

面向 RTL 的 VHDL 语言模拟系统设计与实现

孙凌宇¹, 冷 明^{1,2}, 魏斯民¹, 杨 威¹

(1 井冈山大学 计算机科学系, 江西 吉安 343009; 2 上海大学 计算机工程与科学学院, 上海 200072)

摘 要: 设计并实现了一种面向寄存器传输级的 VHDL 语言模拟系统(RTL-based VHDL Simulator, RVS). 介绍了 RVS 系统的处理流程和组成模块. RVS 系统定义了面向寄存器传输级的 VHDL 语言子集, 在编译阶段采用了一种基于递归的自顶向下语法分析算法, 在模拟阶段采用了一种具有调试功能的基于进程的事件驱动模拟调度算法. RVS 系统在 Windows 平台下用 Visual Studio 2003 进行了实现. 实验表明, RVS 系统对组合逻辑控制和微程序控制的 SAP-CPU 设计电路文件进行了正确地编译和模拟.

关键词: 寄存器传输级; VHDL 语言; 模拟; 编译; 调度算法

中图分类号: TP391

文献标识码: A

文章编号: 1000-7180(2010)02-0167-04

Design and Implementation of RTL-Based VHDL Simulator

SUN Ling-yu¹, LENG Ming^{1,2}, WEI Si-min¹, YANG Wei¹

(1 Computer Science Department, Jinggangshan University, Ji'an 343009, China;

2 School of Computer Engineering and Science, Shanghai University, Shanghai 200072, China)

Abstract: Register Transfer Level (RTL)-based VHDL Simulator is proposed and implemented, named RVS. The RVS's flow, composition modules as well as their functions is described. Firstly, the RTL subset of VHDL language in RVS is defined. Furthermore, a top-down syntax analysis algorithm based on recursion is developed during the compilation phase. During the simulation phase, an event-driven schedule algorithm based on process is proposed with debugging functions. We give the implementation of RVS in Windows OS and Visual Studio 2003. Finally, the experiment and analysis show that the compiler and simulator of RVS perform well and produce encouraging solutions correctly on two SAP-CPU designs controlled by combinational logic and micro-program.

Key words: register transfer level (RTL); VHDL Language; simulation; compilation; schedule algorithm

1 引言

VHDL (Very High Speed Integrated Circuit Hardware Description Language) 语言作为国际标准硬件描述语言, 在电子设计自动化中已成为主要的描述工具, 可用于数字电路系统的描述、模拟和综合等^[1]. 文中设计并实现了一种面向寄存器传输级的 VHDL 语言模拟系统, 它输入面向寄存器传输级的 VHDL 语言描述的 VLSI 设计, 经过相应的编译器生成中间代码, 然后从波形文件读取外部激励信号, 施加在中间代码抽象出的行为模型上进行模拟, 通

过观察该行为模型在外部激励信号作用下的反应, 判断 VLSI 设计是否实现预期的功能^[2-3].

2 面向寄存器传输级的 VHDL 语言模拟系统 RVS

2.1 RVS 系统的组成及处理流程

基于面向寄存器传输级的 VHDL 语言子集, 设计并实现了其模拟系统 RVS, 用于对寄存器传输级的 VLSI 设计的验证和查错. 如图 1 所示, RVS 系统主要包括两个部分: (1) RVS 系统前端的编译器; (2) RVS 系统内核的模拟器. RVS 输入的是采用面

收稿日期: 2009-01-07; 修回日期: 2009-03-18

基金项目: 上海市教育委员会科研创新项目(08YZ13); 上海市教育委员会发展基金项目(01A05)

向寄存器传输级的 VHDL 语言描述^[4]的 VLSI 设计电路文件,由 RVS 编译器生成内部中间代码,然后由 RVS 模拟器基于中间代码抽象出行为模型,读取外部激励信号并施加在行为模型上,调用模拟调度算法,得到相应的模拟结果,并显示输出。

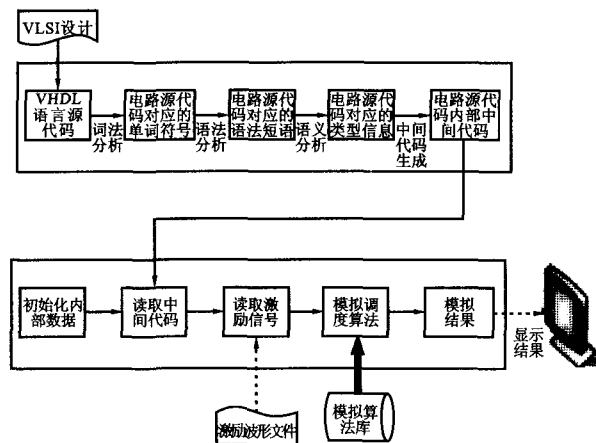


图1 面向RTL的VHDL语言模拟系统的流程图

2.2 RVS 组成模块

2.2.1 RVS 编译器

RVS 编译器实现的步骤是:首先对面向寄存器传输级的 VHDL 语言描述的 VLSI 设计电路文件进行词法分析,即从左到右逐个读入源代码,对构成源代码的字符流进行扫描和分解,从而识别出一个一个单词,得到对应的单词序列;然后进行语法分析,依据 VHDL 语言的语法规则,将单词序列分解成各类语法短语;接着进行语义分析,审核源代码有无语义错误,为中间代码的生成阶段收集类型信息;最后是中间代码生成,将源代码生成为用内部中间格式表示的中间代码^[5]。

在 RVS 编译器语法分析阶段,文中采用了一种基于递归的自顶向下语法分析算法,具体描述请见文献^[6]。该算法的基本思想是结合 VHDL 语言的自身特点,将基于对文法要求唯一确定产生式的确定思想和基于穷举试探方法的不确定思想相结合,降低对文法的要求,允许多个候选产生式存在,同时将穷举试探的范围局限于该非终结符的多个候选产生式中,从而实现在最小范围内对各种可能性的穷举试探方法。其特点是:VHDL 语言的语法规则容易转换到文法的描述,增强其可读性,减低维护难度,并且能克服不确定的自顶向下分析方法具有的回溯分析代价较高的缺点。

2.2.2 RVS 模拟器

RVS 模拟器采用的是具有调试功能的基于进

程的事件驱动模拟调度算法。它基于 RVS 编译器抽象出的行为模型,依据事件驱动进行模拟调度。如图 2 所示,整个行为模型可采用进程集合来表示,RVS 模拟器基于时间顺序的时间映射事件表来模拟进程之间的并行性。在面向寄存器传输级的 VHDL 语言中,整个寄存器传输级的 VLSI 设计行为模型 B 由若干独立的、并行执行的进程 P 及若干信号 S 组成: $B = (P, S)$ 。其中,每个操作定义为一个进程 $p(p \in P)$,它由一组顺序语句 $q_1, q_2, \dots, q_n$ 组成: $p = (q_1, q_2, q_3, \dots, q_n)$;信号 $s(s \in S)$ 是协调各进程之间通信机制的数据通路,进程 p 可以看成 s 的函数 $p(s)$;由信号值来控制各进程的运行,并确定系统的输入和输出;信号 s 的值 v 是关于时间 t 的序列: $v = (v_1, v_2, v_3, \dots, v_t, \dots)$,称为信号 s 的波形,其中每个新值 v_i 称为 s 的一个事件 e ,记为 $e = (v_i, t_i)$ ^[7-8]。

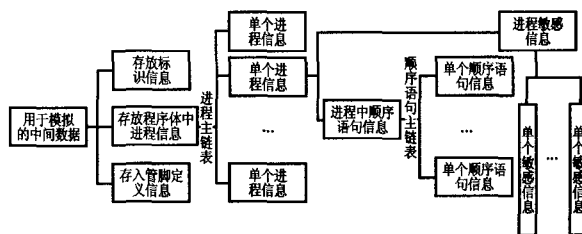


图2 面向RTL的VHDL语言模拟调度算法的行为模型

3 具有调试功能的基于进程的事件驱动模拟调度算法

该模拟调度算法的模拟周期处理流程如图 3 所示。模拟过程从零时刻开始,每当一个新的模拟时刻开始,首先对该模拟时刻激励信号事件的信号新值与该信号记录的旧值进行比较;如果新值与旧值相同,则把该事件删除,不再对它进行处理;否则,用新值取代旧值记录下来,同时查找被激活的进程,并计算模型中被激活进程的信号值;任意信号的值的变化将导致新事件产生,将新事件加入到当前事件列表,直至因当前时刻事件列表无事件、所有被激活的进程都被挂起,此时,该模拟时刻模拟结束,进入到下一个模拟时刻。

面向寄存器传输级的 VHDL 语言的并发进程机制在模拟算法中是通过模拟周期来实现的。每一个模拟时刻的模拟包括 3 个步骤:①当前模拟时刻进入到下一个时刻,此时驱动源被激活,或者等待延迟的进程被恢复。②更新信号值。这一个步骤中,当前模拟时刻内得到新值的所有信号被更新。任意信

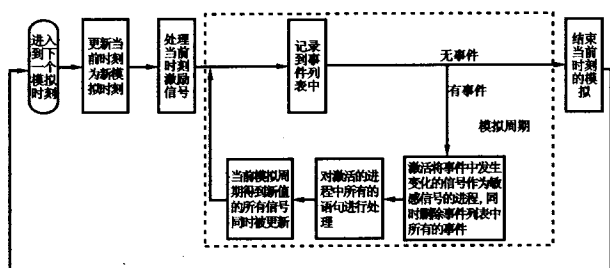


图3 面向 RTL 的 VHDL 语言模拟调度算法
模拟周期处理流程

号的值的变化都导致一个新事件的产生,并激活所有等待该信号的进程。③计算被激活的进程。如果计算结果是给定信号产生的一个事件,则把该事件放置到当前时刻事件列表中,在下一个模拟周期中对该信号敏感的进程语句就会重新计算,直至当前时刻事件列表无事件、所有被激活的进程都被挂起。每一个周期完成之后,模拟时钟被设置为下一个事件发生的时刻,模拟周期再次开始循环^[3]。

具有调试功能的基于进程的事件驱动模拟调度算法的基本步骤如下:

- 步骤 1:清空未来时刻事件列表;
- 步骤 2:接收外部激励信号事件,加入未来时刻事件列表;
- 步骤 3:当前时刻=0;
- 步骤 4:While (未来时刻事件列表存在事件 and 当前时刻不等于模拟结束时间)
 - 步骤 4.1:清空当前时刻的事件列表;
 - 步骤 4.2:While(在未来时刻事件列表中的当前时刻事件不为空)
 - 步骤 4.2.1:在未来时刻事件列表中删除该事件;
 - 步骤 4.2.2:If (事件中保存的值与旧值不同) 保存该事件到当前时刻事件列表中,并更新相应信号值;
 - 步骤 4.2.3:连接计数器加1;
 - 步骤 4.3:按被观察对象列表中的要求,记录发生变化信号的新值;
 - 步骤 4.4:While(当前时刻事件列表不为空)
 - 步骤 4.4.1:激活将当前时刻事件列表中的事件当作敏感信号的进程,同时清空当前时刻事件列表;
 - 步骤 4.4.2:计算被激活的进程,将新事件插入到当前时刻事件列表中;
 - 步骤 4.4.3:更新当前时刻事件列表的事件对应的信号值;
 - 步骤 4.4.4:按被观察对象列表中的要求,记录发生变化的信号值;
 - 步骤 4.5:If (该时刻有时间断点) 暂停模拟处理调试命令;
 - 步骤 4.6:当前时刻=下一个模拟时刻。

其中,步骤 4.2 通过对激励信号事件列表按照时间排序,使得在处理完当前模拟时刻的同时,指向下一个模拟时刻,提高了模拟时刻切换的效率;步骤 4.3 通过增加被观察对象列表,在模拟过程中只记载被观察对象的信号变化,减低了对存储空间的需求;步骤 4.4.1 通过增加进程的敏感信号与相应被激活的进程之间的关联,使得在模拟过程中可以快速判定出被激活的进程,提高了模拟器的速度;步骤 4.4.2 通过在编译器中间代码的语句列表中增加中断标识,提供了语句断点的调试功能;步骤 4.4.4 通过在以时间排序的激励信号事件列表中增加中断标识,提供了时间断点的调试功能。

4 RVS 实验设计及结果分析

针对 RVS 的实验,文中选用的两个实例为组合逻辑控制和微程序控制的 SAP-CPU,其体系结构分别如图 4 和图 5 所示。其中,SAP-CPU 指令格式低 4 位为地址码,支持直接寻址和零位寻址;指令格式高 4 位为操作码,包含 LAD 装载指令、ADD 加法、SUB 减法指令、OUT 输出指令和 HLT 停机指令。

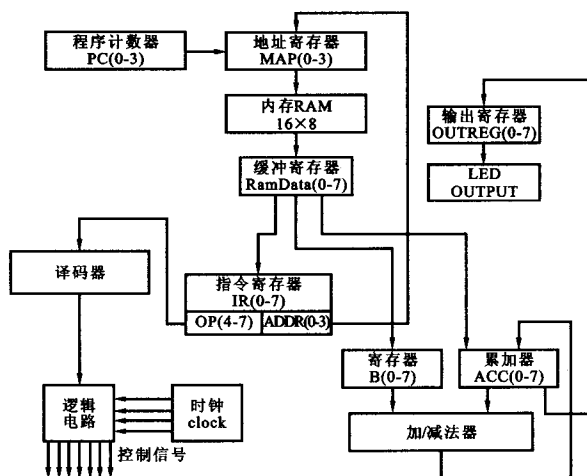


图4 组合逻辑控制的 SAP-CPU 体系结构

首先采用面向寄存器传输级的 VHDL 语言对 SAP-CPU 实例进行描述,具体设计文件请见文献[4]。然后借助 FPGA 硬仿真来验证设计文件的正确性。其中,在 SAP-CPU 运行的测试程序如表 1 所示;硬仿真选用的硬件平台是 Altera 的 Nios 开发板,软件平台是 Quartus 工具;在 Quartus 硬仿真测试床搭建参数设置为:①时钟信号采用开发板提供的晶振信号,频率为 33.3MHz;②复位信号由开发板提供的开关来控制,而输出信号 output 由开发板提供的数码管显示。③在 Quartus 中将管脚映射完毕之后,将综合产生的文件烧入至开发板上的 FP-

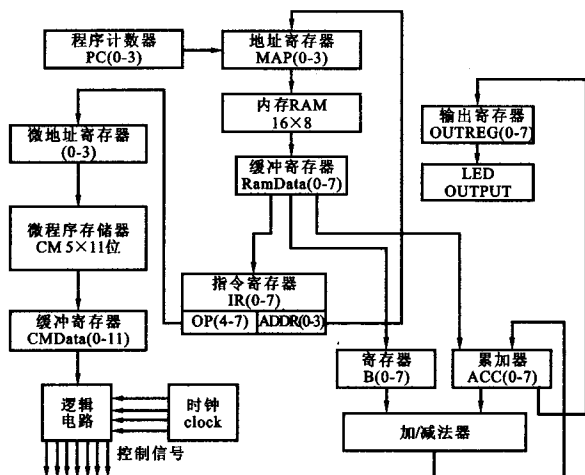


图5 微程序控制的 SAP-CPU 体系结构

GA 器件 APEX 20K 200E FC484-2X. ④对芯片复位之后, Nios 开发板进入工作状态, 从数码管读取的数据是“00011100”, 和测试程序的预期结果 1CH, 即 $10H + 14H + 18H - 20H$ 结果是相符的。

表1 SAP-CPU 测试程序

地址	汇编程序	说明
0H	LDA 9H	将 9H 内存的内容加载到累加器
1H	ADD AH	将 AH 内存的内容与累加器相加, 结果回送累加器
2H	ADD BH	将 BH 内存的内容与累加器相加, 结果回送累加器
3H	SUB CH	将 CH 内存的内容与累加器相减, 结果回送累加器
4H	OUT	将累加器内容输出到输出寄存器
5H	HLT	结束 CPU 执行
6H	XXX	
7H	XXX	
8H	XXX	
9H	10H	9H 内存单元内容是 10H
0AH	14H	0AH 内存单元内容是 14H
0BH	18H	0BH 内存单元内容是 18H
0CH	20H	0CH 内存单元内容是 20H

最后, 在通过 FPGA 硬仿真验证后, 将设计文件输入至 RVS 系统, 并施加如下外部激励信号: Time 为 0 时复位信号置“1”, 一直延续到 Time 为 20, 复位信号转变成“0”; 时钟信号每 25 个 Time 为一个周期, 其中“0”占据了 12 个 Time, “1”占据了 13 个 Time. 通过设置时间断点和被观察对象, 观察到测试程序的运行结果为 1CH, 与 FPGA 硬仿真结果及测试程序预期结果相符。

5 结束语

设计并实现了一种面向寄存器传输级的 VHDL 语言模拟系统 RVS, 定义了面向寄存器传输级的 VHDL 语言子集; RVS 编译器语法分析阶段采用了

一种基于递归的自顶向下语法分析算法, 增强了文法的可读性和可维护性; RVS 模拟器采用了一种具有调试功能的基于进程的事件驱动模拟调度算法, 提供时间断点、语句断点等调试功能. 实验数据表明 RVS 系统编译和模拟模块设计与实现的正确性. 今后进一步的研究: 在编译器和模拟器的实现过程中, 对中间代码标准化以提高 RVS 系统的可扩展性; 增加 RVS 编译器错误处理能力, 提高其健壮性; 在已经实现的基于进程的事件驱动模拟调度算法基础上, 通过离散事件的并行模拟算法研究, 将模拟计算量分散到并行机的多个节点上, 减少模拟时间, 提高 RVS 模拟速度。

参考文献:

- [1] IEEE Computer Society. IEEE Std 1076-1987: IEEE standard VHDL language reference manual [S]. New York: The Institute of Electrical And Electronics Engineers, 1994.
- [2] 张俊涛, 陈晓莉. 锁相环的 VHDL-AMS 建模与仿真分析 [J]. 微电子学与计算机, 2006, 23(2): 169-171.
- [3] 周喜明, 吴悦, 杨洪斌, 等. VHDL 逻辑级模拟系统中模拟模块的设计和实现 [J]. 微电子学与计算机, 2004, 21(3): 106-108.
- [4] 冷明. VHDL-R 语言的编译与模拟系统 [D]. 上海: 上海大学, 2003.
- [5] 吕映芝, 蒋维杜. 编译原理 [M]. 北京: 清华大学出版社, 2000.
- [6] 孙凌宇, 冷明. 一种基于递归算法的自顶向下语法分析方法 [J]. 井冈山师范学院学报: 自然科学, 2004, 25(6): 43-47.
- [7] 边计年, 卢峰, 郭芳. 适应调试功能的 VHDL 模型及模拟算法 [J]. 计算机学报, 1997, 20(11): 996-1002.
- [8] 吴清平, 刘明业. VHDL 编译型事件驱动模拟算法 [J]. 计算机学报, 2002, 25(1): 30-35.

作者简介:

孙凌宇 女, (1976-), 硕士, 副教授. 研究方向为 VLSI 算法设计等.

冷明 男, (1975-), 博士, 副教授. 研究方向为算法分析与设计等.

魏斯民 男, (1961-), 教授. 研究方向为 VLSI 算法设计等.

杨威 男, (1978-), 硕士, 讲师. 研究方向为算法分析与设计等.