EDA技术

第9章 Verilog Test Bench仿真与时序分析



《EDA技术实用教程——Verilog HDL版》
（第七版）
配套课件

EDA Lab

9.1 Verilog HDL仿真流程

- 基于EDA 工具的Verilog HDL设计的仿真，可以称为Verilog HDL仿真

- 从是否考虑电路时延特性对Verilog仿真进行分类
  - 功能仿真
  - 时序仿真

9.1 Verilog HDL仿真流程

- 从电路逻辑描述层次的角度分类：



图9-1 HDL系统设计描述层次

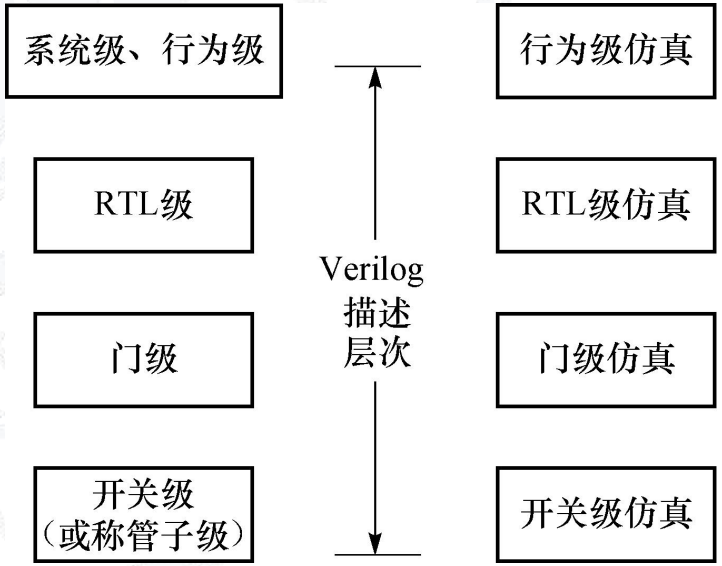
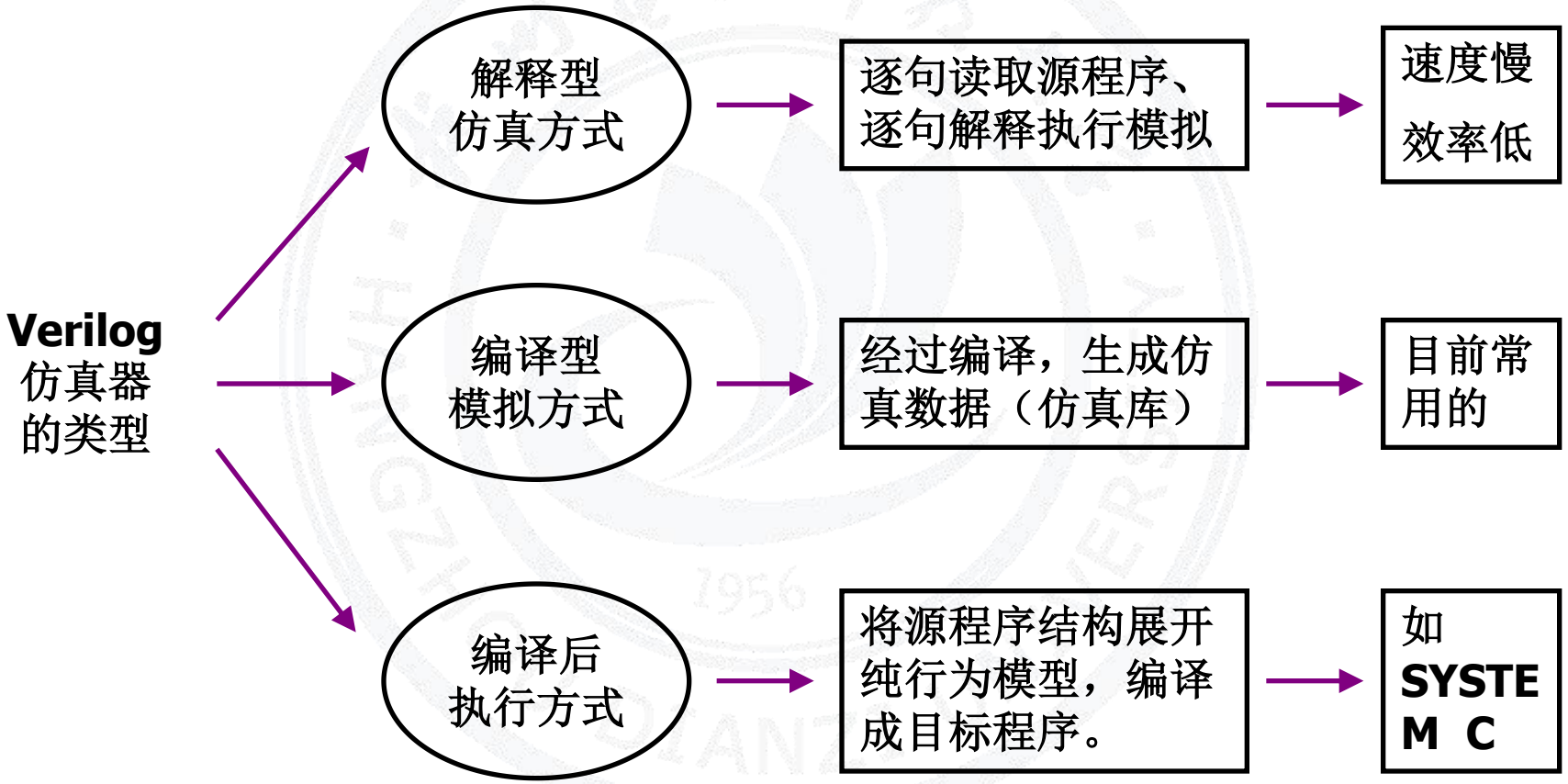


图9-2 Verilog描述层次与对应的仿真层次

9.1 Verilog HDL仿真流程



9.1 Verilog HDL仿真流程

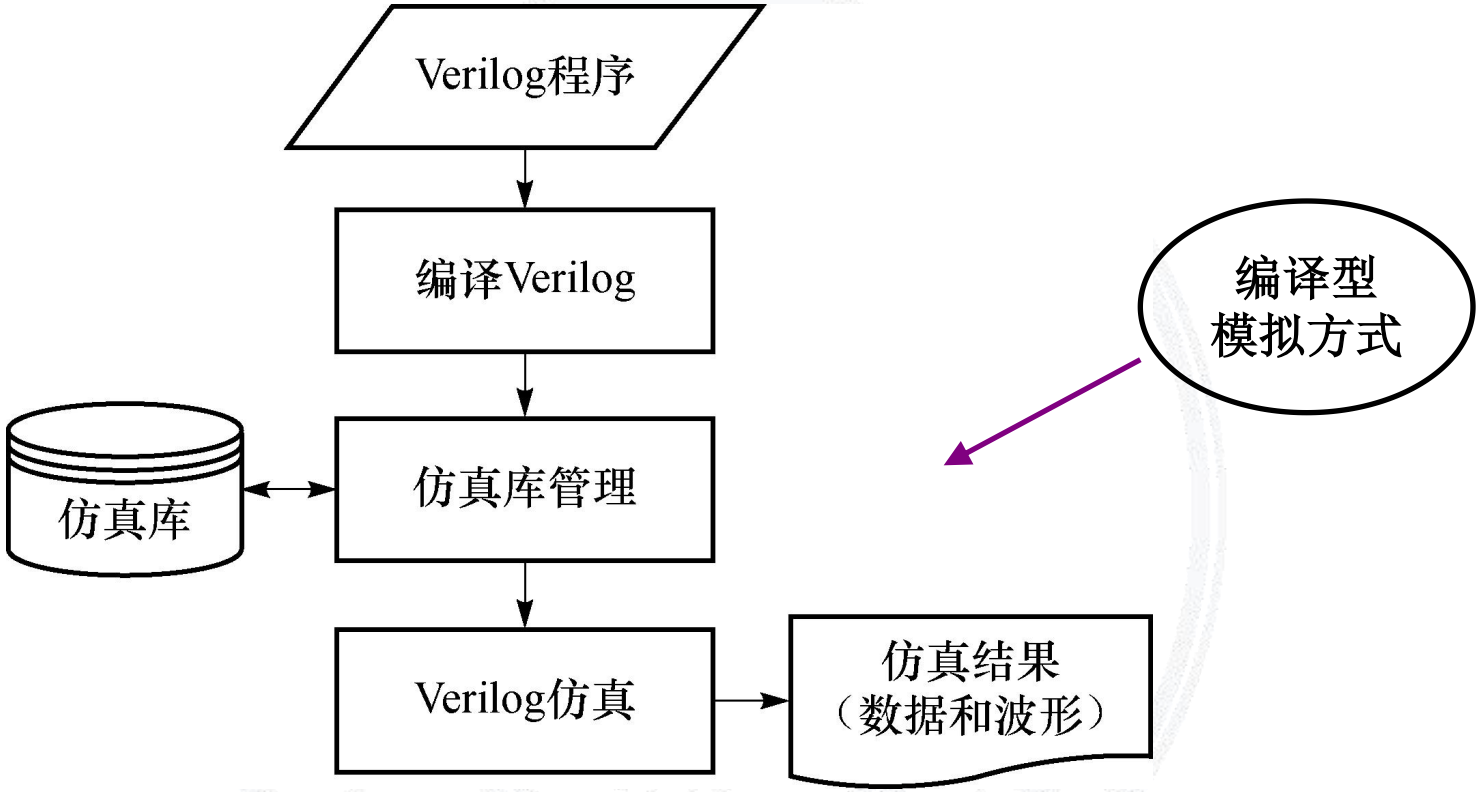


图9-3 Verilog仿真流程

为了实现Verilog仿真，首先可用文本编辑器完成Verilog源程序的设计，送入Verilog仿真器的编译器中进行编译。

9.2 Verilog HDL Test Bench仿真



- Verilog Test Bench是设计者在给定激励并预设结果匹配方法前提下的一种完全不干预仿真器仿真过程的仿真手段，是用来测试一个HDL设计实体的程序。

EDA Lab

9.2 Verilog HDL Test Bench仿真



- Verilog Test Bench 的主要功能有四种：

（1）例化待验证的模块实体。

（2）通过Verilog 程序的行为描述，为待测模块实体提供激励信号。

（3）收集待测模块实体的输出结果，必要时将该结果与预置的所期望的理想结果进行比较，并给出报告。

（4）根据比较结果自动判断模块的内部功能结构是否正确。

EDA Lab

9.2 Verilog HDL Test Bench仿真

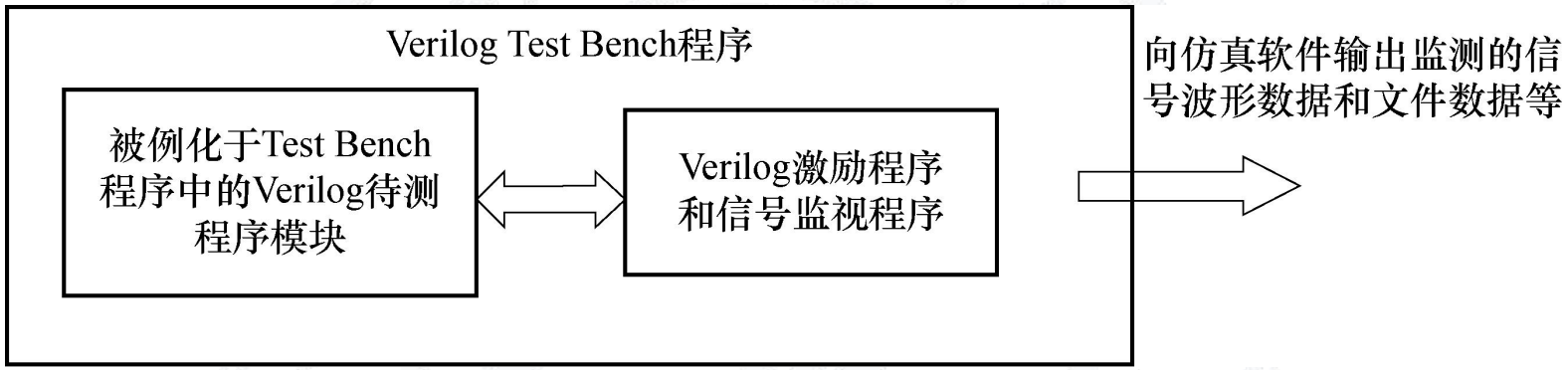


图9-4 Verilog Test Bench结构

- Verilog Test Bench可以使用不可综合的Verilog语句进行描述

EDA Lab

9.2 Verilog HDL Test Bench仿真

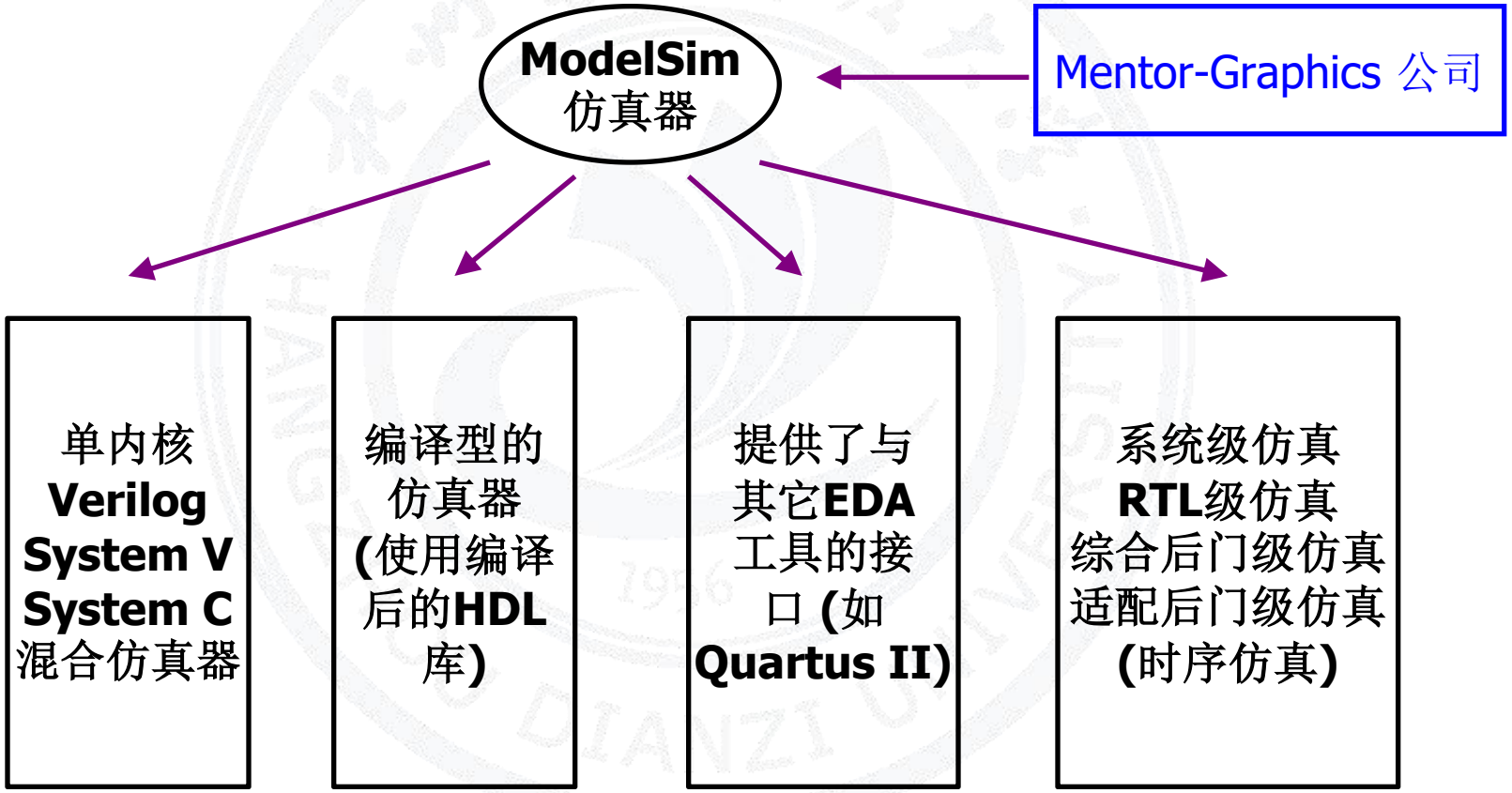


【例9-1】Test Bench文件名：CNT10_TB.v

```
`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
```

EDA Lab

9.3 HDL仿真实例


ModelSim
仿真器
Mentor-Graphics 公司
单内核
Verilog
System V
System C
混合仿真器
编译型的
仿真器
(使用编译
后的HDL
库)
提供了与
其它EDA
工具的接
口 (如
Quartus II)
系统级仿真
RTL级仿真
综合后门级仿真
适配后门级仿真
(时序仿真)

9.3 HDL仿真实例

1. 安装ModelSim
2. 为Test Bench仿真设置参数

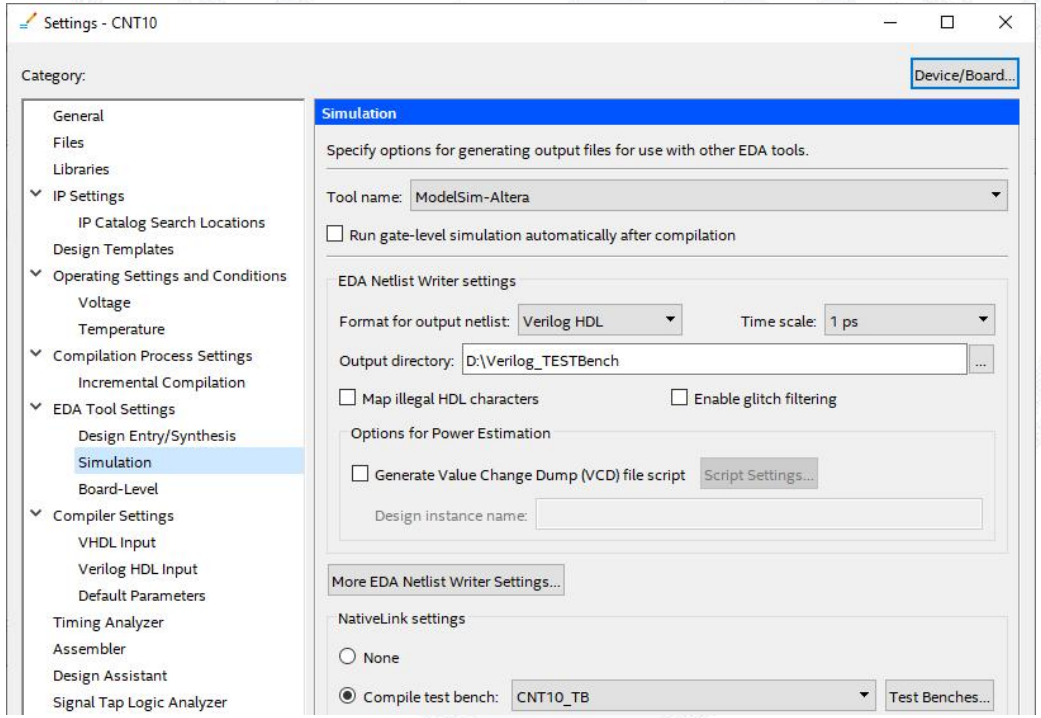


图9-5　选择仿真工具名称和输出网表语言形式

9.3 HDL仿真实例

2．为Test Bench仿真设置参数

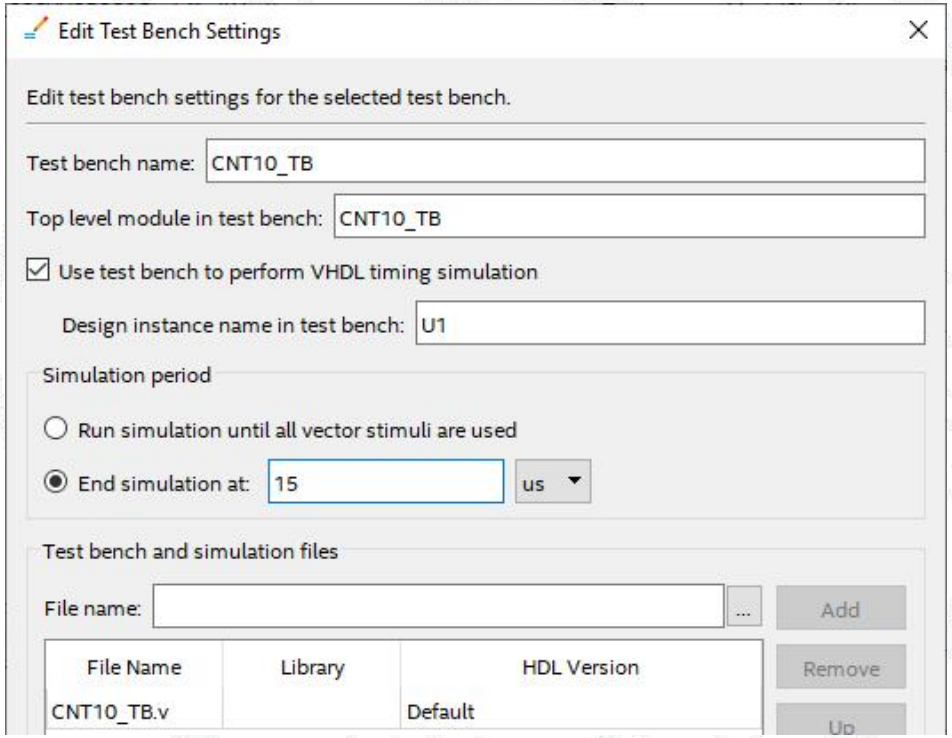


图9-6 为Test Bench仿真设置参数

3．启动Test Bench仿真

9.3 HDL仿真实例

4．分析Test Bench仿真结果

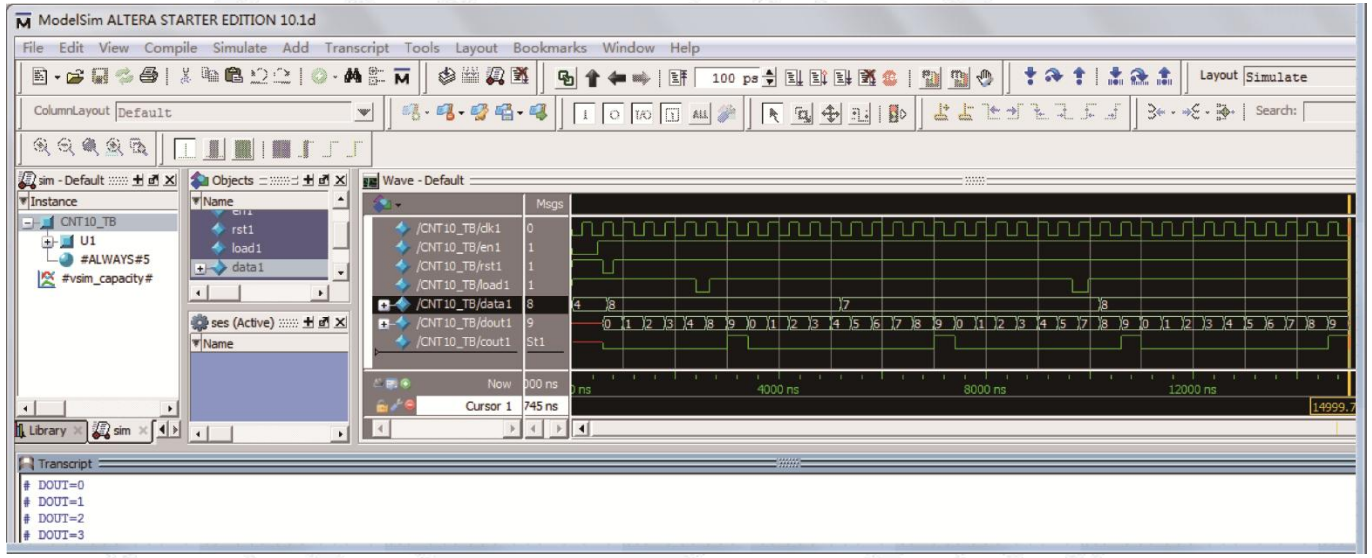


图9-7　启动仿真后弹出的ModelSim界面（时序波形清晰画面如图9-8所示）

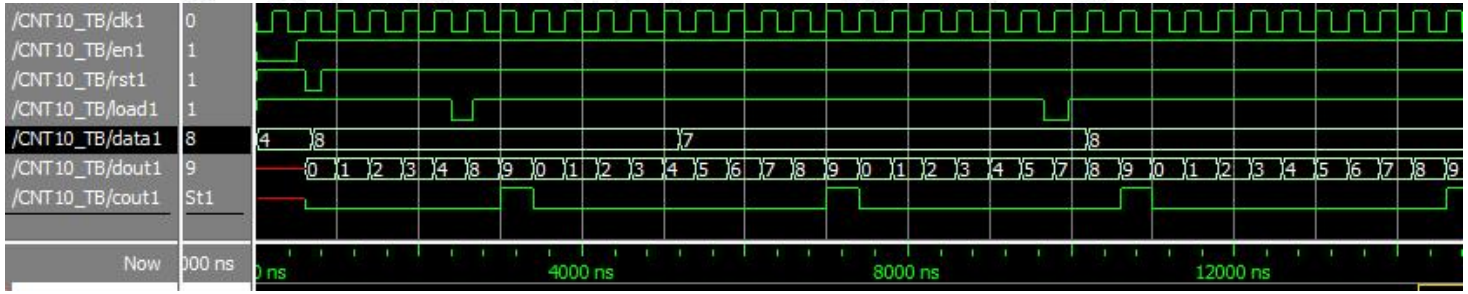


图9-8　Test Bench输出的仿真波形详图

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
```

EDA Lab

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
```

EDA Lab

9.3 HDL仿真实例

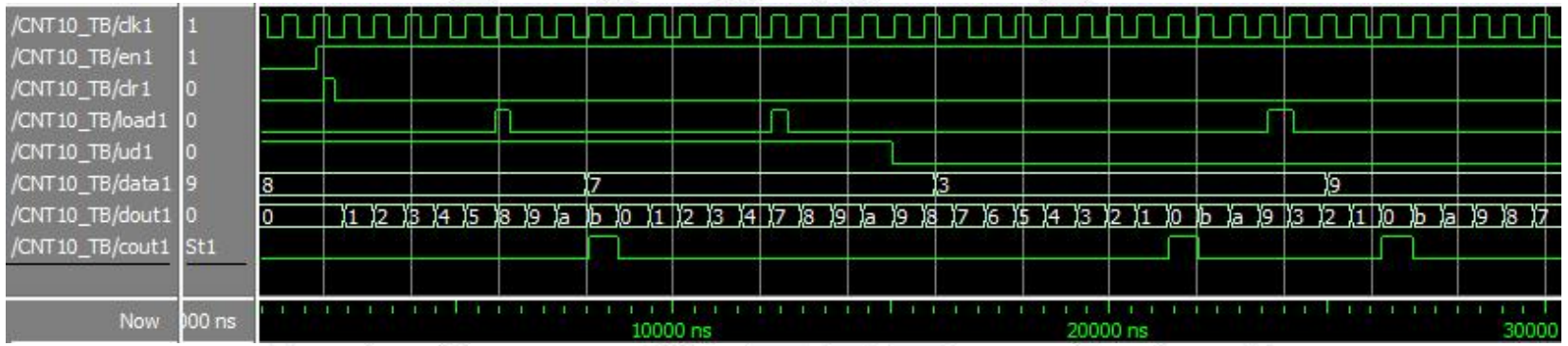


图9-9 ModelSim（直接）输出的Test Bench仿真波形

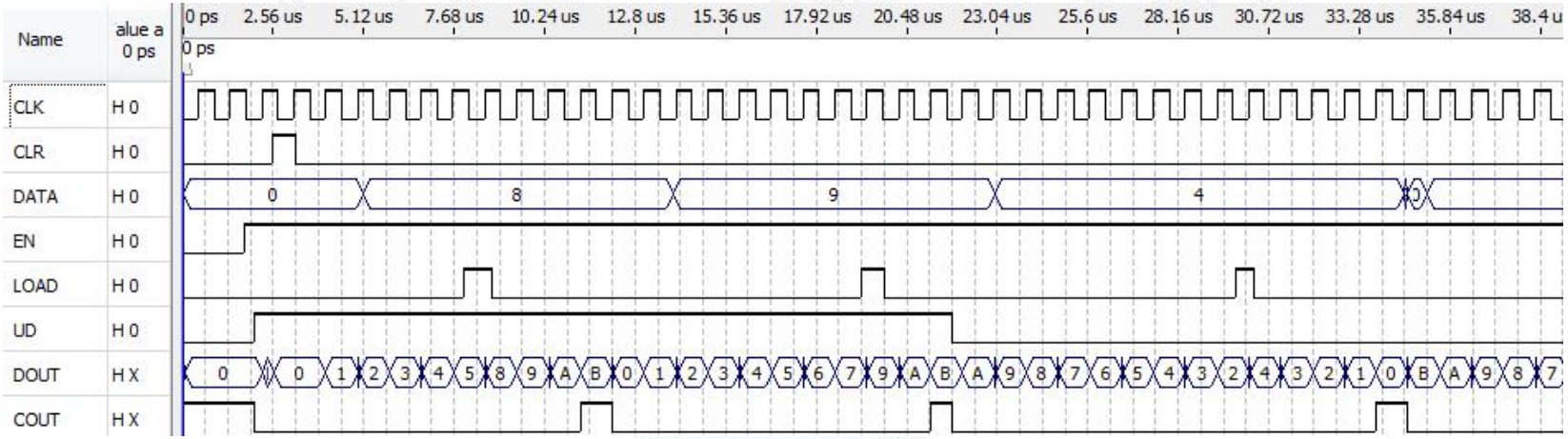
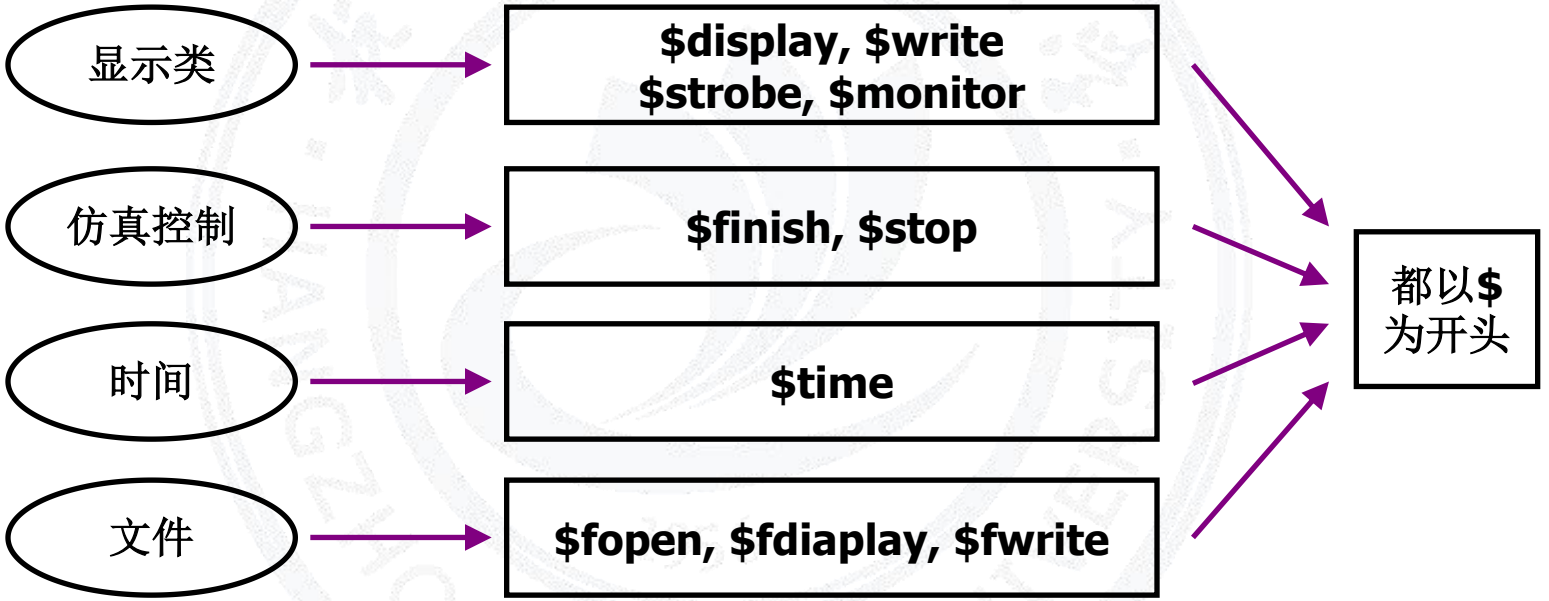


图9-10 波形仿真器的仿真波形（ModelSim间接输出波形）

9.4 HDL系统任务和系统函数



9.4 HDL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
```

命令行窗口返回的结果：

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

EDA Lab

9.4 HDL系统任务和系统函数

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 HDL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
```

仿真结果：

```
# 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 HDL系统任务和系统函数



2. $write

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

$display会在文本后加一个换行，而$write是不加的

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, ... );
```

$strobe 的功能是，当该时刻的所有事件处理完后，在这个时间的结尾显示格式化字符串。
$monitor 的功能是当一个或多个指定的信号变量改变值时，显示格式化字符串，且仿真过程中，只能允许一个$monitor 被执行

这两个函数在有些情况下可以弥补$display函数的不足

EDA Lab

9.4 HDL系统任务和系统函数



3. $strobe和$monitor

【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
```

仿真结果：

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

9.4 HDL系统任务和系统函数



4. $finish和$stop

```
$finish;
$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
```

仿真结果：

```
#          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
```

EDA Lab

9.4 HDL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
```

EDA Lab

9.4 HDL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
```

EDA Lab

9.4 HDL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
```

EDA Lab

9.4 HDL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
```

EDA Lab

9.4 HDL系统任务和系统函数



7．预编译语句

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

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

EDA Lab

9.5延时模型



“#10 rout = cin; 表示cin延迟10个时间单位后，赋值给rout

- # 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
```

仿真结果：

```
# 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
```

EDA Lab

9.5延时模型

【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
```

用以指定一条路径的延迟

9.6其他仿真语句

1 fork_ join块语句

fork_join块中的语句是并行启动执行的，但该块语句的执行终结要等待语句块中执行最慢的语句结束。

【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
```

EDA Lab

9.6其他仿真语句

1 fork_ join块语句

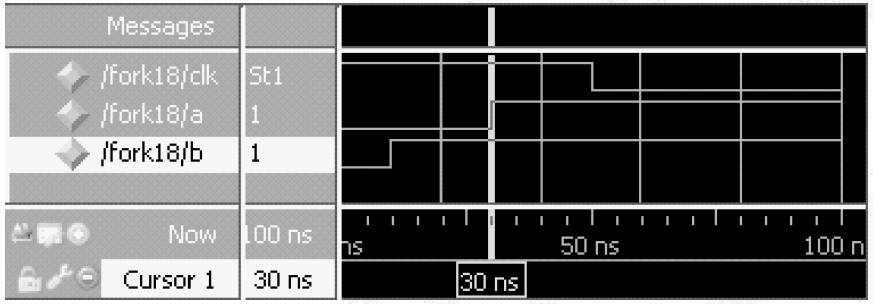


图9-11 例9-11仿真波形

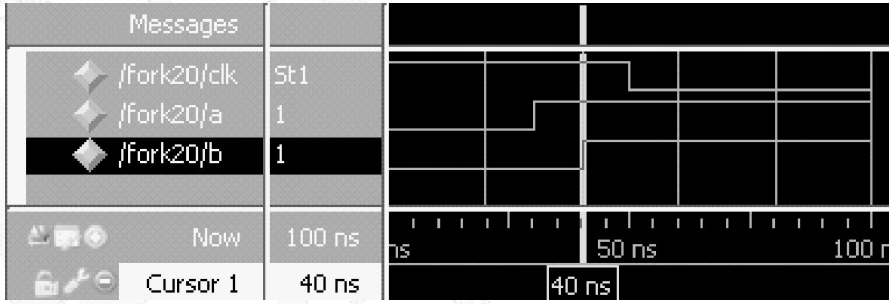


图9-12 例9-12仿真波形

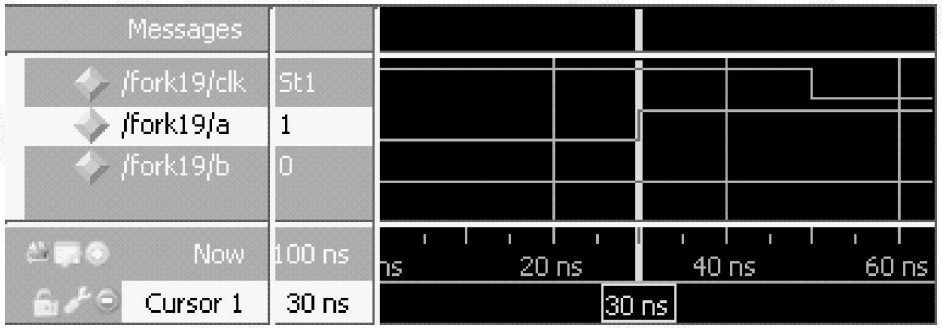


图9-13 例9-13仿真波形

9.6其他仿真语句



2 wait语句

wait语句不可综合，只能用于仿真。语法格式如下：

```
wait (条件表达式) 语句;
```

可以使用以下方式：

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

等待启动信号start为true，在go上输出周期为20ns的方波。

EDA Lab

9.6其他仿真语句



3 force和release语句

- force是Verilog语句中的强制信号赋值，该语句只能出现在initial或always块中。
- force可对wire或reg型变量赋值。
- 如果要取消强制赋值，可用release语句释放。

【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
```

ModelSim仿真，输入run -all

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

仿真器暂停，再次输入run -all

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

9.6其他仿真语句



4 deassign语句

在使用assign语句进行连续赋值后，还可以使用deassign语句来取消assign的赋值

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

- 当clear有效，q赋值0；当clear无效而preset有效，q赋值1；否则，放弃对q的赋值。
- 该写法只能在仿真器上使用，综合器不支持deassign语句

EDA Lab

9.7仿真激励与Verilog数字系统仿真



【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仿真激励与Verilog数字系统仿真



方法一：用Verilog产生激励信号

【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
```

【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
```

EDA Lab

9.7仿真激励与Verilog数字系统仿真

方法二：仿真器的波形设置命令产生激励信号

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

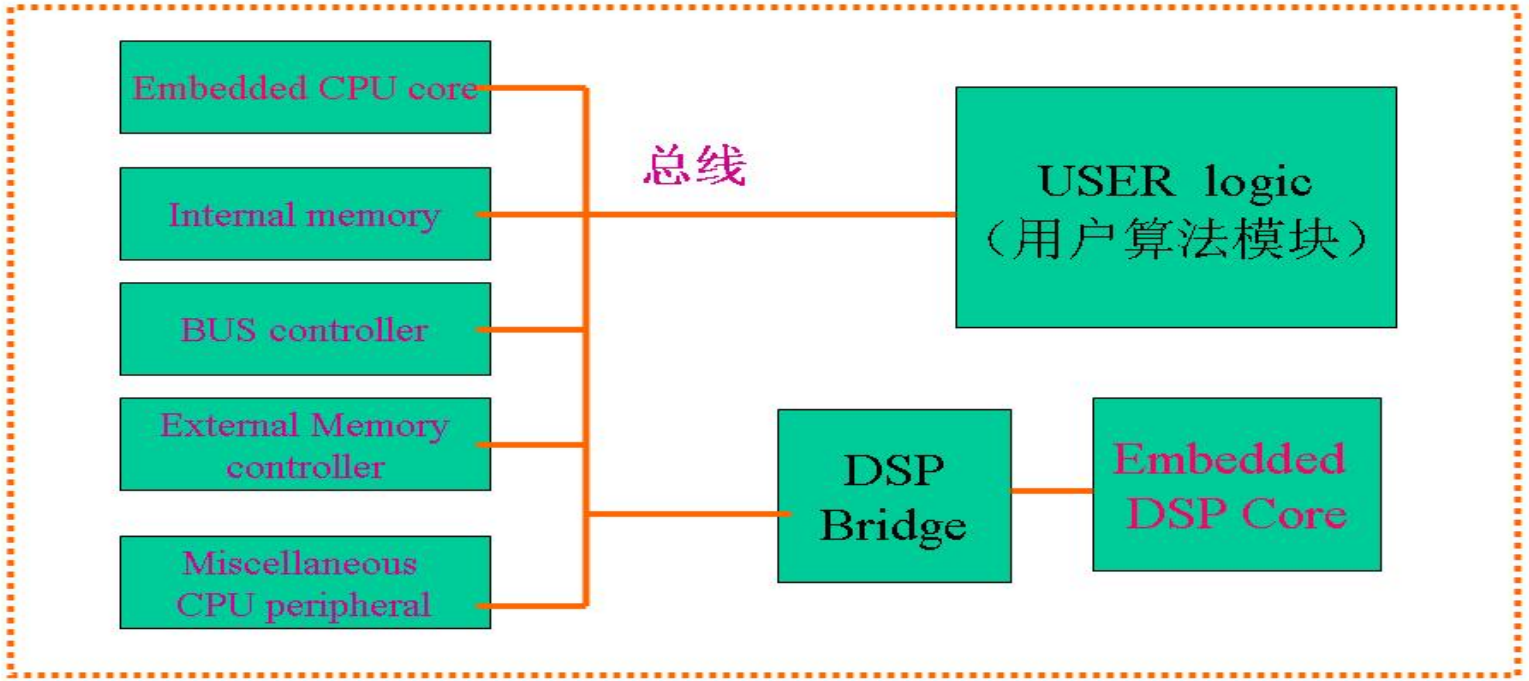
```
force a 0                          （强制信号的当前值为0）
force b 0 0, 1 10                  （强制信号b在时刻0的值为0，在时刻10的值为1）
force clk 0 0, 1 15 -repeat 20     （clk为周期信号，周期为20）
```

可以直接用模块adder4 的结构体进行仿真：

```
force a 10 0, 5 200, 8 400
force b 3 0, 4 100, 6 300
```

EDA Lab

9.7仿真激励与Verilog数字系统仿真



SOC(SOPC)芯片的架构

9.7仿真激励与Verilog数字系统仿真

- 对整个数字系统都进行仿真，可使芯片设计风险大为减小
- 为支持数字系统仿真，目前大多数Verilog仿真器支持标准的编程语言接口PLI，可制作通用的仿真模块
- 即使没有使用Verilog 设计的数字集成电路，同样可以设计出Verilog 仿真模型

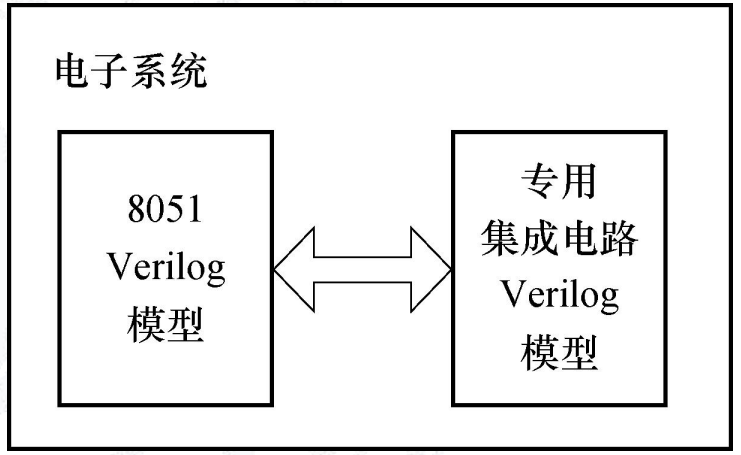


图9-14 Verilog系统仿真模型

EDA Lab

实验与设计



- 9-1 在ModelSim上对计数器的Test Bench进行仿真
- 9-2 在ModelSim上进行16位累加器设计仿真

HANGZHOU DIANZI UNIVERSITY 1956



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
```

EDA Lab