Verilog HDL阻塞属性探究及其应用

郭宝增，汪祥春
（河北大学 电子信息工程学院，河北 保定 071002）

摘 要：阻塞赋值与非阻塞赋值语句作为verilog HDL语言的最大难点之一，一直困扰着FPGA设计者，而其中的错误又隐晦莫测，理解不透彻会直接导致运用不当，使设计工程达不到预期效果，而排错又相当麻烦。阻塞赋值与非阻塞赋值语句既血脉相连，又有本质的区别。透过原理和实际应用，从不同侧面对阻塞赋值与非阻塞赋值进行剖析，并阐述了阻塞赋值与非阻塞赋值的各自特点及其应用。

关键词：Verilog HDL；阻塞赋值；非阻塞赋值；层积事件列
中图分类号：TP332 **文献标识码**：B

Blocking assignment property and its application in Verilog HDL

GUO Bao Zeng, WANG Xiang Chun
(College of Electronic and Information Engineering, Hebei University, Baoding 071002, China)

Abstract: Blocking assignment and nonblocking assignment is one of the most misunderstood constructs in Verilog HDL, which usually bewilders the designer of FPGA. Incorrectly use maybe cause the project fail directly, and this mistake is not very apparent and debug is more difficult. Blocking assignment has close relationship with nonblocking assignment, but the difference is also apparent. This paper details difference between them and their application through the theory.

Key words: Verilog HDL; blocking assignment; nonblocking assignment; stratified event queue

Verilog HDL中，有两种过程赋值方式，即阻塞赋值（blocking）和非阻塞赋值（nonblocking）。阻塞赋值执行时，RHS（right hand statement）估值与更新LHS（left hand statement）值一次执行完成，计算完毕，立即更新。在执行时阻塞同块中的其他语句的执行。阻塞式（blocking）的操作符为“=”。它的执行很像传统程序设计语言。非阻塞赋值RHS估值与更新LHS值分两步执行。在单位仿真周期开始时RHS估值，在同一单位仿真周期末更新LHS值，不阻塞同块中其他语句的执行。非阻塞式（non-blocking）的操作符为“<=”，它的执行更像并行电路，使描述电路更自然。阻塞赋值与非阻塞赋值是Verilog HDL程序设计的难点，它们既有共同点，也有差异，深入剖析其异同，对于硬件程序的开发具有重大意义。

1 Verilog事件处理机制

层积事件列（The Stratified Event Queue）是一个事件管理概念模型，而非硬件逻辑。模型内事件的具体实现与EDA软件生产商的算法策略有关。在IEEE-2001中，Verilog把事件分为5个不同部分，按照时间顺序如图1所示。

触发的任何事件可以加入到这5个事件列中的任何事件列中，但只能从活跃事件列中移出。即上面的5个事件列中的事件最后都将被激活而放入活跃事件列中。层积事件列是层次模型，层积事件列的执行顺序是按优先级排列的。任何EDA软件都只能执行活跃事件。其他事件列都按优先级级别依次激活本列事件以供执行。

1.1 活跃事件列

由图1可见，大部分事件都被放入活跃事件列。活跃事件列里包括非阻塞赋值RHS估值。但是，非阻塞赋值的更新不是在活跃事件列，它被列成独立的非阻塞更新事件列。活跃事件列是仿真的执行源，从一开始执行活跃事件列到活跃事件列执行完毕称为一个仿真周

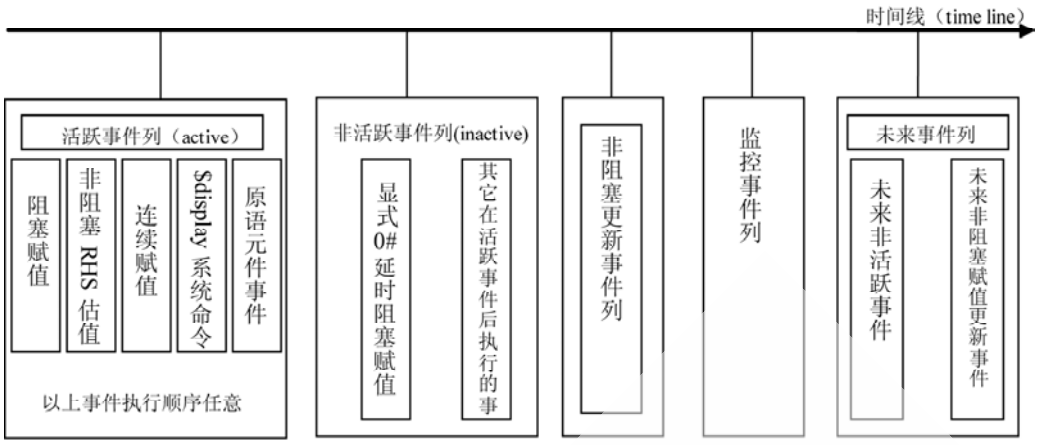

图 1 层积事件列模型

期。活跃事件列中的事件可以触发活跃或非活跃等其他事件。当活跃事件列中的所有事件执行完后，EDA 软件会按优先级依次触发其余事件列以供仿真执行。但在当前活跃事件列中的事件执行顺序是不确定的。

1.2 非活跃事件列

发生在当前仿真时间里并且在活跃事件列执行完后执行的事件列，即非活跃事件列执行优先级仅次于活跃事件列。如带 PLI 例程的回调过程（tf_synchronize()、vpi_register_cb(cb_readwrite)）。非活跃事件列中的事件亦可以触发其他事件。如果触发了优先级更高的活跃事件，非活跃事件列中的其余事件执行后移。

1.3 非阻塞赋值更新事件列

活跃事件列中的每个非阻塞赋值 RHS 估值，都会触发一个与之对应的非阻塞赋值更新事件，这些事件被放在非阻塞赋值更新事件列中，执行优先级次于活跃与非活跃事件列。非阻塞赋值更新事件亦可以触发其他事件。若在非阻塞赋值更新事件列中，存在多个对同一变量的先后赋值，只有最后一个有效，其余值将被覆盖。

1.4 监控事件列

监控事件列被放在非阻塞赋值更新事件列后。由此可见，用监控事件列中的监控命令监控得到的值都是赋值后的值，活跃事件列 $display 系统命令则可以查看非阻塞更新前的值。

1.5 未来事件列

在执行事件时，如果事件含有延时，为不阻碍仿真的继续执行,该事件将被挂起而放入未来事件列。未来事件包含未来非活跃事件和未来非阻塞赋值更新事件。

理解阻塞与非阻塞赋值就需要深入理解层积事件列，层积事件列反应了 Verilog 事件处理机制。

2 应用及分析

通常非阻塞赋值产生寄存器等存储元件，对应的物理器件是带存贮功能的元件，如寄存器、触发器等。阻塞赋值则对应网线（wire）类型，通常与物理连线对应。这是两种赋值方式的最明显的差异，也是时序逻辑用非阻塞、组合逻辑用阻塞的重要原因。但这并不是绝对的，事实上阻塞赋值对应网线（wire）型，亦可对应寄存器（reg）型；阻塞赋值也能生成存贮元件，因此不能片面理解。在组合逻辑里，锁存器可能引发测试问题，带来隐患。说明在建模时，首先要从硬件出发来考虑问题，应先在头脑中形成电路结构，由于赋值方式的不同，综合结果差异甚大，运用不当很可能会导致建模失败。阻塞赋值在时序逻辑中亦有着重要应用，在需要实时更新的组合逻辑中只有阻塞赋值能满足要求。

以下示例代码的功能是计算传送过来的 data 中 1 和 0 的个数。

```
reg [5:0]count0，count1;
always @(posedge clk,negedge Rst_n)
  begin
    if(!Rst_n)
     ...
    else
     begin
       count0 = 0;                    //语句 1
       count1 = 0;                    //语句 2
       for(i = 0;i <= 11;i = i+1)
         begin
           if(data[i] == 1)
             count1 = count1 + 1;       //语句 3
           else if(data[i] == 0)
                 count0 = count0 - 1;     //语句 4
                 else
                     count0 = count0 + 0;   //防止生成锁存器
         end
     end
  end
```

在这段代码里，count0、count1 的值必须在每次计数

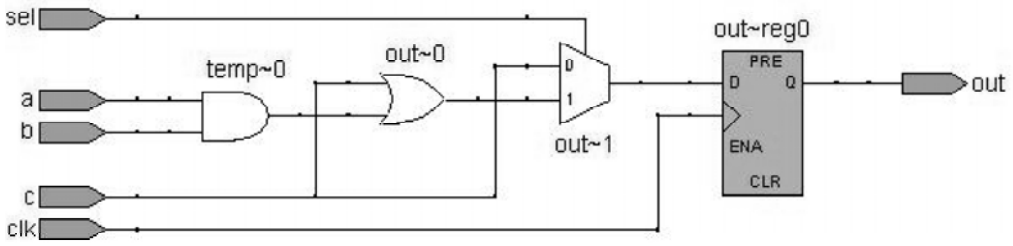

图 2 纯组合逻辑

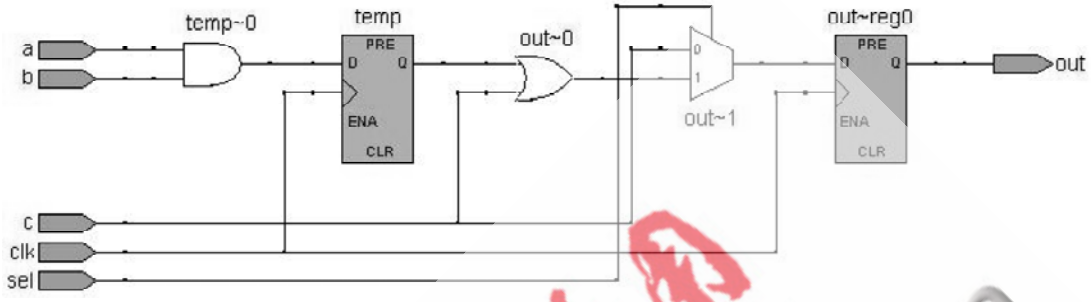

图 3 流水线组合逻辑

之前被清零，count0、count1 必须实时更新。显然，只有阻塞赋值能满足要求。非阻塞赋值分两步完成，所有的更新事件在单位仿真周期末同时执行，只有最后一个值有效，所以非阻塞赋值无法完成计数任务。阻塞赋值却能很好地胜任，因为阻塞赋值估值和更新一次性完成。

事件上，在时序逻辑中经常碰到上述实时更新问题，非阻塞赋值往往无法实现，如用阻塞赋值则可很好地解决问题。

正如阻塞赋值在时序逻辑中有重要应用一样，非阻塞赋值在组合逻辑中亦有不可替代的应用。在组合逻辑中用非阻塞赋值可以把组合逻辑改造成流水线。可执行如下所示纯组合逻辑代码，将生成纯组合逻辑，综合结果如图 2 所示。

```
input a,b,c,clk,sel;
output out;
reg out,temp;
always @(posedge clk)
 begin
  temp  = a & b;    //语句 1
  if(sel)
   out  = temp | c;  //语句 2
  else
   out  = c;       //语句 3
 end
```

若把上面代码中语句 1、语句 2、语句 3 阻塞赋值（" = "）改为非阻塞赋值（" <= "），则综合结果如图 3 所示。

流水线设计方法在高性能、需经常进行大规模运算的组合逻辑中可以到广泛运用。

在组合逻辑中，如在 begin、end 块中同时有许多非阻塞赋值，则它们的赋值顺序是并发的。实际上它们赋予的都是上一个时钟送入寄存器的值。这与使用同一时钟沿触发的许多在同一个使能控制信号下赋值完全一致，并且这种赋值因为数据保存在寄存器中，当时钟沿到来时都已稳定，所以存入的数值是可靠的。用这种方法可以避免由组合逻辑产生的竞争冒险[2]。

在相关应用中，非阻塞赋值能较好地解决零时刻竞争冒险问题。因为非阻塞赋值分两步完成，非阻塞赋值更新事件是在所有活跃与非活跃事件执行完之后执行，能确保所有敏感变量值在零时刻都被触发[3]。

在同一 always 块混合使用阻塞赋值与非阻塞赋值，利弊共存，混合使用的结果可能事半功倍，亦可能功亏一篑。只有了解其处理机制，深刻理解阻塞与非阻塞赋值底层实现的异同，方可灵活运用。

本文通过 Verilog 事件处理机制，详细讨论了阻塞与非阻塞赋值的区别、联系及其应用示例。由本文可知，阻塞与非阻塞赋值灵活多变，底层实现也差异甚大。因而在数字电路设计时，依据预期功能，从硬件实现出发，斟酌差异，仔细选用阻塞与非阻塞赋值才能有效避免出错，缩短开发周期。

参考文献

[1] IEEE Standard verilog hardware description language based on the verilog hardware description language[R]. IEEE Computer Society. IEEE. New York. NY. IEEE Std 1364-2001.

[2] 夏宇闻.Verilog 数字系统设计教程[M]. 北京: 北京航空航天大学出版社，2003.

[3] CLIKFFORD E.C. nonblocking assignment in verilog synthesis. coding styles that kill[J]. SNUG (Synopsys Users Group) 2000 User Papers，2000(3).

(收稿日期：2009-03-19)