Verilog 语言形式化语义研究*

李勇坚[1,2], 孙永强[1], 何积丰[2]

[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 国际软件研究所,澳门)

E-mail: lyj238@yahoo.com

http://www.cs.sjtu.edu.cn/chinese/member/index.htm

摘要: 在连续离散混合时间模型中考虑 Verilog 的语义行为,将混合模型中的一个区间作为 Verilog 程序一次运行过程的指称.提出了一种扩展的 ITL 来描述这种混合区间,从而给出 Verilog 的形式语义.这种语义定义不仅考虑到了各种语言成分的最终执行结果,而且能够很好地给出语言成分执行的时序特征.

关键词: 混合系统;Verilog;区间时态逻辑;离散事件调度;交叠式并发语义

中图法分类号: TP312 **文献标识码**: A

Verilog HDL 是目前应用最为广泛的一种硬件描述语言,可用于数字电子系统设计.设计者可用它进行各种级别的逻辑设计,并可进行数字逻辑系统的仿真、验证与逻辑综合.但是,基于 C 风格的 Verilog 语言及其另一种使用也极为广泛,与基于 Ada 风格的硬件描述语言 VHDL 相比,Verilog 的语义形式化工作显然做得很不够.这就使得语言的使用者、设计者对语言产生了不同的理解,给语言的实施与使用带来了一定的困难.

Verilog 语言成分可以很整洁地分为两大类.(1) 用于算法、行为级描述的顺序过程式语言成分.主要包含变量赋值、顺序、分支、循环等.这些系统高级层次的设计成分的语义主要是基于离散状态的,它们主要考虑的是系统状态在运行前后的变化关系,而没有具体考虑到程序执行的时间因素.(2) 用于硬件、实时特征的语言成分.主要包含卫式、并行、信号同步、连续赋值等.这些成分的语义主要是基于连续时间的.这两种不同的成分体现了 Verilog 的混合系统特征,即系统的变化必须由连续区间与离散区间混合描述[1,2].这种混合特征在语言的基于离散事件调度的仿真过程中体现得非常明显.在某一个宏观时刻,某些进程的敏感信号发生变化、产生事件,触发这些进程的若干顺序语句执行,直至进程结束或被阻塞.当某个进程在这个时刻被触发后,很可能执行多条语句,这些语句在宏观上是不占任何时间的,但在微观上,它要产生一系列离散状态的迁移(这也是所谓的零延时状态迁移),这些状态迁移可以用一系列离散变化区间来描述.在这个宏观时刻的所有事件被处理完毕之后,若没有后继事件,则仿真结束.否则,系统仿真将进入下一事件发生时刻,即宏观时刻必须向前推移一个连续区间.这种面向连续时间的变化特征必须通过连续区间来刻画.

经典 ITL 完全是面向离散状态变化的[3],它不能描述系统在连续时间的变化.经典 Duration Calulus 完全是面向连续时间域的[4],它不能很好地描述系统的离散状态变化,最明显的特点就是

* 收稿日期: 1999-10-10; 修改日期: 2000-05-26

基金项目: 澳门联合国大学国际软件研究所资助项目

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;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;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.

不能描述上述的零延时状态迁移.所以,我们需要一种新的逻辑,它能够同时描述系统的离散状态变化和连续时间变化.在这方面典型的工作有文献[5],它描述了一种具有时间标签的转移系统,这种时间模型能够很好地刻画操作系统的调度等问题.Gerardo Schneider 等人以这种时间模型作为语义模型,并且在经典 DC 的基础上扩充了不动点公式等,从而形成了相应的 μWDC,利用 μWDC 给出 Verilog 的一个子集的语义[6].

1 扩展的 ITL

综上所述,我们所要描述的混合系统同时包含基离散区间与连续区间的变化.这样的系统在一个二维时间域内运行.该时间域的一个点为一个二元序对(r,n).其中 n 表示系统变化过程中的离散状态标号,可以称之为微观时间;r 表示实际时间,也可以称之为宏观时间,表示系统具有该离散状态的实际时间.图 1 直观地显示了这种时间模型.定义 1.1 给出了这个二元时间域的形式化定义.

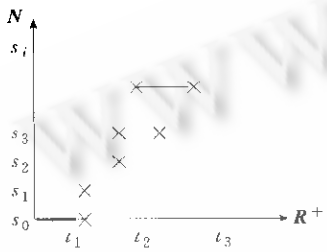

Fig. 1 The time model of a hybrid system
图1 混合系统的时间模型

定义 1.1. 一个二元时间域是这样一个全序$\langle T,<\rangle$,它满足:

(1) $T\subseteq(\mathscr{R}^+\cup\{\infty\})\times(\mathscr{N}\cup\{\infty\})$,并且$(0,0)\in T$;

(2) $(r_1,n_1)<(r_2,n_2)$当且仅当$(r_1\neq\infty\wedge r_1\leqslant r_2\wedge n_1<n_2)\vee(r_1<r_2\wedge n_1\neq\infty\wedge n_1\leqslant n_2)$;

(3) 对于任何 $t\in T$,存在 $t'\in T$,使得 $t<t'$并且 $t'\not<t$;

(4) $\lim_{i\to\infty}((r_i,n_i))\in T$.

其中 $\mathscr{R}^+$ 表示非负实数集,$\mathscr{N}$ 表示自然数集.

上述中前两个条件定义了混合时间集的序关系.第 3 个条件表示该全序是最大的,添加任意一个不属于 T 的序对将导致 T 不是全序,这说明了两点:(1) $(r,\infty),(\infty,n),(\infty,\infty)$中三者必只有一个在 T 中.(2) 若$(r,n)\in T,(r',n)\in T$,并且 $r<r'$,则任取 $r<r_1<r'$,$(r_1,n)\in T$.这一点说明连续时间的迁移将不会导致系统变量的任何变化,这与 Verilog 的仿真语义是一致的.第 4 个条件表示,若有一个无穷序列$(r,n_1),(r,n_2),\ldots,(r,n_i),\ldots$,则$(r,\infty)$必在此序列中.对任意 $t=(r,n)$,$\pi_1(t)=_{\mathrm{def}}r,\pi_2(t)=_{\mathrm{def}}n$.$t=\infty$当且仅当 $r=\infty\vee n=\infty$.一个区间为一个 T 时间对$[b,e]$,$b\neq\infty$并且 $e\leqslant\infty$,若 $m\neq\infty$,$[b,m]^\frown[m,e]$记为区间$[b,m]$与$[m,e]$的连接.

扩展的 ITL 包含以下符号:

全局变量:$GVars=\{x,y,z,\ldots\}$.状态变量:$SVars=\{u,v,\ldots\}$;

区间变量:$TVars\cup Pletters$,其中 $TVars=\{l,k,\ldots\}$,$Pletters=\{X,Y,\ldots\}$;

项的构成语法为 $t::=x\mid k\mid\overleftarrow{v}\mid\overrightarrow{v}\mid f(t_1,t_2,\ldots,t_n)$.$x$ 为全局变量,k 为区间变量,v 为状态变量,f 表示的是 n 元的函数,$t_1,t_2,\ldots,t_n$ 为项.

公式的语法为 $X\mid p(t_1,t_2,\ldots,t_n)\mid\neg\varnothing\mid\varnothing_1\wedge\varnothing_2\mid\varnothing_1;\varnothing_2\mid\exists\, v.\varnothing\mid\exists\, x.\varnothing\mid\mu X.\varnothing$.$x$ 是一个全局变量,v 是状态变量,X 属于 $Pletters$,其中在最小不动点公式中,X 必须出现在偶数个否定符号的后面,以保证该公式是合法的,而且根据公式的构成可知,扩展的 ITL 在下面定义 1.2 所述的模型中是单调的,这样就保证了不动点公式定义的正确性.

定义 1.2. 一个模型 M 是一个五元组$(\langle T,<\rangle,I,J,V,[b,e])$.

$(T,<)$是一个在定义 1.1 中所述的时间域;

$[b,e]$为 T 上的一段区间;

$I:SVars\to T\to Values$. I 为状态变量的解释. $Values=\boldsymbol{R}^+\cup\boldsymbol{X}\cup Bool$;

$J:Pletters\cup Tvars\to Intv\to Values$,为区间变量的解释. $Intv$ 为所有的区间集合. 其中 $J:Pletters\to Intv\to Bool$; $J:Tvars\to Intv\to\boldsymbol{R}^+\cup\boldsymbol{X}$;

$V:GVars\to Values$,为全局变量的解释.

项的解释为 $x(T,I,J,V,[b,e])=V(x)$.

$k(T,I,J,V,[b,e])=\pi_2(e)-\pi_2(b)$,表示区间的微观时间分量的长度;

$l(T,I,J,V,[b,e])==\pi_1(e)-\pi_1(b)$,表示区间的宏观时间分量的长度;

$\overleftarrow{v}(T,I,J,V,[b,e])=I(v)(b)$; $\overleftarrow{v}$ 表示状态变量 v 在区间$[b,e]$的开始时候的值;

$\overrightarrow{v}(T,I,J,V,[b,e])=I(v)(e)$; $\overrightarrow{v}$ 表示状态变量 v 在区间$[b,e]$的结束时候的值.

$f(t_1,t_2,\ldots,t_n)(T,I,J,V,[b,e])-f_*(t_1(T,I,J,V,[b,e]),t_2(T,I,J,V,[b,e]),\ldots,t_n(T,I,J,V,[b,e]))$.

f^*表示模型对函数 f 的解释.

为了通过 Tarski 定理来定义最小不动点公式,我们在完全格$(2^{Intv},\subseteq)$上定义公式的语义. 给定状态变量的解释 I,区间命题变量的解释 J,区间变量的解释 T,全局变量的解释 V,定义一个从全体扩展 ITL 公式集到 2^{Intv}的函数 F,$F(\varnothing)$记为 $\|\varnothing\|_{J,V}^{T,I}$.

$$\|X\|_{J,V}^{T,I}=_{\text{def}}V(X),$$

$$\|p(t_1,t_2,\ldots,t_n)\|_{J,V}^{T,I}=_{\text{def}}\{[b,e]\mid\{b,e\}\subset T\wedge p^*(t_1(T,J,V,I,[b,e]),\ldots,t_n(T,I,J,V,[b,e]))\},$$

p^*表示模型对谓词 p 的解释.

$$\|\neg\varnothing\|_{J,V}^{T,I}=_{\text{def}}2^{Intv}-\|\varnothing\|_{J,V}^{T,I}\quad\|\varnothing_1\wedge\varnothing_2\|_{J,V}^{T,I}=_{\text{def}}\|\varnothing_1\|_{J,V}^{T,I}\cap\|\varnothing_2\|_{J,V}^{T,I},$$

$$\|\exists\ v.\varnothing\|_{J,V}^{T,I}=_{\text{def}}\bigcup_{a\in Values}\|\varnothing\|_{J,V}^{T,I(v\mapsto a)},\text{若 } v \text{ 为一个状态变量},$$

$$\|\exists\ x.\varnothing\|_{J,V}^{T,I}=_{\text{def}}\bigcup_{a\in Values}\|\varnothing\|_{J,V(x\mapsto a)}^{T,I},\text{若 } x \text{ 为一个全局变量},$$

$$\|\varnothing_1;\varnothing_2\|_{J,V}^{T,I}=_{\text{def}}\{[b,e]\mid\exists\ m.\{b,e,m\}\subset T\wedge b\leqslant m\leqslant e\wedge(m\neq\infty\wedge[b,m]\in\|\varnothing_1\|_{J,V}^{T,I}\wedge[m,e]\in\|\varnothing_2\|_{J,V}^{T,I})\vee(e=\infty\wedge[b,e]\in|\varnothing_1\|_{J,V}^{T,I})\},$$

$$\|\mu X.\varnothing\|_{J,V}^{T,I}-_{\text{def}}\bigcap\{A\mid\|\varnothing\|_{J(X\mapsto A),V}^{T,I}\subseteq A\}.$$

$J(X\mapsto A)$指的是这样一个区间变量的解释:该解释除了在 X 上与 J 可能不同以外,在其他区间变量上与 J 均一致. $V(x\mapsto a)$,$I(v\mapsto a)$的定义类似.

最大不动点公式的定义可以通过最小不动点公式来定义:$\nu X.\varnothing=_{\text{def}}\neg\mu Y.\neg\varnothing[\neg Y/X]$.

公式的可满足性与有效性定义如下:

$(T,I,J,V,[b,e])\models\varnothing$当且仅当$[b,e]\in\|\varnothing\|_{J,V}^{T,I}$,$\models\varnothing$当且仅当$\|\varnothing\|_{J,V}^{T,I}=2^{Intv}$(对任意 T, I,J,V).

从上面分析可知,一个混合区间是混合系统一次运行过程的刻画. 其中 $\overleftarrow{v},\overrightarrow{v}$ 分别用于刻画一个区间开始和结束时状态变量 v 的值,所以 $\overleftarrow{v},\overrightarrow{v}$ 描述了系统一次运行前后状态变量 v 的变化情况(假如从此运行终止)[7];并且由于该区间离散,连续混合性质同时存在,所以在扩展的 ITL 中需要同时引进 k,l 两个区间变量,同时刻画区间的微观和宏观时间长度. 另外,混合系统往往是实时的交互系统,这样的系统运行区间是无限的,所以必须在扩展的 ITL 中考虑到允许区间无穷这个因素

对公式定义的影响,如在定义顺序操作符";"时,定义的第2部分正是考虑到当$[b,e]=\infty \wedge [b,e] \in \| \varnothing_1 \|_{I,V}^{T,J}$这种情况.另外,由于要刻画递归及循环,因此要在扩展的ITL中引进最小、最大不动点公式的定义.扩展的ITL实质上是文献[6]中μWDC的一种简化形式,与μWDC最大的不同之处在于,在引入存在算子时,允许对状态变量进行限制,之所以要引入这一点,我们在后文中将加以阐述.

2 扩展ITL的一些性质

为了便于以后的说明,我们先定义一些派生的公式.

$\Diamond A =_{def} \text{true}; A; \text{true}$ $[b,e] \models \Diamond A$ 表示 A 在$[b,e]$的某段子区间为真.

$A =_{def} \neg\Diamond(\neg A)$,$[b,e] \models A$ 表示 A 在$[b,e]$的所有子区间恒真.

$\text{fin} -_{def} \exists\, x(l<x \wedge k<x)$,$x$ 为一个全局变量.

$\text{inf} -_{def} \neg\text{fin}$ $\text{point} -_{def} l=0 \wedge k=0$, $\text{dint} =_{def} l=0 \wedge k=1$, $\text{cint} =_{def} l>0 \wedge k=0$.

fin,inf,point 分别表示如下性质:区间的宏观时间与微观时间皆有限、区间无限、区间为一点.dint,cint 分别表示一段离散和一段连续区间.

令 e 为有关状态变量的Verilog程序表达式,$\vec{e}(\overset{\leftarrow}{e})$为一个EX_ITL表达式,它将 e 中所有的状态变量 v 改为 $\vec{v}(\overset{\leftarrow}{v})$.

$$v=_{e} e =_{def} \text{dint} \wedge (\vec{v}=\vec{e}), \quad stb(e) =_{def} (\overset{\leftarrow}{e}=\vec{e}),$$

$$stb(Vars) =_{def} ((\overset{\leftarrow}{v_1}=\overset{\rightarrow}{v_1} \wedge (v_2=\overset{\rightarrow}{v_2} \ldots \wedge (\overset{\leftarrow}{v_i}) \ldots \wedge \overset{\leftarrow}{v_n}=\overset{\rightarrow}{v_n}),$$

其中 $Vars=\{v_1,v_2,\ldots,v_n\}$.

$[b,e] \models stb(e)$表示 e 的值在区间$[b,e]$上稳定,$stb(Vars)$与 $stb(e)$的含义类似.

关于最小及最大不动点公式,有以下两条定理,这两条定理对理解用不动点公式来刻画循环语句的语义很有帮助.在以下两条定理的证明中,关键在于最小、最大不动点公式的计算上,其中 $\| \mu X.(\varnothing_1;X) \vee \varnothing_2 \|_{I,V}^{T,J} = \bigcup \{W_n \mid n \geqslant 0\}$.当 $W_0=\varnothing$时,

$$W_{n+1} - \| (\varnothing_1;X) \vee \varnothing_2 \|_{J[X \mapsto W_n],V}^{T,I}; \quad \| \nu X.(\varnothing_1;X) \vee \varnothing_2 \|_{I,V}^{T,J} = \bigcap \{W_n \mid n \geqslant 0\},$$

当 $W_0=2^{\text{int}V}$时,

$$W_{n+1} = \| (\varnothing_1;X) \vee \varnothing_2 \|_{J[X \mapsto W_n],V}^{T,I}.$$

定理1. $\varnothing_1,\varnothing_2$ 为两个扩展ITL的公式.如果$\varnothing_1,\varnothing_2$ 不包含状态命题变量 X 的自由出现,则对任意 $T,I,J,V,[b,e] \in \| \mu X.(\varnothing_1;X) \vee \varnothing_2 \|_{I,V}^{T,J}$当且仅当存在一个有限自然数 n(可能为0)与一个非递减的时间序列 $b_0,b_1,\ldots,b_n$,该序列满足:(1) $\forall\, b_i, b_i \in T-\{\infty\}$,其中 $b_0=b$;(2) $\forall\, i$, $[b_i,b_{i+1}] \in \| \varnothing_1 \|_{I,V}^{T,J}$;(3) $[b_n,e] \in \| \varnothing_1 \|_{I,V}^{T,J}$,或者 $e=\infty \wedge [b_n,e] \in \| \varnothing_1 \|_{I,V}^{T,J}$.

定理1表明,$[b,e]$若满足 $\mu X.(\varnothing_1;X) \vee \varnothing_2$,则要么在有限次迭代后,$\varnothing_2$ 被满足,要么在若干有限次迭代后,$\varnothing_1$ 公式的计算进入无限.

定理2. $\varnothing_1,\varnothing_1$ 的定义与定理1类似,则对任意 T,I,J,V, $\| \nu X.(\varnothing_1;X) \vee \varnothing_2 \|_{I,V}^{T,J} = \| \mu X.(\varnothing_1;X) \vee \varnothing_2 \|_{I,V}^{T,J} \cup E$,其中$[b,e] \in E$ 当且仅当存在一个无限非递减的时间序列 $b_0,b_1,\ldots,b_n$.该序列满足:$\forall\, i, b_i \leqslant b_{i+1}$, $\forall\, i, [b_i,b_{i+1}] \in \| \varnothing_1 \|_{I,V}^{T,J}$.

定理2表明,$[b,e]$若满足 $\nu X.(\varnothing_1;X) \vee \varnothing_2$ 形式的公式,则或者$[b,e]$满足 $\mu X.(\varnothing_1;X) \vee \varnothing_2$,或者$[b,e] \in E$.后者的特征表明,$[b,e]$由无限个满足$\varnothing_1$ 的区间构成.

3 Verilog 的形式化语义

我们将考虑 Verilog 的一个子集 V,V 已经包含了 Verilog 的核心部分,包含了上述的两大特征:用于算法、行为级的过程式语言成分;用于硬件、实时特征的语言成分.V 也包含了文献[6]所提出的 Verilog 子集 V^-,而且允许并发操作符定义在任何层次上,而不是像文献[6]所规定的那样,必须在最外层次才能定义并发操作符.另外,对文献[6]没有涉及的带惰性延迟的赋值语句及非阻塞赋值等都给出了定义.

3.1 V 的语法

为了描述方便,V 的语法与 Verilog 的语法有一些细微的差别.

⟨number⟩,⟨variable⟩,⟨exp⟩⟨bexp⟩为语法终结符.

⟨var-list⟩::= ⟨variable⟩|⟨variable⟩,⟨var-list⟩

⟨exp-list⟩::=⟨exp-list⟩|⟨exp⟩,⟨exp-list⟩

⟨guard⟩::= #⟨number⟩ |@⟨variable⟩ |@postdge⟨variable⟩
|@negedge ⟨variable⟩ |wait⟨variable⟩ |⟨guard⟩or⟨guard⟩

⟨statement⟩::=skip |⟨var-list⟩=⟨exp-list⟩|⟨guard⟩
|⟨var-list⟩=⟨guard⟩⟨exp-list⟩ ⟨guard⟩⟨var-list⟩=⟨exp-list⟩
|variable⟨=⟨guard⟩⟨exp⟩ if⟨bexp⟩then⟨statement⟩else⟨statement⟩
|while⟨bexp⟩do⟨statement⟩ forever⟨statement⟩
|begin⟨stmt-list⟩end fork ⟨statement⟩ || ⟨stament⟩join

⟨stmt-list⟩::=⟨statement⟩ ⟨statement⟩;⟨stmt-list⟩

⟨module⟩::=intial⟨stmt-list⟩ |always ⟨stmt-list⟩
|assign ⟨variable⟩=⟨exp⟩ |assign #⟨number⟩⟨variable⟩=⟨exp⟩
|⟨moudule⟩ || ⟨module⟩

从顶层上来讲,一个 Verilog 程序由若干并发进程构成,这些进程通过共享变量进行通信.当一个进程的敏感信号产生变化,产生一个新的事件时,将触发进程的执行,进程的执行将导致状态的迁移,并且将产生新的事件,并传播下去,程序将一直执行下去,直至没有后继事件发生.所以,Verilog 程序的语义可以通过这样一个区间集 $runs(runs\subseteq 2^{Intv})$,对于每一个$[b,e]\in runs$,$[b,e]$反映的是电路受到触发后状态迁移的情况.对于$[b,e]$的一个子区间$[b_i,b_{i+1}]$,其中若$[b_i,b_{i+1}]$为一个离散区间,则$[b_i,b_{i+1}]$刻画了系统在一个宏观时刻的状态迁移,若$[b_i,b_{i+1}]$为一个连续区间,则$[b_i,b_{i+1}]$刻画了系统正在被阻塞的情况,这是由于执行了延时等卫式语句;若$[b,e]$有限,表明电路能够在有限的时间内达到稳定,否则电路将是振荡的.这样一个扩展的 ITL 公式能很方便地刻画一个 Verilog 程序的语义.以上是利用扩展 ITL 来刻画 Verilog 程序的基本思想.任何 Verilog 程序都满足以下公理:

AX1:$\forall v\in Vars.(\text{cint}\rightarrow\vec{v}=\overleftarrow{v})$.$Vars$ 表示一个 Verilog 程序所使用的状态变量集合.

该公式表明,系统的状态变化只发生在离散区间,而不可能发生在连续区间.

另外,由于我们在此所定义的是 Verilog 基于软件仿真的语义,所以我们采用交叠式并发语义,即在同一宏观时刻,若有多个进程同时处于运行态,则按某种顺序将这些进程依次一一调度并执行,某个进程被调度后将一直执行到被阻塞为止,所以当进程之间存在共享输出时,调度次序的

不同将引起不确定性.交叠式语义的特征体现在:每一个离散区间的状态迁移仅仅由一个进程(被调度的那个进程)的当前执行语句引起,整个系统在某一宏观时刻的离散状态变化区间等于构成此系统的所有子进程在这个时刻离散状态变化区间按照调度次序顺序交叠组合起来.为了刻画这种交叠调度的语义,我们为每一个进程赋予了一个状态变量 *scheduled*,该变量在一个离散区间的初始值为真表明该进程在该离散区间被调度.这种定义方式克服了文献[6]中对并行操作子定义的局限性,文献[6]实际上只允许定义顶层并行操作子,因为它要显式地为每一个顶层进程赋予一个表明进程的标号,系统有一个标号变量 $e.\delta$.在某一个离散区间,$e.\delta=i$,则表明在此离散区间上,标号为 i 的进程执行了一条语句.这种定义方式实际上限制了每一个进程只能是一个顺序进程,在这个进程中不能再包含互相并行的子进程.而对 Verilog 程序而言,一个进程可以由若干并行的子进程构成,而且对于如下的合法 Verilog 程序,文献[6]不可能给出定义:begin $s_1;s_2;\ldots;s_n$ end,其中 s_2 语句为 fork $P \| Q$ join 或者 s_2 为诸如 $v<=e$ 的形式.对于顶层用于仿真的进程,我们要求其在每个离散区间都被调度,即满足以下公理:

AX2:$(\text{dint}\rightarrow\overleftarrow{scheduled})$.

(2) 语句的形式化语义

(I) 原子语句

延时卫式:

$$M(\#n)=idle\wedge l\leqslant n\wedge(\text{fin}\rightarrow l=n),$$

其中 $idle=(\text{dint}\rightarrow\neg\overleftarrow{scheduled})$,一个延时为 n 的卫式语句并不一定能等到宏观时间前进 n 个单位,因为其他进程有可能在此时刻前陷入了某一个微观时间的死循环.

事件卫式:

$$@(v)=_{\text{def}}\vec{v}\neq\overleftarrow{v}\quad @(\text{postedge } v)=_{\text{def}}\vec{v}>\overleftarrow{v}\quad @(\text{negedge } v)=_{\text{def}}\vec{v}<\overleftarrow{v}.$$

$$@(\text{event}_1 \text{ or } \ldots \text{event}_n)=_{\text{def}}@(\text{event}_1)\vee\ldots\vee@(\text{event}_n)$$

$$M(@\text{event})=_{\text{def}}(idle\wedge((\neg@(\text{event}));(\text{dint}\wedge@\text{event}))\ M(\text{Wait } v)$$

$$=_{\text{def}}idle\wedge((\neg\vec{v});(\text{dint}\wedge\vec{v})).$$

空语句:

$$M(skip)=_{\text{def}}(idle;(\text{dint}\wedge stb(Vars)\wedge\overleftarrow{scheduled}))\wedge l=0.$$

过程赋值:

$$M(v=e)=_{\text{def}}(idle;(v=o\ e\wedge\overleftarrow{scheduled}\wedge stb(Vars-\{v\})))\wedge l=0.$$

(II) 复合语句

$M(v=g\ e)=_{\text{def}}\exists\ u.\ (\vec{u}=\overleftarrow{e};(M(g)\wedge stb(u));M(v=u))$,$u$ 为一个状态变量.

$$M(gv=e)=_{\text{def}}M(g);M(v=e).$$

顺序组合:

$$M(\text{begin } s_1;s_2;\ldots;s_n \text{ end})=_{\text{def}}M(s_1);M(\text{begin } s_2;\ldots;s_n \text{ end}).$$

条件语句:

$$M(\text{if } (eb)s_1 \text{ else } s_2)=_{\text{def}}((\overleftarrow{ed}\wedge M(s_1))\vee(\neg\overleftarrow{ed}\wedge M(s_2)).$$

循环语句:

$$M(\text{while } (eb)S)=_{\text{def}}\nu X.((\overleftarrow{ed};M(s);X)\vee\neg\overleftarrow{ed}).$$

$$\text{forever } P\equiv\text{while(true)}P.$$

语义定义的难点在于并行组合符的定义.为了体现交叠式并发语义,我们引入了两个状态变量 s_1 和 s_2,分别表示进程 P_1 和 P_2 的调度情况.

$$M(P_1 \parallel P_2) =_{\text{def}} \exists\ s_1, s_2 \cdot ((\text{dint} \rightarrow (\overleftarrow{scheduled} \leftrightarrow ((\overleftarrow{s_1} \vee \overleftarrow{s_2}) \wedge \overleftarrow{s_1} \neq \overleftarrow{s_2})))$$
$$\wedge (M(P_1)[scheduled/s_1]; idle_1) \wedge M(P_2)[scheduled/s_2]$$
$$\vee M(P_1)[scheduled/s_1] \wedge M(P_2)[scheduled/s_2]; idle_2)),$$

其中 $s_1, s_2, scheduled$ 都为状态变量.$idle_1 =_{\text{def}} (\text{dint} \rightarrow \neg \overleftarrow{s_1})$,$idle_2 =_{\text{def}} (\text{dint} \rightarrow \neg \overleftarrow{s_2})$.

从上述定义可以看出,当 P_1, P_2 有共享输出变量,并在同一宏观时刻写共享变量时,不同的调度次序则产生不同的结果.这即为竞争冒险,在实际设计中一定要避免这种情况.

$$M(\text{fork } P \parallel Q \text{ join}) = M(P \parallel Q),$$
$$M(\text{intial } P) \equiv M(P); \text{idle},$$
$$\text{always } P = \text{intial } (\text{forever } P).$$
$$M(\text{assign } v = e) \equiv \text{always@}(v_1 \text{or } v_2 \ldots \text{or } v_n)(v = e),$$

其中 $v_1, v_2, \ldots, v_n$ 为表达式 e 中所使用的状态变量.若要想定义带延迟的连续赋值语句,必须先引进一个表达惰性延迟的赋值公式:$v \# n - e$.

$$v \# n = e =_{\text{def}} \nu X.((stb(e) \wedge l = n \wedge M(v = \# ne)) \vee ((l < n \wedge stb(e) \wedge idle); X)).$$

其中 $stb(e) =_{\text{def}} stb(e); (\text{dint} \wedge \overleftarrow{e} \neq \overleftarrow{e})$.注意,$v \# n = e$ 与 $v = \# ne$ 的写法上的区别.

在惰性延迟模型中,若右边表达式 e 的值在某时刻 t 发生了变化,则在 $t+n$ 时刻,左边 wire 信号的值可能会出现两种情况:(1) 若在 t 至 $t+n$ 时刻之间,表达式 e 的值保持稳定,则 v 的值在 $t+n$ 时刻变为表达式 e 在 t 时刻的值,这是公式第 1 部分表示的含义.(2) 若在 t 至 $t+n$ 时刻之间 t',表达式 e 的值发生变化,则 v 的值在 $t+n$ 时刻保持不变,即取消 $t+n$ 时刻对 v 赋值事件的调度,并重新安排 $t'+n$ 时刻对 v 赋值的事件,这是公式第 2 部分表示的含义.

$$M(\text{assign } \#n\ v = e) =_{\text{def}} ((l < n \wedge \neg \overleftarrow{v}) \vee ((l = n \wedge \neg \overleftarrow{v}); \text{true}))$$
$$\wedge M(\text{always}(@(v_1, v_2, \ldots, v_n) v\ \#n = e)).$$

其中 $v_1, v_2, \ldots, v_n$ 为表达式 e 中所使用的状态变量,等式第 1 部分表示的是初始的 n 时间单位信号 v 的情况,第 2 部分表示在 v_i 变化后执行的惯性延迟赋值语句.

最难定义的是非阻塞式赋值,非阻塞赋值的本质是并行,即该非阻塞式赋值语句的执行与其后面的语句并行执行.即该特点可以有 $v <= e; p$ 实质上是 $v = e \parallel P$.但与一般的并行不同的是,表达式 e 的值并不立即赋给 v,而是要经过一个所谓的无穷小延迟 δ.这样,在进程使用的 V 的初始值必然为 V 发生 e 赋值之前的值.上述关于非阻塞赋值的无穷小延迟的性质是在 IEEE 标准之中有定义的,不过,非阻塞赋值引起的非确定因素太多,例如,如果进程 P 中又有关于 v 的非阻塞赋值.下面的例子很好地说明了这个问题.$v <= 1; v <= 2$ 执行后的结果肯定是不确定的,IEEE 标准中也没有确定这样的程序的执行结果.所以,在具体的实现中,各个商业产品对非阻塞赋值的实现并不相同,为了简化起见,我们对非阻塞赋值的使用进行一定的限制.该定义是

$$M(v <= ge; P) =_{\text{def}} \exists\ v_p (M(v = g\ e \parallel P[v/v_p]) \wedge \overleftarrow{v} = \overleftarrow{v_p}),$$

其中,要求 P 中没有对 v 再进行写的语句,v_p 是一个状态变量.

4 结论以及将来的工作

综上所述,我们的主要研究思想是:在连续离散混合时间模型中考虑 Verilog 的语义行为,将

混合模型中的一个区间作为系统一次运行过程的指称,扩展的ITL正好是描述这种混合区间的有利工具,所以,利用扩展的ITL能较好地给出Verilog的语义.这种语义定义不仅考虑到了各种语言成分的最终执行结果(通过状态变量的左右区间值给出),而且能够很好地给出语言成分执行的时序特征.这种时序特征不仅是刻画实时程序语义必不可少的内容,而且符合人们从执行过程理解程序的习惯,从而使得语义描述更容易被人们理解.引入时序特征不仅增加了语义的表达能力,而且增加了语义求精的范围,从而使得这种语义定义方式能够较好地适用于处理真正的实时程序设计语言.

通过以上的语义工作,我们分析了Verilog语言中的两种不同性质的语言成分,并且在一个统一的理论框架下给出了语言的语义,并且对惯性延迟以及非阻塞赋值等较易引起混淆的语义成分作了细致的分析和刻画.另外,对语言中导致不确定性结果的成分也作了很好的分析.我们进一步的工作是希望确定Verilog的一个子集,该子集直接面向电路综合,可消除不确定的语义成分(如通过限制进程间的共享输出等).在给出了该子集的语义后,我们将继续研究该语义的若干性质(通过一系列代数等式或不等式规则给出这些语义的性质),通过这些代数等式规则,实现从行为级Verilog源程序(语言规范)到RTL或网表级(低层实现)的语义求精,从而为高层综合提供一种形式化的方法.

References:

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

[2] Thomas, D. E., Moorby, P. R. The Verilog Hardware Description Language (3rd edition). New York: Kluwer Academic Publishers, 1996.

[3] Moszkowski, B. C. Executing Temporal Logic Programs. Cambrdige: Cambridge University Press, 1986.

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

[5] Henzinger, T. A., Zohar, Manna, Amir Pnueli. Timed transition systems. In: de Bakker, J. W., *et al.*, eds. Real-Time: Theory in Practice. LNCS 600. New York: Springer-Verlag, 1992. 226~251.

[6] Gerardo, Schneider, Xu, Qi-wen. Towards a formal semantics of verilog using duration calculus. Technical Report, 133, UNU/IIST, Macau, 1998.

[7] He, Ji feng. A behavioural model for co-design. Technical Report, 166, UNU/IIST, Macau, 1999.

Study on the Formal Semantics of Verilog*

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

[1](*Department of Computer Science and Engineering, Shanghai Jiaotong University, Shanghai 200030, China*);
[2](*International Institute of Software Technology, United Nations University, Macau*)

E-mail: lyj238@yahoo.com

http://www.cs.sjtu.edu.cn/chinese/member/index.htm

Abstract: In this paper, the semantics of Verilog program are studied in a discrete-continuous hybrid time model, a hybrid interval is denoted as the description of a run of Verilog program. And an extended ITL is presented to describe this kind of hybrid interval. This semantics not only considers the ultimate execution effects of all kinds ingredients of language, but also gives their temporal characrestics.

Key words: hybrid system; Verilog; interval temporal logic; the scheduling of discrete event; interleaving concurrency

* Received October 10, 1999; accepted May 26, 2000
Supported by the UNU/IIST Research Project