



第10章

Verilog HDL仿真及相关语句

10.1 Verilog HDL仿真流程



图 10-1 HDL 系统设计描述层次

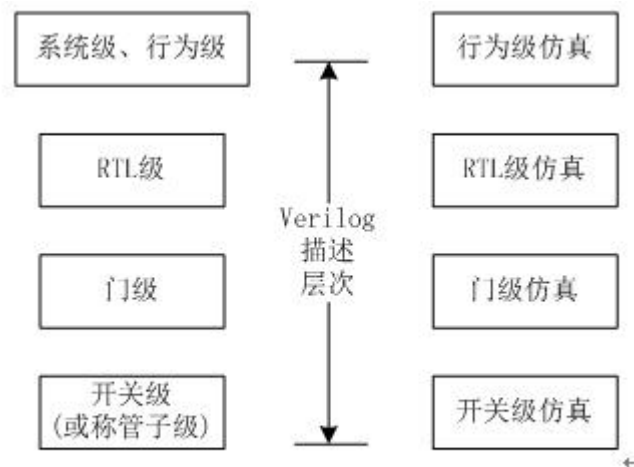


图 10-2 Verilog 描述层次与对应的仿真层次

10.1 Verilog HDL仿真流程

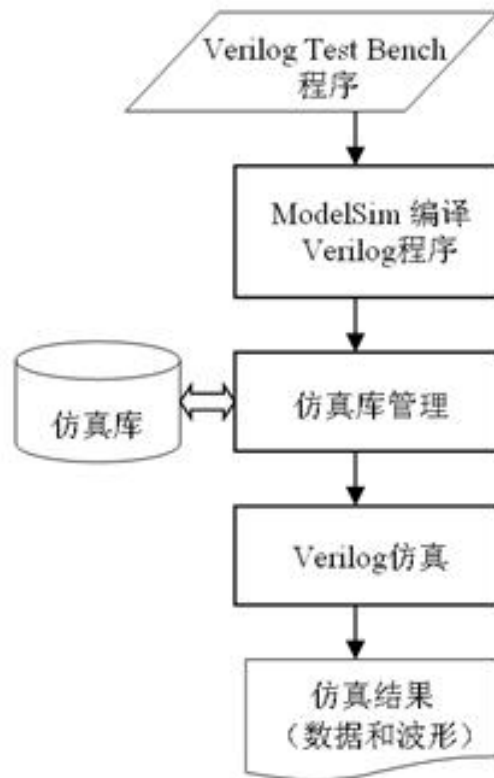


图 10-3 Verilog 仿真流程

10.2 Verilog 测试基准示例



图 10-4 Verilog Test Bench 结构

【例 10-1】 //Test Bench 文件名: CNT10_TB.v

```
`timescale 10ns/1ns //Test Bench 中, 此仿真时间标度语句必须存在
module CNT10_TB (); //注意此 module 不必给出端口描述
    reg clk, en, rst, load; reg [3:0] data ;//定义激励信号的数据类型是 reg
    wire [3:0] dout ; wire cout ; //定义激励信号的数据类型是 reg
    always #3 clk=~clk;//产生时钟的语句, 每隔 3 个时间单元, 即 30ns, clk 翻转一次。
    initial
    $monitor ("DOUT=%h",dout); //以十六进制形式打印待测模块 DOUT 的输出数据
    initial begin //一次性过程语句
        #0 clk=1'b0; //0 时间单元时, 设定 clk 电平是 0
        #0 rst=1'b1; #20 rst=1'b0; #2 rst=1'b1; //注意, 是顺序赋值语句
    end
    initial begin
        #0 en = 1'b0; #5 en = 1'b1;
    end
    initial begin
        #0 load=1'b1; #49 load=1'b0; #3 load=1'b1;
    end
    initial begin
        #0 data=4'h7; #30 data=4'h2; #30 data=4'h5; #30 data=4'h4;
    end
    CNT10 U1 (.CLK(clk), .RST(rst), .DATA(data), .LOAD(load),
        .EN(en), .COUT(cout), .DOUT(dout)); //例化语句
endmodule
```

10.3 Verilog Test Bench测试流程

1. 安装ModelSim

2. 为Test Bench仿真设置参数

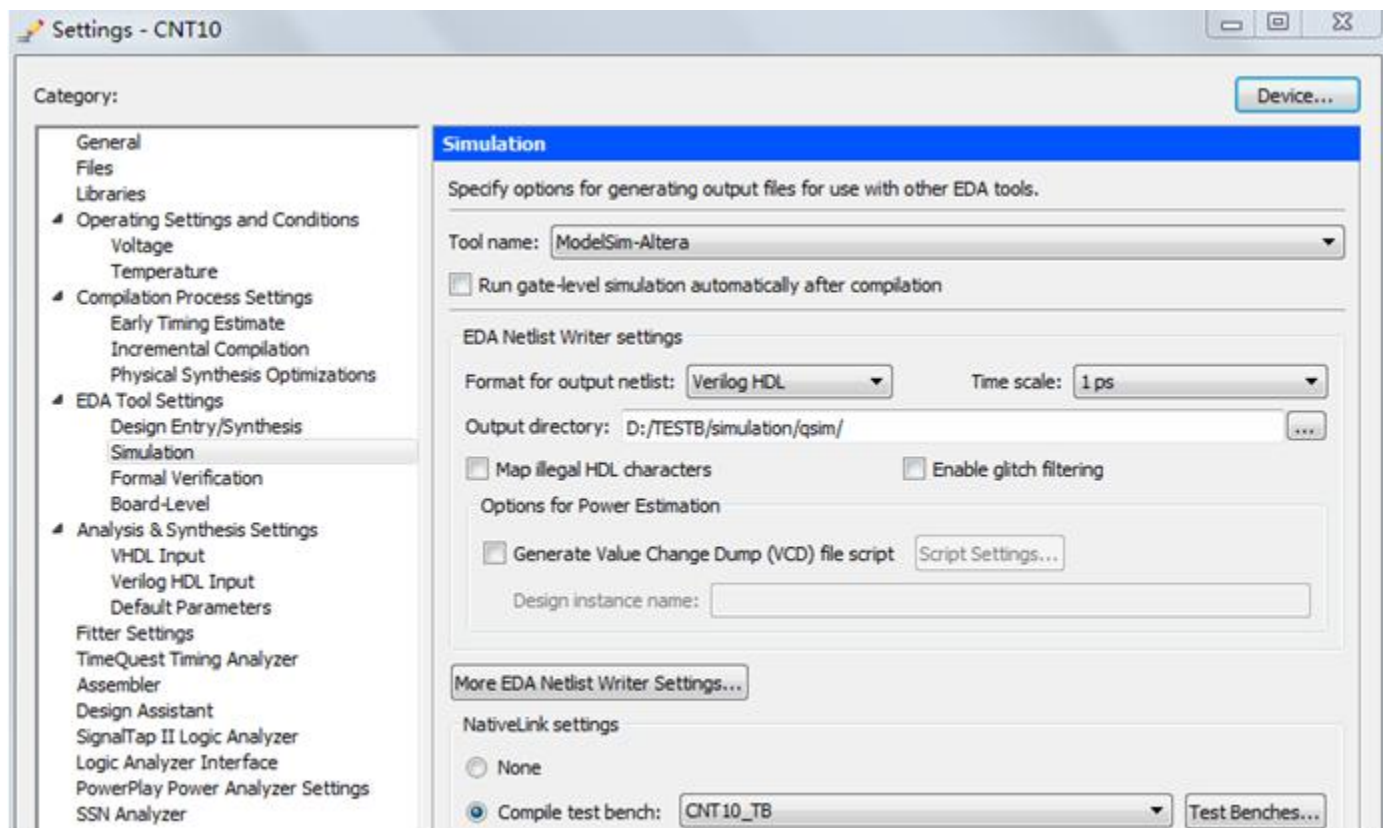


图 10-5 选择仿真工具名称和输出网表语言形式

10.3 Verilog Test Bench测试流程

2. 为Test Bench仿真设置参数

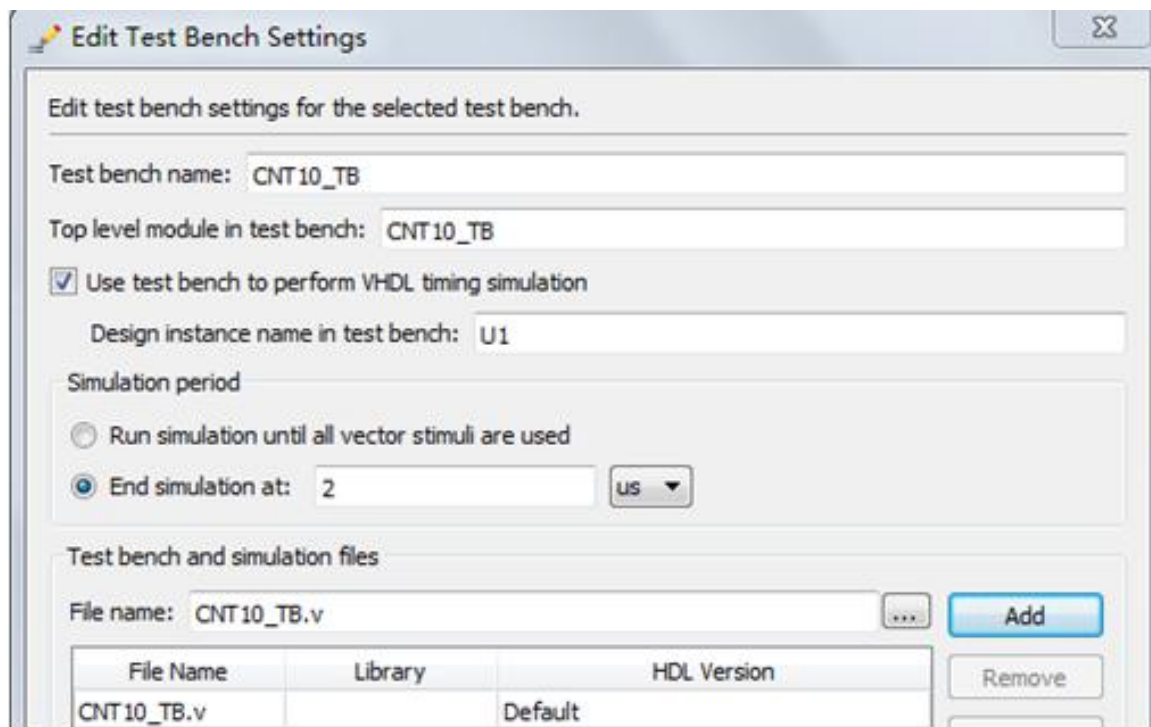


图 10-6 为 Test Bench 仿真设置参数

10.3 Verilog Test Bench测试流程

- 3. 启动Test Bench仿真
- 4. 分析Test Bench仿真结果

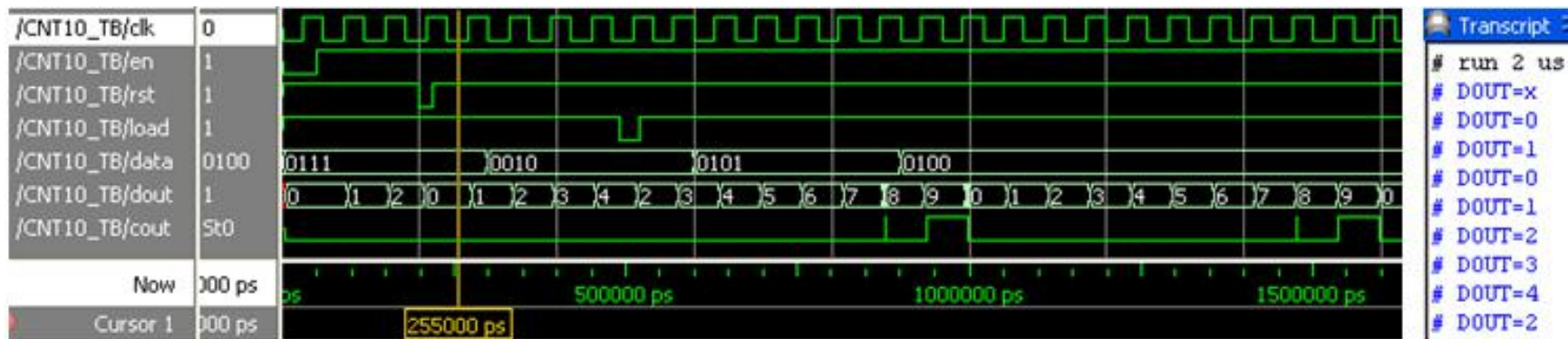


图 10-7 Test Bench 输出的仿真波形及数据（右侧）

10.4 Verilog系统任务和系统函数

10.4.1 系统任务和系统函数

1. \$display

```
$display("带格式字符串", 参数1, 参数2, ... );
```

【例 10-2】

```
module sdisp1;  
  integer i;    // i为整型  
  reg [3:0] x;  // x为4位  
  initial begin // initial块, 只执行一次。  
    i=21;      x=4'he;  
    $display("1\t%d\n2\t%h\\", i, x); // 输出显示  
  end endmodule
```

10.4 Verilog系统任务和系统函数

10.4.1 系统任务和系统函数

1. \$display

表 10-1 Verilog 转义符

| 转义符 | 含义 | 转义符 | 含义 | 转义符 | 含义 |
|------|----------------------|-----|-----------|-----|----------------|
| \n | 换行 | %b | 二进制格式 | %v | 显示信号强度 |
| \t | Tab | %o | 八进制格式 | %m | 显示层次名 |
| \\ | 字符\ | %d | 十进制格式 | %s | 字符串格式 |
| \" | 字符" | %h | 十六进制格式 | %t | 显示当前时间 |
| \ddd | 1~3 位八进制表示的 ASCII 字符 | %l | 显示：库绑定信息 | %u | 未格式化二值数据 |
| %% | 字符% | %c | 字符格式 | %z | 未格式化四值数据 |
| %e | 以科学计数法显示实数值 | %f | 以十进制显示实数值 | %g | 取%e、%f格式中最短的显示 |

10.4 Verilog系统任务和系统函数

10.4.1 系统任务和系统函数

1. \$display

【例 10-3】

```
module sdisp2;          // 注意无输入输出端口
reg [31:0] rval;       // 32 位 reg 类型
pulldown (pd);        // pd 接下拉电阻, plldown 用法见本章后续内容
initial begin         // initial 块
rval = 101;           // 赋整数 101
$display("rval = %h hex %d decimal",rval,rval); // 十六进制、十进制显示
$display("rval = %o octal\nrval = %b bin",rval,rval); // 八进制、二进制显示
$display("rval has %c ascii character value",rval); // 字符格式显示输出
$display("pd strength value is %v",pd);           // pd 信号强度显示
$display("current scope is %m");                 // 当前层次模块名显示
$display("%s is ascii value for 101",101);       // 字符串显示
$display("simulation time is %t", $time);        // 显示当前仿真时间
end
endmodule
```


10.4 Verilog系统任务和系统函数

10.4.1 系统任务和系统函数

2. \$write

```
$write ("带格式字符串", 参数 1, 参数 2, ... );
```

3. \$strobe和\$monitor

```
$strobe ("带格式字符串", 参数 1, 参数 2, ... );  
$monitor ("带格式字符串", 参数 1, 参数 2, ... );
```

10.4 Verilog系统任务和系统函数

10.4.1 系统任务和系统函数

3. \$strobe和\$monitor

【例 10-4】

```
module sdisp3;                                // 无输入输出信号
reg [1:0]a;                                   // a为2位reg
reg b;
initial $monitor("\$monitor: a = %b", a);     // $monitor 监测 a 的变化
initial begin                                 // initial 块, 只执行一次
b = 0;  a = 0;                                // b、a 赋值 0, 阻塞赋值
$strobe ("\$strobe : a = %b", a);            // $strobe 显示 a 的赋值
a = 1;                                        // a 赋值 1
$display ("\$display: a = %b", a);           // $display 显示 a 的当前赋值
a = 2;                                        // a 赋值 2
$monitor("\$monitor: b = %b", b);           // $monitor 取代前一个$monitor
a = 3;                                        // a 赋值 3
#30 $finish;                                  // 延时 30 个时间单位后, 仿真终止
end
always #10 b = ~b;                            // b 每隔 10 个时间单位, 值反转, Clock 信号
endmodule
```

10.4 Verilog系统任务和系统函数

10.4.1 系统任务和系统函数

3. \$strobe和\$monitor

```
# $display: a = 01  
# $strobe : a = 11  
# $monitor: b = 0  
# $monitor: b = 1  
# $monitor: b = 0
```

10.4 Verilog系统任务和系统函数

10.4.1 系统任务和系统函数

4. \$finish和 \$stop

```
$finish;
```

```
$stop;
```

【例 10-5】

```
module sdisp4();  
    reg [3:0]a,b;    // a, b 都为 4 位 reg  
    initial $monitor($time," \ $monitor:a=%0d,b=%d",a,b); //显示变化及当前时间  
    initial begin    // initial 块, 只执行一次  
        b = 0;        // b 赋值 0  
        $strobe ($time," \ $strobe : a = %0d", a);    // 显示 a 的赋值结果  
        $monitoron;    // 开启 $monitor  
        a = 1;        // a 赋值 1, 阻塞赋值  
        a <= 2;        // a 赋值 2, 非阻塞赋值  
        $display ($time," \ $display: a = %d", a);    // 显示 a 的当前值  
        a = 3;        // a 赋值 3, 阻塞赋值  
        #25 $monitoroff; // 关闭 $monitor  
        #10 $stop;    // 10 个时间单位后, 暂停仿真器仿真  
    end  
    always #10 b = b+1; // b 每过 10 个时间单位, 加 1  
endmodule
```


10.4 Verilog系统任务和系统函数

10.4.1 系统任务和系统函数

4. \$finish和 \$stop

```
#           0 $display: a = 1
#           0 $strobe : a = 2
#           0 $monitor:a=2,b= 0
#          10 $monitor:a=2,b= 1
#          20 $monitor:a=2,b= 2
```

5. \$time

```
$time           //返回一个64位整数时间值
$time          //返回一个32位整数时间值
$realttime     //返回一个实数时间值↵
$timeformat    //控制时间的显示方式↵
```

10.4 Verilog系统任务和系统函数

10.4.1 系统任务和系统函数

6. 文件操作

```
文件句柄 = $fopen("文件名") //打开文件↵
$fstrobe(文件句柄, "带格式字符串", 参数列表) //strobe到文件 ↵
$fdisplay(文件句柄, "带格式字符串", 参数列表t) //display到文件 ↵
$fmonitor(文件句柄, "带格式字符串", 参数列表t) //monitor到文件, 可以多个进程
$fwrite(文件句柄, "带格式字符串", 参数列表) //write到文件↵
$fclose(文件句柄); //关闭文件↵
$feof(文件句柄); //查询是否已到文件末尾↵
```

10.4 Verilog系统任务和系统函数

10.4.1 系统任务和系统函数

6. 文件操作

```
$dumpfile("文件名"); //导出到文件, 这里文件后缀为vcd↵
$dumppvar; //导出当前设计的所有变量↵
$dumppvar(1, top); //导出顶层模块中的所有变量↵
$dumppvar(2, top); //导出顶层模块和顶层下第1层模块的所有变量↵
$dumppvar(n, top); //导出顶层模块到顶层下第n-1层模块的所有变量↵
$dumppvar(0, top); //导出顶层模块和所有层次模块的所有变量↵
$dumpon; //导出初始化↵
$dumppoff; //停止导出↵
```

10.4 Verilog系统任务和系统函数

10.4.1 系统任务和系统函数

6. 文件操作

【例 10-6】

```
module fileio_demo;           // 文件读写
integer fp_r, fp_w, cnt;     // 定义文件句柄, 整型
reg [7:0] reg1, reg2, reg3; // 3个8位reg值
initial begin
    fp_r = $fopen("in.txt", "r"); // 以只读方式打开 in.txt
    fp_w = $fopen("out.txt", "w");// 以写方式打开 out.txt
    while(!$feof(fp_r)) begin    // 循环读写文件, 直到 in.txt 末尾
        cnt = $fscanf(fp_r, "%d %d %d", reg1, reg2, reg3); // 读一行
        $display("%d %d %d", reg1, reg2, reg3);           // 显示读到的值
        $fwrite(fp_w, "%d %d %d\n", reg3, reg2, reg1);   // 反序写一行
    end
    $fclose(fp_r); // 关闭文件 in.txt
    $fclose(fp_w); // 关闭文件 out.txt
end
endmodule
```

10.4 Verilog系统任务和系统函数

10.4.2 预编译语句

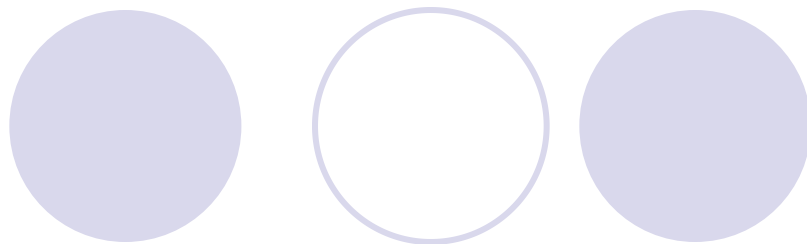
1. `define 宏定义

```
`define dnand(dly) nand #dly
`dnand(2) g121 (q21, n10, n11);
`dnand(5) g122 (q22, n10, n11);
```

2. translate_on与translate_off

```
//synthesis translate_off
//synthesis translate_on
```

10.5 延时模型



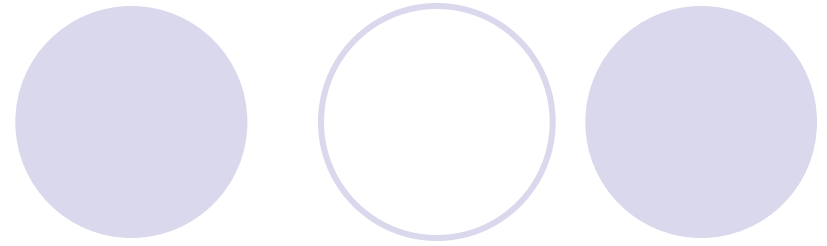
10.5.1 # 延时和门延时

- # 延时时间单位数。
- # (上升延迟, 下降延迟)。
- # (上升延迟, 下降延迟, 转换到 z 的延迟)。

【例 10-7】

```
module dnot ();
reg in; wire out;
not #(3,4) (out,in); // 例化 not, 同时说明门延时
initial begin
    $monitor ("%g in = %b out=%b", $time, in, out); // 监控 in、out
    in = 0; // 初始赋值 0
    #10 in = 1; // 10 个时间单位后, 赋值 1
    #10 in = 0; // 10 个时间单位后, 赋值 0
    #10 $stop; // 10 个时间单位后, 暂停仿真
end endmodule
```

10.5 延时模型



10.5.1 # 延时和门延时

```
# 0 in = 0 out=x  
# 3 in = 0 out=1  
# 10 in = 1 out=1  
# 14 in = 1 out=0  
# 20 in = 0 out=0  
# 23 in = 0 out=1
```

10.5 延时模型

10.5.2 延时说明块

【例 10-8】

```
module veridelay(output out, input a,b,c,d);  
wire e,f;  
specify          // specify 延时说明块  
    (a=>out)=3;  // a 到 out 延时 3 个时间单位  
    (b=>out)=3;  // b 到 out 延时 3 个时间单位  
    (c=>out)=5;  // c 到 out 延时 5 个时间单位  
    (d=>out)=51; // d 到 out 延时 51 个时间单位  
endspecify  
and U1(e,a,b);  and U2(f,c,d);  and U3(out,e,f); //例化 3 个元件  
endmodule
```


10.6 其他仿真语句

10.6.1 fork-join块语句

【例 10-9】

```
module forkA(clk,a,b);  
  input clk;  
  output reg a, b;  
  initial begin  
    a=0; b=0; end  
  always @(posedge clk)  
    fork  
      #30 a = 1;  
      #10 b = 1;  
    join  
endmodule
```

【例 10-10】

```
module forkB(clk,a,b);  
  input clk;  
  output reg a, b;  
  initial begin  
    a=0; b=0; end  
  always @(posedge clk)  
    begin  
      #30 a = 1;  
      #10 b = a;  
    end  
endmodule
```

【例 10-11】

```
module forkC(clk,a,b);  
  input clk;  
  output reg a, b;  
  initial begin  
    a=0; b=0; end  
  always @(posedge clk)  
    fork  
      #30 a = 1;  
      #10 b = a;  
    join  
endmodule
```

10.6 其他仿真语句

10.6.1 fork-join块语句

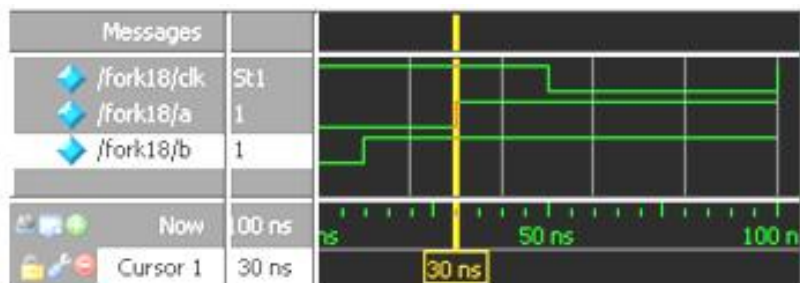


图 10-8 例 10-9 的仿真波形

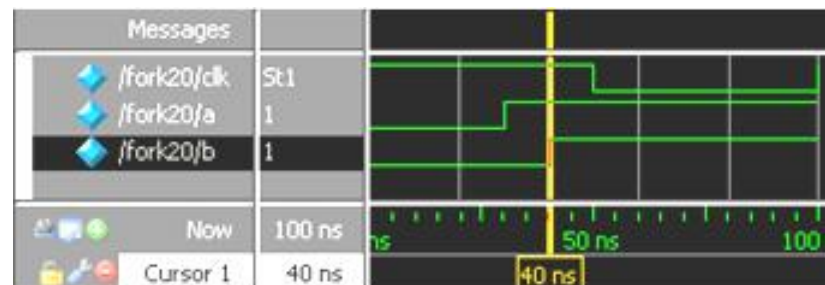


图 10-9 例 10-10 的仿真波形



图 10-10 例 10-11 的仿真波形

10.6 其他仿真语句

10.6.2 wait 语句

`wait (条件表达式) 语句;`

```
forever wait(start) #10 go = ~go;
```

10.6 其他仿真语句

10.6.3 force、release语句

【例 10-12】

```
module testforce; // force 语句测试示例
reg a, b, c, d;   wire e;
and and1 (e, a, b, c);
initial begin // 监控 d、e 的变化
$monitor("%d d=%b,e=%b", $stime, d, e);
assign d = a & b & c; // 连续赋值 d
a = 1; b = 0; c = 1;
#10; // 延迟 10 个时间单位
force d = (a | b | c); // 强制赋值 d
force e = (a | b | c); // 强制赋值 e
#10 $stop; // 暂停仿真
release d; // 释放 d
release e; // 释放 e
#10 $stop; // 暂停仿真
end
endmodule
```

```
#          0 d=0,e=0
#          10 d=1,e=1
```

```
#          20 d=0,e=0
```

10.6 其他仿真语句

10.6.4 deassign语句

```
always @(clear or preset)
    if (clear)
        assign q = 0;
    else if (preset)
        assign q = 1;
    else
        deassign q;
always @(posedge clock) q = d;
```

10.7 仿真激励信号的产生

【例 10-13】 // 4 位加法器

```
module adder4(input[3:0] a, input[3:0] b,  
              output reg[3:0] c, output reg co);  
always @*  
  {co,c} <= a + b;           // co 为进位, c 为和  
endmodule
```

10.7 仿真激励信号的产生

10.6.4 deassign语句

1. 方法一

【例 10-14】

```
`timescale 10ns/1ns // 时间设置
module signal_gen(output reg [3:0] sig1,output reg [3:0] sig2);
initial begin
    sig1 <= 4'd10; // 依序列出输入信号变化
    sig2 <= 4'd3;
    #10 sig2 <=4'd4;      #10 sig1 <=4'd11;      #10 sig2 <=4'd6;
    #10 sig1 <=4'd8;      #10 $stop;
end
endmodule
```

【例 10-15】

```
module test_adder4(); // 用于仿真的顶层文件
wire [3:0] a,b,c;    wire co;
adder4 U1(.a(a),.b(b),.c(c),.co(co)); // 例化被测元件 DUT
signal_gen TU1(.sig1(a),.sig2(b)); // 例化激励发生模块
endmodule
```

10.7 仿真激励信号的产生

10.6.4 deassign语句

2.方法二

```
force <信号名> <值> [<时间>][, <值> <时间> ...] [-repeat <周期>]
```

例如:

```
force a 0 (强制信号的当前值为 0)
```

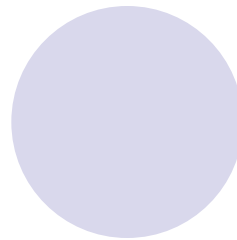
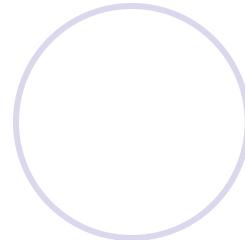
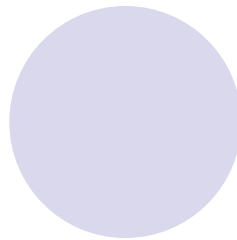
```
force b 0 0, 1 10 (强制信号 b 在时刻 0 的值为 0, 在时刻 10 的值为 1)
```

```
force clk 0 0, 1 15 -repeat 20 (clk 为周期信号, 周期为 20)
```

```
force a 10 0, 5 200, 8 400
```

```
force b 3 0, 4 100, 6 300
```


10.8 数字系统仿真





实验与设计

实验10-1 在ModelSim上对计数器的Test Bench进行仿真

实验10-2 在ModelSim上进行16位累加器设计仿真

【例 10-16】

```
module acc16( input [15:0] a, input rst, input clk,  
             output reg [15:0] c );  
    always @(posedge clk, negedge rst) if(!rst) c=0; else c=c+a;  
endmodule
```