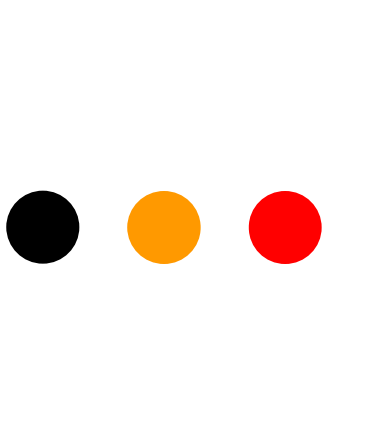

第9章

Verilog Test Bench仿真与时序分析

9.1 Verilog HDL仿真流程

图 9-1 HDL 系统设计描述层次

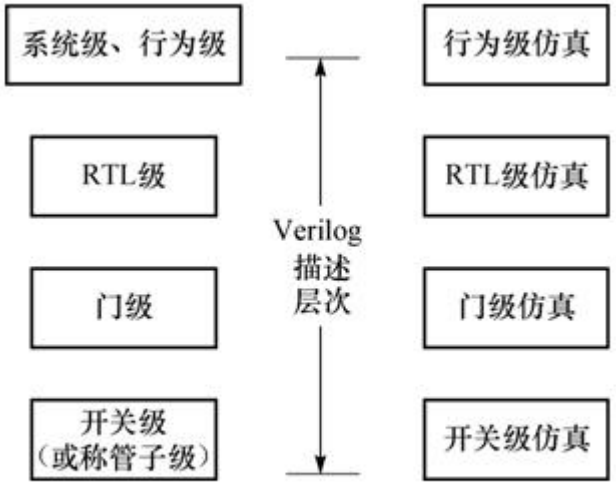

图 9-2 Verilog 描述层次与对应的仿真层次

9.1 Verilog HDL仿真流程

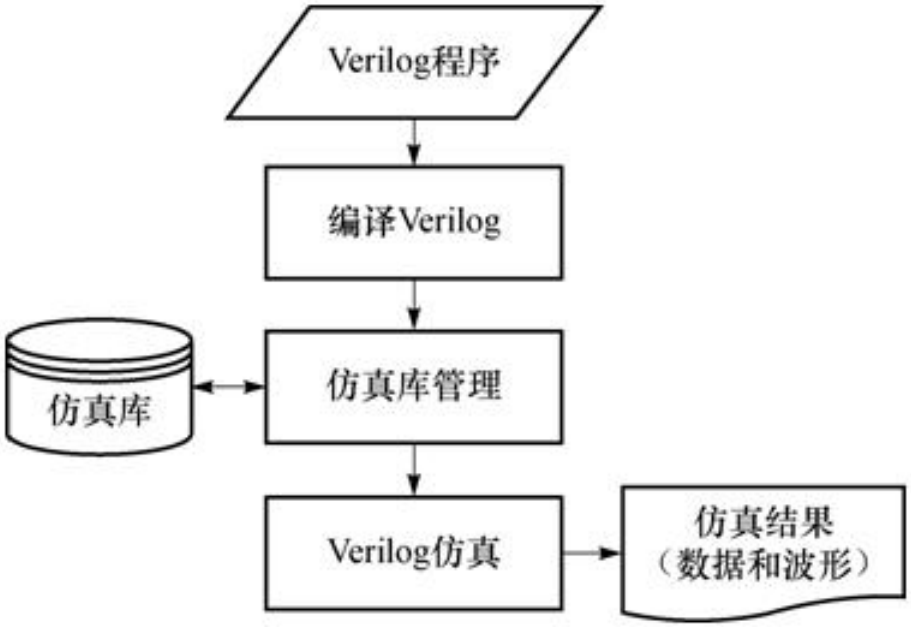

图 9-3　Verilog 仿真流程

9.2 Verilog HDL Test Bench仿真

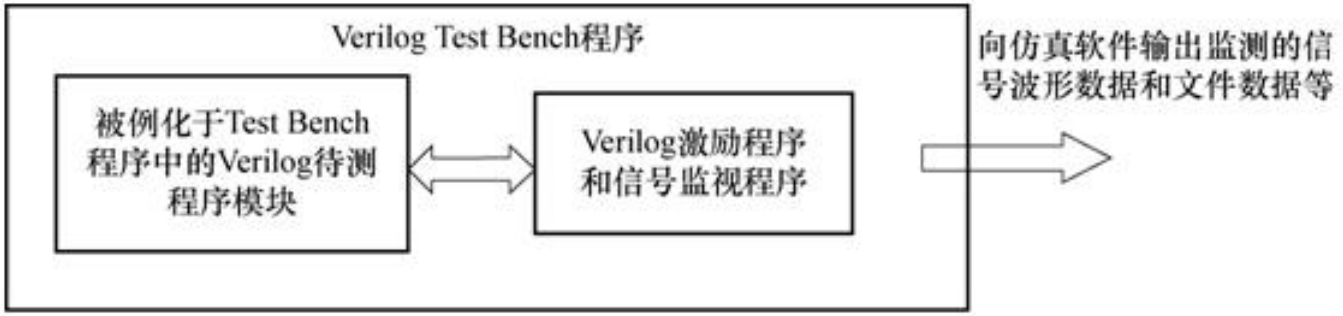

图 9-4　Verilog Test Bench 结构

9.2 Verilog HDL Test Bench仿真

【例 9-1】Test Bench 文件名：CNT10_TB.v

```
`timescale 10ns/1ns
module CNT10_TB ();
reg clk1, en1, rst1, load1; reg [3:0] data1;//定义输出的激励信号用 reg 类型
wire [3:0] dout1 ;  wire cout1 ; //定义来自被测模块输入信号的数据类型是 reg
always #20 clk1=~clk1; //产生时钟的语句，每隔 20 个单元，即 200ns,clk 翻转一次
initial
$monitor ("DOUT=%h",dout1); //以十六进制形式打印待测模块 DOUT1 的输出数据
initial begin                //一次性过程语句
 #0  clk1=1'b0;              //0 时间单元时，设定 clk1 电平是 0
 #0  rst1=1'b1;  #61 rst1=1'b0;  #20  rst1=1'b1; end //注意，是顺序赋值语句
initial begin
   #0   en1  = 1'b0;  #50   en1  = 1'b1;    end
initial begin
  #1  load1=1'b1; #240 load1=1'b0; #25  load1=1'b1;
                  #700 load1=1'b0; #30  load1=1'b1;   end
initial begin
#0  data1=4'h4; #70 data1=4'h8; #450 data1=4'h7; #500 data1=4'h8; end
CNT10 U1 (.CLK(clk1), .RST(rst1), .DATA(data1), .LOAD(load1),
         .EN(en1), .COUT(cout1), .DOUT(dout1));     //例化语句
endmodule
```

9.3 HDL仿真实例

1. 安装ModelSim

2. 为Test Bench仿真设置参数

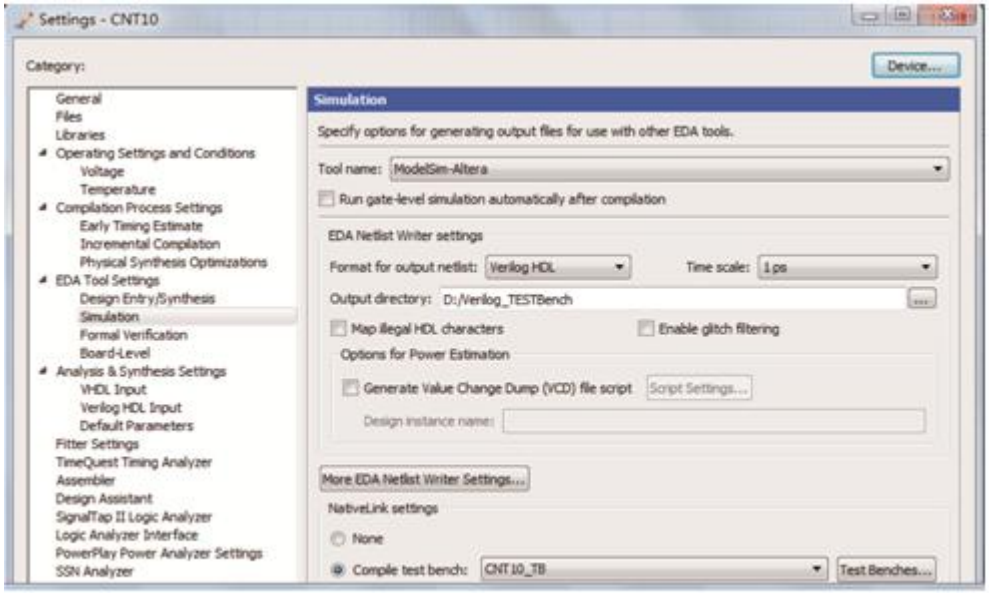

图 9-5　选择仿真工具名称和输出网表语言形式

9.3 HDL仿真实例

2．为Test Bench仿真设置参数

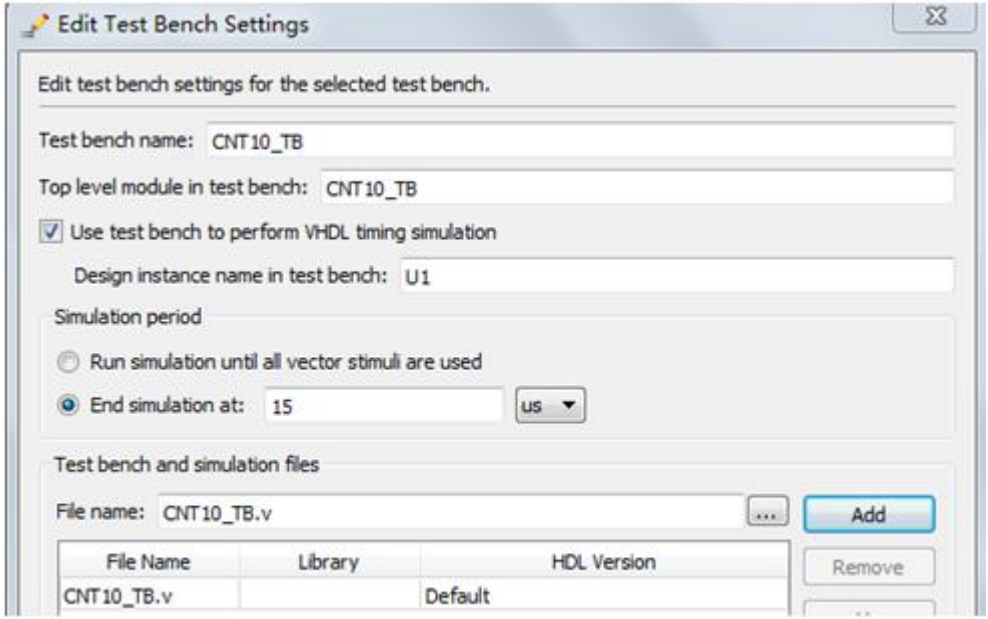

图 9-6　为 Test Bench 仿真设置参数

3．启动Test Bench仿真

9.3 HDL仿真实例

4. 分析Test Bench仿真结果

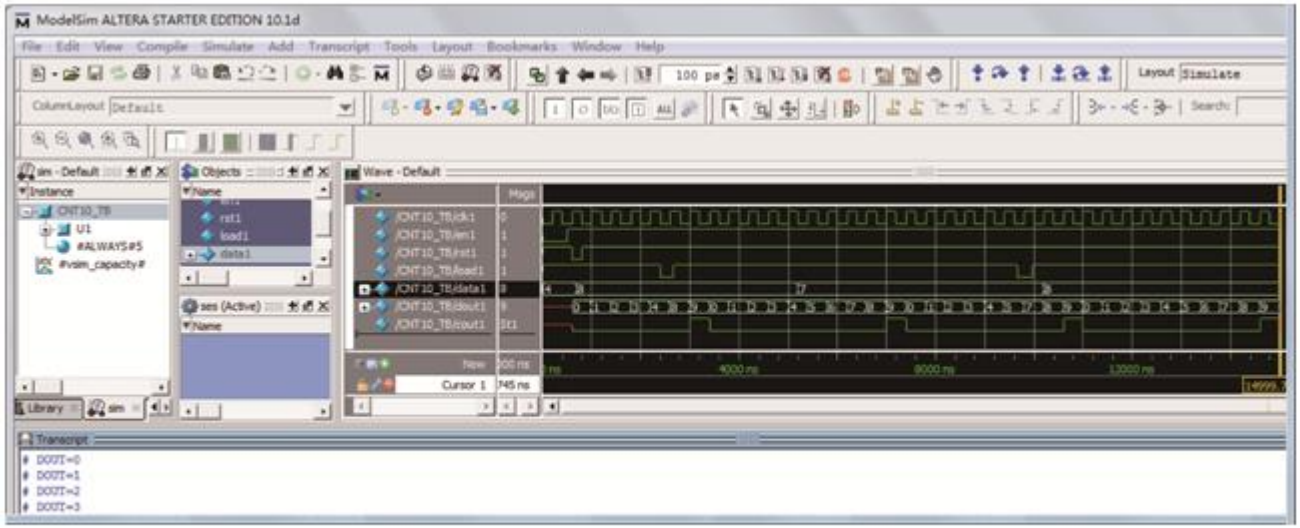

图 9-7　启动仿真后弹出的 ModelSim 界面（时序波形清晰画面如图 9-8 所示）

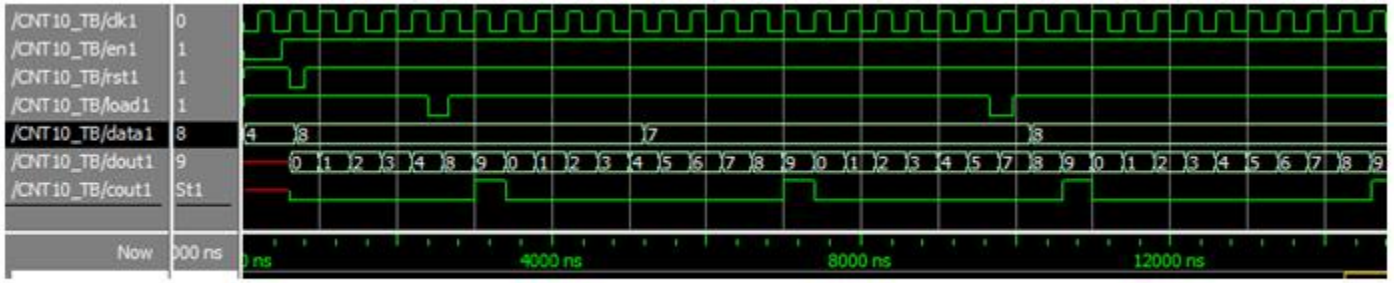

图 9-8　Test Bench 输出的仿真波形详图

9.3 HDL仿真实例

【例 9-2】

```
module CNT4B (CLR, EN, CLK, DATA, LOAD, UD, COUT, DOUT);
     input CLR,EN,CLK,LOAD, UD;
     input[3:0] DATA ;
     output COUT; output[3:0] DOUT;
     wire sub_wire0; wire[3:0] sub_wire1;
     wire COUT = sub_wire0;      wire[3:0] DOUT = sub_wire1[3:0];
   lpm_counter U1(
     .sload(LOAD), .clk_en(EN), .aclr(CLR),
     .data(DATA), .clock(CLK), .updown(UD),
     .cout(sub_wire0), .q(sub_wire1), .aload(1'b0),
     .aset(1'b0), .cin(1'b1), .cnt_en(1'b1),
     .eq(), .sclr(1'b0), .sset(1'b0));
  defparam
    U1.lpm_direction = "UNUSED",
    U1.lpm_modulus = 12,
    U1.lpm_port_updown = "PORT_USED",
    U1.lpm_type = "LPM_COUNTER",
    U1.lpm_width = 4;
endmodule
```

9.3 HDL仿真实例

【例 9-3】

```
`timescale 10ns/1ns
 module CNT4B_TB ();
    reg clk1, en1, clr1, load1,ud1; reg[3:0] data1 ;
    wire[3:0] dout1;  wire cout1;
 always #35 clk1=~clk1;
 initial
$monitor ("DOUT=%h",dout1);
initial begin
    #0  clk1=1'b0;
    #0  clr1=1'b0; #200 clr1=1'b1;  #25  clr1=1'b0;  end
initial begin
    #0  en1=1'b0;  #180   en1=1'b1;    end
initial begin
    #0  ud1=1'b1;  #1500   ud1=1'b0;   end
initial begin
   #0  load1=1'b0;  #590  load1=1'b1 ; #35  load1=1'b0;
                    #600  load1=1'b1 ; #39 load1=1'b0;
                    #1100 load1=1'b1 ; #60 load1=1'b0;  end
initial begin
#0  data1=4'h8; #800 data1=4'h7; #800 data1=4'h3; #900 data1=4'h9; end
CNT10 U1(.CLK(clk1), .CLR(clr1), .DATA(data1), .LOAD(load1), .UD(ud1),
          .EN(en1), .COUT(cout1), .DOUT(dout1) );
endmodule
```

9.3 HDL仿真实例

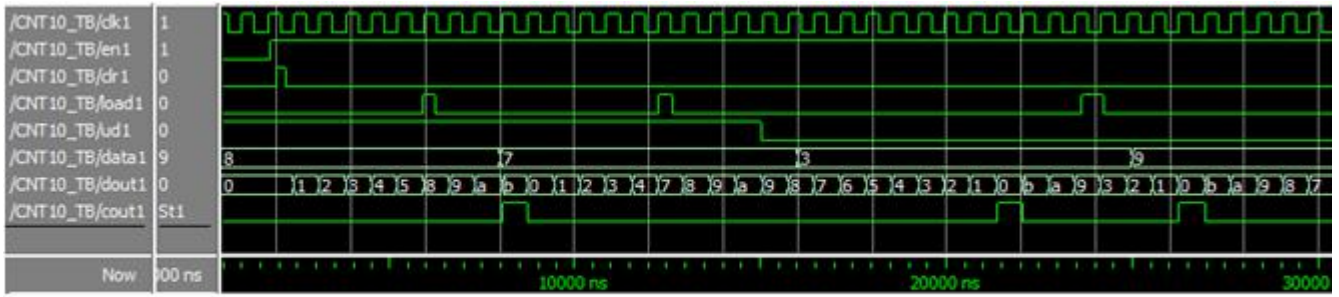

图 9-9　ModelSim（直接）输出的 Test Bench 仿真波形

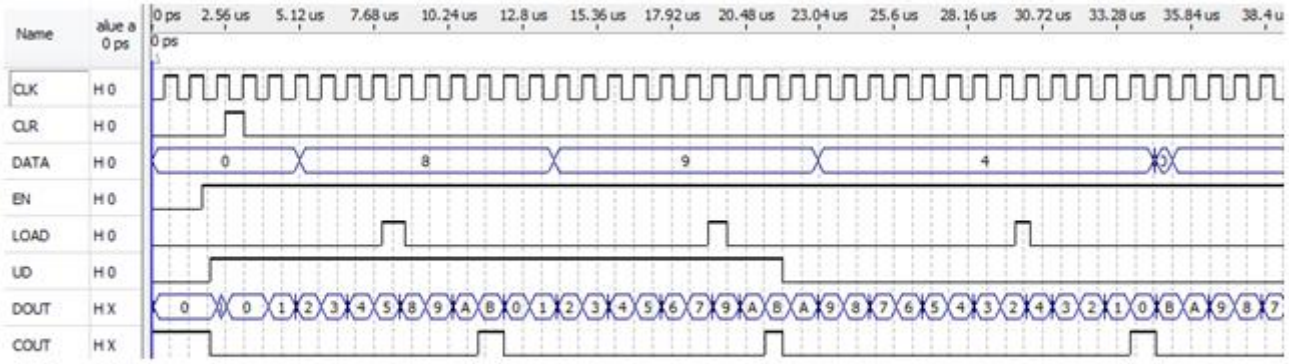

图 9-10　波形仿真器的仿真波形（ModelSim 间接输出波形）

9.4 Verilog系统任务和系统函数

9.4.1 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 9-4】

```
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
```

```
# 1          21
# 2  e\
```

9.4 Verilog系统任务和系统函数

9.4.1 系统任务和系统函数

1. $display

表 9-1 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

9.4 Verilog系统任务和系统函数

9.4.1 系统任务和系统函数

1. $display

【例 9-5】

```
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
```

9.4 Verilog系统任务和系统函数

9.4.1 系统任务和系统函数

1. $display

```
# rval = 00000065 hex       101 decimal
# rval = 00000000145 octal
# rval = 00000000000000000000000001100101 bin
# rval has e ascii character value
# pd strength value is StX
# current scope is sdisp2
#    e is ascii value for 101
# simulation time is                    0
```

9.4 Verilog系统任务和系统函数

9.4.1 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, ... );
```

9.4 Verilog系统任务和系统函数

9.4.1 系统任务和系统函数

【例 9-6】

```
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
```

9.4 Verilog系统任务和系统函数

9.4.1 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
```

9.4 Verilog系统任务和系统函数

9.4.1 系统任务和系统函数

```
$finish;
$stop;
```

4. $finish和$stop

【例 9-7】

```
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
```

9.4 Verilog系统任务和系统函数

9.4.1 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
```

9.4 Verilog系统任务和系统函数

9.4.1 系统任务和系统函数

5. $time

$time 返回一个 64 位整数时间值

$stime 返回一个 32 位整数时间值

$realtime 返回一个实数时间值

$timeformat 控制时间的显示方式

```
$monitor("%d d=%b,e=%b", $stime, d, e); //$time 显示当前时间的另一种形式
```

9.4 Verilog系统任务和系统函数

9.4.1 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
```

9.4 Verilog系统任务和系统函数

9.4.1 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
$dumpvar;             //导出当前设计的所有变量
$dumpvar(1, top);     //导出顶层模块中的所有变量
$dumpvar(2, top);     //导出顶层模块和顶层下第 1 层模块的所有变量
$dumpvar(n, top);     //导出顶层模块到顶层下第 n-1 层模块的所有变量
$dumpvar(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
$dumpoff;             //停止导出
```

9.4 Verilog系统任务和系统函数

9.4.1 系统任务和系统函数

6. 文件操作

【例 9-8】

```
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
```

9.4 Verilog系统任务和系统函数

9.4.2 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
```

9.5 延时模型

9.5.1 #延时和门延时

【例 9-9】

```
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
```

9.5 延时模型

9.5.2 延时说明块

```
# 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
```

9.5 延时模型

9.5.2 延时说明块

【例 9-10】

```
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
```

9.6 其他仿真语句

9.6.1 fork_join块语句

【例 9-11】

```
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 9-12】

```
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 9-13】

```
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
```

9.6 其他仿真语句

9.6.1 fork_join块语句

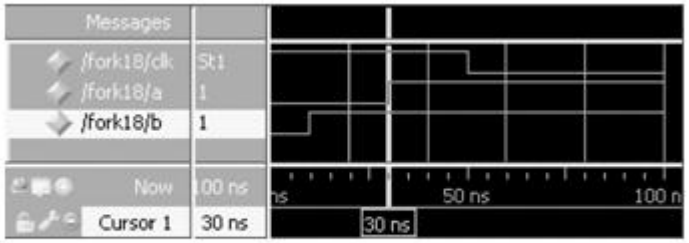

图 9-11　例 9-11 仿真波形

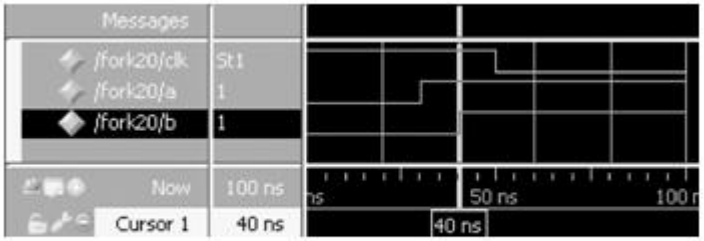

图 9-12　例 9-12 仿真波形

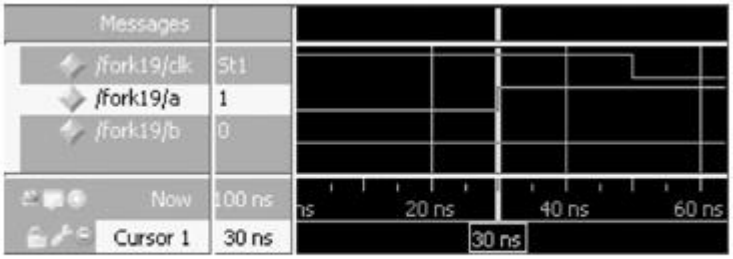

图 9-13　例 9-13 仿真波形

9.6 其他仿真语句

9.6.2 wait语句

```
wait (条件表达式) 语句;
```

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

9.6 其他仿真语句

9.6.3 force语句和release语句

【例 9-14】

```
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
```

9.6 其他仿真语句

9.6.4 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;
```

9.7 仿真激励信号的产生

9.6.4 deassign语句

【例 9-15】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
```

9.7 仿真激励信号的产生

9.6.4 deassign语句

1. 方法一

【例 9-16】

```
`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
```

9.7 仿真激励信号的产生

9.6.4 deassign语句

1. 方法一

【例 9-17】

```
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
```

9.7 仿真激励信号的产生

9.6.4 deassign语句

2. 方法二

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

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
```

9.8 Verilog数字系统仿真

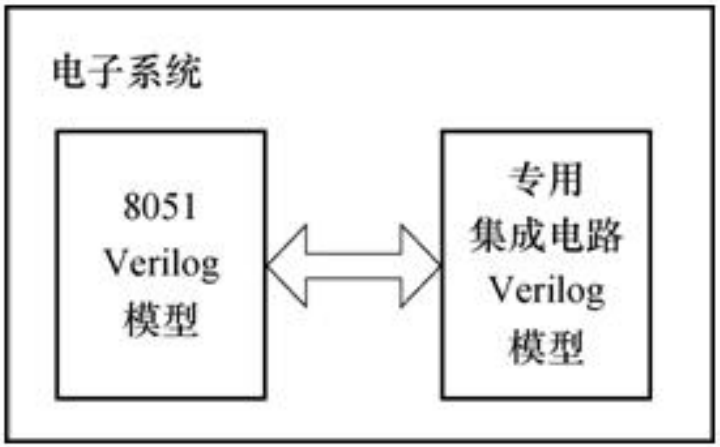

图 9-14　Verilog 系统仿真模型

实验与设计

9-2 在ModelSim上进行16位累加器设计仿真

【例 9-18】

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