基于层次化事件队列的赋值操作应用①

孙 健, 张莎莎, 时鹏飞, 张 雪

(西安微电子技术研究所, 西安 710054)

摘 要: 随着 ASIC、SoC 和 FPGA 在工业生产和科研工作中的广泛应用, 作为硬件描述语言的 Verilog HDL 已成为实现集成电路设计的重要方式, 而设计师对其中赋值操作的错误使用, 也导致了部分设计的不正确. 因此, 对于 Verilog HDL 中赋值操作的深入了解也成为解决设计中隐藏问题的瓶颈. 基于层次化事件队列, 对设计中经常遇到的赋值操作问题进行了深入剖析, 并结合工作实际给出了避免设计中竞争问题出现的方法.

关键词: Verilog HDL; 竞争; 层次化事件队列; 编码规范

Assignment Based on Stratified Event Queue

SUN Jian, ZHANG Sha-Sha, SHI Peng-Fei, ZHANG Xue

(Xi'an Microelectronics Technology Institute, Xi'an 710054, China)

Abstract: With the development of ASIC, SoC and FPGA in research and industry, Verilog HDL has been the main method to design the integrate circuit. For the designer miusing the assignments in Verilog HDL, more and more designs have been with some invisible bugs. That the designers fully understand how the assignments are scheduled in the program has been the main method to solve the bugs. This paper is based on the stratified event queue to detail how the assignment works in design, and gives important coding guidelines to infer correct logic and avoid race condition in the design.

Key words: verilog HDL; race condition; stratified event queue; coding style

1 前言

随着 ASIC、SoC 和 FPGA 在工业生产和科研工作中的广泛应用, 作为实现其功能的硬件描述语言(Hardware Description Language)得到了广泛的应用, 其中的 Verilog HDL 已成为主要的硬件实现语言.

与此同时, 国外的 CLIFFORD Cummings 等人进行了相关的研究工作, 并取得了一定的成果. 但是, 目前国内许多工程师和设计人员在使用 Verilog HDL 进行设计时, 并没有对语言本身有足够的理解, 也并没有对此进行深入的研究. 由于对于语言本身的理解偏差或者不合理的编码方式导致设计结果与期望不相符的情况也越来越多, 从而大大影响了整个设计的正确性、可靠性和稳定性. 因此只有对 Verilog HDL 语言本身的正确理解和使用, 才能从根本上避免由于错误使用语言导致的设计错误.

本文主要基于层次化事件队列对 Verilog HDL 的赋值操作在设计中遇到的各种常见问题进行了论述, 并根据论述结果结合工作实际给出相应的编码规则和指导建议, 从而可以帮助相关技术人员正确进行硬件设计, 避免设计错误.

2 问题的产生

2.1 竞争

基于 Verilog HDL 的设计中, 当两条或多条语句在同一个仿真时间(time-step)中执行时, 如果语句的执行顺序不同, 会产生不同的仿真结果, 这种现象被称为竞争(Race Condition). 而在设计中对赋值操作执行机理的准确理解和正确应用将直接影响到设计的正确性和可靠性.

2.2 层次化事件队列(Stratified Event Queue)

Verilog 中事件队列的执行是按照一定的算法结构执行, 在 IEEE std 1364[5][1,2]中描述了 Verilog 仿真的参

① 基金项目:国家高技术研究发展计划(863)(2011AA120202)
收稿时间:2013-06-13;收到修改稿时间:2013-07-15

考模型(如图 1 所示), 但没有提出具体的执行时序关系.

```
T refers to the current simulation time, and all events are held in the event
queue, ordered by simulation time.
while (there are events) {
        if (no active events) {
                if (there are inactive events) {
                        activate all inactive events;
                } else if (there are non blocking assign update events) {
                        activate all non blocking assign update events;
                } else if (there are monitor events) {
                        activate all monitor events;
                } else {
                        advance T to the next event time;
                        activate all inactive events for time T;
                }
        }
        E = any active event;
        if (E is an update event) {
                update the modified object;
                add evaluation events for sensitive processes to event queue;

        } else { /* shall be an evaluation event */
                evaluate the process;
                add update events to the event queue;
        }
}
```

图 1 Verilog 仿真参考模型

根据 IEEE 中提出的仿真参考模型的定义, 可以将该仿真参考模型从逻辑执行顺序上分为五个不同部分(如图 2), 形成各种语句操作执行的层次化事件队列.

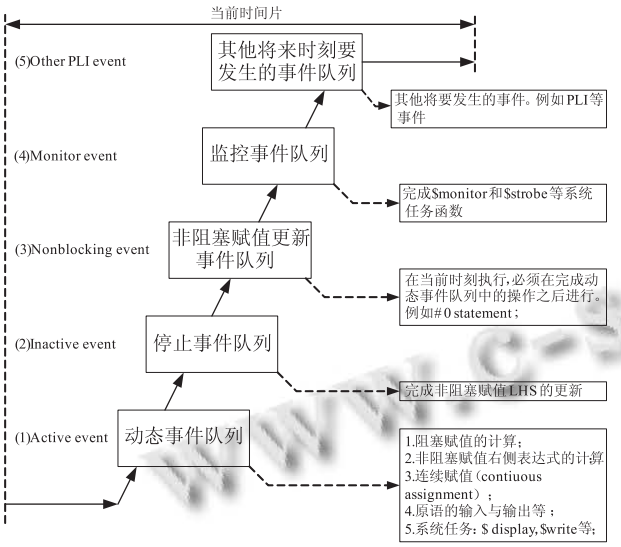

图 2 Verilog HDL 层次化事件队列的划分

从图 2 可以看到, 层次化事件队列与 Verilog 参考模型的对应关系如下:

① 动态事件队列对应于 Verilog 参考模型中的 active events;

② 停止事件队列对应于 Verilog 参考模型中的 inactive events;

③ 非阻塞赋值更新事件队列对应于 Verilog 参考模型中的 non blocking assign updata events;

④ 监控事件队列对应于 Verilog 参考模型中的 monitor events;

⑤ 将来时刻将要发生的事件队列在所有 active event、inactive event 和 monitor event 事件队列之后进行;

通过该层次化事件队列的应用, 将有助于解决设计编程中遇到的各种问题和困惑, 有助于硬件设计人员更好的进行正确的设计.

2.3 阻塞赋值(Blocking Assignment)的执行顺序问题

阻塞赋值首先执行赋值符号右边(RHS)的表达式, 再把 RHS 的表达式的值赋给赋值符号左侧(LHS), 在执行过程中不允许任何其他语句的干扰, 直到 RHS 的值完全赋给 LHS. 因此可以理解为, 在同一个执行过程中, 只有阻塞赋值语句执行完成后, 在阻塞赋值语句之后的语句才能得到执行[3].

但是在进行组合逻辑建模时, 很多工程师忽略了阻塞赋值语句执行的顺序, 造成执行结果不正确, 如图 3 代码所示.

```
// --- Code_A --- //
always @(posedge clk or negedge rst )
begin
   if (rst == `b0)
   begin
      ……
   end
   else
   begin
      tmp = d;
      q   = tmp;
   end
end

// --- Code_B --- //
always @(posedge clk or negedge rst )
begin
   if (rst == `b0)
   begin
      ……
   end
   else
   begin
      q   = tmp;
      tmp = d;
   end
end
```

图 3 不同顺序的阻塞赋值代码

在 Code_A 中, 按照 Verilog HDL 层次化事件队列执行的顺序, 阻塞赋值按照顺序执行, 且只有当前语句执行完后, 其后的语句才能够执行. 因此, 在该段

代码中, “tmp = d”中的 tmp 的值在动态事件队列中被立即更新, 更新完成后, 执行“q = tmp”, 其中此时的 tmp 为上一条阻塞赋值语句执行后更新的值, 因为阻塞赋值的整个赋值过程都被安排在动态事件队列, 所以此时 q 的值被更新后的 tmp 的值更新. 因此, 在同一个仿真时间中, q 的值被更新为 d, 其仿真行为可用图 4 来模拟解释.

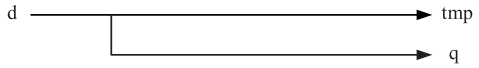

图 4　Code_A 仿真行为

在 Code_B 中, 因为“q = tmp”先于“tmp = d”执行, 之前仿真时刻寄存于 tmp 中的值更新至 q 端, 在 q 端被更新后, “tmp = d”得到执行, tmp 中的值被 d 更新, 从而可以实现触发器功能. 虽然前仿可以实现触发器的功能, 但是经过综合后, 可能会造成其中的存储功能可能被优化掉, 形成图 4Code_A 的仿真行为, 造成前后仿结果不一致, 建议不要使用这种风格的编码方式编写触发器, Code_B 仿真行为可用图 5 来模拟解释.

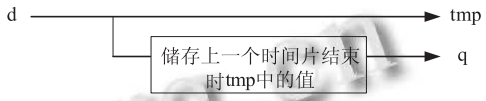

图 5　Code_B 仿真行为

Code_A 和 Code_B 前仿真波形如图 6 所示.

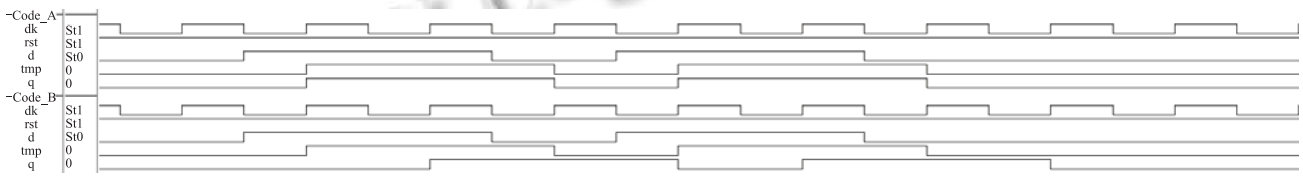

图 6　阻塞赋值 Code_A 和 Code_B 仿真波形

从以上结果可以看到, 当阻塞赋值语句的执行顺序不同时, 执行结果完全不同, 从而很容易导致设计结果的不同. 因此, 在使用阻塞赋值语句时, 需要特别注意语句的执行顺序, 否则, 实际结果可能与期望结果不一致, 影响设计正确性、可靠性和规范性.

2.4 非阻塞赋值(NBA: Nonblocking Assignment)输出结果显示不一致问题

非阻塞赋值是用来模拟时序逻辑电路中的传输延迟(tclk-q)[4], 因此, 非阻塞赋值符号右侧表达式的值在 tclk-q 之后, 将左侧表达式的值更新. 在计算 RHS 和更新 LHS 期间, 其他语句的执行不受影响. 因此可以认为, 非阻塞赋值语句执行时, 其他语句的执行可以同时进行操作, 其具体实现如图 7 中 Code_A 和 Code_B 所示.

Code_A 和 Code_B 仿真波形如下图 8 所示.

从上述仿真结果可以看出, 在进行非阻塞赋值时, 在仿真开始时首先计算的是非阻塞赋值符号“<=”RHS 的值, 完成动态事件队列中的事件之后, 进入非阻塞赋值更新事件队列时才更新非阻塞赋值符号“<=”LHS 的值.

而在实际建模时, 使用$display 和$monitor 系统任务对可以观测到不同事件队列中的赋值操作如何进行, 如图 9 所示.

```
// --- Code_A --- //
always @(posedge clk or negedge rst )
begin
   if (rst ==  'b0)
   begin
      ……
   end
   else
   begin
      tmp <= d;
      q    <= tmp;
   end
end

// --- Code_B --- //
always @(posedge clk or negedge rst )
begin
   if (rst ==  'b0)
   begin
      ……
   end
   else
   begin
      q    <= tmp;
      tmp <= d;
   end
end
```

图 7　不同顺序的非阻塞赋值代码

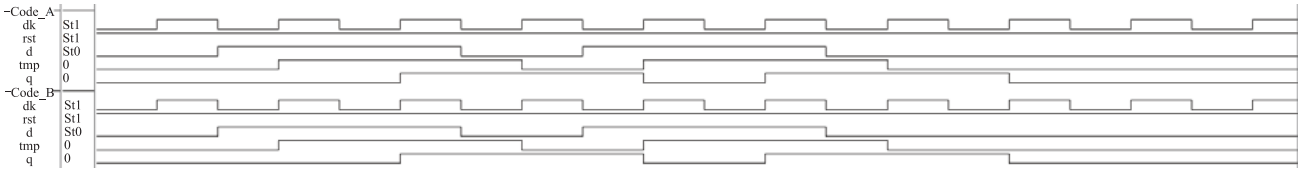

图 8 非阻塞赋值 Code_A 和 Code_B 仿真波形

```
reg a;
initial begin
   $monitor(“\$monitor : a = %b” , a);
end
initial begin
   a = ‘bz;
   $display(“First \$display : a = %b”, a);
   a = ‘b1;
   a <= ‘b0;
   $display(“Second \$display : a = %b” , a);
end

运行结果：

# First $display : a = z;
# Seconf $display : a = 1;
# $monitor : a = 0;
```

图 9 不同事件队列中观测到的信号变化

造成对同一变量显示不同结果是因为非阻塞赋值表达式左侧的值得更新发生在动态事件队列之后，所以此时使用属于动态事件队列中的$display 系统任务观测到的是未被非阻塞赋值更新的 a 中的值，而$monitor 属于监测事件队列，排在非阻塞赋值 LHS 更新队列之后，通过此任务可观测到 a 是非阻塞赋值更新之后的值.

因此，在使用非阻塞赋值语句进行建模时，还需要注意使用显示和监控任务的输出数值有可能不一致的情况.

2.5 混合赋值(mix assignment)输出结果不一致

在 IEEE 中并没有禁止阻塞赋值和非阻塞赋值同时出现在一个过程性语句中，但是当这两种赋值出现在同一个过程性语句中时，可能会导致输出结果不一致[5,6].

在实际建模时，会遇到图 10 中 Code_A 和 Code_B 中同时使用阻塞赋值与非阻塞赋值的情况，造成仿真结果完全不一样.

```
// --- Code_A --- //
always @(posedge clk or negedge rst )
begin
   if (rst == ‘b0)
   begin
      ……
   end
   else
   begin
      tmp  = d;
      q    <= tmp;
   end
end

// --- Code_B --- //
always @(posedge clk or negedge rst )
begin
   if (rst == ‘b0)
   begin
      ……
   end
   else
   begin
      tmp <= d;
      q    = tmp;
   end
end
```

图 10 混合赋值操作代码

在 Code_A 中，虽然动态事件队列中的事件执行顺序是任意的，但是此处 begin-end 中的语句是按顺序执行(在同一个时间片中)的，所以，在非阻塞赋值语句“q <= tmp”中右侧表达式 tmp 值被更新之前，首先执行了“tmp = d”，即 tmp 中的值被 d 更新，所以非阻塞赋值语句在计算 RHS 时，tmp 值为被阻塞赋值语句更新后的值，因此，q 的输出为 d.

在 Code_B 中，“q = tmp”中 tmp 为之前时间片中计算的 tmp 值，所以此时观测到的 q 的输出为上一时间片中 tmp 的值，而在 tclk-q 之后，d 中的值将 tmp 进行更新.

Code_A 和 Code_B 的仿真结果如图 11 所示.

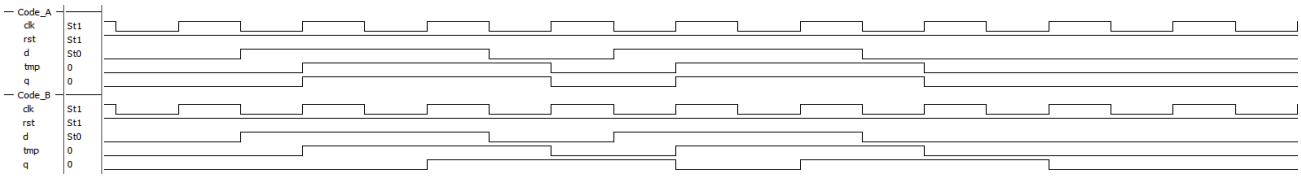

图 11 混合赋值 Code_A 和 Code_B 仿真波形

因此, 在进行编码时, 应该避免将阻塞赋值和非阻塞赋值在同一个过程性语句中进行操作, 否则将导致不一致的设计结果. 所以, 在同一个过程性语句中全部使用同一种赋值方式, 以保证结果与预期结果一致.

3 解决方法

针对以上设计中遇到的各种问题, 依据层次化事件队列执行顺序的特点, 给出以下指导方法, 可以减少 90%以上的竞争问题和设计仿真结果与综合不一致的问题[7-10]:

① 在进行时序逻辑和锁存器建模时, 建议采用如下方式进行编码:

```
always @(posedge clk or negedge rst)
begin
   if (rst == ‘b0)
   begin
      ……
   end
   else
   begin
     q <= d;
   end
end
```

图 12　时序逻辑模型

② 在进行组合逻辑建模时, 建议采用如下方式进行编码:

```
always @(d)
begin
      q  = d;
end
```

图 13　组合逻辑模型

③ 在建立时序和组合逻辑的混合逻辑时, 建议采用如下方式进行编码:

```
always @(posedge clk or negedge rst )
begin
   if (rst == ‘b0)
   begin
      ……
   end
   else
   begin
     q  <= d & d_1;
   end
end
```

图 14　混合逻辑建模

④ 在同一个过程语句中, 不要混合使用阻塞赋值和非阻塞赋值(如 2.5 节所示);

⑤ 不要在多个过程性语句中对同一个变量进行赋值操作, 将会导致竞争状况的出现, 影响输出结果的正确性;

⑥ 在进行仿真时, 选用正确的显示任务记录要观测的变量, 建议使用$strobe 和$monitor 等命令观测相应的信号(如 2.4 节所示);

⑦ 在进行赋值时, 不要使用零延迟(#0), 以免影响正常的语句执行顺序;

⑧ 因为 begin-end 结构中的语句是按照顺序执行的, 无论该语句是阻塞赋值还是非阻塞赋值, 其在 begin-end 中是按照自上而下的顺序分析每条语句的;

⑨ 敏感信号列表中只加入过程性语句中的输入信号, 不要加入参与运算的中间信号, 因为中间信号的变化将会再次触发敏感信号列表, 从而打断原有事件队列的执行, 导致前后仿真结果不一致.

4 总结

本文通过基于层次化事件队列对硬件设计中使用的 Verilog HDL 阻塞赋值与非阻塞赋值遇到的各种问题进行了剖析, 总结并且给出减少和避免设计中竞争情况出现的指导方法, 同时可以使设计综合出正确的结果, 确保设计仿真功能与综合结果一致, 保证设计的可靠性、稳定性和正确性.

参考文献

1 IEEE Standard Verilog Hardware Description Language, IEEE Computer Society. IEEE. New York. IEEE std 1364–2001.

2 IEEE Standard Verilog Hardware Description Language, IEEE Computer Society. IEEE. New York. IEEE std 1364–1995.

3 Bhasker J. A Verilog HDL Primer. Star Galaxy Publishing. 2008. 141–149.

4 Clifford C. Nonblocking Assignments in Verilog Synthesis, Coding Styles That Kill!. SNUG 2000 User Paper. 2000. section-MC1(1st paper).

5 Stuart S & Don M. Verilog and SystemVerilog Gotchas. USA: Springer. 2007. 62–72.

6 Janick B. Writing Testbenches, Functional Verification of HDL Models. USA: Kluwer Academic Publishers. 2000. 28–35.

(下转第 52 页)

2.40GHz、48G 内存、3T SATA 硬盘、千兆以太网、操作系统采用 Linux 2.6.18-8.el5PAE 内核，脚本采用 Linux bash shell 编程语言，版本 3.1.17，数据库为 MYSQL (版本 5.0.22)，前台网页系统为 PHP (版本 5.1.6).

我们在骨干运营网络进行实际部署，从骨干路由器上生成 Netflow 数据，并发给系统服务器，系统服务器经过数据预处理、数据筛选、存储、分析等处理操作，进行前端展示.

从图 4 可知，某时刻 139 和 445 端口发生扫描异常的 IP 地址，该地址扫描计数和扫描判别概率值明显偏高，并在系统中高亮预警显示，通过用 Netflow 数据的具体数据整理验证，该地址在该时间段内确实有大量的目的地址段扫描情况发生(如图 5)，经过后续现场排查工作发现，该主机感染了僵尸木马程序，正在利用微软 ms06-040 高危漏洞进行扫描探测并渗透传播，该系统的运行效果良好.

图 4　扫描判别概率结果

5　结语

本文将数值分析的方法应用到异常扫描行为的监测中，大大提高了海量网管数据对异常网络行为的筛查和甄别效率，提升了互联网运营单位的服务质量和

ID	源IP	目的IP	协议	源端口	目的端口	字节	包	流	包大小
12991	159.[redacted].85	222.56.118.22	6	1581	445	4800	100	100	48
12993	159.[redacted].85	222.56.118.22	6	1992	445	8300	100	100	83
12996	159.[redacted].85	210.67.99.19	6	2293	445	4800	100	100	48
12999	159.[redacted].85	210.22.195.34	6	3633	445	8300	100	100	83
13000	159.[redacted].85	222.56.118.22	6	1749	445	38800	100	100	388
13001	159.[redacted].85	121.52.49.151	6	2037	445	4800	100	100	48
13003	159.[redacted].85	222.56.118.23	6	2128	139	4800	100	100	48
13004	159.[redacted].85	222.56.118.23	6	3816	445	4000	100	100	40
13005	159.[redacted].85	210.22.195.34	6	4661	445	21600	100	100	216
13006	159.[redacted].85	61.10.206.40	6	1937	445	4800	100	100	48
13007	159.[redacted].85	210.75.203.5	6	4014	445	4800	100	100	48
13008	159.[redacted].85	210.3.239.66	6	1232	445	8300	100	100	83
13009	159.[redacted].85	210.3.239.66	6	3551	139	4800	100	100	48
13010	159.[redacted].85	114.92.121.213	6	2536	445	4800	100	100	48
13012	159.[redacted].85	210.3.239.66	6	2759	139	4800	100	100	48
13013	159.[redacted].85	222.56.118.23	6	3306	445	4800	100	100	48
13014	159.[redacted].85	222.56.118.23	6	1167	139	4000	100	100	40
13015	159.[redacted].85	114.92.37.239	6	2453	445	4800	100	100	48
13016	159.[redacted].85	210.3.239.66	6	4077	445	8300	100	100	83
13019	159.[redacted].85	210.78.57.127	6	2795	445	4800	100	100	48
13020	159.[redacted].85	114.92.147.7	6	4599	445	4800	100	100	48
13021	159.[redacted].85	210.22.195.34	6	3759	445	4000	100	100	40
13024	159.[redacted].85	210.75.25.5	6	2603	445	4800	100	100	48
13026	159.[redacted].85	114.92.209.149	6	3371	445	4800	100	100	48
13027	159.[redacted].85	24.76.10.198	6	1801	445	2160	10	10	216
13028	159.[redacted].85	75.236.25.237	6	2011	445	480	10	10	48
13029	159.[redacted].85	20.33.98.245	6	1753	445	480	10	10	48
13030	159.[redacted].85	215.98.183.109	6	1769	445	480	10	10	48

图 5　疑似扫描地址通信地址对连接情况

支撑保障能力，但系统目前采用的数值分析函数仅为线性函数，且对权重的考量还需进行深入的研究，仍需在未来研究工作中进行完善和充实.

参考文献

1 CNNIC.第 31 次“中国互联网络发展状况统计报告”.http://www.cnnic.cn,2013,5.

2 CNCERT.“2012 年我国互联网网络安全态势综述”.http://www.cert.org.cn,2013:2–9.

3 李军,曹文君,李扬.FB-NBAS:一种基于流的网络行为分析模型.计算机工程,2008,34(3):165–167.

4 牛国林,管晓宏,龙毅,秦涛.多源流量特征分析方法及其在异常检测中的应用.解放军理工大学学报(自然科学版),2009,10(4):350–355.

5 杨奇.基于异常行为特征的僵尸网络检测方法研究[学位论文].西安:陕西师范大学,2010.

(上接第 18 页)

7 Cliff C. Correct Methods For Adding Delays To Verilog Behavioral Models. International HDL Conference 1999 Proceedings. 1999. 23–29.

8 Navabi Z. Verilog Digital System Design: Register Transfer Level Synthesis, Testbench, and Verification. 2nd ed., USA: McGRAW-Hill Companies, 2007: 83–85.

9 Samir P. Verilog HDL: A Guide To Digital Design and Synthesis. USA: PEARSON Education. 2003. 82–85.

10 刘波.精通 Verilog HDL 语言编程.北京:电子工业出版社,2007.247–250.