Verilog HDL 语言中的特殊数据类型及其赋值

林　媛

(厦门大学 计算机与信息工程学院　福建 厦门　361005)

摘　要: 阐述了 Verilog HDL 语言中的两种特殊数据类型, 并通过示例源程序, 归纳总结了对于这两种特殊数据类型的赋值。对 Verilog HDL 用户具有普遍指导意义。

关键词: Verilog HDL; 连线类型变量; 寄存器类型变量; 赋值

中图分类号: TP312　　文献标识码: B　　文章编号: 1004-373X (2004) 21-024-03

Two Special Data Types in Verilog HDL and Evaluation of the Variables

LIN Yuan

(College of Computer and Information Engineering, Xiamen University, Xiamen, 361005, China)

Abstract: Present two special data types in Verilog HDL. And the source programs of Verilog HDL is given. Then get some conclusions about how to evaluate these two kinds of variables in different situation. It has common guide significance to the Verilog HDL users.

Keywords: Verilog HDL; wire variable; register variable; evaluate

Verilog HDL 语言支持保持不变的常量和在仿真中可以修改的变量。常量同其他编程语言一样, 可以是十进制、十六进制、八进制或二进制。而 Verilog HDL 语言中的变量除了其他语言中常见的整数、实数等以外, 还引入了一些特定的数据类型: 连线类型和寄存器类型。其中连线型代表单元之间的互连; 寄存器代表存储单元。以下主要讨论这两种特殊的, 也是 Verilog HDL 中最主要的两种数据类型。

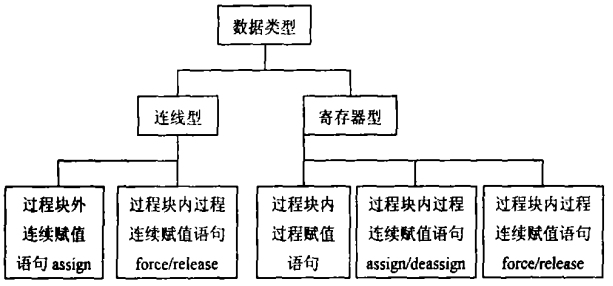

图 1　Verilog HDL 特殊数据类型与赋值

1　对连线类型变量的赋值

assign 连续赋值语句 (Continuous Assignment Statement)　在过程块外完成连线类型变量的赋值。语句中的赋值算符只有 "=" 即阻塞型一种。一旦连线变量被连续赋值语句赋值后, 只要表达式中的信号有任何变化, 都将随时反映到连线变量中 (此特点不同于过程赋值语句, 只有在语句被执行时赋值才进行一次)。

force/release 过程连续赋值语句　可以在过程块内完成连线类型变量的赋值。赋值算符也是 "="。该语句对连线类型变量赋值的基本工作过程同 assign 连续赋值语句。但是, force/release 语句只对整体变量赋值, 而不能像连续赋值语句 assign 对变量的一位或其中几位进行赋值, 而且 force/release 赋值语句具有更高的优先级。若存在此赋值语句对某连线型变量进行赋值, 那么他可以覆盖掉其他所有对这条连线的驱动。

2　对寄存器类型变量的赋值

过程赋值语句, 即过程块中的赋值语句。他只能在块中对寄存器类变量进行赋值。可以是对整个变量赋值, 也可以是对其中的某一位或某几位进行赋值。过程赋值语句有两种赋值方式: 阻塞型过程赋值与非阻塞型过程赋值。二者均可实现对寄存器变量的赋值, 但赋值算符不同, 语句执行过程不同。出于可综合化设计的角度考虑, 在同一块语句中不允许同时出现阻塞型过程赋值语句与非阻塞型过程赋值语句。

(1) 阻塞型过程赋值

其算符是 "="。在串行块的执行过程中, 若一条语句没有完成赋值, 那么接下来的语句不可能被执行。

(2) 非阻塞型过程赋值

其算符是 "<="。在串行块的执行过程中, 若一条非阻塞型赋值语句没有完成赋值, 那么其他块内语句的执行不受影响。也就意味着一个完全由非阻塞型赋值语句构成的串行块的执行效果与并行块的执行效果完全一样。

assign/deassign 过程连续赋值语句　在过程块中, 使用连续赋值语句 assign 对寄存器变量进行赋值。当使用

收稿日期: 2004-04-15

assign 语句对寄存器变量赋值后, 原有对给寄存器变量进行的过程赋值语句不再起作用, 直到 deassign 语句将此强制过程释放, 正常的过程赋值语句才重新起作用。值得注意的是, 使用 assign/deassign 语句赋值的寄存器变量只能被整体赋值, 不能只对其中的某一位或某几位进行赋值。

force/release 过程连续赋值语句 在过程块中, 可使用此语句对寄存器变量进行赋值。该赋值语句具有最高优先级。既使过程块中出现由此赋值语句赋值的寄存器变量, 那么其他赋值语句对变量的赋值均无效, 直至 release 语句执行。

3 连续赋值语句与过程赋值语句的比较

[示例源程序一]

```
// 用连续赋值语句实现或门
module demo_or_assign(c,a,b);
input a,b;
output c;
assign c= a| b;      //连续赋值语句,且默认 c 为连线型变量
endmodule
```

[示例源程序二]

```
// 用过程赋值语句实现或门
module demo_or_procedure(c,a,b);
input a,b;
output c;
reg c;
always @(a or b) c= a| b;
                    //过程赋值语句,c 被定义为寄存器变量
endmodule
```

通过执行示例源程序一、二可以比较出连续赋值语句和过程赋值语句的不同:

程序一中 c 作为连线型变量由 assign 赋值, 只要 a 和 b 信号有任何变化都马上反映到 c。

程序二中的 c 则定义为寄存器变量, a 或 b 发生变化, 则过程赋值语句通过 always 循环控制, 从而完成对 c 的赋值。

4 过程赋值语句中阻塞型与非阻塞型赋值语句的比较

[示例源程序三]

```
//使用阻塞型过程赋值语句对寄存器变量赋值
module demo_blocking;
reg     a,b,c;
initial
  begin
  a= # 10   1;
  b= # 20   0;                    //阻塞型过程赋值语句
  c= # 5    0;
  end
endmodule
```

[示例源程序四]

```
//使用非阻塞型过程赋值语句对寄存器变量赋值
module demo_im_blocking;
reg     a,b,c;
initial
  begin
  a<= # 10   1;
  b<= # 20   0;                   //非阻塞型过程赋值语句
  c<= # 5    0;
  end
endmodule
```

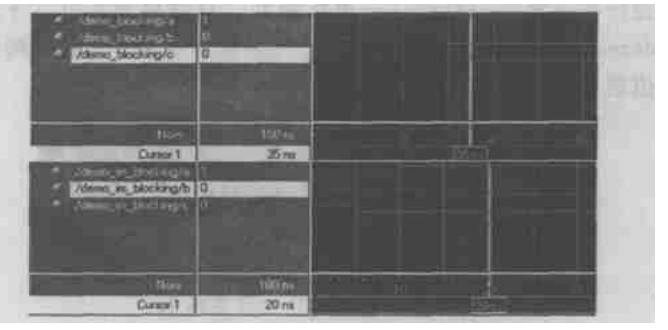

图 2 示例源程序三、四仿真结果

通过执行示例源程序三、四可以比较出阻塞型过程赋值语句和非阻塞型过程赋值语句的不同:

(1) 赋值符不同

阻塞型过程赋值为“=”;

非阻塞型过程赋值为“<=”。

(2) 执行结果不同

从图2 中看出, 程序三的结果是: 在模拟到达 10 单位时刻, a 得 1 值; b 在模拟到达 30 单位时刻才获得 0 值; c 则在 35 单位时刻才获得 0 值。而程序四的运行波形结果是: 在模拟到达 5 单位时刻, c 即获得 0 值; 在模拟到达 10 单位时刻, a 获得 1 值; 在模拟到达 20 单位时刻, b 获得 0 值。由此可得, 非阻塞型过程赋值语句构成的串行块的效果类似于 fork/join 并行块的作用。以下给出若使用 fork/join 实现程序四功能的示例源程序。

[示例源程序五]

```
//使用并行块实现程序四功能
module demo_blocking;
reg     a,b,c;
fork
      a= # 10   1;
      b= # 20   0;
      c= # 5    0;
join
endmodule
```

5 过程连续赋值语句进阶

[示例源程序六]

```
//异步清零的上升沿 D 触发器
module demo_asyn(q,d,clear,clk);
output q;
input   d,clear,clk;
reg     q;
always @(clear)
  if(! clear)
    assign   q= 0;
//过程连续赋值语句 assign/deassign 对 q 寄存器变量赋值
  else
    deassign q;
  always @ (posedge clk)
    q= d;                         //过程赋值语句对 q 赋值
```

endmodule

通过对程序六的仿真如图 3 所示，清楚地看到，只要 clear 出现跳变，结果为 0 时，assign 过程连续赋值语句会强行把 q 置 0，直到 clear 再次跳变，且结果非 0 时，才由 deassign 释放对 q 的强行赋值，另一过程块中对 q 的过程赋值语句赋值才开始有效。

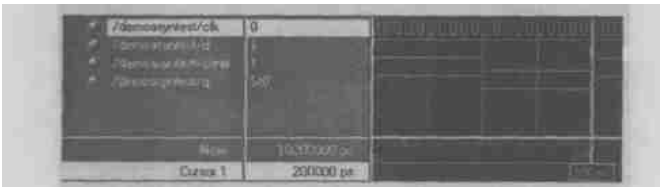

图 3　示例源程序六仿真结果

通常情况下，出现过程连续赋值语句 assign 的地方，必存在一条 deassign 语句，否则系统无法恢复到正常的过程赋值状态。若在同一过程块中出现有两个过程连续赋值语句 assign 对同一寄存器变量赋值，则后执行的语句将覆盖前一过程赋值语句产生的作用。无论块中出现了多少个过程连续赋值语句 assign 对同一寄存器变量的赋值，只需在最后一个 assign 后出现一条 deassign，该变量的语句就能释放此变量了。

6　结　语

综上所述，连线类型变量和寄存器类型变量均能通过赋值语句接受赋值，但驱动方式不同。尤其是连线类型变量由于没有电荷保持作用，所以当他没有驱动时，所处状态是高阻态 Z；而寄存器变量对应的是具有状态保持作用的硬件电路元件，因此当他没有驱动时，处于不定态 X。

参　考　文　献

[1] 袁俊泉，孙敏琪，曹瑞．Verilog HDL 数字系统设计及其应用［M］．西安：西安电子科技大学出版社，2002.

[2] 张亮．数字电路设计与 Verilog HDL［M］．北京：人民邮电出版社，2000.

[3] 张明．Verilog HDL 使用教程［M］．成都：电子科技大学出版社，1999.

作者简介　林　媛　女，厦门大学计算机与信息工程学院电路与系统专业研究生。研究方向为芯片设计和 ASIC 设计。

（上接第 19 页）

由表 1 可知，采用准同步采样实现对谐波的测量，其效果是比较令人满意的。另外，由于采用了准同步采样技术，使得硬件实现电路得到了简化，降低了成本；同时由于采用了加权处理，使得实际测量的计算量大为减少，用一般的处理器就可得到较快的速度。

4　结　语

参数自适应的准同步采样谐波分析法是在准同步采样的基础上，对采样次数 N 与采样时间间隔 t_s 两个参数进行自适应调整，并对存在采样误差的原始数据进行一次加权处理，不仅减小误差，而且提高了运算速度。该算法经仿真分析证明是有效的，并能够实现相对快速、准确的测量，具有实用性。

参　考　文　献

[1] 周海波，王学伟，白杨．数据插值在谐波分析算法中的应用及其实现［J］．电测与仪表，2002，39（8）：5－7.

[2] 王震宇，王学伟．基于参数自适应的准同步采样法［J］．哈尔滨理工大学学报，2001，6（2）：36－39.

[3] 杨网娟．一种非正弦周期信号谐波分析的方法—准同步采样法［J］．河海大学机械学院学报，1995，9（2）：17－20.

[4] 沈国峰，王祁．进一步提高准同步采样谐波分析法准确度的方案［J］．仪器仪表学报，2001，22（5）：455－457.

[5] 张盎然，陈明华，杨杨．基于准同步算法的谐波分析方法［J］．电测与仪表，2002，39（1）：10－12.

作者简介　李　勇　男，1978 年出生，湖北鄂州人，硕士。现从事测控仪表的智能化及智能监测控制算法研究。
赵　建　男，1956 年出生，陕西西安人，教授，硕士生导师。现从事智能化仪器仪表的研究工作。

（上接第 23 页）

参　考　文　献

[3] Microsoft．SQL Server2000 Training Kit 系列教程［M］．美国：微软出版社，2001.

[4] Microsoft．Microsoft SQL Server 2000 Referenct Library［M］．北京：北京大学出版社，2001.

[5] 萨师煊，王珊．数据库系统概论（第 3 版）［M］．北京：高等教育出版社，2000.

[6] Scott Meyers．Effective C++中文版［M］．武汉：华中科技大学出版社，2001.

[7] Bjarne Stroustrop．C++程序设计语言［M］．北京：高等教育出版社，2001.

作者简介　梁德胜　男，1972 年出生，陕西蒲城人，在职硕士。主要研究方向为管理信息系统、计算机网络。
杨晓燕　女，1977 年出生，湖北襄樊人，硕士。主要研究方向为管理信息系统。