文章编号:1001 - 9081(2007)08 - 1991 - 03

# 基于 SCV 的 MIPS 指令集指令随机生成工具

尚利宏,田 冰
(北京航空航天大学 计算机学院,北京 100083)
(shanglh@buaa.edu.cn)

**摘 要**:介绍了在 SystemC 的系统级建模与验证环境中,使用 SCV 实现用于 MIPS 指令集指令随机生成的工具。该随机指令生成工具包括静态生成与动态生成两种工作模式。并针对随机指令生成中会遇到的数据访问越界、分支/跳转地址越界等问题提出了解决办法。此外还研究了针对流水线冲突进行验证的方法。

**关键词**:随机指令生成;验证;SystemC;SCV;MIPS
**中图分类号**: TP311 **文献标志码**:A

# SCV based random instruction generation tool for MIPS instruction set

SHANG Li-hong, TIAN Bing
(*Computer Science and Engineering School, Beihang University, Beijing* 100083, *China*)

**Abstract:** A SCV-based random instruction generation tool used for generation of MIPS instructions in a system level modeling and simulation environment of SystemC was introduced. The tool consisted of two kinds of generation mode, the static mode and the dynamic mode. The problems in random instruction generation, such as data access violation, branch/jump violation and others, were solved. Additionally, a method with respect to the verification of pipeline conflict was provided.

**Key words:** random instruction generation; verification; SystemC; SystemC Verification (SCV); MIPS

## 0 引言

现在,嵌入式处理器正变得越来越复杂,所以对此类处理器的功能验证已逐渐成为整个设计流程中的瓶颈环节。仿真是在实际使用中用于大规模集成电路验证的重要方法[1]。要达到令人满意的验证效果,大量的测试用例是必不可少的[2]。在处理器的功能验证过程中,测试用例主要是符合该处理器的指令集规范的机器指令,可以手工编写,也可以使用工具自动生成。显然,除了少数特殊目的的验证以外,自动生成指令有着显著的优势。在 IBM 的 PowerPC、DEC 的 Alpha 21264 等处理器的验证中,都有报道使用了随机指令生成的方法[3,4]。

使用随机生成指令的方法可以迅速得到大量的测试用于处理器功能验证,但也存在着不足:1)所得到的指令序列可能难以深入某些功能特征,例如,RISC 处理器流水线中的依赖关系;2)随机生成的指令很容易引起处理器执行异常,诸如存储器访问越界、跳转地址越界等。因此,实际使用中的随机指令生成方法往往结合了一定的约束条件[6]。

## 1 使用 SystemC、SCV 验证 HDL 模型

SystemC 是一种基于 C/C++ 的建模语言,支持门级、RTL 级、系统级等各个抽象层次上硬件的建模和仿真。SCV 是紧密集成于 SystemC,并提供了有关验证功能的 C++ 类库。同时, SystemC 也是硬件描述语言,可以同已有的 HDL 模型进行有效的交互并协同仿真[7]。图 1 表示基于 SystemC 的用于在仿真环境下验证硬件模型的系统的基本结构。

本文介绍了一种用于图 1 所示验证环境,且基于 SCV 的随机指令生成工具,其中的待验证模块是符合 MIPS32 指令集规范的嵌入式处理器。

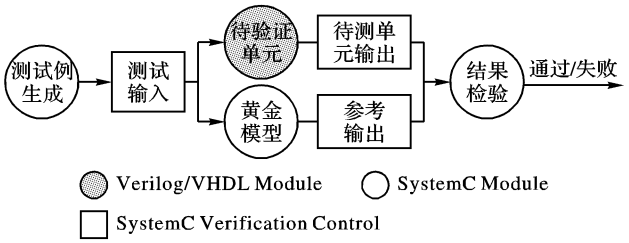


图 1 基于 SystmC 的系统协同仿真

## 2 仿真验证环境

### 2.1 基于 SystemC 的 MIPS 指令集仿真器

所介绍的验证系统中,需要一个待验证单元的“黄金模型”,其功能同待验证单元一致,用于产生待验证单元的预期输出。在处理器验证系统中,“黄金模型”常常是该处理器的 ISA 仿真器。为此专门实现了一个 MIPS ISA 仿真器,能够执行符合 MIPS 规范的机器指令。

### 2.2 基于 SCV 的随机指令生成器

该基于 SCV 的随机指令生成器属于带有一定约束条件的随机指令生成工具。

随机指令生成器的基本工作流程包括:读入配置文件并解析在配制文件中定义的约束条件,继而随机生成机器指令及对应的汇编指令。它可以工作于“静态指令生成”与“动态指令生成”两种模式。当进行“静态指令生成”时,该随机指

**收稿日期**:2007 - 02 - 07;**修回日期**:2007 - 04 - 19。

**作者简介**:尚利宏(1971 - ),男,甘肃平凉人,讲师,博士,主要研究方向:容错技术、容错处理器验证、软硬件协同设计; 田冰(1981 - ),男,辽宁沈阳人,硕士,主要研究方向:容错技术、容错处理器验证。

令生成工具作为与验证平台无关的独立程序而执行。而进行“动态指令生成”时,该随机指令生成工具作为软件模块嵌入到该基于 SystemC 的验证环境中。每当待验证处理器发出取指令请求,则调用该随机指令生成模块,并得到其随机生成的一条机器指令。在“动态指令生成”过程中,生成的指令与处理器所请求的目标地址无关,因而克服了“静态指令生成”中可能遇到的分支/跳转目标地址越界问题。但“动态指令生成”过程会影响验证系统整体执行速度。

还可以将该随机指令生成器配置为诸如“RANDOM”、“DEPENDENCY”等模式。“RANDOM”模式下,指令生成器随机生成指令;“DEPENDENCY”模式下,指令生成器生成用于验证流水线冲突的指令序列。此外还可以配置输出文件名,所需要生成的指令数量,指令类型,生成指令中使用的寄存器以及相关寄存器的初始值等内容。

## 3 实施方案

### 3.1 使用 SCV 类库生成带约束的随机序列

在 SCV 中,为随机序列生成提供了基础支持。scv_smart_ptr〈T〉类型的对象具有生成随机序列的基本功能,并可以通过 keep_only 和 keep_out 方法设置基本的约束,以使所生成的随机序列的取值被约束在合理的范围之内。而通过 scv_bag〈T,T〉类型的对象,可以定义随机序列中的每个元素出现的不同概率。

在 MIPS 指令中,有操作码、RT、RS、RD 寄存器、立即数等不同字段,为每个字段定义不同的 scv_smart_ptr〈T〉类型对象,并分别地设置约束条件,以生成合法的 MIPS 指令。而且,不同类型的指令具有不同的字段,应分别处理。而实际程序中,不同类型指令所占的比例显然大不相同,因此,还要为其设置不同的出现概率。

### 3.2 随机指令序列生成流程

指令随机生成过程如图 2 所示:首先,在所指定的约束条件下,随机生成指令的操作码,由一般操作码和扩展操作码两部分构成,一般操作码和扩展操作码共同确定一条指令的类型。在确定了指令类型之后,再依据不同的指令类型为该指令的其他字段设置合适的约束条件。

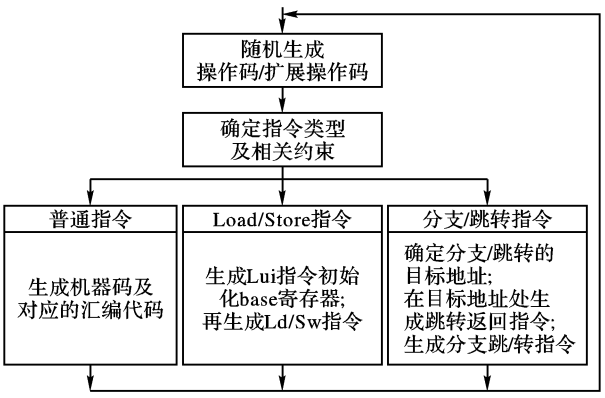


图 2 指令随机生成算法流程

#### 3.2.1 Load/Store 指令的处理

在生成的指令中,对存储器访问指令和分支/跳转两类需要进行特殊处理。如果所生成的指令是 Load/Store 类型的存储器访问指令,则需要避免出现存储器访问越界的情况[5]。以 Load Word 指令为例,其基本格式为:

LW rt, base( offset)

其中,rt 是目的寄存器,base 表示基址寄存器编号, offset 是 16 位立即数。在执行过程中,处理器将基址寄存器的内容与偏移值 offset 相加,得到目标地址。因此,为保证随机生成的指令的目标地址处于合法的地址空间内,需要对寄存器号 base 和立即数 offset,以及 base 号寄存器的内容[base]进行额外地约束。因此,在生成一条 Load Word 指令之前,还需要生成一条为该基址寄存器赋值的指令。在生成了 Load Word 指令之后,还要插入一条填充延迟槽的指令。

#### 3.2.2 分支/跳转系列指令的处理

对于分支/跳转指令,则需要考虑防止跳转目标地址超越了合法的指令空间。简单的将分支偏移值或者跳转目的地址设定在较小的范围内的方法是不可取的。例如,对于如下指令:

J target

如果标号 target 恰好为该指令所处地址之前的某地址,则会在所生成的指令序列中形成了死循环,这就无法利用随机生成的指令序列得到较高的验证覆盖率。同样道理,对于“B -2”这样的向后分支指令,也形成了在指令序列的局部重复的死循环。由于上述问题的存在,已有的静态指令随机生成工具往往回避对分支/跳转类指令的生成。在文献[8]中给出了一种解决方法,包括避免后向跳转,结合为每一合法地址生成记录等。本文使用另一种方法克服该问题。

具体方法是,在所生成的代码序列的尾部开辟一段空间专门用于存放分支/跳转类型的指令。在指令随机生成过程中,每当遇到分支/跳转指令时,则该分支/跳转指令的目标地址为指令序列尾部空间中的某一特定地址,在该地址处存在一条同当前分支/跳转指令相对应的分支/跳转指令,并且该相对应的分支/跳转指令将确保控制流转向该当前分支/跳转指令的下两条指令(即越过延迟槽)。

需要注意的是:1)在正常指令序列中的分支/跳转指令可以是条件分支指令,但在尾部跳转区中的分支/跳转指令则必须是无条件分支/跳转指令,以保证控制流能够顺利地返回;2)由于随机生成的分支/跳转指令的数目是不定的,因此尾部跳转区的长度也是不定的。所以,需要使用一定的数据结构保存有关尾部跳转区的信息,以便于计算每条分支/跳转指令所对应的尾部跳转区入口地址。该方法虽然固定了单条分支/跳转指令的目标地址和跳转方向,但所生成的随机指令序列中包含多条分支/跳转指令,而处于尾部跳转区中的分支/跳转指令又均为后向跳转,所以能够满足验证过程中对分支/跳转指令的目标地址及跳转方向等因素的覆盖率需求。

### 3.3 流水线冲突

在生成用于验证流水线功能的指令序列时,要考虑流水线冲突问题来确定相应的约束条件。流水线冲突通常被分为三类:结构冲突、数据冲突以及控制冲突。结构冲突已经通过使用分离的指令 Cache 和数据 Cache 而得以消除,而控制冲突则是在处理分支/跳转指令中必然涉及,因此在生成导致流水线冲突的指令序列时,仅考虑数据冲突。数据冲突的发生集中在指令序列中后一条指令的取指(ID)阶段,同时对照 MIPS 指令集结构中对于没有延迟的两条指令的流水线状态,某条指令的译码阶段在时间轴上与前两条指令各自的执行和访存阶段重叠,并且在前面第 3 条指令的写回阶段之后发生。在验证中,应重点考察某指令与其后两条指令的操作数字段

是否存在冲突[9]。

MIPS 指令集结构的处理器已经在硬件上消除了大部分的流水线冲突,而仅存在如下三种可能的情况:加载延时、乘法单元冒险以及 CP0 冒险。加载延时指紧随 Load 指令之后的指令不能访问 Load 的数据。乘法单元冒险指 MFHI 和 MFLO 指令之后的两条指令必须不是 MUL、DIV、MTHI 以及 MTLO。指令 CP0 冒险则同具体硬件实现相关,在此暂不考虑。

### 3.4　与原系统整合用于动态指令生成

图 3 表示在进行动态指令生成时,指令生成模块与原验证系统的结构关系。此时,用动态指令生成模块替代了原验证系统中的"Instruction Memory"部件,即在该 MIPS ISA 看来,它所面对的依然是一个指令存储器,只是该存储器的空间似乎是无限大,无论其请求怎样的分支目标地址,总是能从该存储器中获得指令。这也克服了在静态生成方式中遇到的分支/跳转地址越界问题。

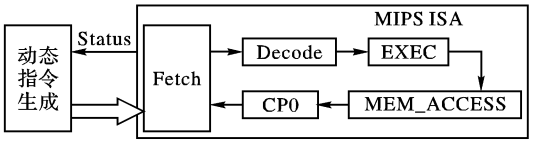


图 3　动态指令生成系统结构

## 4　实验

表 1 为一次静态指令生成过程中生成 4000 条指令(不包括软件异常、浮点、协处理器等指令类型)的情况下,所生成的各种指令的统计数据。"b"类型指令出现较多是因为对应于每条分支指令,都在尾部跳转区中生成了一条对应的"b"指令,"jr"对应于所生成的每条函数调用指令。而"lui"指令用于为 Load/Store 指令初始化基址寄存器,"nop"指令则用于填充访存指令、分支/跳转指令之后的延迟槽。

图 4 表示使用所生成的指令作为 MIPS ISA 的输入序列时,对其中的译码部件的功能覆盖率情况。随机生成的指令数在 600 条之后(约为指令种类的 10 倍),覆盖率超过了 90%。这表明该随机指令生成工具所生成的指令序列在空间上也具有较好的随机性,并可以随序列长度的增加而获得较高的相对于指令类型的覆盖率。

表 1　随机生成指令统计

| 指令 | 数目 | 指令 | 数目 | 指令 | 数目 | 指令 | 数目 | 指令 | 数目 | 指令 | 数目 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| add | 46 | bgezl | 5 | clz | 23 | lui | 362 | nop | 511 | sra | 43 |
| addi | 40 | bgzall | 14 | div | 41 | lw | 11 | nor | 51 | srav | 38 |
| addiu | 35 | blez | 9 | divu | 34 | lwl | 19 | or | 33 | srl | 45 |
| addu | 41 | blezl | 10 | j | 52 | lwr | 16 | ori | 30 | srlv | 35 |
| and | 32 | bltz | 20 | jal | 27 | mfhi | 40 | sh | 63 | sub | 42 |
| andi | 50 | bltzal | 4 | jarl | 40 | mhlo | 44 | sll | 35 | subu | 35 |
| b | 127 | bltzall | 9 | jr | 101 | movn | 44 | sllv | 47 | sw | 21 |
| beq | 19 | bltzl | 18 | lb | 18 | mthi | 32 | slt | 50 | swl | 31 |
| beql | 12 | bne | 13 | lbu | 12 | mtlo | 41 | slti | 44 | swr | 60 |
| bgez | 9 | bnel | 12 | lh | 18 | mult | 44 | sltiu | 29 | xor | 45 |
| bgezal | 7 | clo | 21 | lhu | 10 | multu | 34 | sltu | 56 | xori | 40 |

通过定义每类指令随机生成的概率,有助于进行带有针对性的功能验证。而在使用所实现的MIPS ISA测试该随机指令生成工具所生成的指令序列过程中,意外的通过一次执行异常,发现了所实现的 MIPS ISA 中有关 SRL 指令的缺陷(SRL,向右移位指令,当移位的次数小于 0 时出现异常)。

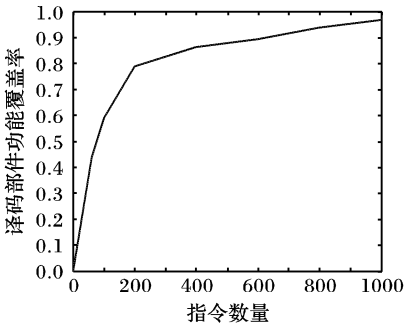


图 4　指令数量—功能覆盖率曲线

## 5　结语

基于 SCV 的随机指令生成工具可以较方便地实现具有一定约束条件的指令随机生成。在随机指令生成过程中,每类指令的出现概率可以控制。所提出新的方法较好地克服了已有指令随机生成工具中有关分支/跳转指令难于控制目标地址方面的不足,并且能够满足对目标地址及跳转方向的覆盖率需求。此外,该随机指令生成工具可以作为系统模块嵌入到基于 SystemC 的系统级验证环境之中,并提供符合 SystemC 标准的接口。

**参考文献:**

[1] LIANG Z S, YAN X L, WANG J B, *et al.* A dynamic random instruction and stimulus generation for functional verification of embedded processor[C]// Proceedings of 5th International Conference on ASIC, Beijing. [S.l.]: IEEE Press, 2003,1: 459 - 462.

[2] PIXLEY C, STRADER N, BRUCE W, *et al.* Commercial design verification: methodology and tools[C]// Proceedings of the IEEE International Test Conference on Test and Design Validity. Washington, DC: IEEE Computer Society, 1996: 839 - 848.

[3] AHARON A, GOODMAN D, LEVINGER M, *et al.* Test program generation for functional verification of PowerPC processor in IBM [C]// Proceedings of the 2nd ACM/ IEEE Design Automation Conference. San Francisco: ACM Press, 1995: 279 - 285.

[4] KANTROWITZ M , NOACK L M . Functional verification of a multiple-Issue, pipelined, superscalar Alpha processor: the Alpha 21164 CPU chip[J]. Digital Technical Journal, 1995,37(1): 89 - 99.

[5] 王涛. 一种新型微处理器功能验证[J]. 电子测量技术,2003,22(5): 47 - 48.

[6] CHANG T-C, IYENGAR V, RUDNICK E M. A biased random instruction generation environment for architectural verification of pipelined processors[J]. Journal of Electronic Testing, 2000, 16(1/2): 13 - 27.

[7] ROSE J, SWAN S. SCV randomization[EB/OL]. [2006 - 12 - 26]. www. systemc. org.

[8] LEVHARI Y. Dynamic pseudo-random assembly test generation in CPU verification[EB/OL]. [2006 - 12 - 26]. http://www. veri-sure. com/papers/cpuver. doc.

[9] SWEETMAN D. MIPS 处理器设计透视: See MIPS Run[M]. 赵俊良,张福新,陶品,译. 北京: 北京航空航天大学出版社, 2005: 363 - 365.