在 RTL 代码中我们知道如果表达组合逻辑时使用“=”赋值，表达时序逻辑时使用“<=”赋值，如果我们不按照这种规则来设计往往会得到意想不到的答案。

虽然说在 Testbench 中我们对赋值号的要求并不是很在意，使用“=”和“<=”赋值均可，都能够仿真出来结果，且最后不会被综合成实际的电路，不会影响功能。网络上的各种资料教程也各有不同的写法，难道在 Testbench 中随便使用“=”和“<=”赋值真的对测试没有任何影响吗？经过下面的测试验证我们得到了出乎意料的答案。

被测试测 RTL 代码：一个简单的两输入 1bit 数据相与后通过寄存器输出

```
//-----------------------------------------------------
01  module  test(
02      input   wire    sys_clk     ,
03      input   wire    sys_rst_n   ,
04      input   wire    in1         ,
05      input   wire    in2         ,
06
07      output  reg     out
08  );
09
10  always@(posedge  sys_clk or negedge sys_rst_n)
11      if(sys_rst_n == 1'b0)
12          out <= 1'b0;
13      else
14          out <= in1 & in2;
15
16  endmodule
//-----------------------------------------------------
```

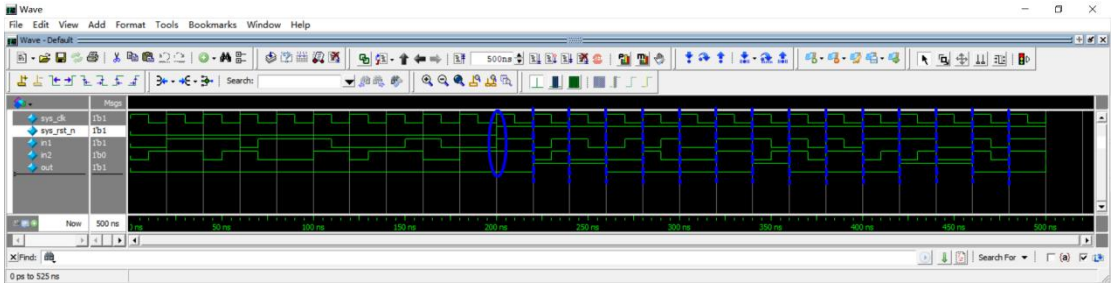
一、时钟初始值为 1’b1，仿真时间为 500ns

1、时钟用“=”赋值，输入信号用“<=”赋值(correct)

```
//-----------------------------------------------------
01  `timescale  1ns/1ns
02
03  module  tb_test();
04
05  reg     sys_clk;
06  reg     sys_rst_n;
07  reg     in1;
08  reg     in2;
09
10  wire    out;
```

```
11
12  //初始化
13  initial begin
14      sys_clk       =  1'b1;
15      sys_rst_n    <=  1'b0;
16      in1          <=  1'b0;
17      in2          <=  1'b0;
18      #200
19      sys_rst_n    <=  1'b1;
20  end
21
22  //产生 50Mhz 的时钟
23  always #10 sys_clk = ~sys_clk;
24
25  always #10 in1 <= {$random};
26  always #10 in2 <= {$random};
27
28  //------------------------test_isnt------------------------
29  testtest_inst(
30      .sys_clk     (sys_clk     ),  //input     sys_clk
31      .sys_rst_n   (sys_rst_n   ),  //input     sys_rst_n
32      .in1         (in1         ),  //input     in1
33      .in2         (in2         ),  //input     in2
34
35      .out         (out         )   //output    out
36  );
37
38  endmodule
//-----------------------------------------------------
```

图中蓝色虚线位置后的一段波形是正确的，蓝色椭圆处为复位结束标志。我们通过观察分析，仿真结果同 RTL 逻辑代码实现的功能一致。



同时还测试了该情况下的“|”、“+”、“^”运算均正确。

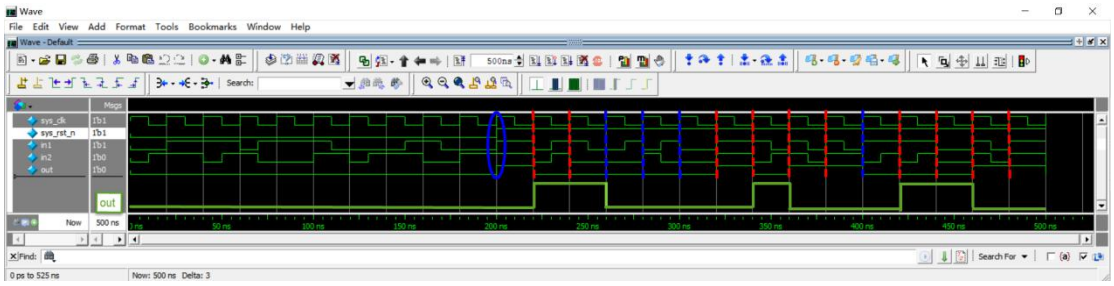
2、时钟和输入信号都用“<=”赋值(error)

```
//-------------------------------------------------
01  `timescale  1ns/1ns
```

```
02
03  module   tb_test();
04
05  reg      sys_clk;
06  reg      sys_rst_n;
07  reg      in1;
08  reg      in2;
09
10  wire     out;
11
12  //初始化
13  initial begin
14      sys_clk      <=  1'b1;
15      sys_rst_n    <=  1'b0;
16      in1          <=  1'b0;
17      in2          <=  1'b0;
18      #200
19      sys_rst_n    <=  1'b1;
20  end
21
22  //产生 50Mhz 的时钟
23  always #10 sys_clk <= ~sys_clk;
24
25  always #10 in1 <= {$random};
26  always #10 in2 <= {$random};
27
28  //------------------------test_isnt-----------------------
29  testtest_inst(
30      .sys_clk     (sys_clk     ),  //input      sys_clk
31      .sys_rst_n   (sys_rst_n   ),  //input      sys_rst_n
32      .in1         (in1         ),  //input      in1
33      .in2         (in2         ),  //input      in2
34
35      .out         (out         )   //output     out
36  );
37
38  endmodule
//-----------------------------------------------------
```

图中蓝色虚线位置后的一段波形是正确的，红色虚线位置后的一段波形是错误的，蓝色椭圆处为复位结束标志。我们通过观察分析，仿真结果同 RTL 逻辑代码实现的功能不一致，有大量的错误。

下图中绿色为正确的 out 输出波形，和第一种仿真结果 out 输出完全一致。



同时还测试了该情况下的“|”、“+”、“^”运算均错误。

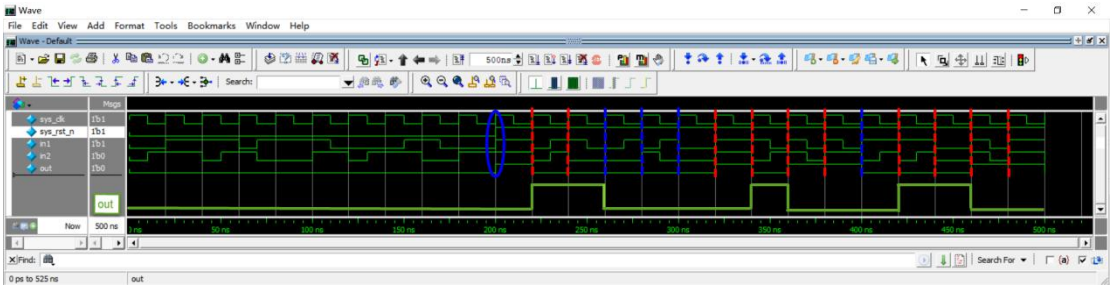
3、时钟和输入信号都用“=”赋值(error)

```
//------------------------------------------------
`timescale  1ns/1ns

module  tb_test();

reg     sys_clk;
reg     sys_rst_n;
reg     in1;
reg     in2;

wire    out;

//初始化
initial begin
    sys_clk     =   1'b1;
    sys_rst_n   =   1'b0;
    in1         =   1'b0;
    in2         =   1'b0;
    #200
    sys_rst_n   =   1'b1;
end

//产生 50Mhz 的时钟
always #10 sys_clk = ~sys_clk;

always #10 in1 = {$random};
always #10 in2 = {$random};

//----------------------test_isnt-----------------------
testtest_inst(
    .sys_clk    (sys_clk    ),  //input     sys_clk
    .sys_rst_n  (sys_rst_n  ),  //input     sys_rst_n
    .in1        (in1        ),  //input     in1
```

```
33      .in2        (in2        ),  //input     in2
34
35      .out        (out        )   //output    out
36  );
37
38  endmodule
//--------------------------------------------------
```

图中蓝色虚线位置后的一段波形是正确的，红色虚线位置后的一段波形是错误的，蓝色椭圆处为复位结束标志。我们通过观察分析，仿真结果同 RTL 逻辑代码实现的功能一致，有大量的错误。

下图中绿色为正确的 out 输出波形，和第一种仿真结果 out 输出完全一致。



同时还测试了该情况下的“|”、“+”、“^”运算均错误。

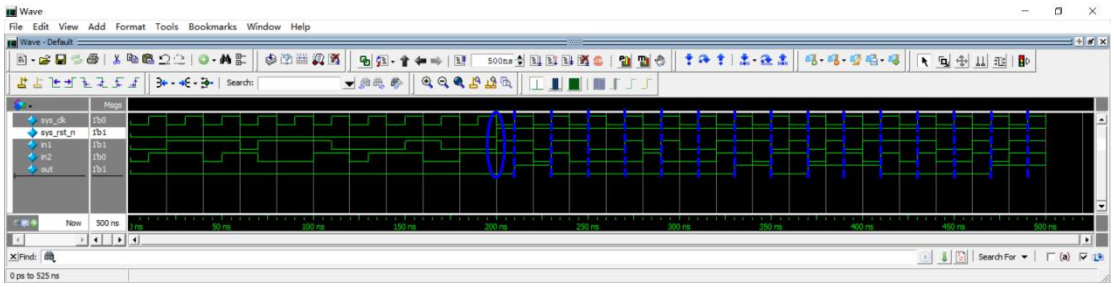
二、时钟初始值为 1’b0，仿真时间为 500ns

1、时钟用“=”赋值，输入信号用“<=”赋值(correct)

```
//--------------------------------------------------
01  `timescale  1ns/1ns
02
03  module  tb_test();
04
05  reg     sys_clk;
06  reg     sys_rst_n;
07  reg     in1;
08  reg     in2;
09
10  wire    out;
11
12  //初始化
13  initial begin
14      sys_clk     =  1'b1;
15      sys_rst_n   <=  1'b0;
16      in1         <=  1'b0;
17      in2         <=  1'b0;
18      #200
19      sys_rst_n   <=  1'b1;
```

```
20  end
21
22  //产生 50Mhz 的时钟
23  always #10 sys_clk = ~sys_clk;
24
25  always #10 in1 <= {$random};
26  always #10 in2 <= {$random};
27
28  //----------------------test_isnt-----------------------
29  testtest_inst(
30      .sys_clk     (sys_clk     ),  //input     sys_clk
31      .sys_rst_n   (sys_rst_n   ),  //input     sys_rst_n
32      .in1         (in1         ),  //input     in1
33      .in2         (in2         ),  //input     in2
34
35      .out         (out         )   //output    out
36  );
37
38  endmodule
//-----------------------------------------------------
```

图中蓝色虚线位置后的一段波形是正确的，蓝色椭圆处为复位结束标志。我们通过观察分析，仿真结果同 RTL 逻辑代码实现的功能一致。



同时还测试了该情况下的“|”、“+”、“^”运算均正确。

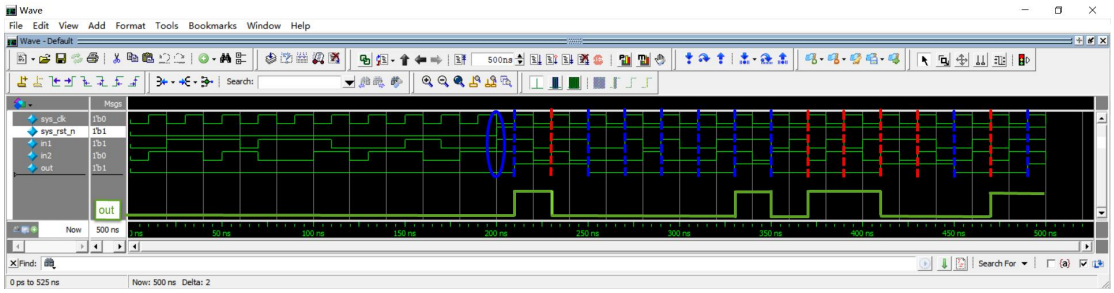
2、时钟和输入信号都用“<=”赋值(error)

```
//-----------------------------------------------------
01  `timescale  1ns/1ns
02
03  module  tb_test();
04
05  reg     sys_clk;
06  reg     sys_rst_n;
07  reg     in1;
08  reg     in2;
09
10  wire    out;
```

```
11
12  //初始化
13  initial begin
14      sys_clk      <=  1'b1;
15      sys_rst_n    <=  1'b0;
16      in1          <=  1'b0;
17      in2          <=  1'b0;
18      #200
19      sys_rst_n    <=  1'b1;
20  end
21
22  //产生 50Mhz 的时钟
23  always #10 sys_clk <= ~sys_clk;
24
25  always #10 in1 <= {$random};
26  always #10 in2 <= {$random};
27
28  //------------------------test_isnt------------------------
29  testtest_inst(
30      .sys_clk     (sys_clk     ),  //input     sys_clk
31      .sys_rst_n   (sys_rst_n   ),  //input     sys_rst_n
32      .in1         (in1         ),  //input     in1
33      .in2         (in2         ),  //input     in2
34
35      .out         (out         )   //output    out
36  );
37
38  endmodule
//------------------------------------------------------
```

图中蓝色虚线位置后的一段波形是正确的，红色虚线位置后的一段波形是错误的，蓝色椭圆处为复位结束标志。我们通过观察分析，仿真结果同 RTL 逻辑代码实现的功能不一致，有大量的错误。

下图中绿色为正确的 out 输出波形，和第一种仿真结果 out 输出完全一致。



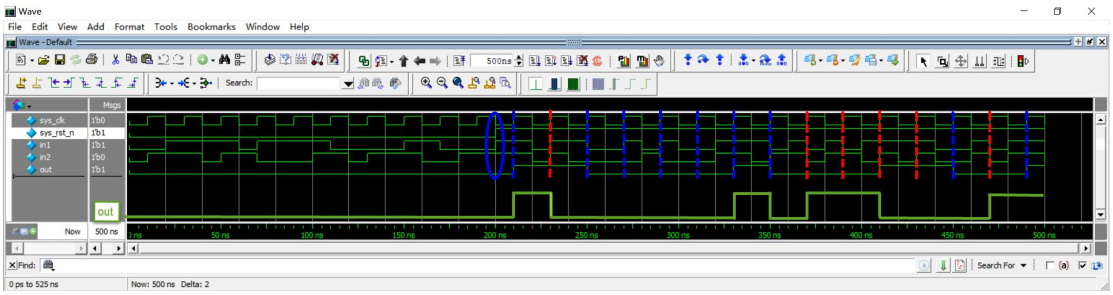
同时还测试了该情况下的“|”、“+”、“^”运算均错误。

3、时钟和输入信号都用“=”赋值(error)

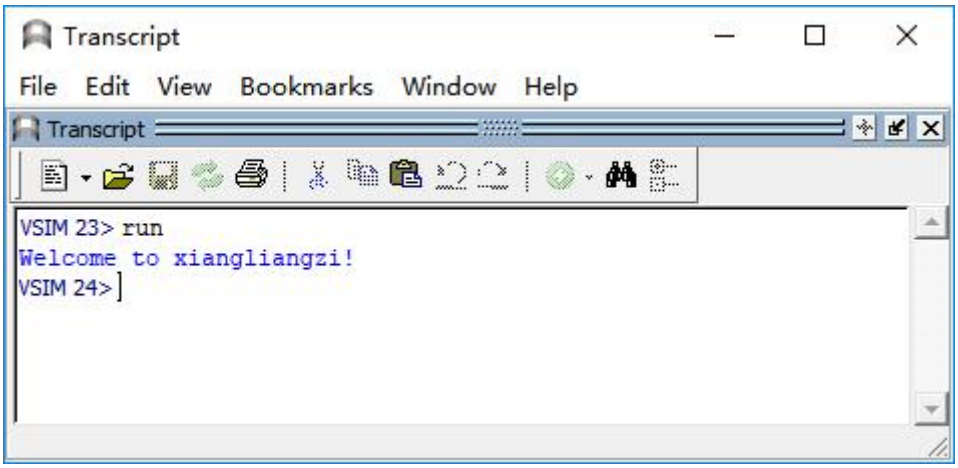
```
//-------------------------------------------------
01  `timescale  1ns/1ns
02
03  module  tb_test();
04
05  reg     sys_clk;
06  reg     sys_rst_n;
07  reg     in1;
08  reg     in2;
09
10  wire    out;
11
12  //初始化
13  initial begin
14      sys_clk     =   1'b1;
15      sys_rst_n   =   1'b0;
16      in1         =   1'b0;
17      in2         =   1'b0;
18      #200
19      sys_rst_n   =   1'b1;
20  end
21
22  //产生 50Mhz 的时钟
23  always #10 sys_clk = ~sys_clk;
24
25  always #10 in1 = {$random};
26  always #10 in2 = {$random};
27
28  //-----------------------test_isnt-----------------------
29  testtest_inst(
30      .sys_clk    (sys_clk    ),  //input     sys_clk
31      .sys_rst_n  (sys_rst_n  ),  //input     sys_rst_n
32      .in1        (in1        ),  //input     in1
33      .in2        (in2        ),  //input     in2
34
35      .out        (out        )   //output    out
36  );
37
38  endmodule
//-------------------------------------------------
```

图中蓝色虚线位置后的一段波形是正确的，红色虚线位置后的一段波形是错误的，蓝色椭圆处为复位结束标志。我们通过观察分析，仿真结果同 RTL 逻辑代码实现的功能不一致，有大量的错误。

下图中绿色为正确的 out 输出波形，和第一种仿真结果 out 输出完全一致。



同时还测试了该情况下的“|”、“+”、“^”运算均错误。



总结：所以我们推荐在写 Testbench 的时候时钟用“=”赋值，输入信号用“<=”赋值才能够避免这种情况，这种问题的根源其实是产生的数据没有同步时钟的原因，导致产生一些错乱，这种情况在 System Verilog 中就不会差生，这也是 System Verilog 更适合作为仿真验证语言的原因之一。至于时钟的初始值是 0 还是 1 对仿真的正确性影响不大，但是推荐大家把时钟的初始值幅值为 1，方便数据的变化都是在时钟的上升沿进行，和我们的 RTL 代码更接近。