Verilog 操作语义研究*

李勇坚[1], 何积丰[2], 孙永强[1]

[1](上海交通大学 计算机科学与工程系,上海 200030);
[2](澳门联合国大学 国际软件技术研究所,澳门)
E-mail: lyj238@ios.ac.cn; hjf@iist.unu.edu; sun-yq@cs.sju.edu.cn
http://www.sjtu.edu.cn

摘要: 提出了一个结构化操作语义模型,用于描述 Verilog 核心子集的语言特征,此子集包含了事件驱动、基于共享变量的并发特性、时间延迟等 Verilog 的主要语言成分.在此操作语义模型中,所有的 Verilog 程序将被统一地认为是开放式系统,所以在此操作语义模型的基础上能够进一步提出 Verilog 开放进程的观察模型,并提出基于互模拟的观察等价概念来判定进程之间的等价关系.最后证明了所定义的观察等价关系对所有的 Verilog 构造子而言是一个同余关系,从而为发展相应的进程代数理论提供了一个可靠性基础.

关 键 词: Verilog;事件调度;操作语义;观察模型;互模拟;同余性
中图法分类号: TP301 文献标识码: A

Verilog 是当前应用最为广泛的硬件设计语言之一,它可以用于硬件系统各种级别的设计、综合、仿真.为了保证设计的正确性,Verilog 语言应该具有一个形式化语义标准,以确保语言的设计者和使用者对语言有相同而准确的理解.Verilog 语言具有非常丰富的语言特征.它主要包括:(1) 事件驱动的计算过程.事件既可以是导线或寄存器值的变化,也可以是抽象事件.Verilog 程序的语义是基于事件的调度与传播的.(2) 基于共享变量的并发特性.Verilog 并发进程之间的相互影响可以被认为是通过共享变量来进行的.(3) 同时具有离散状态变化以及连续时间变化的混合特征.所以,要能够给出 Verilog 的形式化语义并不是很容易的.

Verilog 的语言 IEEE 标准使用的就是非形式化的仿真语义[1],为了使语言具有一个更加严格准确的标准,人们开始了 Verilog 的形式化语义工作.以下是对已有工作的简要总结:(1) 操作语义方法,最早开始 Verilog 形式化语义工作的是 Mike Gordon[2],他最早提出了一个标准算法来描述 Verilog 程序仿真的操作过程,对 Verilog 形式化语义的进一步工作提出了很好的建议.Xu 与 He 等人以上述算法为基础,给出了 Verilog 的一个子集的操作语义[3].在此操作语义模型中,他们把程序分为两个层次:顺序部件和并发程序.一个并发程序作为一个封闭系统,由若干顺序部件组成.虽然他们考虑了一个封闭并发系统内各种顺序部件的相互交互影响过程,但却忽略了并发系统间的相互影响.正是因为把并发程序作为封闭系统,所以一个封闭系统是不会与外界环境交互的,那么又怎么能够比较封闭系统之间的行为呢?所以他们无法以此操作语义为模型进一步提出互模拟(bisimulation)技术和相应的程序等价概念以及相应的进程代数理论等.(2) 指称语义方法,指称语义的方法主要集中在利用 Duration Calculus[4]来描述 Verilog 程序语义,其中心思想是把 Verilog 程序放在 DC 的语义模型中考虑,将 DC 语义模型的一个区间作为程序一次运行过程的指称,并利用 DC 公式来描述这个运行区间的各种时序特征.但是,作为一种硬件设计语言,Verilog 具有混合特征,即它同时包括连续时间区间的变化,也包括在同一时刻的零延迟

* 收稿日期: 2001-02-13; 修改日期: 2001-04-24
基金项目: 澳门联合国大学国际软件技术研究所“实时混成系统的研究技术”(Design Techniques for Real-Time Hybrid Systems)研究计划的资助项目.
作者简介: 李勇坚(1974－),男,山东青岛人,博士生,主要研究领域为程序语义,并行理论;何积丰(1946－),男,上海人,教授,博士生导师,主要研究领域为程序语义,并行计算;孙永强(1931－),男,浙江嘉善人,教授,博士生导师,主要研究领域为程序语言,函数式语言,并行计算.

状态迁移过程.目前,已有利用扩展 DC 来给出 Verilog 指称语义的工作[5~7],但是为了刻画零延迟状态迁移,所有以上工作使用的逻辑系统都要在原有 DC 基础上进行扩展,扩展部分很多涉及高阶成分(如对状态变量允许使用存在量词),这样使得该逻辑系统变得极为复杂,而且我们在证明程序的等价性以及提出相应的进程代数理论时,必须从最底层的逻辑公式出发,利用该逻辑系统的证明规则进行推导.由于逻辑系统的复杂性,使得程序代数定理的证明过程已经复杂到证明不下去了.可以说,以上的指称语义工作在进程代数推导工作中都没有取得实质性的进展.

利用指称语义方法给出 Verilog 的语义后,我们在以此为基础研究程序等价性以及有关进程代数时遇到了很大的困难.这些困难促使我们重新转向了操作语义方法.首先,我们希望所提出的操作语义模型能够很好地描述以上 Verilog 的语言特征,并且把 Verilog 的所有程序(包括顺序以及并发程序)作为开放系统,用统一的观点来看待所有程序与环境交互作用的方式以及进程内部各部件之间的交互方式;然后,我们希望在此操作语义模型的基础上提出相应的 Verilog 程序的实时观察模型,即确定一个 Verilog 程序的哪些行为是外部的或者是可观察的,然后在观察模型的基础上提出相应的互模拟概念,从而导出相应的观察等价关系.观察等价性(互模拟关系)描述了程序间的等价关系,证明两个程序等价是要构造这样一个等价关系,使得这个程序对在这个关系中.如果我们再能证明此等价关系的同余性,那么我们在此基础上就可以发展 Verilog 相应的进程代数,并且利用代数方法进行相应的验证和综合工作.

1 Verilog HDL 的语法

本文所研究的 Verilog 子集的语法如下:

$$P::=v=e|g|P;P|\ \text{if}\ b\ P\ \text{else}\ P|\ \text{while}\ b\ P|\ \text{always}\ P|\ \text{begin}\ P\ \text{end}\ |P||P,$$

其中 g 为用于事件调度的卫式控制语句,主要有延时卫式#n(n 为大于 0 的整数)和事件卫式@(η).

$$g::=@(\eta)|\#n\ \ \eta::=v|\text{posedge}\ v|\text{negedge}\ v|@(\eta_1\ \text{or}\ \eta_2\ldots\text{or}\ \eta_n).$$

为了代数推理的需要,我们对 Verilog 语言进行了相应的扩展.扩展成分主要包括:

(1) 原子进程 skip,chaos,stop;

(2) 终止程序ε;

(3) 布尔赋值卫式语句@(b&v=e);

(4) 与复合事件卫式语句@(η_1 and η_2…and η_n);

(5) 卫式选择语句 $G_1[]G_2[]\ldots[]G_n$,其中 $G_i=g$,或者 $G_i=g\rightarrow P$;

(6) IF 程序 IF($c_1P_1,c_2P_2,\ldots,c_nP_n$).

我们采用了非展平(unflat)式的语法形式,并发程序可以出现在任何嵌套层次上,而不像展平结构那样,并发程序只能出现在语法的最外层.这主要是为开放式系统语义而设计的.另外,除原子语句外,复合程序可分为两大类:并发复合与顺序复合程序.

我们定义 PTEXT 为所有符合推理语言语法的 Verilog 程序集合.

2 操作语义模型以及观察模型

2.1 事件调度语义与操作语义模型

本文的操作语义以文献[8]提出的仿真算法为基础,该算法是基于事件调度和传播的.在某一仿真时刻,如果系统中有若干进程处于激活状态(即该进程的头一条语句是能够立即执行的赋值、分支等顺序成分,这些成分的执行是零延迟,或者是瞬时的,即称为瞬时动作),则仿真调度程序非确定地选择一个进程执行,并且将当前的数据状态记录下来,作为一个新事件的起始状态,该进程执行这些瞬时成分直至遇到卫式语句为止.这时,该进程所执行的语句可能导致新的数据状态,这个由起始数据状态到当前数据状态的变化构成了一个新的事件,该事件的产生解放了那些等待此事件的若干进程.然后仿真进入下一循环周期.这个过程将进行至在此宏观时刻所有的事件被处理完毕为止.如果没有后继事件了,则仿真结束,否则系统仿真进入下一仿真时刻,即下一事件

的发生时刻.

首要的一个问题是,根据上述仿真语义,我们如何用一种统一的观点对 Verilog 进程建立起开放系统的概念.所谓开放式系统,指的是能够与外界环境发生交互的系统;而所谓的统一指的是所有 Verilog 程序都用相同的方式来观察与描述.在我们的操作语义模型中,不像文献[2]那样区分所谓的串行部件与并发程序,而是以统一的方式来看待进程内部各部件的组成与交互以及进程与环境之间的组成与交互.所以我们应该忘记所谓进程、环境、系统等的区别,在我们的理论中,它们统一地都被认为是进程,并且在一个统一的模型下被考察与描述.一个进程 P 可能由两个子进程(或称为线程)P_1,P_2 经过串行组合构成,同时也可能由它们并发组合起来构成;同时,进程 P 自身也能够与另外一个进程 Q 经过串行或者并发组合构成一个新的更高层次的系统.这样,我们就有可能在一个串行环境下来考察程序的语义,即考察此程序对其顺序后继程序的影响,或者作为一个顺序后继在其前驱施加了一定的作用后观察它会发生什么样的行为;另外,我们也可以在并发环境下来考察进程的行为,即考察该进程与环境在并发运行时,会对环境产生什么样的影响以及如何被环境影响.

事件是 Verilog 语义中最重要的概念:它用于描述数据变量的变化,而不是描述数据变量本身.值得说明的是事件产生(或者说是输出)过程,通常一个事件的产生并不仅仅由一条赋值语句执行后就能完成,而是由一个所谓的原子动作所包含的所有瞬时动作所积累的状态变化.该原子动作从一个进程被调度后开始,一直执行到阻塞为止.

事件产生过程是通过那些能执行瞬时操作的进程 P 从非调度状态转向调度状态而开始的(以后我们称之为 P 进入瞬时区),并且 P 确定是由自己的第 i 个线程去执行瞬时动作(假如进程的两个线程可以,则非确定地选择其中一个),这时,当前的每一个数据变量的值都被拷贝到相应的一个局部数据变量当中,在事件产生的过程中,所有的数据操作都在局部变量上进行;另外,真正执行瞬时操作的线程标号也应该被记录下来.

进程的状态至少应包括程序正文(program text)和当前数据状态分量(为当前变量名到其值的一个映射).另外,为了描述进程基于事件调度的语义,我们还可能需要引入以下状态分量:(1) 一个控制状态分量 i,表明程序是否已经被调度.它的值域为{0,1},若为 0,表明程序还未被调度;否则表明该程序已经被调度.(2) 记录事件产生过程中的局部的数据状态,若对那些无事件产生的状态而言此分量记为$\varnothing$.(3) 一个线程标号分量,表明到底是由被调度进程的哪一个线程去执行瞬时操作,它的值域为{1,2}.

一个进程所处的状态及其在该状态下所能进行的状态转移如下表示.

描述 **1**. 操作语义模型的非形式化描述

A. 进程处于非调度状态,并且此时也没有事件产生.我们称进程处于消极状态,此时进程的转移行为是:

(a) 如果该进程的某一个线程的头一条语句是赋值卫式语句,那么它可以执行一个原子动作,产生一个事件,若此时进程还未变成终止程序,那么它直接转向状态 D,否则转向状态 E.

(b) 如果它的某一线程能够执行瞬时操作,那么它可以进入瞬时区,转到状态 B.此时的每一个全局数据变量要被拷贝到一个局部变量中去,并且真正进入瞬时区的线程标号也应该被记录下来.

(c) 此进程等待其环境向它发出事件激励,导致其程序正文或全局数据状态发生相应变化,最后该进程仍处于消极状态.

(d) 进程本身及其环境都不能产生事件,时间将进行推移,最后该进程仍处于消极状态.

B. 进程的某一个线程已处于调度状态,并且该线程产生一个瞬时操作,导致相应的程序正文与局部数据状态发生变化.这时如果该线程还未终止,那么该进程转向状态 C;否则,如果进程由两个线程组成(即该进程是一并发程序),该进程将转入状态 D;否则该进程本身就是此线程(如该进程为一赋值语句,或者是循环语句等),那么它将转入状态 E.

C. 进程仍处于调度状态.

(a) 这或者是由于它的某一线程刚刚执行完一个瞬时动作所致.这时我们称该进程处于积极状态,并且该线程根据自己现在的程序性质决定该事件是否继续产生过程,如果该线程已经变成阻挡程序,那么它使整个进程离开调度状态并且转向状态;否则该线程将再次进入瞬时区以继续事件产生过程,并且转向状态 B.

(b) 或者是由于该进程的顺序前驱将控制权自然地传递给它,该进程同样根据自己是否是阻挡程序来决定

继续该事件的产生过程.

D. 进程已离开瞬时区,并且将已经产生的事件传播出去,从而转向状态 A.

E. 进程处于终止态(即程序正文变为ε),不会发生任何动作.

从上述操作语义模型中可以看出,事件的产生与结束和进程调度权的竞争、转让紧密地联系在一起.一个进程的调度状态由 0 变 1 意味着事件产生过程的开始,而由 1 变 0 意味着事件的产生过程的结束.

2.2 事件调度语义与操作语义模型,观察模型与观察等价

观察模型是对外界观察者而言所能观察到的进程行为模型,操作语义模型中进程的很多状态与状态迁移是内部的,并不能被外界观察者观察到.换句话来说,操作语义模型的动作太细了,我们需要定义更为抽象的所谓观察动作.程序等价性指的是在观察模型中程序之间可观察行为的等价性.观察模型中的两个基本问题是:(1) 我们从什么样的进程状态开始观察该进程;(2) 在这样的状态,该进程将发生什么样的行为(这种行为可能包括很多内部的状态转移),并最终转移到什么样的状态上去.观察模型定义如下:

描述 **2**. 观察模型

A. 我们可能从操作语义模型中的进程状态 1(消极状态)开始观察该进程,此时该进程未被调度,并且此时没有事件产生.在该状态下:

(a) 该进程发散,即它可能进行无限的瞬时动作以及自我触发动作,永远被调度.

下面的(b)~(f)是针对进程不发散的情况下的行为.

(b) 该进程能够进行瞬时操作,那么它可能被调度上,并且将进行产生事件的一个原子行为,然后将事件传播出去,并将处于一个新的消极状态.

(c) 该进程能够进行瞬时操作,那么它可能被调度上,并且进行一系列瞬时操作,但是当这些瞬时操作还未能产生事件时,这个进程就终止了,此时程序正文变为ε,调度状态仍保持为 1.

(d) 该进程能够进行瞬时操作,那么它可能被调度上,并且进行一系列瞬时操作,产生一个事件,但是还未将事件传播出去,这时进程就终止了,程序正文变为ε,调度状态变为 0.

(e) 该进程不能进行瞬时操作,或者它没被调度上,等待环境向它发出一个事件激励.

(f) 如果该进程不能进行瞬时操作,而且环境也未向其发出事件激励,那么时间将推进一个单位.

B. 我们可能从进程状态 3(积极状态)开始观察该进程,此时进程已经自然地从其前驱者获得被调度状态,并且已有一个事件的产生过程正在进行当中,在该状态下:

(a) 该进程发散,即它可能进行无限的瞬时动作以及自我触发动作,永远被调度.

下面的(b)~(f)是针对进程不发散的情况下的行为.

(b) 该进程能够进行瞬时操作,那么它将产生事件的过程继续进行下去,直至事件产生并且将事件传播出去,然后它将处于一个新的消极状态.

(c) 该进程能够进行瞬时操作,那么它将产生事件的过程继续进行下去,并进行一系列瞬时操作,但是当这些瞬时操作还未能产生事件时,这个进程就终止了,此时程序正文变为ε,调度状态仍保持为 1.

(d) 该进程能够进行瞬时操作,那么它将产生事件的过程继续进行下去,并且进行一系列瞬时操作,并且产生一个事件,但是还未将事件传播出去,这时进程就终止了,程序正文变为ε,调度状态变为 0.

(e) 阻挡程序终结事件的产生过程并且事件传播出去,然后它将处于一个新的消极状态.

进程等价指的是在观察模型中两者的等价,而观察等价又是通过互模拟的概念来定义的.所谓两个进程构成互模拟关系,不仅是指任何一方执行的可观察行为另一方也能执行,而且要求双方的子孙(即执行完一个或多个动作蜕变后的程序)也是如此.可以说在思想上我们借鉴了 CCS(calculus of communication system)[9]所提出的互模拟思想.但在互模拟的具体定义上,我们与 CCS 有很大的不同.首先,定义两个进程在给定的一个初始观察状态下是互模拟的,如果这两个进程在所有的初始观察状态下都互模拟,那么我们就称这两个进程是互模拟的或者观察等价的.另外,在定义两个进程在某一个初始观察状态下构成互模拟关系时,实际上又要分成两种情况处理:先是比较两者的发散性质是否相同,然后比较两者都不发散的情况下两者的行为,这与 CCS 也是不同的.

3 状态转移系统

在给出上述 Verilog 子集的形式化语义之前,我们需要引入一些辅助定义.

定义 **1**(数据状态集合,线程标记集合).

(1) 令 Var 为所有变量名集合,$BOOL=\{0,1\}$,则数据状态集合 $DSTAS$ 定义为 $Var\to BOOL$.

(2) 线程标记集合 $THREADS=\{1,2\}$.

类似 Plotokin 所采用的结构化操作语义方法[10],我们给出上述 Verilog 子集的操作语义.该操作语义模型通过一个标签转移系统(labelled transition system)$\langle S,T,\to\rangle$给出,其中

- S 为状态集合,并且 $S=S_1\cup S_2,S_1=PTEXT\times DSTAS\times(DSTAS\cup\varnothing)\times\{0\},S_2=PTEXT\times DSTAS\times DSTAS\times\{1\}\times THREADS$;
- T 为标签集合,并且 $T=\{\langle\sigma,\sigma'\rangle|\sigma,\sigma'\in DSTAS\}\cup N$,其中 N 代表自然数集;
- $\to$为转移规则集合,并且$\to\subseteq(S\times S)\cup(S\times T\times S)$.

通过这些转移规则能够将进程状态的变化一步步地产生出来.

与我们在描述 1 中对操作语义模型的非形式化描述相对应,该转移系统中相应的状态种类大致如下:

- $\langle P,\sigma,\varnothing,0\rangle$:表示该进程处于消极状态,此时它既未被调度上,也没有事件产生.P 表示程序当前的正文,σ表示进程当前所处的全局数据状态,而第 3 分量为∅表示事件还未开始产生.
- $\langle P,\sigma,\sigma',1,i\rangle$:表示该进程的第 i 个线程已经进入瞬时区,σ用于记录事件产生过程开始时的数据状态,而 σ'表示的是进程产生事件过程所用的局部数据状态,i 用于记录该线程标号.
- $\langle P,\sigma,\sigma',1\rangle$:表示该进程自然地从其前驱获得调度权,该进程将决定继续或者终结已经发生的事件产生过程.
- $\langle P,\sigma,\sigma',0\rangle$:表示该进程已离开瞬时区,并准备传播事件$\langle\sigma,\sigma'\rangle$.

相应的状态转移规则如下:

- 产生事件过程的各类瞬时操作

T1. 在初始状态为消极状态的情况下,进程 P 的第 i 个线程被调度,即该线程进入瞬时区,并准备执行相应的瞬时操作.那么状态元组的第 3 分量由∅变为当前数据状态,并且此分量用于表示事件产生过程中的局部数据状态,而状态元组的第 2 分量用于保存事件产生过程开始时的全局数据状态.

$$\langle P,\sigma,\varnothing,0\rangle\to\langle P,\sigma,\sigma,1,i\rangle.$$

T2. 在初始状态为积极状态的情况下,进程 P 不是阻挡程序,所以它将继续事件的产生过程.

$$\langle P,\sigma,\sigma,1\rangle\to\langle P,\sigma,\sigma',1,i\rangle.$$

T3. 在瞬时区内,一个并发程序的第 i 个线程执行完相应的操作,进程程序正文从 P 变为 P',数据状态也从σ变为σ',并且该线程以及整个进程仍然保持被调度状态,我们这时仍然要用元组的第 5 分量来记住到底是进程的哪一个线程保持在瞬时区内.

$$\langle P,\sigma_0,\sigma,1,i\rangle\to\langle P',\sigma_0,\sigma',1,i\rangle.$$

T4. 在瞬时区内,一个进程执行完相应的操作,程序正文从 P 变为 P',并且数据状态从σ变为σ',进程 P'仍然保持被调度状态,但是还未决定是否将事件产生过程进行下去.

$$\langle P,\sigma_0,\sigma,1,i\rangle\to\langle P',\sigma_0,\sigma',1\rangle.$$

T5. 在瞬时区内,并发程序 P 的第 i 个线程终止或者该线程的程序正文变成阻挡程序,从而迫使该程序离开瞬时区.

$$\langle P,\sigma_0,\sigma',1,i\rangle\to\langle P',\sigma_0,\sigma',0\rangle.$$

T6. 在瞬时区内,进程 P 的程序正文是阻挡程序,从而迫使自身离开瞬时区.

$$\langle P,\sigma_0,\sigma',1\rangle\to\langle P,\sigma_0,\sigma',0\rangle.$$

T7. 布尔赋值卫式语句在一个消极状态进行一个原子动作,产生一个事件,然后又直接转入非调度态.

$$\langle P,\sigma,\varnothing,0\rangle\to\langle P',\sigma,\sigma',0\rangle.$$

• 事件卫式语句对应的状态转移

T8. 该进程传播其顺序前驱已经产生的事件,此事件可能导致该进程的程序正文发生变化,事件在被传播后就不存在了,同时我们将σ'拷贝至元组的第 2 分量,而将元组第 3 分量清空.

$$\langle P,\sigma,\sigma',0\rangle \xrightarrow{\langle\sigma,\sigma'\rangle} \langle P',\sigma',\varnothing,0\rangle.$$

T9. 进程并发的环境产生一个事件来影响该进程.

$$\langle P,\sigma,\varnothing,0\rangle \xrightarrow{\langle\sigma,\sigma'\rangle} \langle P',\sigma',\varnothing,0\rangle.$$

• 在消极状态下进程让时间推移,同时数据状态保持不变

T10. $$\langle P,\sigma,\varnothing,0\rangle \xrightarrow{t} \langle P',\sigma,\varnothing,0\rangle.$$

受篇幅所限,在此我们仅给出赋值、顺序、并发语句的规则,为清楚起见,我们使用了标号“Ti”,并给出了此规则所属的种类,其中 Ti 正是上述对各类规则的讨论中所用的标号.注意,对于一种程序而言,它并不是所有种类的动作都能执行的.

(1) 赋值的转移规则

T1. $\langle v=e,\sigma,\varnothing,0\rangle \to \langle v=e,\sigma,\sigma,1,1\rangle$.

T2. $\langle v=e,\sigma,\sigma',1\rangle \to \langle v=e,\sigma,\sigma',1,1\rangle$.

T3. $\langle v=e,\sigma_0,\sigma,1,1\rangle \to \langle \varepsilon,\sigma_0,\sigma(v\mapsto e(\sigma)),1\rangle$.

T8. $\langle v=e,\sigma,\sigma',0\rangle \xrightarrow{\langle\sigma,\sigma'\rangle} \langle v=e,\sigma',\varnothing,0\rangle$.

T9. $\langle v=e,\sigma,\varnothing,0\rangle \xrightarrow{\langle\sigma,\sigma'\rangle} \langle v=e,\sigma',\varnothing,0\rangle$.

(2) 顺序语句的转移规则

T1. $$\frac{\langle S_1,\sigma,\varnothing,0\rangle \to \langle S_1,\sigma,\sigma,1,i\rangle}{\langle S_1;S_2,\sigma,\varnothing,0\rangle \to \langle S_1;S_2,\sigma,\sigma,1,i\rangle, i\in\{1,2\}}.$$

T2. $$\frac{\langle S_1,\sigma_0,\sigma,1\rangle \to \langle S_1,\sigma_0,\sigma',1,i\rangle}{\langle S_1;S_2,\sigma_0,\sigma,1,i\rangle \to \langle S_1;S_2,\sigma_0,\sigma',1,i\rangle, i\in\{1,2\}}.$$

T3. $$\frac{\langle S_1,\sigma_0,\sigma,1,i\rangle \to \langle S_1',\sigma_0,\sigma',1,i\rangle}{\langle S_1;S_2,\sigma_0,\sigma,1,i\rangle \to \langle S_1';S_2,\sigma_0,\sigma',1,i\rangle, i\in\{1,2\}}.$$

T4. $$\frac{\langle S_1,\sigma_0,\sigma,1,i\rangle \to \langle S_1',\sigma_0,\sigma',1\rangle}{\langle S_1;S_2,\sigma_0,\sigma,1,i\rangle \to \langle seq(S_1',S_2),\sigma_0,\sigma',1\rangle, i\in\{1,2\}}.$$

T5. $$\frac{\langle S_1,\sigma_0,\sigma,1,i\rangle \to \langle S_1',\sigma_0,\sigma',0\rangle}{\langle S_1;S_2,\sigma_0,\sigma,1,i\rangle \to \langle seq(S_1',S_2),\sigma_0,\sigma',0\rangle, i\in\{1,2\}}.$$

T6. $$\frac{\langle S_1,\sigma,\sigma',1\rangle \to \langle S_1,\sigma,\sigma',0\rangle}{\langle S_1;S_2,\sigma,\sigma',1,i\rangle \to \langle S_1;S_2,\sigma,\sigma',0\rangle, i\in\{1,2\}}.$$

T7. $$\frac{\langle S_1,\sigma,\varnothing,0\rangle \to \langle S_1',\sigma,\sigma',0\rangle}{\langle S_1;S_2,\sigma,\varnothing,0\rangle \to \langle seq(S_1',S_2),\sigma,\sigma',0\rangle}.$$

T8. $$\frac{\langle S_1,\sigma,\sigma',0\rangle \xrightarrow{\langle\sigma,\sigma'\rangle} \langle S_1',\sigma',\varnothing,0\rangle}{\langle S_1;S_2,\sigma,\sigma',0\rangle \xrightarrow{\langle\sigma,\sigma'\rangle} \langle seq(S_1',S_2),\sigma',\varnothing,0\rangle}.$$

T9. $$\frac{\langle S_1,\sigma,\varnothing,0\rangle \xrightarrow{\langle\sigma,\sigma'\rangle} \langle S_1',\sigma',\varnothing,0\rangle}{\langle S_1;S_2,\sigma,\varnothing,0\rangle \xrightarrow{\langle\sigma,\sigma'\rangle} \langle seq(S_1',S_2),\sigma',\varnothing,0\rangle}.$$

T10. $$\frac{\langle S_1,\sigma,\varnothing,0\rangle \xrightarrow{t} \langle S_1',\sigma',\varnothing,0\rangle}{\langle S_1;S_2,\sigma,\varnothing,0\rangle \xrightarrow{t} \langle seq(S_1',S_2),\sigma',\varnothing,0\rangle}.$$

(3) 并发程序的转移规则

T1. $$\frac{\langle S_i,\sigma,\varnothing,0\rangle \to \langle S_i,\sigma,\sigma,1,j\rangle}{\langle S_1 \parallel S_2,\sigma,\varnothing,0\rangle \to \langle S_1 \parallel S_2,\sigma,\sigma,1,i\rangle, i,j\in\{1,2\}}.$$

T3. $$\frac{\langle S_i,\sigma,\sigma',1,j\rangle \to \langle S_i',\sigma,\sigma'',1\rangle, \langle S_i',\sigma,\sigma'',1\rangle \to \langle S_i'',\sigma,\sigma'',1,i\rangle}{\langle S_1 \parallel S_2,\sigma,\sigma',1,i\rangle \to \langle S_i'' \parallel S_{3-i},\sigma,\sigma'',1,i\rangle, i,j\in\{1,2\}},$$

$$\frac{\langle S_i,\sigma,\sigma',1,j\rangle\to\langle S_i',\sigma,\sigma'',1,j\rangle, S_i'\neq\varepsilon}{\langle S_1\parallel S_2,\sigma,\sigma',1,i\rangle\to\langle S_i''\parallel S_{3-i},\sigma,\sigma'',1,i\rangle, i,j\in\{1,2\}}.$$

T5. $$\frac{\langle S_i,\sigma,\sigma',1,j\rangle\to\langle S_i',\sigma,\sigma'',1\rangle,\langle S_i',\sigma,\sigma'',1\rangle\to\langle S_i'',\sigma,\sigma'',0\rangle}{\langle S_1\parallel S_2,\sigma,\sigma',1,i\rangle\to\langle S_i''\parallel S_{3-i},\sigma,\sigma'',1,i\rangle, i,j\in\{1,2\}},$$

$$\frac{\langle S_i,\sigma,\sigma',1,j\rangle\to\langle S_i',\sigma,\sigma'',0\rangle}{\langle S_1\parallel S_2,\sigma,\sigma',1,i\rangle\to\langle par(S_i'',S_{3-i}),\sigma,\sigma'',0\rangle, i,j\in\{1,2\}},$$

$$\frac{\langle S_i,\sigma,\sigma',1,j\rangle\to\langle\varepsilon,\sigma,\sigma'',1(0)\rangle}{\langle S_1\parallel S_2,\sigma,\sigma',1,i\rangle\to\langle S_{3-i},\sigma,\sigma'',0\rangle, i,j\in\{1,2\}}.$$

T6. $\langle S_1\parallel S_2,\sigma,\sigma',1\rangle\to\langle S_1\parallel S_2,\sigma,\sigma',0\rangle$.

T7. $$\frac{\langle S_i,\sigma,\varnothing,0\rangle\to\langle S_i',\sigma,\sigma'',0\rangle}{\langle S_1\parallel S_2,\sigma,\varnothing,0\rangle\to\langle par(S_i',S_{3-i}),\sigma,\sigma',0\rangle}.$$

T8. $$\frac{\langle S_i,\sigma,\sigma',0\rangle\xrightarrow{\langle\sigma,\sigma'\rangle}\langle S_i',\sigma,\varnothing,0\rangle, i\in\{1,2\}}{\langle S_1\parallel S_2,\sigma,\sigma',0\rangle\to\langle par(S_1',S_2'),\sigma,\varnothing,0\rangle}.$$

T9. $$\frac{\langle S_i,\sigma,\varnothing,0\rangle\xrightarrow{\langle\sigma,\sigma'\rangle}\langle S_i',\sigma,\varnothing,0\rangle, i\in\{1,2\}}{\langle S_1\parallel S_2,\sigma,\varnothing,0\rangle\xrightarrow{\langle\sigma,\sigma'\rangle}\langle par(S_1',S_2'),\sigma',\varnothing,0\rangle}.$$

T10. $$\frac{\langle S_i,\sigma,\varnothing,0\rangle\xrightarrow{t}\langle S_i',\sigma,\varnothing,0\rangle, i\in\{1,2\}}{\langle S_1\parallel S_2,\sigma,\varnothing,0\rangle\xrightarrow{t}\langle par(S_1',S_2'),\sigma',\varnothing,0\rangle}.$$

在上述转移规则中,为方便起见,我们分别定义简写方式 *seq*,*Par*:

$$seq(P,Q)=_{\mathrm{df}}\begin{cases}P;Q, P,Q\neq\varepsilon\\ P,Q=\varepsilon\\ Q,P=\varepsilon\end{cases},\quad Par(P,Q)=_{\mathrm{df}}\begin{cases}P\parallel Q, P,Q\neq\varepsilon\\ P\parallel\varepsilon, Q=\varepsilon\\ \varepsilon\parallel Q, P=\varepsilon\end{cases}.$$

4 互模拟与观察等价

首先给出前面观察模型中所谓的可观察行为的形式化定义,然后再给出互模拟与观察等价的严格定义.我们引入 $\xrightarrow{a}$ 来表示一个进程从开始观察状态起直至遇到阻挡程序为止的一连串瞬时动作.

定义 **2**(连续瞬时动作). 若 P 是一个 Verilog 程序,

(1) 如果状态元组$\langle P,\sigma,\varnothing,0\rangle$,$\langle P',\sigma,\sigma',0\rangle$满足下列关系,那么它们将被记为$\langle P,\sigma,\varnothing,0\rangle\xrightarrow{a}\langle P',\sigma,\sigma',0\rangle$,它表示从消极状态$\langle P,\sigma,\varnothing,0\rangle$起将会发生一系列瞬时动作,直至$\langle P',\sigma,\sigma',0\rangle$为止.该条件是:$\langle P,\sigma,\varnothing,0\rangle\to\langle P',\sigma,\sigma',0\rangle$或者存在一个有限状态序列$\{c_i|0\le i\le n,n>1\}$满足条件:(a) $c_0=\langle P,\sigma,\varnothing,0\rangle$,$c_0\to c_1$;(b) 若 $1\le i<n$,那么 $c_i\to c_{i+1}$,$\pi_2(c_i)=\sigma,\pi_4(c_i)=1$;(c) $c_n=\langle P',\sigma,\sigma',0\rangle$.

(2) 如果状态元组$\langle P,\omega,\sigma,1\rangle$,$\langle P',\omega,\sigma',0\rangle$之间满足下列条件,那么将被记为$\langle P,\omega,\sigma,1\rangle\xrightarrow{a}\langle P',\omega,\sigma',0\rangle$,它表示从积极状态$\langle P,\omega,\sigma,1\rangle$起,$P$ 将会发生一系列瞬时动作,直至$\langle P',\omega,\sigma',0\rangle$为止.该条件是,存在这样一个有限状态序列$\{c_i|0\le i\le n\}$满足:(1) $c_0=\langle P,\omega,\sigma,1\rangle$;(2) $c_i\to c_{i+1}$,$\pi_2(c_i)=\omega,\pi_4(c_i)=1$,其中 $0\le i\le n$.(3) $c_n=\langle P',\omega,\sigma',0\rangle$.其中,$\pi_j(c_i)$指的是状态元组 c_i 的第 j 个元素.

然后我们引进$\Rightarrow$来定义观察模型中的事件产生动作和终止动作(即观察模型中两种状态下的(b)(c)(d)).

定义 **3** (事件输出与终止动作). 若 P 是一个 Verilog 程序,

(1) 若状态元组$\langle P,\sigma,\varnothing,0\rangle$(或$\langle P',\sigma,\sigma',0\rangle$)与状态元组$\langle P,\sigma'',\varnothing,0\rangle$之间满足下列条件将被记为$\langle P,\sigma,\varnothing,0\rangle(\langle P,\sigma,\sigma',1\rangle)\Rightarrow\langle P,\sigma'',\varnothing,0\rangle$,它表示从$\langle P,\sigma,\varnothing,0\rangle(\langle P,\sigma,\sigma',1\rangle)$状态起,直至状态元组$\langle P',\sigma'',\varnothing,0\rangle$的一个事件产生及传播动作.该条件是:(a) $\langle P,\sigma,\varnothing,0\rangle(\langle P,\sigma,\sigma',1\rangle)\xrightarrow{a}\langle P_0,\sigma,\sigma'',0\rangle$(这表示事件产生过程);(b) $\langle P_0,\sigma,\sigma'',0\rangle\xrightarrow{\langle\sigma,\sigma'\rangle}\langle P',\sigma'',\varnothing,0\rangle$(这表示事件传播过程).

(2) 若状态元组$\langle P,\sigma,\varnothing,0\rangle$(或$\langle P,\sigma,\sigma',1\rangle$)与状态元组$\langle\varepsilon,\sigma,\sigma'',0\rangle$之间满足下列条件将被记为$\langle P,\sigma,\varnothing,0\rangle(\langle P,\sigma,\sigma',1\rangle)\Rightarrow\langle\varepsilon,\sigma,\sigma'',0\rangle$,它表示从$\langle P,\sigma,\varnothing,0\rangle$或$(\langle P,\sigma,\sigma',1\rangle)$状态起,直至状态元组$\langle\varepsilon,\sigma,\sigma'',0\rangle$的一个事件产

生但还未传播就终止的动作,该条件是: $\langle P,\sigma,\varnothing,0\rangle(\langle P,\sigma,\sigma',1\rangle) \xrightarrow{a} \langle \varepsilon,\sigma,\sigma'',0\rangle$.

(3) 若状态元组$\langle P,\sigma,\varnothing,0\rangle$(或$\langle P,\sigma,\sigma',1\rangle$)与状态元组$\langle \varepsilon,\sigma,\sigma'',1\rangle$之间满足下列条件将被记为: $\langle P,\sigma,\varnothing,0\rangle(\langle P,\sigma,\sigma',1\rangle)\Rightarrow\langle \varepsilon,\sigma,\sigma'',1\rangle$,它表示从$\langle P,\sigma,\varnothing,0\rangle$(或$\langle P,\sigma,\sigma',1\rangle$)状态起,直至状态元组$\langle \varepsilon,\sigma,\sigma'',1\rangle$的一个还未产生事件就终止的动作,该条件是,存在一个有限状态序列$\{c_i|0\le i\le n\}$满足:(a) $c_0=\langle P,\sigma,\varnothing,0\rangle(\langle P,\sigma,\sigma',1\rangle)$, $c_0\to c_1$; (b) $c_i\to c_{i+1}$, $\pi_1(c_i)\ne\varepsilon$, $\pi_2(c_i)=\sigma$, $\pi_4(c_i)=1$, $1\le i<n$; (c) $c_n=\langle \varepsilon,\sigma,\sigma'',1\rangle$.

现在我们来定义所谓的状态元组发散动作.

定义 **4**(发散). 状态元组$\langle P,\sigma,\sigma',1\rangle(\langle P,\sigma,\varnothing,0\rangle)$被称为发散的,如果存在这样一组无限序列$\{c_i|i\in N\}$满足:

(1) $c_0=\langle P,\sigma,\sigma',1\rangle(\langle P,\sigma,\varnothing,0\rangle)$;

(2) $c_i\to c_{i+1}$;

(3) 或者 $c_i=\langle P_i,\sigma_i,\sigma_i',0\rangle$, $\sigma_i'\ne\varnothing$, $c_i \xrightarrow{\langle\sigma,\sigma'\rangle} c_{i+1}$.若状态元组$\langle P,\sigma,\sigma',1\rangle$ $(\langle P,\sigma,\varnothing,0\rangle)$是发散的,则将被记为$\uparrow\langle P,\sigma,\sigma',1\rangle(\uparrow\langle P,\sigma,\varnothing,0\rangle)$.

特别应该指出的是,一个进程在发散时,只有内部瞬时动作还有自我触发(即前驱产生的事件在触发其后继)动作,因为在自我触发后,后继程序若能执行瞬时动作,那么它还可能被调度.即我们采用的是可能发散就是发散的观点.

定义 **5**(单位时间延迟动作).

若状态元组$\langle P,\sigma,\varnothing,0\rangle$与$\langle P',\sigma,\varnothing,0\rangle$满足下列条件,那么将被记为$\langle P,\sigma,\varnothing,0\rangle \xrightarrow{\#1} \langle P',\sigma,\varnothing,0\rangle$,它表示从状态$\langle P,\sigma,\varnothing,0\rangle$到$\langle P',\sigma,\varnothing,0\rangle$的一个可观察的单位时间推移动作.该条件为:存在一个有限状态序列$\{\langle P_i,\sigma,\varnothing,0\rangle|0\le i\le n, n>1\}$满足:(1) $P_0=P$;(2) $\langle P_i,\sigma,\varnothing,0\rangle\Rightarrow\langle P_{i+1},\sigma,\varnothing,0\rangle$,其中 $0\le i\le n$;(3) $\langle P_n,\sigma,\varnothing,0\rangle \xrightarrow{1} \langle P',\sigma,\varnothing,0\rangle$.

现在我们来定义本文最重要的一个概念:互模拟.它用于判断两个 Verilog 进程是否观察等价.前已提及,我们可能在两种起始状态下考察进程的行为,所以首先定义在两类起始状态下,两个进程互模拟的概念.

定义 **6**(状态互模拟).

若令 $ActCS=\{\langle P,\sigma,\sigma',1\rangle|P\in PTEXT,\sigma,\sigma'\in DSTAS\}$, $PasCS=\{\langle P,\sigma,\varnothing,0\rangle|P\in PTEXT,\sigma\in DSTAS\}$, 二元关系$\Re\subseteq(ActCS\times ActCS)\cup(PasCS\times PasCS)$若满足以下条件,它将被称为是一个互模拟关系.这些条件是

(1) 它是对称的.

(2) 在以下讨论中,我们假设 c_1,c_2 为两消极(积极)状态元组,而且$\pi_1(c_1)=P$, $\pi_1(c_2)=Q$.

$$c_1\Re c_2\Rightarrow\uparrow c_1\equiv\uparrow c_2\wedge\forall i\in\{2,3,4\}[\pi_i(c_1)=\pi_i(c_2)].$$

(3) 若 c_1,c_2 都不发散,并且 $c_1\Re c_2$,

(3a) 若 $c_1\Rightarrow\langle P',\sigma',\varnothing,0\rangle$,那么存在 Q'使得 $c_2(\Rightarrow\cup Id)\langle Q',\sigma',\varnothing,0\rangle$,与$\langle P',\sigma',\varnothing,0\rangle\Re\langle Q',\sigma',\varnothing,0\rangle$(其中 Id 表幂等关系).

(3b) 若 $c_1\Rightarrow\langle \varepsilon,\sigma,\sigma',0\rangle$,那么 $c_2\Rightarrow\langle \varepsilon,\sigma,\sigma',0\rangle$.

(3c) 若 $c_1\Rightarrow\langle \varepsilon,\sigma,\sigma',1\rangle$,那么 $c_2\Rightarrow\langle \varepsilon,\sigma,\sigma',1\rangle$.

(3d) 若 $c_1 \xrightarrow{\langle\sigma,\sigma'\rangle} \langle P',\sigma',\varnothing,0\rangle$,那么存在 Q'满足 $c_2 \xrightarrow{\langle\sigma,\sigma'\rangle} \langle Q',\sigma',\varnothing,0\rangle$与$\langle P',\sigma',\varnothing,0\rangle\Re\langle Q',\sigma',\varnothing,0\rangle$,其中 $c_1,c_2\subseteq PasCS$.

(3e) 若 $c_1 \xrightarrow{\#1} \langle P',\sigma,\varnothing,0\rangle$,那么存在 Q'满足 $c_1 \xrightarrow{\#1} \langle Q',\sigma,\varnothing,0\rangle$,与$\langle P',\sigma,\varnothing,0\rangle\Re\langle Q',\sigma,\varnothing,0\rangle$,其中 $c_1,c_2\subseteq PasCS$.

条件(2)说的是两个状态元组的发散性质相同,而条件(3a)是说 P 与 Q 在相应的状态下都能完成一个事件的输出与传播动作,然后两者完成该动作蜕变后的程序在对应的状态又互模拟.条件(3b)是说 P 与 Q 在相应的状态下都能执行一个原子动作产生一个相同的事件,然后两者都终止了.条件(3c)则是说两者在相应的状态下都能导致相同的状态迁移,但还未构成事件之前两者就都终止了.(3d)是说 P 与 Q 在消极状态下从其并行环境中接受一个事件后,它们蜕变后的程序在对应的状态仍然互模拟.(3e)是说 P 与 Q 在消极状态下进行一个单位时间推移动作之后它们蜕变后的程序在对应的状态仍然互模拟.

定义 **7**(观察等价).

(1) 若存在一个互模拟关系$\Re$使得 $c_1\Re c_2$,那么我们称状态元组 c_1,c_2 是互模拟的,并记为 $c_1\approx c_2$.

(2) 对于两个 Verilog 程序 P,Q,若对任意的数据状态 $\sigma,\sigma'\in DSTAS$. 有 $\langle P,\sigma,\varnothing,0\rangle\approx\langle Q,\sigma,\varnothing,0\rangle$ 与 $\langle P,\sigma,\sigma',1\rangle\approx\langle Q,\sigma,\sigma',1\rangle$,那么我们称 P 与 Q 是互模拟或观察等价的,并记为 $P\approx Q$.

显而易见,互模拟关系是等价关系,但我们同时要求它是同余关系.所谓同余关系就是指该关系在推理语言所有的复合构造子下封闭,正如以下定理所述:

定理 **1**(同余性). 若 $P\approx Q$,那么对所有的程序复合构造子 F 有,$F(P)\approx F(Q)$.

定理 1 是本文最重要的结果之一,它保证了我们所定义的观察等价关系能够在所有的上下文关系中得到保持,从而使得代数的替换规则能够成立,这样我们推导出的等式就能够真正地构成一个代数系统.

本文最大的进展在于互模拟(观察等价)关系以及观察等价关系满足同余性等结果.我们所提出的互模拟思想不仅在理论上为 Verilog 程序的等价性概念提供了一个比较完美的框架,而且也在实践上为证明程序等价提供了一种有效的技术.同余性的证明以及我们所举的几个例子说明了这种证明技术的可行性,证明过程非常直观、简洁、明了.这与在扩展的 DC 语义框架下的繁琐的逻辑推导构成鲜明的对比.另外,观察等价关系是同余关系也是一个及其重要的结果,它保证了观察等价关系在代数的替换规则下能够得到保持,即 $P\approx Q$ 就意味着 $F(P)\approx F(Q)$,这样我们推导出的等式就能够真正构成一个代数系统,或者说我们的操作语义模型能够为以后的代数语义提供一个具体的模型(把代数语义的等式关系映射到操作语义观察模型中的互模拟关系).

最后我们提一下 Verilog所具有的其他并发语言所没有的特征.首先,如果我们把事件产生动作看成是输出,而把事件触发看成输入的话,Verilog 的输入输出是不对称的,即产生事件的动作根本不需要输入方的同意,但反过来,输入方必须得到事件信号后方能得到解放.这与 CCS 或 CSP(communicating sequential processes)[11]的管道通信不同,管道通信的输出动作同样也要得到输入方的同意,两者必须经过严格的同步方能通信.另外,Verilog 的通信方式属于一对多的广播式通信方式,在某一时刻,只能有一个进程产生信号,而此信号将对系统内各进程的所有线程都将产生作用.而 CCS 或 CSP 中通信方式是严格的一对一的,在某一时刻,只能有两个进程(或者更严格地说,是两个进程的两个线程)进行通信.另外,Verilog 语言还有事件自我触发,即一个进程产生事件后直接触发其顺序后继进程,为了清晰地严格地将以上特征表述出来,我们的办法是利用进程代数,这也是我们反复强调进程代数的一个重要原因.

实际上,Verilog 的进程代数有很丰富的内容,特别是那些反映 Verilog 特性的一些规则在一般并发理论的进程代数中是没有的.我们将在相应的后继文章中专门进行讨论.

References:

[1] IEEE Computer Society. IEEE Standard Based on the Verilog Hardware Description Language (IEEE std 1364-1995). 1995.

[2] Gordon, M. The semantic challenge of Verilog HDL. In: Proceedings of the 10th Annual IEEE Symposium on Logic in Computer Science. San Diego: IEEE Computer Society Press, 1995. 136~145.

[3] He, Ji-feng, Xu, Qi-wen. An operational semantics of a simulator algorithm. In: Proceedings of the 2000 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2000). Las Vegas: CSREA Press, 2000. 203~209.

[4] Zhou, Chao-chen, Hoare, C.A.R., Ravn, A.P. A calculus of duration. Information Processing Letter, 1991,40(5):269~275.

[5] Pace, G.J. Hardware design based on Verilog HDL [Ph.D. Thesis]. Oxford University, 1997.

[6] Schneider, G., Xu, Qi-wen. Towards a formal semantics of Verilog using duration calculus. LNCS 1486, Berlin: Spinger-Verlag, 1998.

[7] Zhu, Hui-biao, He, Ji-feng. A DC-based semantics for Verilog. In: Feng, Yu-lin, Notkin, D., Gaudel, M-C., eds. Proceedings of the 16th IFIP World Computer Congress 2000: Theory and Practice (ICS 2000). Beijing: Publishing House of Electronics Industry, 2000. 421~432.

[8] Gordon, M. Event and cycle semantics of hardware description languages. VFE Project Report, Cambridge University, 1998.

[9] Miler, R. Communication and Concurrency. Prentice Hall, 1989.

[10] Plotkin, G. A structural approach to operational semantics. Technical Report, DAIMI FN-19, Department of Computer Science, Aarhus University.

[11] Hoare, C.A.R. Communicating Sequential Processes. Prentice Hall, 1985.

Study on the Operational Semantics of Verilog*

LI Yong-jian[1], HE Ji-feng[2], SUN Yong-qiang[1]

[1](*Department of Computer Science and Engineering, Shanghai Jiaotong University, Shanghai* 200030, *China*);

[2](*International Institute for Software Technology, United Nations University, Macau, China*)

E-mail: lyj238@ios.ac.cn; hjf@iist.unu.edu; sun-yq@cs.sju.edu.cn

http://www.sjtu.edu.cn

Abstract: In this paper, a structural operational semantic model is presented for a core subset of Verilog, and the subset has the main features of Verilog such as event-driven computation, shared-variable concurrency, time-delay, and so on. And all the Verilog processes are seen as open systems in this operational semantic model, so a model of observation is provided for open Verilog processes, and use observation equivalence based on bisimulation to identify the equivalence between programs. The observation equivalence can be proved to be a congruence for all Verilog operators, so it provides a sound base for deriving the algebraic laws for Verilog processes.

Key words: Verilog; event scheduling; operational semantics; observation model; bisimulation; congruence

* Received February 13, 2001; accepted April 24, 2001

Supported by the Project DTfRTS (Design Techniques for Real-Time Hybrid Systems) of the International Institute for Software Technology, United Nations University

****

第 3 届中国 Rough 集与软计算学术研讨会(CRSSC 2003)

征 文 通 知

由中国计算机学会人工智能与模式识别专业委员会、中国人工智能学会 Rough 集专业委员会筹备组主办，重庆邮电学院承办的第 3 届中国 Rough 集与软计算学术研讨会（CRSSC 2003）于 2003 年 5 月 26 日~29 日在重庆举行。

征文范围

Rough 集理论及应用,计算智能,机器学习,文字计算,Fuzzy 集理论及应用,粒度计算,软计算及其应用,进化计算,Petri 网,软计算的逻辑基础,非经典逻辑,神经网络,软计算复杂性,空间推理,统计推理,智能 Agent,多标准决策分析,决策支持系统,知识发现与数据挖掘,多 Agent 技术,网络智能,集成智能系统,近似推理与不确定推理,数据仓库,模式识别与图像处理,其他有关领域。

征文要求

论文必须未公开发表过，一般不超过 6000 字;论文包括题目、作者姓名、单位、地址、邮编、Email 地址、联系电话、中英文摘要（一般不超过 200 字）、关键词、正文和参考文献;论文一律为 A4 打印稿，一式两份，用 Word 排版，欢迎用电子版投稿（Email 递交）和通过会议网站投稿。

征文请寄：重庆邮电学院计算机科学与技术研究所 王国胤 收（邮编 400065）

电子版投稿请送：wanggy@cqupt.edu.cn

网站在线投稿：http://210.77.169.211/Information for Chinese.htm

关键日期

截稿日期：2002 年 12 月 31 日（收到日期）

录用通知：2003 年 1 月 31 日（发出日期）

清样付印日期：2003 年 2 月 28 日（收到日期）

联系方式

王国胤 023-62460066(O)，023-62460159(H)，13508331272(手机),传真：023-62461882