一套自动比对结果的机制来加快验证进程.如图 5 所示,是本文采用的结果检查机制框图.

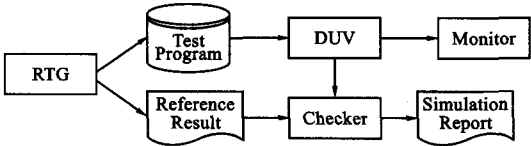


图 5 结果检查机制框图

本文采用的自动检测的原理是,随机测试生成在产生测试程序的同时也会生成一个参考结果文件,将测试程序输入 DUV,并将参考结果输入验证环境的检查器中,然后与 DUV 的结果相对比.结果检测机制的意义就在于将测试程序生成和结果检测无缝结合,使整个随机验证过程完全自动化,提高了验证效率.本文的实现方案是如下:①随机测试生成在一定的间隔处插入一个指令标识,以此作为一个 ISS 与 DUV 的同步点;②每当 ISS 侦测到一个同步点,它就将此时的寄存器数据打印到文件 results.dat 中;③在 RTL 环境下,检查器读入 results.dat,在同步点处取出 DUV 的寄存器数据进行比较.如果所有同步点处的比较都完全一样,则认为运行正确,继续自动验证过程;如果有不一致的结果,仿真过程将停止,从波形中查看错误的原因.由于错误极有可能在同步点之前就已经发生,传播到同步点处才被发现.这时可以用 ISS 来调试此同步点与上一个相邻同步点之间的这段代码,通过 ISS 的信息查明出错的具体位置.

4 实验结果

在实际验证工作中,已经使用了 ARTG 且收到了较好的结果.文中将从随机测试程序的并行度、代码有效长度内操作覆盖率、测试生成时间和覆盖率 4 个方面来说明 ARTG 的性能.图 6 所示的是并行度与代码有效长度内操作覆盖率的对比结果.并行度方面,IME-Diamond 最大并行度是 8,如果不采用本文所述的测试生成算法(此处称作 Random 方法),资源使用度平均仅为 4.4,而使用 ARTG 可以平均达到 7.5.代码有效长度指的是在不引起流水线冲突的情况下,所产生的代码长度.Random 方法

只有 35 条,而 ARTG 使用了冲突预测算法,可以完全避免冲突,从而一直产生有效代码直到满足指令数的配置值.在实验过程中,从语义上总共定义了 106 个操作.在有效代码长度内,Random 方法最多只能覆盖到 33%,而 ARTG 可以全部覆盖到 106 个操作.从表 1 可以看出,ARTG 的时间复杂度虽然高于 Random 方法,但是却远小于编译时间和 RTL 仿真,并不是整个验证环节的关键路径.

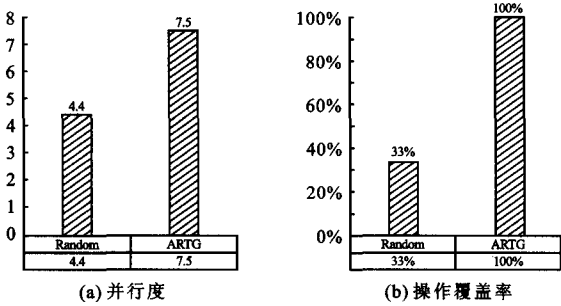


图 6 对比数据

表 1 时间复杂度

指令数	Random	ARTG	Compile	RTL simulation
50 万	7 s	10 s	125 s	>1 h
100 万	14 s	22 s	279 s	>4 h

5 结束语

文中提出了一种基于指令集结构的自动随机测试生成方法.该方法成功的应用于了 IME-Diamond DSP 的验证. ARTG 产生的测试程序被证明是灵活有效的.并且时间复杂度又比较低,提高了验证效率.同时也注意到,在用户所能配置参数部分该实现还有所欠缺,并且缺少当前比较热门的覆盖率驱动部分.在未来的工作中,我们将在这方面做的更加完善.

参考文献:

[1] ANICK Bergeron. Writing testbench: functional verification of HDL models[M]. Boston: Kluwer Academic Publisher, 2001

[2] Anoosh Hosseini. Code generation and analysis for the functional verification of microprocessors 2011 N[R]. Shoreline Blvd, Mountain View, CA:Silicon Graphics Inc, 1989.

[3] Adir A , Almog E, Fournler L, et al. Genesys-Pro: innovations in test program generation for functional processor verification [J]. IEEE Design & Test of Computers,2004,21(2):84-93

应用价值.

参考文献:

- [1] Czys D, Rajski J, Tyszer J. Low power test application with selective compaction in VLSI designs[C]// Test Conference, ITC 2012 IEEE International. Anaheim, CA; IEEE, 2012; 1-10.
- [2] Patrick Girard, Wen Xiaoqing, Nicola Nicolici. Power-aware testing and test strategies for low power devices[M]. Berlin; Springer, 2010; 31-50.
- [3] Lin Xjjiang. Low power testing-what can commercial DFT tools provide? [C]// Green Computing Conference and Workshops (IGCC), Orlando, FL; IEEE, 2011; 1-6.
- [4] Efi Arvaniti, Yiorgos Tsiatouhas. Low power scan by partitioning and scan hold[C]//IEEE 15 th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS), Tallinn; IEEE, 2012; 262-265.

- [5] JanuszRajski, Jerzy Tyszer, Mark Kassab, et al. Embedded deterministic test[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2004, 23(5): 776-792.
- [6] Dariusz Czys, Mark Kassab, Xijiang Lin, et al. Low-power scan operation in test compression environment [J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2009, 28(11): 1742-1755.
- [7] DariuszCzys, GrzegorzMrugalski, JanuszRajski, et al. Low-power test data application in EDT environment through decompressor freeze[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2008, 27(7): 1278-1290.

作者简介:

陈首智 男, (1988-), 硕士. 研究方向为超大规模集成电路可测型设计方向研究.

(上接第 158 页)

- [4] Shen H H, Ma L, Zhang H. CRPG: a configurable random test-program generator for microprocess. [C]//Proceedings of IEEE International Symposium on Circuits and Systems, Kobe; IEEE, 2005; 4171-4174
- [5] 姚英彪, 刘鹏, 姚庆栋, 等. 微处理器功能验证程序生成[J]. 计算机辅助设计与图形学学报, 2006, 18(10): 1484-1490.
- [6] 朱丹, 李瞰, 郭阳, 等. 微处理器体系结构级测试程序自动生成技术[J]. 软件学报, 2005, 16(12): 2172-2180.
- [7] Fine S, Ziv A. Coverage directed test generation for

functional verification using bayesian networks[C]// Proceeding of the 40 th ACM/IEEE Design Automation Conference (DAC)[s. l.]; IEEE, 2003; 286-291.

作者简介:

罗汉青 男, (1985-), 博士研究生. 研究方向为处理器验证与测试.

梁利平 男, (1969-), 博士生导师. 研究方向为高性能数字信号处理器、微处理器和高速混合信号集成电路设计技术.

叶甜春 男, (1965-), 博士生导师. 研究方向为集成电路制造技术、新型器件及微细加工技术.