The background features five light purple circles. One circle is empty and contains the text '第3章'. The other four circles are solid and are positioned around the first one, two above and two below.

# 第3章

# Verilog行为语句

# 3.1 过程语句

## 3.1.1 always过程语句

`always @` (敏感信号及敏感信号列表或表达式)  
包括块语句的各类行为语句

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

# 3.1 过程语句

## 3.1.2 always语句在D触发器设计中的应用

### 【例 3-1】

```
module DFF1 (CLK, D, Q);  
    output Q ;  
    input  CLK, D ;  
    reg Q;  
    always @(posedge CLK )  
        Q <= D;  
endmodule
```

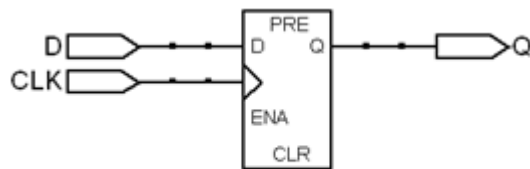


图 3-1 边沿触发型 D 触发器

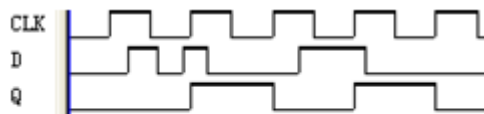


图 3-2 D 触发器时序波形

# 3.1 过程语句

## 3.1.3 多过程应用与异步时序电路设计

### 【例 3-2】

```
module AMOD(D,A,CLK,Q);  
    output Q; input A,D,CLK; reg Q,Q1;  
    always @(posedge CLK) //过程 1  
        Q1 <= ~(A | Q);  
    always @(posedge Q1) //过程 2  
        Q <= D;  
endmodule
```

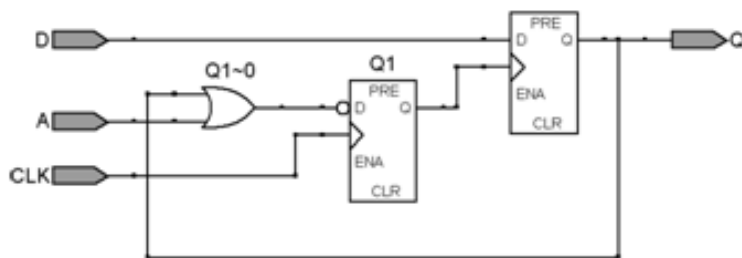


图 3-3 例 3-2 的时序电路图

# 3.1 过程语句

## 3.1.4 简单加法计数器及其Verilog表述

### 【例 3-3】

```
module CNT4 (CLK, Q);  
    output [3:0] Q; input CLK;  
    reg [3:0] Q1 ;  
    always @(posedge CLK)  
        Q1 = Q1+1 ;//注意赋值符号  
    assign Q=Q1;  
endmodule
```

### 【例 3-4】

```
module CNT4 (CLK, Q);  
    output [3:0] Q; input CLK;  
    reg [3:0] Q ;  
    always @(posedge CLK)  
        Q <= Q+1 ; //注意赋值符号  
endmodule
```

# 3.1 过程语句

## 3.1.4 简单加法计数器及其Verilog表述

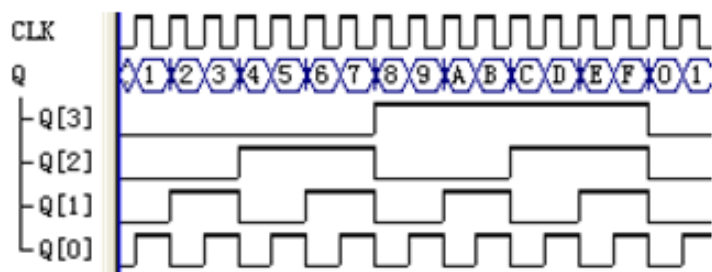


图 3-4 4 位加法计数器工作时序

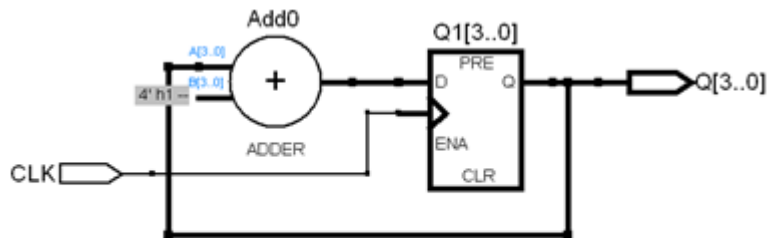


图 3-5 4 位加法计数器 RTL 电路图

# 3.1 过程语句

## 3.1.5 initial过程语句

```
initial  
begin 语句 1; 语句 2; ... end
```

```
initial $readmemh("RAM78_DAT.dat", mem );
```

### 【例 3-5】

```
`timescale 1ns/100ps //声明仿真时间单位是 1ns，仿真精度也是 100ps  
module test; //定义 testbench 名为 test 的测试模块  
reg A, B, C;  
initial //定义 initial 过程语句结构  
begin  
    A=0;B=1;C=0 //在过程中分别定义 A、B、C 在时刻 0 的初始值  
    #50 A=1;B=0; //经过 50ns 延时后，在仿真时刻 50ns 时 A 和 B 的输入值分别是 1,0  
    #50 A=0;C=1; //又经过 50ns 延时后，在时刻 100ns 时 A 和 C 的输入值分别是 0,1  
    #50 B=1; //再经过 50ns 延时后，在时刻 150ns 时 B 的输入值分别是 1  
    #50 B=0;C=0 //再经过 50ns 延时后，在时刻 200ns 时 B 和 C 的输入值都是 0  
    #50 $finish //又经过 50ns 延时后，结束。  
end  
endmodule  
`timescale 仿真时间单位/仿真精度
```

## 3.2 块语句

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



## 3.3 case条件语句

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

### 【例 3-6】

```
case ({s1, s0})
  2'b00 : y <= a;
  2'b01 : y <= b;
endcase
```

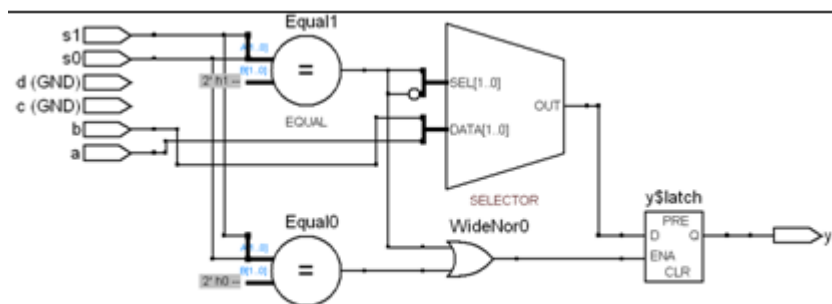


图 3-6 例 3-6 的 RTL 图

# 3.4 if条件语句

## 3.4.1 if语句的一般表述形式

```
if (条件表达式) begin 语句块; end //if语句类型1
if (条件表达式) begin 语句块1; end //if语句类型2
    else begin 语句块2; end
if (条件表达式1) begin 语句块1; end //if语句类型3
    else if (条件表达式2) begin 语句块2; end
    .
    .
    else if (条件表达式n) begin 语句块n; end
        else begin 语句块n+1; end
```

# 3.4 if条件语句

## 3.4.2 基于if语句的组合电路设计

```
if (S) Y = A; else Y = B;
```

### 【例 3-7】

```
module MUX4to1a (A,B,C,D,S1,S0,Y);  
  input A,B,C,D,S1,S0;    output Y;  
  reg [1:0] SEL ;    reg Y;  
  always @(A,B,C,D,SEL)  
  begin  
    SEL = {S1,S0};           //块语句起始  
    if (SEL==0) Y = A;       //把 s1, s0 并位为 2 元素矢量变量 SEL[1:0]  
    //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  
endmodule                   //块语句结束
```

# 3.4 if条件语句

## 3.4.2 基于if语句的组合电路设计

### 【例 3-8】

```
module CODER83 (DIN,DOUT);  
    output [0:2] DOUT;    input [0:7] DIN;  
    reg [0:2] DOUT;  
    always @(DIN)  
        casez (DIN)  
            8'b???????0 : DOUT<=3'b000;  
            8'b???????01 : DOUT<=3'b100;  
            8'b?????011 : DOUT<=3'b010;  
            8'b????0111 : DOUT<=3'b110;  
            8'b???01111 : DOUT<=3'b001;  
            8'b??011111 : DOUT<=3'b101;  
            8'b?0111111 : DOUT<=3'b011;  
            8'b01111111 : DOUT<=3'b111;  
            default : DOUT<=3'b000;  
        endcase  
endmodule
```

### 【例 3-9】

```
module CODER83 (DIN,DOUT);  
    output [0:2] DOUT;  
    input [0:7] DIN;  
    reg [0:2] DOUT;  
    always @(DIN)  
        if (DIN[7]==0) DOUT=3'b000;  
    else if (DIN[6]==0) DOUT=3'b100;  
    else if (DIN[5]==0) DOUT=3'b010;  
    else if (DIN[4]==0) DOUT=3'b110;  
    else if (DIN[3]==0) DOUT=3'b001;  
    else if (DIN[2]==0) DOUT=3'b101;  
    else if (DIN[1]==0) DOUT=3'b011;  
    else DOUT=3'b111;  
endmodule
```



图 3-7 例 3-8 和例 3-9 的时序仿真波形图

# 3.4 if条件语句

## 3.4.2 基于if语句的组合电路设计

表 3-1 8 线-3 线优先编码器真值表

输 入								输 出		
din0	din1	din2	din3	din4	din5	din6	din7	output0	output1	output2
x	x	x	x	x	x	x	0	0	0	0
x	x	x	x	x	x	0	1	1	0	0
x	x	x	x	x	0	1	1	0	1	0
x	x	x	x	0	1	1	1	1	1	0
x	x	x	0	1	1	1	1	0	0	1
x	x	0	1	1	1	1	1	1	0	1
x	0	1	1	1	1	1	1	0	1	1
0	1	1	1	1	1	1	1	1	1	1

注：表中的“x”为任意

# 3.4 if条件语句

## 3.4.3 基于if语句的时序电路设计

### 【例 3-10】

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



图 3-8 锁存器模块

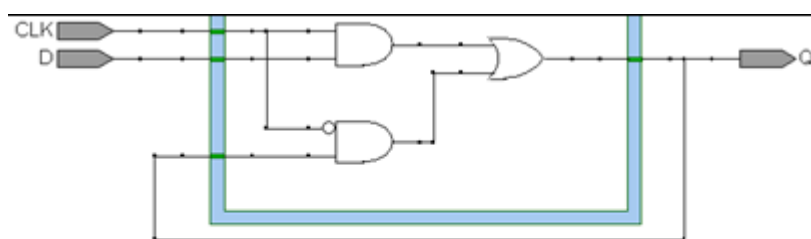


图 3-9 锁存器模块内部逻辑电路



图 3-10 例 3-10 锁存器的时序波形

# 3.4 if条件语句

## 3.4.4 含异步复位和时钟使能的D触发器的设计

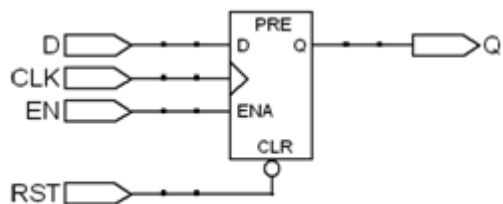


图 3-11 含使能和复位控制的 D 触发器

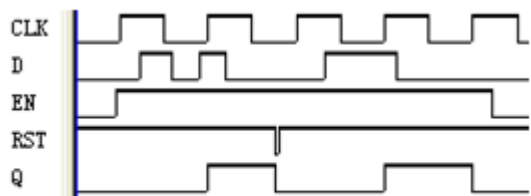


图 3-12 例 3-11 的时序图

### 【例 3-11】

```
module DFF2 (CLK, D, Q, RST, EN);  
    output Q;    input CLK, D, RST, EN;  
    reg Q;  
    always @(posedge CLK or negedge RST)  
        begin  
            if (!RST) Q <= 0;  
            else if (EN) Q <= D;  
        end  
endmodule
```

# 3.4 if条件语句

## 3.4.5 含同步复位控制的D触发器的设计

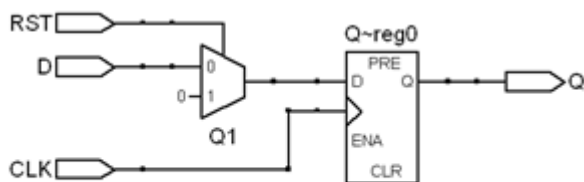


图 3-13 含同步清 0 控制的 D 触发器

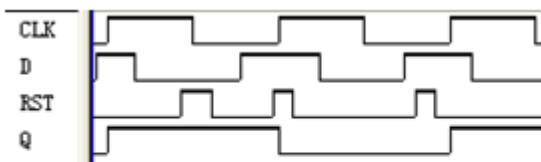


图 3-14 含同步清 0 控制 D 触发器的时序图



## 3.4 if条件语句

### 3.4.5 含同步复位控制的D触发器的设计

#### 【例 3-12】

```
module DFF3 (CLK, D, Q, RST);
    output Q ;
    input CLK, D, RST ;
    reg Q;
    always @(posedge CLK )
        if (RST==1) Q = 0;
    else if (RST==0) Q = D;
        else      Q = Q;
endmodule
```

#### 【例 3-13】

```
module DFF1 (CLK, D, Q, RST);
    output Q; input CLK, D, RST ;
    reg Q, Q1; //注意定义了 Q1 信号
    always @(RST) //纯组合过程
        if (RST==1) Q1=0;
            else Q1=D;
    always @(posedge CLK )
        Q <= Q1;
endmodule
```

# 3.4 if条件语句

## 3.4.6 含清0控制的锁存器的设计

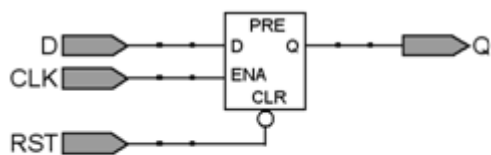


图 3-15 含异步清 0 的锁存器

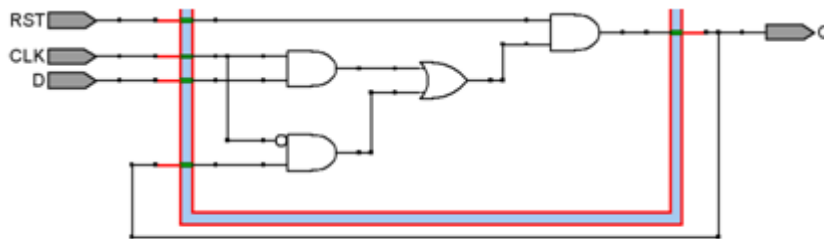


图 3-16 含异步清 0 锁存器的逻辑电路图

## 3.4 if条件语句

### 3.4.6 含清0控制的锁存器的设计

#### 【例 3-14】

```
module LATCH2 (CLK,D,Q,RST);  
    output Q ;    input CLK,D,RST;  
    assign Q = (!RST)? 0:(CLK ? D:Q);  
endmodule
```

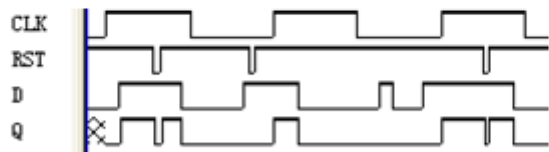


图 3-17 含异步清 0 的锁存器的仿真波形

#### 【例 3-15】

```
module LATCH3 (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
```

# 3.4 if条件语句

## 3.4.7 时钟过程表述的特点和规律

### 【例 3-16】

```
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;  
            else if(RST==0) Q=D;  
        end  
endmodule
```

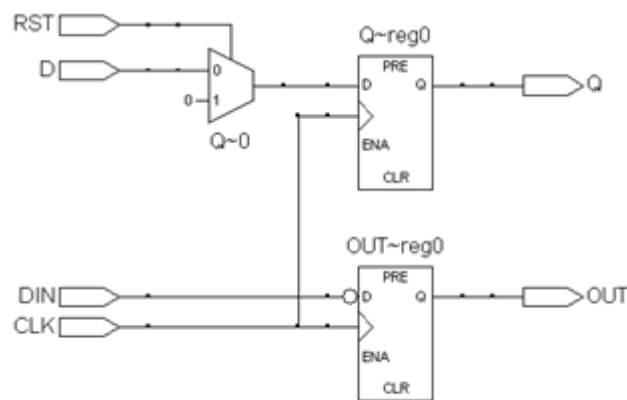


图 3-18 例 3-16 的 RTL 图

## 3.4 if条件语句

### 3.4.8 实用加法计数器设计

#### 【例 3-17】

```
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;
    input [3:0] DATA ;          // 4 位并行加载数据
    output [3:0] DOUT ;          // 4 位计数输出
    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) //时序过程
        begin
            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
        end
    always @(Q1) //组合过程
        if (Q1==4'h9) COUT = 1'b1; else COUT = 1'b0;
endmodule
```

# 3.4 if条件语句

## 3.4.8 实用加法计数器设计

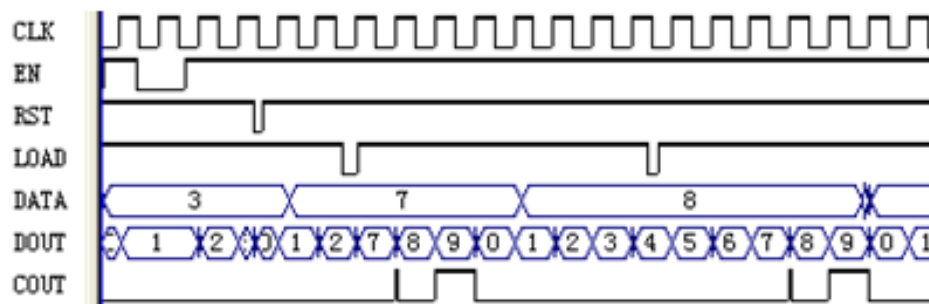


图 3-19 例 3-17 的仿真波形图

# 3.4 if条件语句

## 3.4.8 实用加法计数器设计

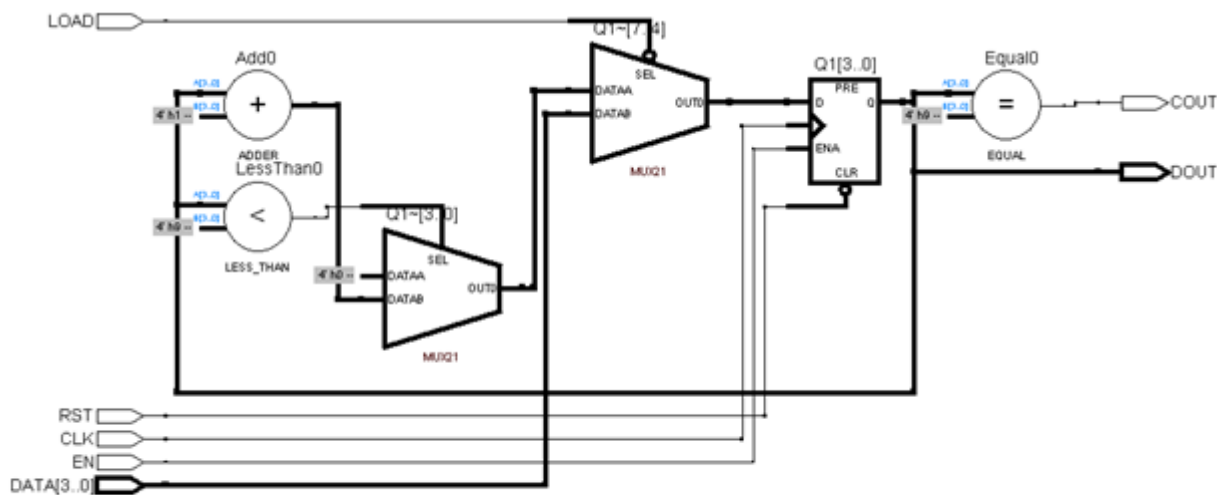


图 3-20 QuartusII 对例 3-17 综合和后得到的 RTL 电路图

# 3.4 if条件语句

## 3.4.9 含同步预置功能的移位寄存器设计

【例 3-18】

```
module SHFT1(CLK,LOAD,DIN,QB);  
    output QB; input CLK,LOAD;  
    input[7:0] DIN; reg[7:0] REG8;  
    always @(posedge CLK )  
        if (LOAD)    REG8<=DIN ;  
        else REG8[6:0]<=REG8[7:1];  
    assign QB = REG8[0] ;  
endmodule
```

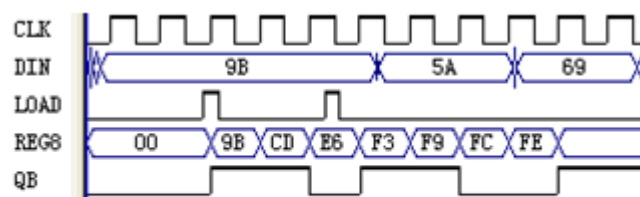


图 3-21 例 3-18 的工作时序图



## 3.4 if条件语句

### 3.4.10 关注if语句中的条件指示

【例 3-19】

```
module andd(A,B,Q);  
    output Q;  
    input A,B;  
    reg Q;  
    always @(A,B )  
        if (A==0)  
            if (B==0) Q=0;  
            else Q=1;  
endmodule
```

【例 3-20】

```
module andd(A,B,Q);  
    output Q;  
    input A,B; reg Q;  
    always @(A,B )  
        if (A==0)  
            begin  
                if (B==0) Q=0;  
            else Q=1; end  
endmodule
```

【例 3-21】

```
module andd(A,B,Q);  
    output Q; input A,B;  
    reg Q;  
    always @(A,B )  
        if (A==0)  
            begin if(B==0) Q=0;  
            end  
        else Q=1;  
endmodule
```

# 3.4 if条件语句

## 3.4.10 关注if语句中的条件指示

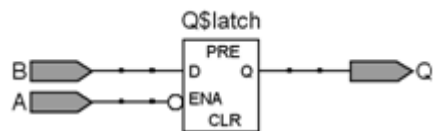


图 3-22 例 3-19/3-20 的 RTL 图

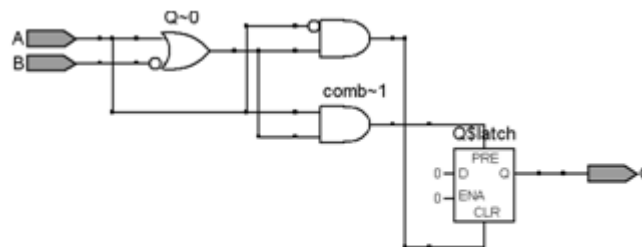
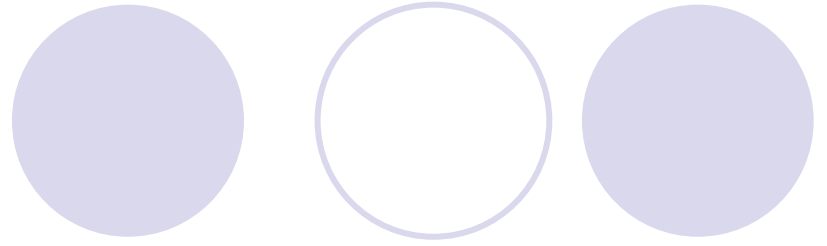


图 3-23 例 3-21 的 RTL 图

## 3.5 过程赋值语句



(1) 阻塞式赋值。

(2) 非阻塞式赋值。

# 3.6 循环语句

## 3.6.1 for语句用法

for (循环初始值设置表达式; 循环控制条件表达式; 循环控制变量增值表达式)  
begin 循环体语句结构 end

### 【例 3-22】

```
module MUL4B(R,A,B);  
    parameter S=4;  
    output[2*S:1] R ;  
    input[S:1] A,B ;  
    reg[2*S:1] R ;  
    integer i;  
    always @(A or B)  
    begin  
        R = 0 ;  
        for(i=1; i<=S; i=i+1)  
            if(B[i])  
                R=R+(A<<(i-1)); //左移(i-1)位  
        end  
    endmodule
```

### 【例 3-23】

```
module MULB (R,A,B);  
    parameter S=4;  
    output[2*S:1] R ;  
    input[S:1] A,B ;  
    reg[2*S:1] R,AT; reg[S:1] BT,CT;  
    always @(A,B) begin  
        R=0; AT = {{S{1'B0}},A};  
        BT = B; CT = S;  
        for(CT=S; CT>0; CT=CT-1)  
            begin if(BT[1]) R=R+AT;  
                AT = AT<<1; //左移 1 位  
                BT = BT>>1; //右移 1 位  
            end  
    end end  
endmodule
```

# 3.6 循环语句

## 3.6.1 for语句用法



图 3-24 四位乘法器时序仿真图

# 3.6 循环语句

## 3.6.2 while语句用法

**while** (循环控制条件表达式)  
begin 循环体语句结构 end

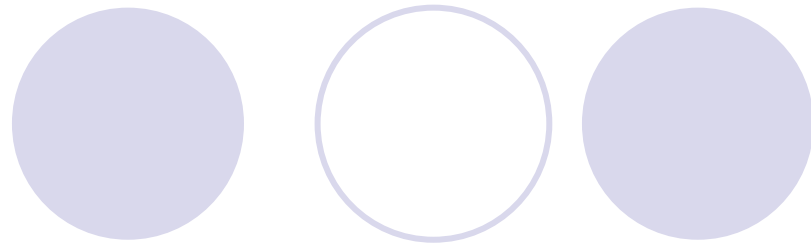
【例 3-24】

```
module MULT4B(A,B,R);  
  parameter S=4;  
  input[S:1] A,B;  
  output[2*S:1] R;  
  reg[2*S:1] R,AT;  
  reg[S:1] BT,CT;  
  always@(A or B) begin  
    R=0; AT={{S{1'b0}},A};  
    BT=B; CT=S;  
    while(CT>0) begin  
      if(BT[1]) R=R+AT; else R=R;  
    end  
    CT=CT-1; AT=AT<<1; BT=BT>>1;  
  end  
end  
endmodule
```

【例 3-25】

```
module MULT4B(R,A,B);  
  parameter S=4;  
  output [2*S:1] R; input [S:1] A,B ;  
  reg [2*S:1] TA,R;  
  reg [S:1] TB;  
  always @(A or B) begin  
    R = 0 ; TA = A ; TB = B ;  
    repeat(S) begin  
      if(TB[1]) begin R=R+TA; end  
      TA = TA<<1; //左移 1 位  
      TB = TB>>1; //右移 1 位  
    end  
  end  
end  
endmodule
```

## 3.6 循环语句



### 3.6.3 repeat语句用法

```
repeat (循环次数表达式)  
begin 循环体语句结构 end
```

### 3.6.4 forever循环语句

```
forever 语句;  
或 forever begin 语句; end
```

# 3.7 任务与函数语句

## 1.任务 (task) 语句

任务 (task) 定义语句格式

```
task <任务名>;  
    端口及数据类型声明语句  
    begin 过程语句; end  
endtask
```

任务调用格式

```
<任务名> (端口 1, 端口 2, ..., 端口 N);
```



# 3.7 任务与函数语句

## 1.任务 (task) 语句

### 【例 3-26】

```
module TASKDEMO (S,D,C1,D1,C2,D2); //主程序模块及端口定义
input S;  input[3:0] C1,D1,C2,D2;
output [3:0] D; //端口定义数目不受限制。
reg [3:0] out1,out2;
task CMP; //任务定义，任务名 CMP，此行不能出现端口定义语句
input [3:0] A,B; output [3:0] DOUT; //注意任务端口名的排序
begin if (A>B) DOUT= A; //任务过程语句描述一个比较电路
else DOUT=B; end //在任务结构中可以调用其他任务或函数，甚至自身。
endtask //任务定义结束
always @(*) begin //主程序过程开始
CMP(C1,D1,out1); //调用一次任务。任务调用语句只能出现在过程结构中
CMP(C2,D2,out2); end //第二次调用任务
assign D=S? out1:out2;
endmodule
```

# 3.7 任务与函数语句

## 2.函数（function）语句

函数定义语句格式

```
function <位宽范围声明> 函数名;  
    输入端口说明, 其他类型变量定义;  
    begin 过程语句; end  
endfunction
```

函数调用格式

```
<函数名> (输入参数 1, 输入参数 2, ...)
```

# 3.7 任务与函数语句

## 2.函数 (function) 语句

### 【例 3-27】

```
module CN (input [3:0] A, output [2:0] OUT);  
function [2:0] GP; //定义一个函数名为 GP 的函数, GP 同时作为位宽为 3 的输出参数  
input [3:0] M; //M 定义为此函数的输入值, 位宽是 4  
reg [2:0] CNT, N;  
begin CNT=0; for (N=0; N<=3; N=N+1) //for 循环语句  
if (M[N]==1) CNT=CNT+1; GP=CNT; end //含 1 的位个数累加  
endfunction  
assign OUT=(~|A) ? 0:GP(A); //主程序输入 A 或非缩位, 若为 1 则输出函数计数结果  
endmodule
```

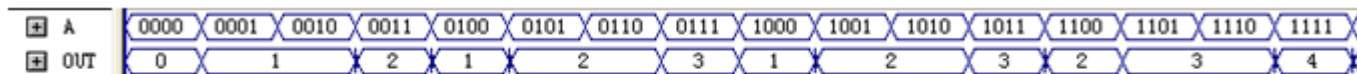


图 3-25 例 3-27 的仿真图

# 习题

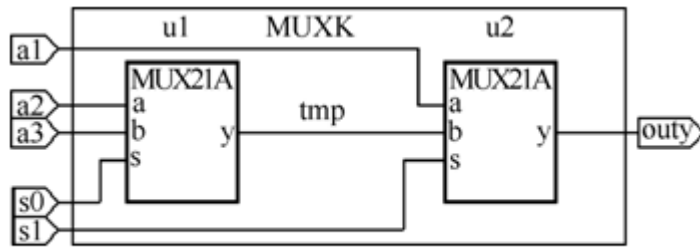


图 3-26 含 2 选 1 多路选择器的模块

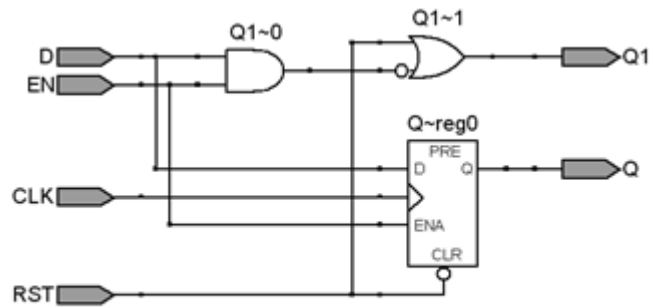


图 3-27 RTL 图 1

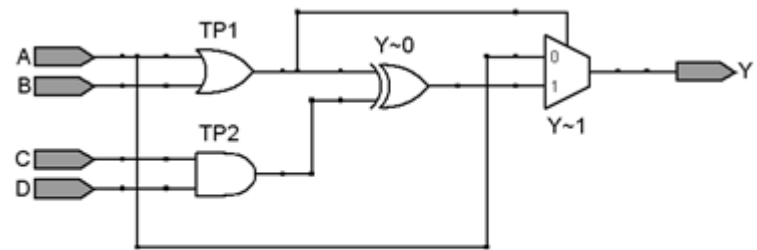


图 3-28 RTL 图 2

# 习 题

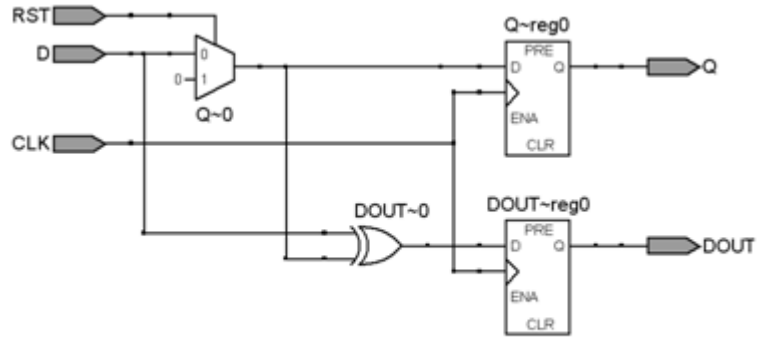


图3-29 RTL图3

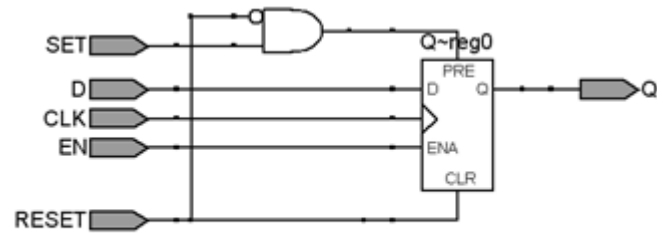


图3-30 RTL图4