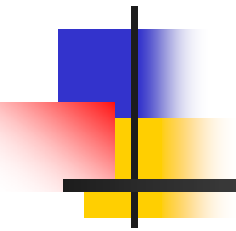


第3章

Verilog设计入门

3.1 组合电路的Verilog描述

3.1.1 2选1多路选择器及其Verilog描述

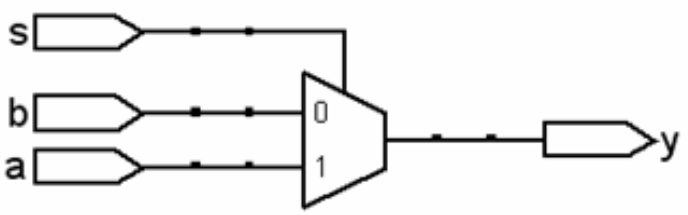


图 3-1　二选一多路选择器 mux21a 电路实体

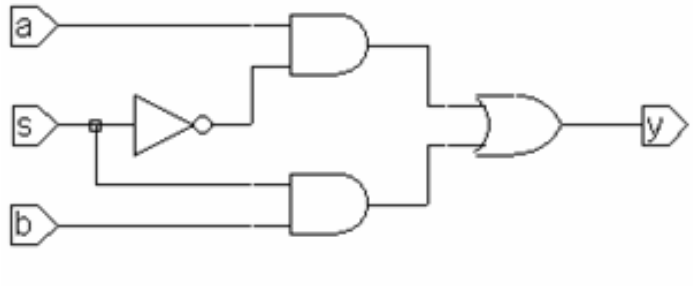


图 3-2　mux21a 逻辑结构之一

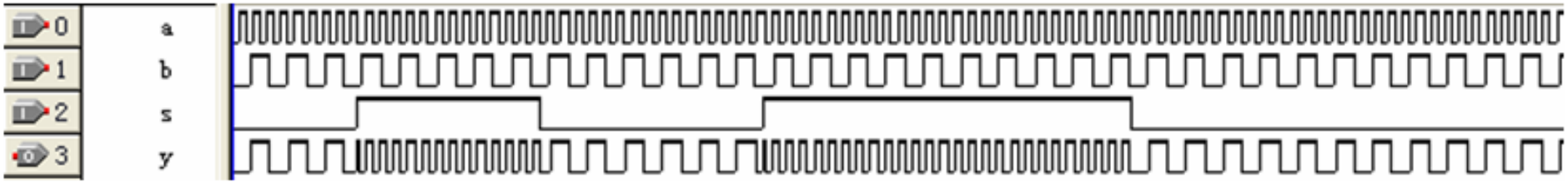


图 3-3　mux21a 电路的时序波形

3.1 组合电路的Verilog描述

3.1.1 2选1多路选择器及其Verilog描述

【例 3-1】

```
module MUX21a (a,b,s,y);
    input a,b,s;
    output y;
    assign  y = (s ? a : b);
endmodule
```

3.1 组合电路的Verilog描述

3.1.1 2选1多路选择器及其Verilog描述

1．模块表达

```
module  模块名 （模块端口名表） ;
    模块端口和模块功能描述 .
endmodule
```

2．端口语句、端口信号名和端口模式

```
input   端口名 1，端口名 2，. . .  ;
output  端口名 1，端口名 2，. . .;
inout   端口名 1，端口名 2，. . .  ;
input [msb : lsb]   端口名 1，端口名 2，. . .;
```

3.1 组合电路的Verilog描述

3.1.1 2选1多路选择器及其Verilog描述

3．赋值语句和条件操作符

```
assign   y = a;          //将信号 a 向 y 赋值
assign   y = a & b ;     //将信号 a 和信号 b 逻辑相与后向 y 赋值
```

条件表达式 ？ 表达式1 ： 表达式2

4．关键字

5．标识符

6．规范的程序书写格式

7．文件取名和存盘

3.1 组合电路的Verilog描述

3.1.2 4选1多路选择器及其case语句表述方式

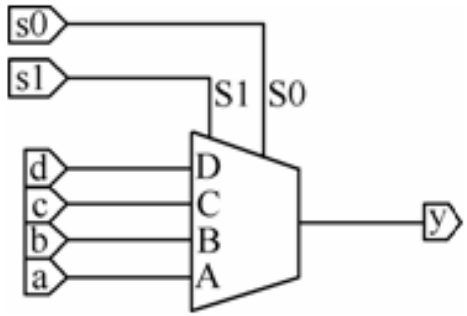


图 3-4 4 选 1 多路选择器

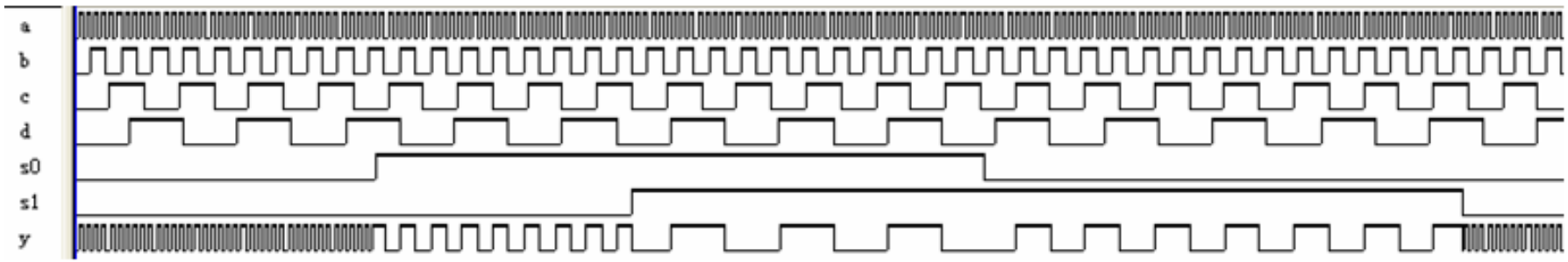


图 3-5 4 选 1 多路选择器 MUX41a 的时序波形

3.1 组合电路的Verilog描述

3.1.2 4选1多路选择器及其case语句表述方式

【例 3-2】

Verilog表述的可综合的完整电路模块

```
module MUX41a (a, b, c, d, s1, s0, y);
    input  a, b, c, d ;          // 电路模块端口说明和定义段
    input  s1, s0 ;
    output y ;
    reg y ;                      // 信号类型定义段
    always @ ( a or b or c or d or s1 or s0 )
        begin  : MUX41           // 电路模块功能描述段
            case( { s1, s0 } )
                2'b00 : y <= a ;
                2'b01 : y <= b ;
                2'b10 : y <= c ;
                2'b11 : y <= d ;
                default : y <= a ;
            endcase
        end
endmodule
```

3.1 组合电路的Verilog描述

3.1.2 4选1多路选择器及其case语句表述方式

1．reg型变量定义

```
reg  变量名1，变量名2，．．．；
reg  [msb:lsb] 变量名1，变量名2，．．．；
```

2．过程语句

always @ （敏感信号及敏感信号列表或表达式）

包括块语句的各类顺序语句

3.1 组合电路的Verilog描述

3.1.2 4选1多路选择器及其case语句表述方式

3．块语句begin_ end

```
begin  [: 块名]
    语句 1; 语句 2; . . .; 语句 n;
end
```

4．case条件语句和4种逻辑状态

```
case （表达式）
    取值 1 : begin 语句 1; 语句 2; ... ; 语句 n;    end
    取值 2 :  begin 语句 n+1; 语句 n+2; ... 语句 n+m;  end
      ...
      default : begin 语句 n+m+1; ... ;  end
endcase
```

3.1 组合电路的Verilog描述

3.1.2 4选1多路选择器及其case语句表述方式

5．并位操作和数字表达

case

{s1，s0}=2'b00

{ a1, b1, 4{a2,b2}}={ a1, b1, {a2,b2},{a2,b2},{a2,b2},{a2,b2}}={ a1,b1,a2,b2,a2,b2,a2,b2,a2,b2}

<位宽>'<进制><数字>

3.1 组合电路的Verilog描述

3.1.3 4选1多路选择器及其数据流描述方式

【例 3-3】

```
module MUX41a (a,b,c,d,s1,s0,y);
    input a,b,c,d,s1,s0;        output y;
    wire [1:0] SEL;          // 定义2元素位矢量SEL为网线型变量wire
    wire AT, BT, CT, DT;     //定义中间变量，以作连线或信号节点
    assign SEL = {s1,s0};    //对s1,s0进行并位操作，即SEL[1]=s1; SEL[1]=s0
    assign AT = (SEL==2'D0); //assign语句中的变量必须是网线型变量
    assign BT = (SEL==2'D1);
    assign CT = (SEL==2'D2);
    assign DT = (SEL==2'D3);
    assign  y = (a & AT)|(b &BT)|(c & CT)|(d & DT);//4个逻辑信号相或
  endmodule
```

3.1 组合电路的Verilog描述

3.1.3 4选1多路选择器及其数据流描述方式

1．按位逻辑操作符

表 3-1 逻辑操作符

逻辑操作符	逻辑功能	A,B 逻辑操作结果	C,D 逻辑操作结果	C,E 逻辑操作结果
~	逻辑取反	~A = 1'b1	~C = 4'b0011	~E = 6'b101001
|	逻辑或	A | B = 1'b 1	C | D = 4'b1111	C | E = 6'b011110
&	逻辑与	A & B = 1'b 0	C & D = 4'b1000	C & E = 6'b000100
^	逻辑异或	A ^ B = 1'b 1	C ^ D = 4'b0111	C ^ E = 6'b011010
~^ 或 ^~	逻辑同或	A ~^ B = 1'b 0	C ~^ D = 4'b1000	C ~^ E =6'b100101

A=1'b0;　B=1'b1;　C[3:0]=4'b1100;　D[3:0]=4'b1011;　E[5:0]=6'b010110;

3.1 组合电路的Verilog描述

3.1.3 4选1多路选择器及其数据流描述方式

2．等式操作符

表 3-2 等式操作符

等式操作符	含义	等式操作示例
==	等于	(3==4)= 0; (A==4'b1011)= 1; (B==4'b1011) =0;
!=	不等于	(D==C) = 0; (D!=C) = 1; (3!=4)= 1;
===	全等	(4'b0z1x == 4'b0z1x1) = 0 ;
!==	不全等	(4'b0z1x=== 4'b0z1x1) = 0;

```
A=4'b1011; B=4'b0010; C=4'b0z10; D=4'b0z10;
```

3.1 组合电路的Verilog描述

3.1.3 4选1多路选择器及其数据流描述方式

3．assign连续赋值语句

```
assign  目标变量名 = 驱动表达式;
```

```
assign DOUT = a & b;
```

```
assign DOUT = a & b | c ;
assign DOUT = e & f | d ;
```

3.1 组合电路的Verilog描述

3.1.3 4选1多路选择器及其数据流描述方式

4. wire定义网线型变量

```
wire  变量名1，变量名2，．．．；
wire  [msb:lsb] 变量名1，变量名2，．．．；
```

```
wire   Y = tmp1 ^ tmp2;
```

```
wire   tmp1,tmp2;   assign   Y = tmp1 ^ tmp2;
```

3.1 组合电路的Verilog描述

3.1.3 4选1多路选择器及其数据流描述方式

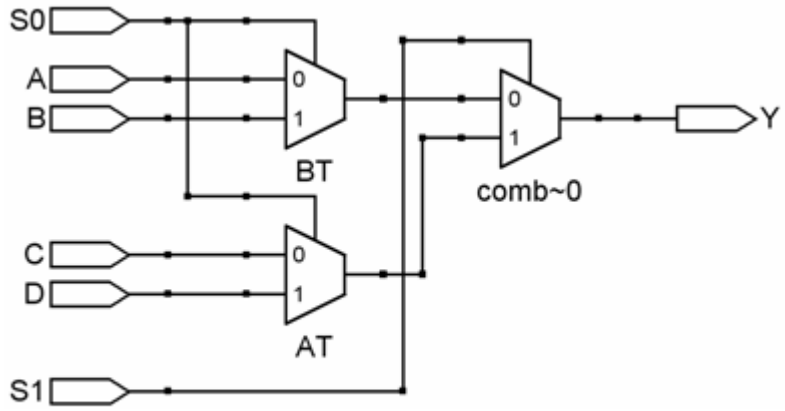


图 3-6 例 3-4 的 RTL 图

【例 3-4】

```
module MUX41a (A,B,C,D,S1,S0,Y);
   input A,B,C,D,S1,S0;    output Y;
   wire AT = S0 ? D : C ;       //如果S0=1成立，则AT=D；如果S0=0成立，则AT=C
   wire BT = S0 ? B : A ;       //如果S0=1成立，则BT=B；如果S0=0成立，则BT=A
   wire  Y = (S1 ? AT : BT);    //如果S1=1，则Y=AT；如果S1=0，则Y=BT
endmodule
```

3.1 组合电路的Verilog描述

3.1.3 4选1多路选择器及其数据流描述方式

5．注释符号

“//”　　/* . . . */

3.1 组合电路的Verilog描述

3.1.4 4选1多路选择器及其if语句描述方式

【例 3-5】

```
module MUX41a (A,B,C,D,S1,S0,Y);
    input A,B,C,D,S1,S0;
    output Y;                          //定义 Y 为输出信号
    reg [1:0] SEL ;                    //定义一个模块内部的暂存变量 SEL[1:0]
    reg Y;                             //定义输出端口信号 Y 为寄存器型变量
    always @(A,B,C,D,SEL)  begin //块语句起始
            SEL = {S1,S0};             //把 S1,S0 并位为 2 元素矢量变量 SEL[1:0]
        if (SEL==0)  Y = A;            //当 SEL==0 成立，即(SEL==0)=1 时，Y=A;
  else  if (SEL==1)  Y = B;            //当(SEL==1)为真，则 Y=B;
  else  if (SEL==2)  Y = C;            //当(SEL==2)为真，则 Y=C;
  else               Y = D;            //当 SEL==3，即 SEL==2'b11 时，Y = D;
  end                                  //块语句结束
   endmodule
```

3.1 组合电路的Verilog描述

3.1.4 4选1多路选择器及其if语句描述方式

1. if_ else条件语句

2. 过程赋值语句

（1）阻塞式赋值

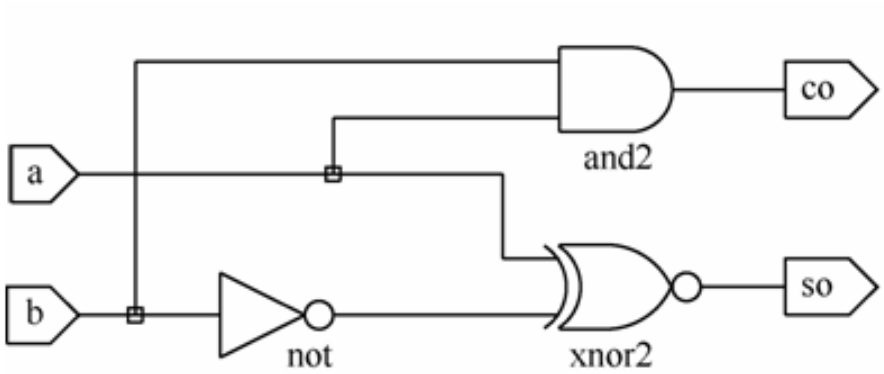
（2）非阻塞式赋值

3. 数据表示方式

3.1 组合电路的Verilog描述

3.1.5 加法器及其Verilog描述

1. 半加器描述



a	b	so	co
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

图 3-7 半加器 h_adder 电路图及其真值表

3.1 组合电路的Verilog描述

3.1.5 加法器及其Verilog描述

1. 半加器描述

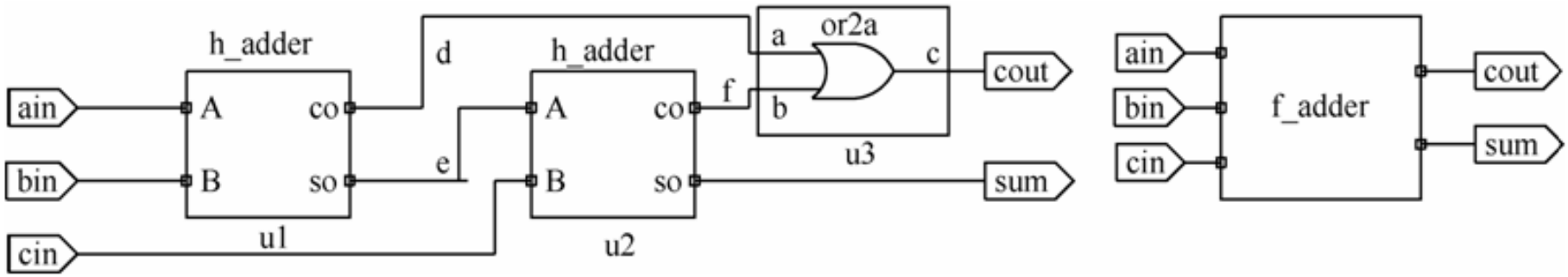


图 3-8 全加器 f_adder 电路图及其实体模块

3.1 组合电路的Verilog描述

3.1.5 加法器及其Verilog描述

1. 半加器描述

【例 3-6】

```
module h_adder(a,b,so,co);//半加器描述(1)，利用图 3-7 的布尔函数描述方法
   input a,b ;
   output so,co ;
  assign so = (a) ^~ (~b) ;  // (a)同或(b 非)
  assign co = a & b ;
endmodule
```

3.1 组合电路的Verilog描述

3.1.5 加法器及其Verilog描述

1. 半加器描述

【例 3-7】

```
module h_adder (a,b,so,co); //半加器描述(2)：基于 case 语句的类真值表描述方法
   input a,b ;      output so,co ;
   reg so,co;
   always @ (a , b , so , co)   begin  //主块开始
      case({a,b})
       0 : begin so=0 ; co=1'b0 ; end  //注意这里使用了块语句
       1 : begin so=1 ; co=1'b0 ; end
       2 : begin so=1 ; co=1'b0 ; end
       3 : begin so=0 ; co=1'b1 ; end //注意条件表达式下的数值的数值类型
    default : begin so=0 ; co=0 ;  end
      endcase
      end      //主块结束
endmodule
```

3.1 组合电路的Verilog描述

3.1.5 加法器及其Verilog描述

1. 半加器描述

表 3-3　算术操作符

逻辑操作符	功 能	说 明	示 例
+	加		S＝A＋B＝8′b00011000
-	减		S＝B－A＝8′b11111110
*	乘		S＝A＊B＝8′b10001111=2′H8F
/	除	结果：小数抛弃	S＝A／3＝8′b00000100
%	求余	除法求余数	S＝A％3＝8′b00000001

A[3:0]=4′b1101; B[3:0]=4′b1011; 定义 S 为 S[7:0]

3.1 组合电路的Verilog描述

3.1.5 加法器及其Verilog描述

1. 半加器描述

【例 3-8】

```
module h_adder(a,b,so,co); //半加器描述(3)：直接用算符完成的描述
    input a,b ;
    output so,co ;
    assign {co,so} = a + b;  //两位二进制数直接相加，进位进入并位后的 co
endmodule
```

【例 3-9】

```
module or2a(a,b,c);          //或门逻辑描述
    output c;  input a,b ;
    assign c = a | b ;
endmodule
```

3.1 组合电路的Verilog描述

3.1.5 加法器及其Verilog描述

2. 全加器顶层文件设计和例化语句

【例 3-10】

```
module f_adder(ain,bin,cin,cout,sum); //1 位二进制全加器顶层设计描述
    output cout,sum ;    input ain,bin,cin ;
    wire  e,d,f ;                          //定义网线型变量用作内部元件间连线
    h_adder  u1(  ain, bin, e, d );        //使用位置关联法进行例化
    h_adder  u2(.a(e), .so(sum), .b(cin),.co(f) );
    or2a  u3(.a(d), .b(f),  .c(cout) );//使用端口名关联法进行例化
endmodule
```

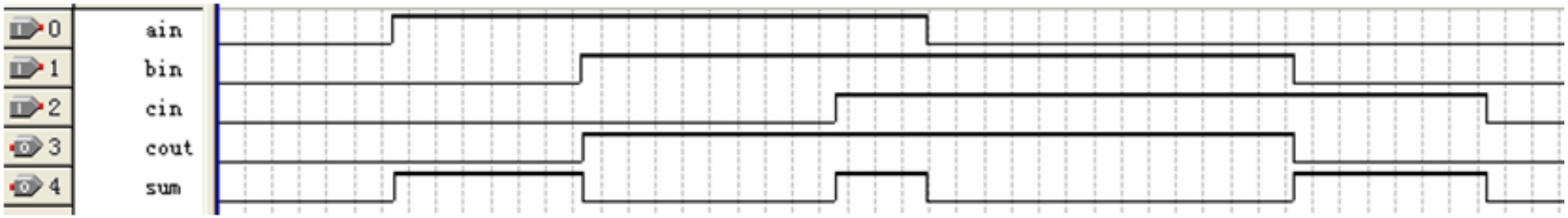


图 3-9　1 位全加器仿真时序

3.1 组合电路的Verilog描述

3.1.5 加法器及其Verilog描述

2. 全加器顶层文件设计和例化语句

<模块元件名>　<例化元件名>（.例化元件端口（例化元件外接端口名）,...）;

```
h_adder  u2(.a(e), .so(sum), .b(cin),.co(f));

h_adder  u2(.b(cin),.co(f),.a(e), .so(sum));

h_adder  u1(  ain, bin, e, d );
```

3.1 组合电路的Verilog描述

3.1.5 加法器及其Verilog描述

3. 8位加法器描述

【例 3-11】

```
module ADDER8B(A,B,CIN,COUT,DOUT);
    output [7:0] DOUT;  output COUT;
     input [7:0] A,B;    input  CIN;     wire [8:0]  DATA;
     assign DATA = A + B + CIN; //加操作的进位自动进入 DATA[8]
     assign  COUT = DATA[8] ;   assign DOUT = DATA[7:0] ;
  endmodule
```

3.1 组合电路的Verilog描述

3.1.5 加法器及其Verilog描述

3．8位加法器描述

【例 3-12】

```
module ADDER8B (A,B,CIN,COUT,DOUT);
    output [7:0] DOUT;  output COUT;
    input [7:0] A,B;    input  CIN;
    assign {COUT,DOUT} = A + B + CIN; //加操作的进位进入并位 COUT
endmodule
```

3.1 组合电路的Verilog描述

3.1.5 加法器及其Verilog描述

3. 8位加法器描述

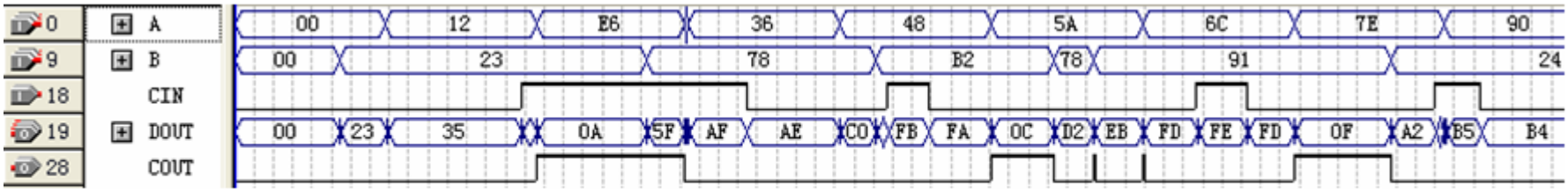


图 3-10 八位加法器仿真波形

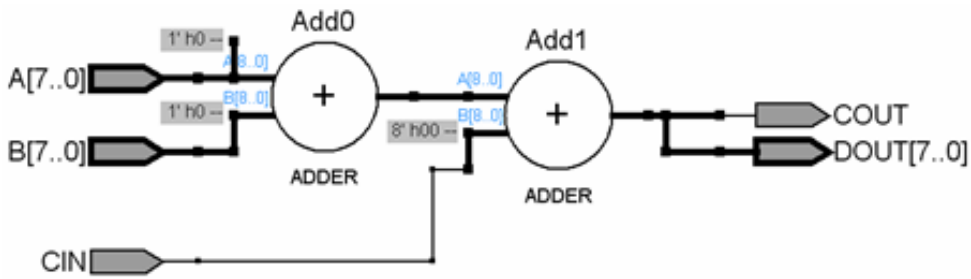


图 3-11 8 位加法器 QuartusII 综合之 RTL 电路

3.2 时序模块及其Verilog表述

3.2.1 边沿触发型触发器及其Verilog表述

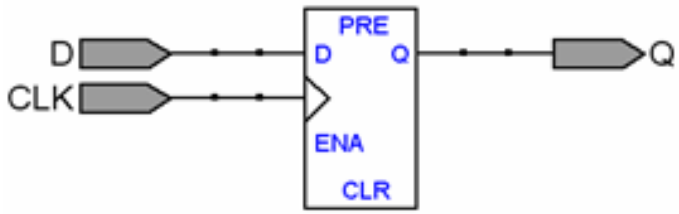


图 3-12　D 触发器模块

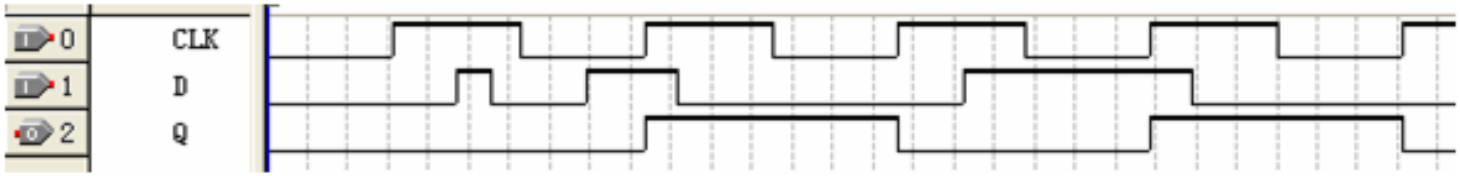


图 3-13 D 触发器时序波形

3.2 时序模块及其Verilog表述

3.2.1 边沿触发型触发器及其Verilog表述

【例 3-13】

```
module DFF1(CLK,D,Q);                    //D 触发器基本模块
   output Q ;
   input  CLK, D ;
   reg Q;
   always @( posedge CLK )               //CLK 上升沿启动
      Q <= D;                            //当 CLK 有上升沿时 D 被锁入 Q
endmodule
```

3.2 时序模块及其Verilog表述

3.2.2 电平触发型锁存器及其Verilog表述

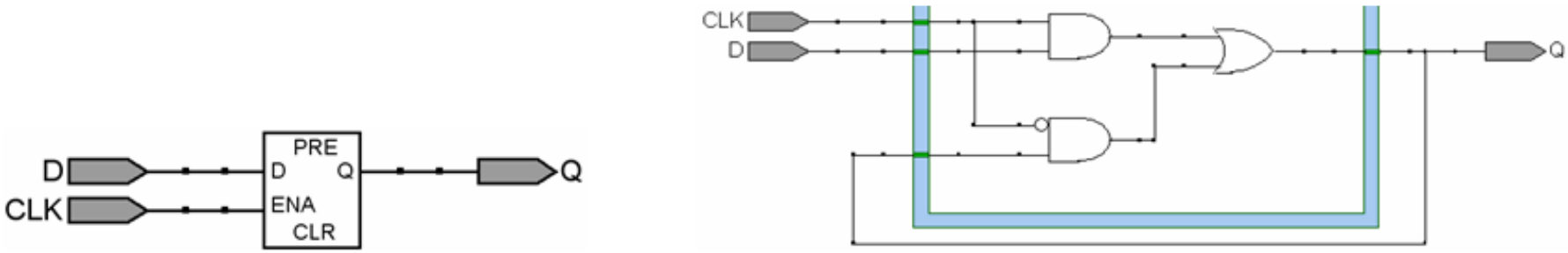


图 3-14 锁存器 LATCH1 模块及其逻辑电路

3.2 时序模块及其Verilog表述

3.2.2 电平触发型锁存器及其Verilog表述

【例 3-14】

```
module LATCH1(CLK,D,Q);
   output Q ; input CLK,D;
   reg Q;
   always @(D or CLK)
      if(CLK)  Q <= D;
endmodule
```

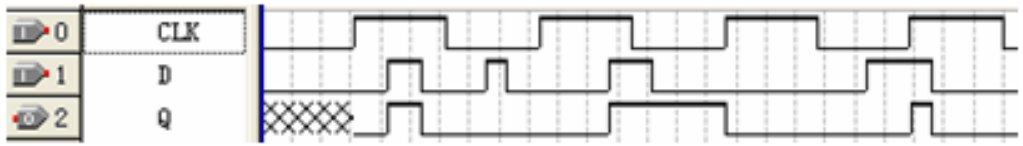


图 3-15 例 3-14 的锁存器时序波形

3.2 时序模块及其Verilog表述

3.2.3 含异步复位/时钟使能型触发器及其Verilog表述

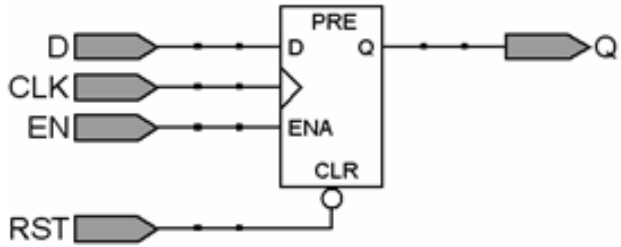


图 3-16 含使能和复位的 D 触发器

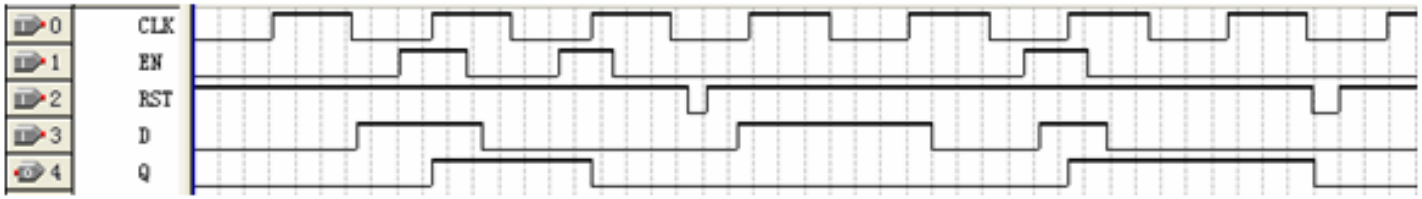


图 3-17 含异步清 0 和时钟使能型 D 触发器的时序图

3.2 时序模块及其Verilog表述

3.2.3 含异步复位/时钟使能型触发器及其Verilog表述

【例 3-15】

```
module DFF2(CLK,D,Q,RST,EN); //含异步清 0 和时钟同步使能的 D 触发器
    output Q ;   input CLK,D,RST,EN;
    reg Q;
      always @(posedge CLK or negedge RST ) begin //块开始
         if (!RST)  Q <= 0;  //如果 RST=0 条件成立，Q 被清 0
    else if (EN)    Q <= D;  //在 CLK 上升沿处，EN=1,则执行赋值语句
    end   //块结束
endmodule
```

3.2 时序模块及其Verilog表述

3.2.4 同步复位型触发器及其Verilog表述

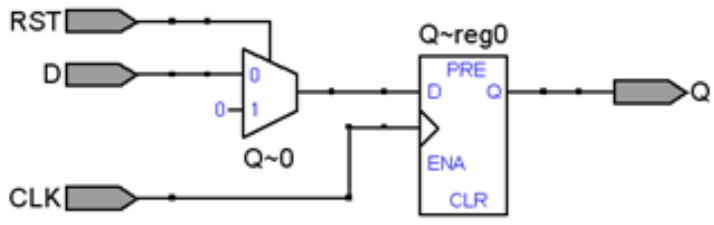


图 3-18　含同步清 0 的 D 触发器

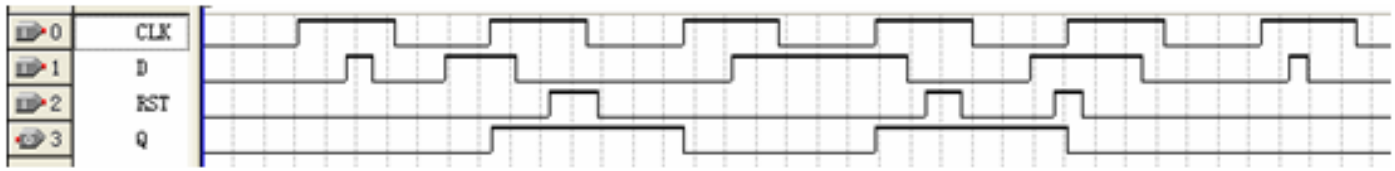


图 3-19 含同步清 0 的 D 触发器的仿真波形

3.2 时序模块及其Verilog表述

3.2.4 同步复位型触发器及其Verilog表述

【例 3-16】

```
module DFF2(CLK,D,Q,RST);      //含有同步清 0 的 D 触发器
   output Q ;
   input CLK,D,RST ;
   reg Q;
   always @(posedge CLK )     //注意，敏感信号表中只放了对 CLK 上升沿的敏感表述
       if (RST==1)  Q = 0;     //当 CLK 有上升沿时，若 RST=0，则 Q 被清 0
  else if (RST==0)  Q = D;     //当 CLK 有上升沿时，若 RST=1，则 Q 被更新为 D 值
         else       Q = Q;     //否则保持 Q 原值。此句可以不要。
endmodule
```

3.2 时序模块及其Verilog表述

3.2.5 异步复位型锁存器及其Verilog表述

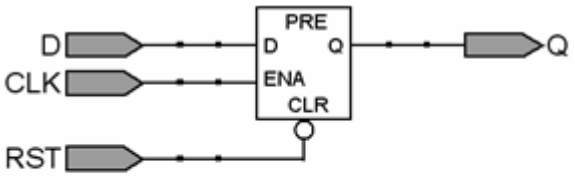


图 3-20 含异步清 0 的锁存器

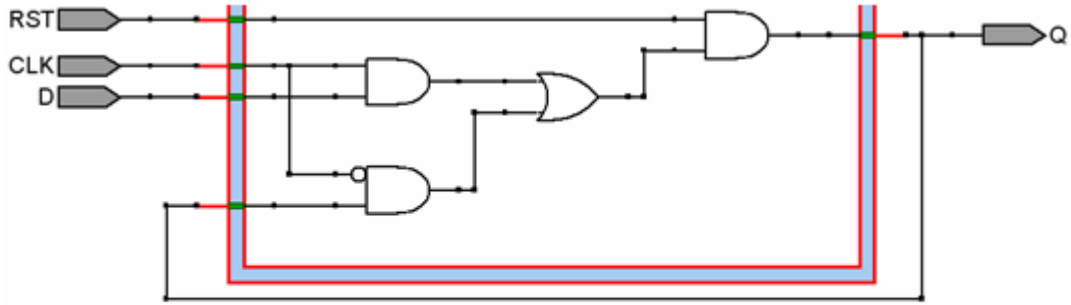


图 3-21 含异步清 0 锁存器的逻辑电路图

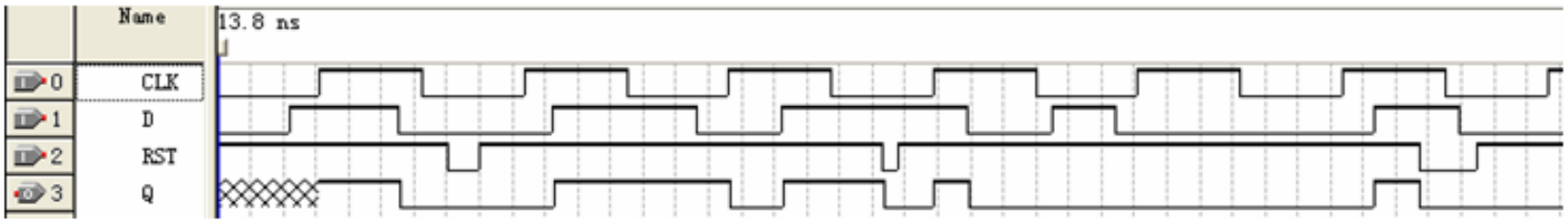


图 3-22 含异步清 0 的锁存器的仿真波形

3.2 时序模块及其Verilog表述

3.2.5 异步复位型锁存器及其Verilog表述

【例 3-17】

```
module LATCH3(CLK,D,Q,RST);
   output Q ;
   input CLK,D,RST;
   assign Q = (!RST)? 0:(CLK ? D:Q);
endmodule
```

【例 3-18】

```
module LATCH4(CLK,D,Q,RST);
    output Q ; input CLK,D,RST;
    reg Q;
    always @(D or CLK or RST)
            if(!RST)  Q<=0;
       else if(CLK)   Q<=D;
endmodule
```

3.2 时序模块及其Verilog表述

3.2.6 Verilog的时钟过程表述的特点和规律

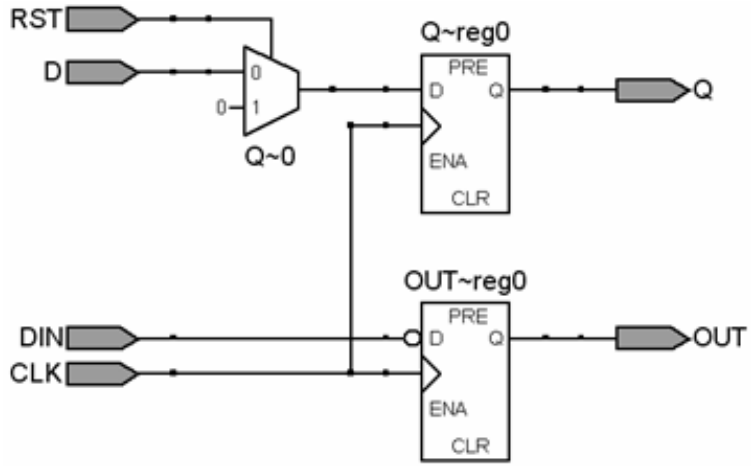


图 3-23 例 3-19 的 RTL 图

【例 3-19】

```
module DFF5(CLK,D,Q,RST,DIN,OUT);
    output Q,OUT ;
    input CLK,D,RST,DIN ;
    reg Q,OUT;
    always @(posedge CLK )
      begin
        OUT = !DIN ;
        if (RST==1)  Q=0;          //当 CLK 有上升沿时，若 RST=1，则 Q 被清 0
        else if (RST==0)  Q=D;//当 CLK 有上升沿时，若 RST=1，则 Q 被更新为 D 值
          else      Q=Q;          //否则保持 Q 原值
      end
    endmodule
```

3.2 时序模块及其Verilog表述

3.2.7 异步时序模块的Verilog表述

【例 3-20】

```
module AMOD(D,A,CLK,Q);                    //含有两个过程语句的异步时序电路
    output Q ;  input A,D,CLK;    reg Q,Q1;
    always @(posedge CLK)                  //过程 1
      begin  Q1 = ~(A | Q);  end
    always @(posedge Q1 )                  //过程 2，将过程 1 的输出作为时钟信号
      begin  Q =  D;          end
endmodule
```

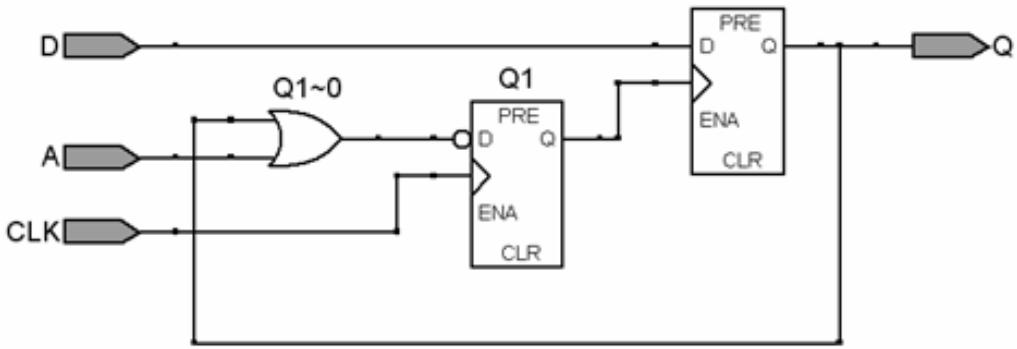


图 3-24　例 3-20 的异步时序电路图

3.3 二进制计数器及其Verilog设计

3.3.1 4位二进制计数器及其Verilog表述

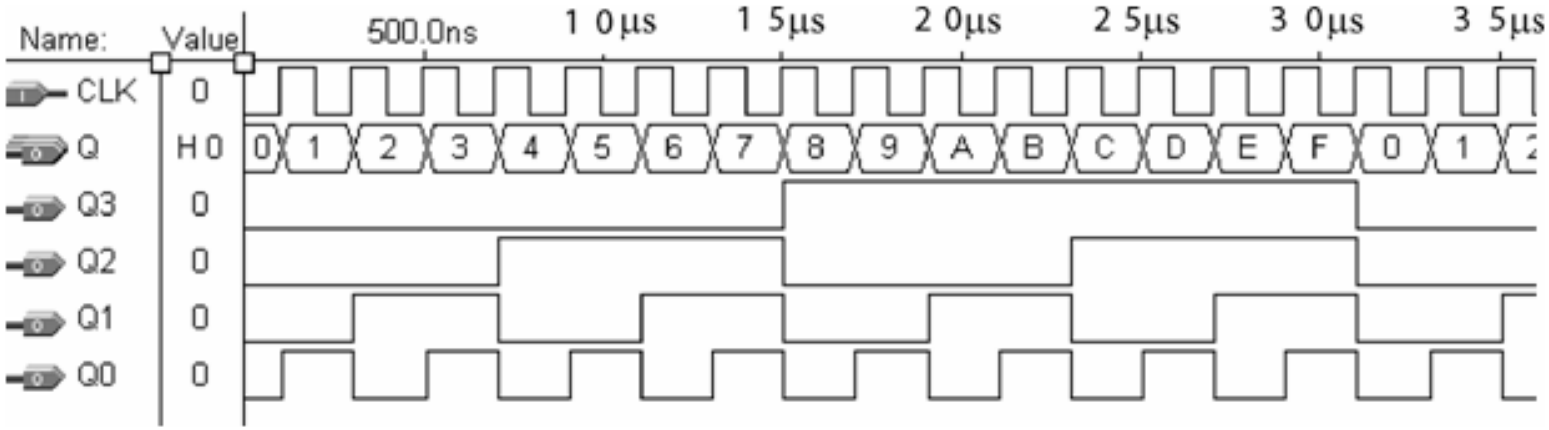


图 3-25　4 位加法计数器工作时序

3.3 二进制计数器及其Verilog设计

3.3.1 4位二进制计数器及其Verilog表述

【例 3-21】

```
module CNT4(CLK,Q);                    //最简单的 4 位二进制加法计数器
     output [3:0] Q ;   input  CLK;
        reg [3:0] Q1 ;                 //定义一个内部 4 位寄存节点
    always @(posedge CLK)
       begin  Q1 <= Q1+1 ; end   //CLK 有上跳沿时，Q1 累加 1，否则保持
    assign Q=Q1;                       //将 Q1 数据向端口 Q 输出
endmodule
```

3.3 二进制计数器及其Verilog设计

3.3.1 4位二进制计数器及其Verilog表述

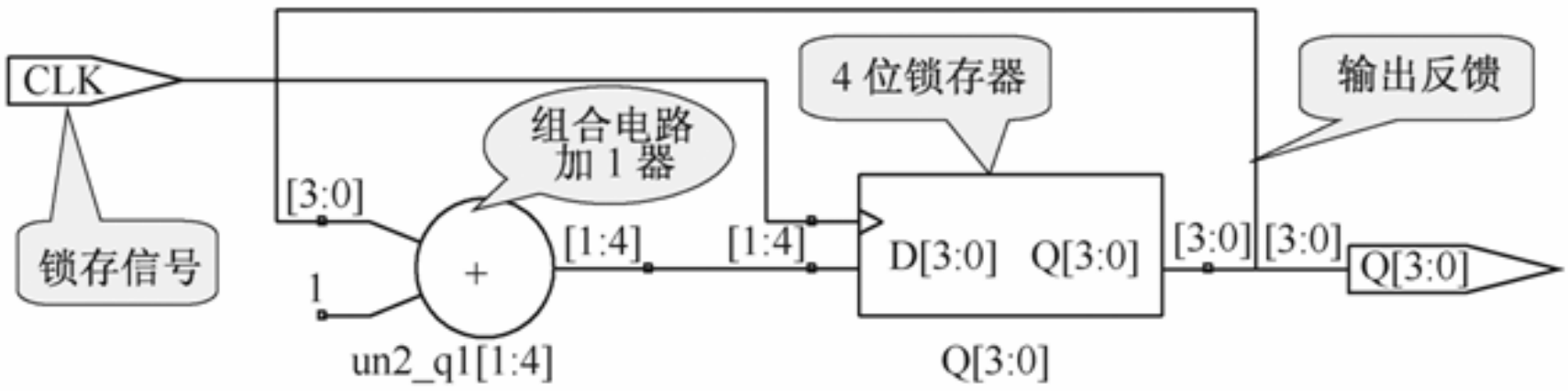


图 3-26 4 位加法计数器 RTL 电路（Symplify 综合）

3.3.2 功能更全面的计数器设计

【例 3-22】

```
module CNT10 (CLK,RST,EN,LOAD,COUT,DOUT,DATA);
   input CLK,EN,RST,LOAD ; // 时钟，时钟使能，复位，数据加载控制信号输入口;
    input [3:0] DATA ;        // 4 位并行加载数据输入口
   output [3:0] DOUT ;       // 计数数据输出信号口
    output COUT ;             // 计数进位输出
   reg [3:0] Q1 ;     reg COUT ;
   assign DOUT = Q1;         // 将内部寄存器的计数结果输出至 DOUT
   always @(posedge CLK or negedge RST) begin  //时序过程
           if (!RST)   Q1 <= 0;      //RST=0 时，对内部寄存器单元异步清 0
     else  if (EN)  begin      //同步使能 EN=1，则允许加载或计数
             if (!LOAD)      Q1 <= DATA; //当 LOAD=0，向内部寄存器加载数据
         else  if (Q1<9)  Q1 <= Q1+1; //当 Q1 小于 9 时，允许累加
         else   Q1 <= 4'b0000; end    //否则一个时钟后清 0 返回初值
           end
   always @(Q1)                          //组合电路之过程
     if (Q1==4'h9)  COUT = 1'b1;  //当 Q1=1001 时，COUT 输出进位标志 1
       else        COUT = 1'b0;    //否则，输出进位标志 0
   endmodule
```

3.3 二进制计数器及其Verilog设计

3.3.2 功能更全面的计数器设计

表 3-4　不等式操作符

等式操作符	含义	操作示例
>	大于	(A < B)= 0;　(A > B)= 1;
<	小于	(A < 20)= 1;　(A > 12) =1;
<=	小于或等于	(A >= 14) = 0;　(A <= 13) = 1;
>=	大于或等于	注，以上示例中，设 A=4'B1101 ;　B=4'B0110

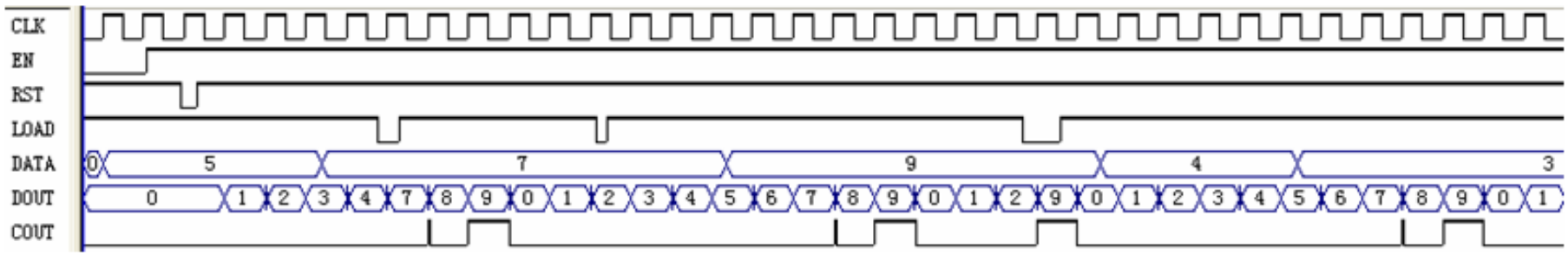


图 3-27　例 3-22 的仿真波形图

3.3 二进制计数器及其Verilog设计

3.3.2 功能更全面的计数器设计

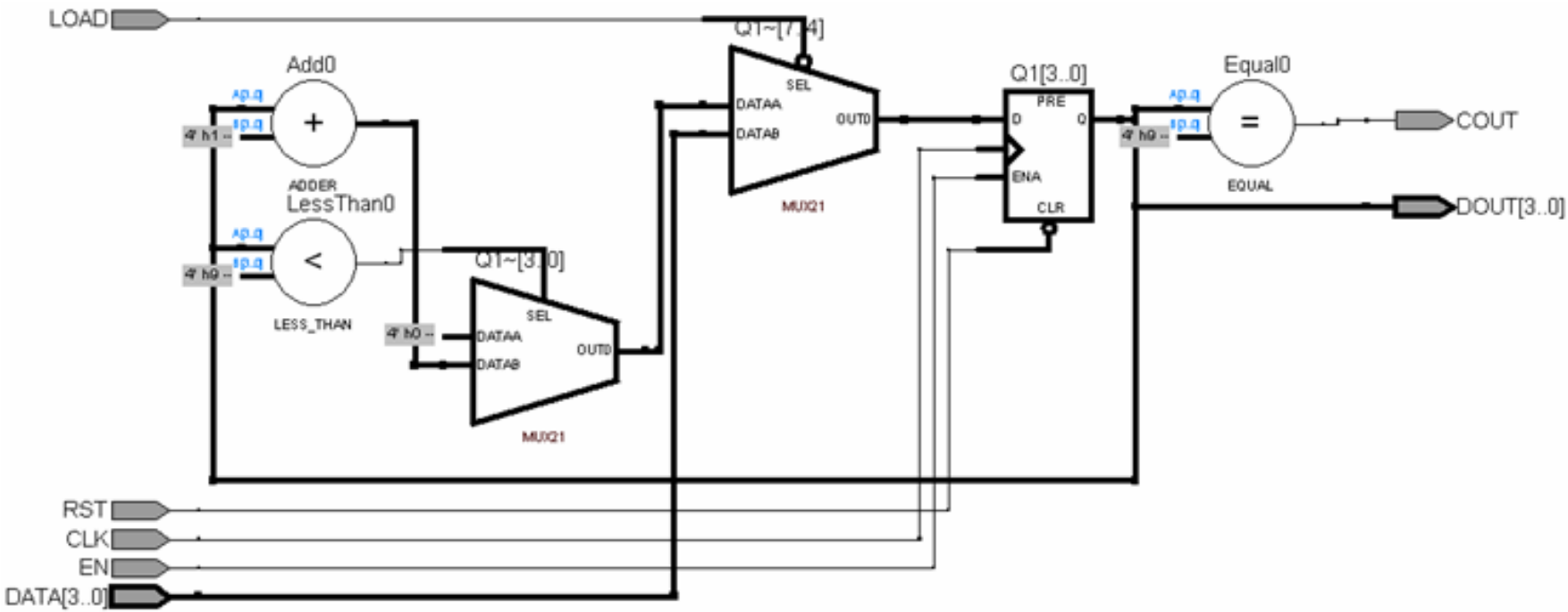


图 3-28　QuartusII 对例 3-22 综合和后得到的 RTL 图

习　题

3-6

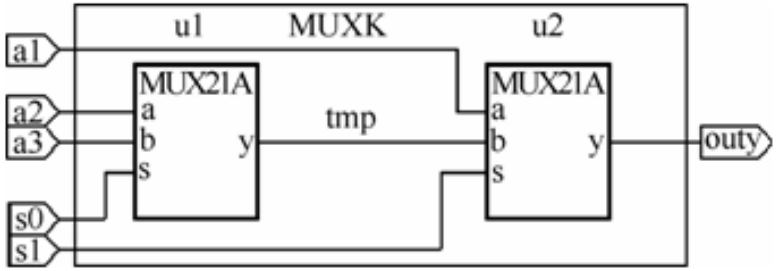


图 3-29　含 2 选 1 多路选择器的模块

习　题

3-7

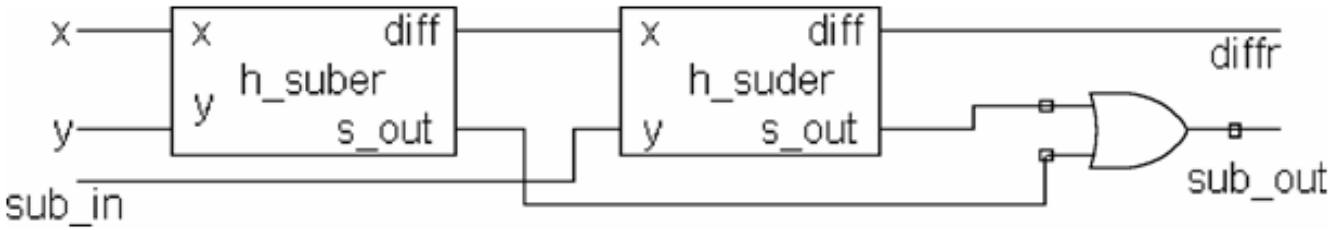


图 3-30　　1 位全减器

习　题

3-13

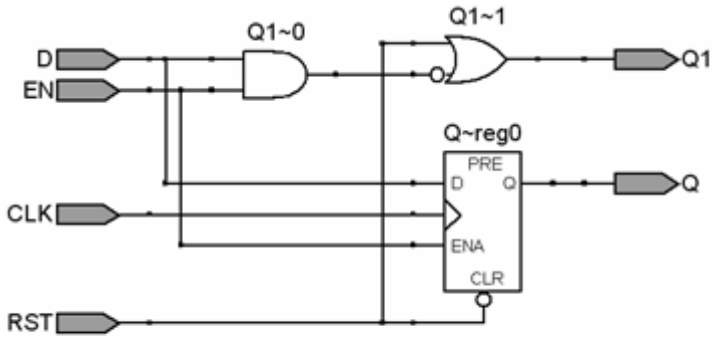


图 3-31　RTL 图 1

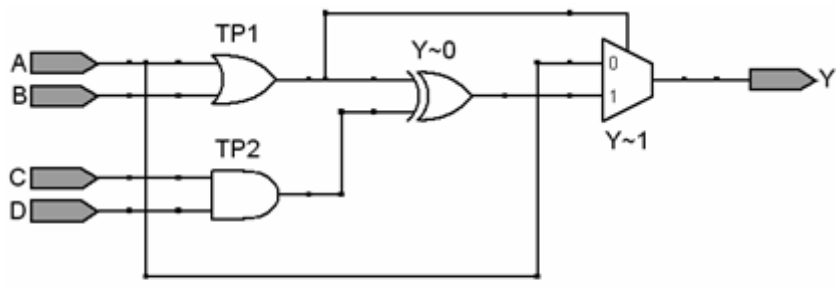


图 3-32　RTL 图 2

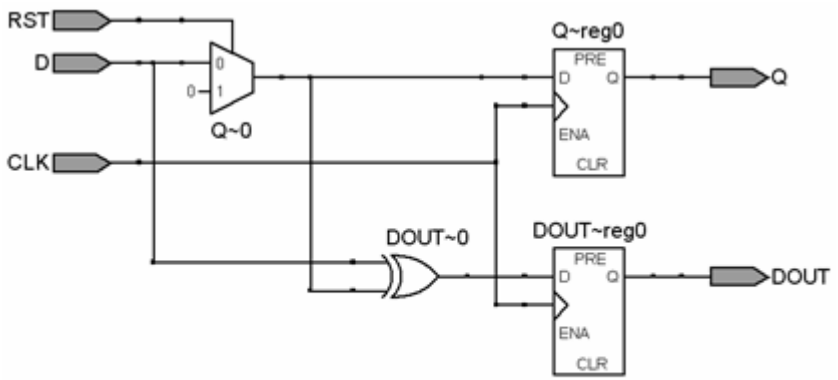


图 3-33　RTL 图 3

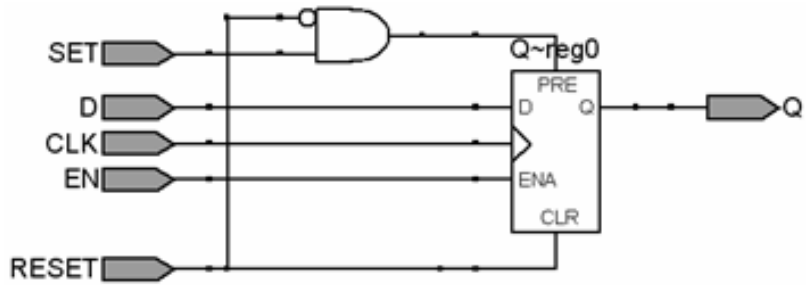


图 3-34　RTL 图 4