实验二 时序逻辑设计

**一、实验目的**

1 掌握Verilog语言和Vivado、Logisim开发平台的使用；

2 掌握基础时序逻辑电路的设计和测试方法。

**二、实验内容（使用Logisim或Vivado实现）**

1 锁存器、触发器的设计与测试

2 寄存器、计数器的设计与测试

3 状态机的设计与测试

**三、实验要求**

1 掌握Vivado或Logisim开发工具的使用，掌握以上电路的设计和测试方法；

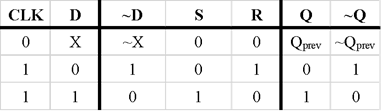
2 记录设计和调试过程（Verilog代码/电路图/表达式/真值表，Vivado仿真结果，Logisim验证结果等）；

3 分析Vivado仿真波形/Logism验证结果，注重输入输出之间的对应关系。

**四、实验过程及分析**

1 D锁存器

D锁存器既能够存储一位二进制状态，同时可以在时钟信号有效时更新输出状态（输出信号既有原变量Q也有反变量QN）。具体为：复位信号RST高有效时，输出Q置零（复位优先）；RST无效，使能信号EN无效时，保持状态，有效时更新Q=D。真值表：



设计代码：

module D\_latch(

Q,QN,D,EN,RST

);

input D,EN,RST;

output reg Q,QN;

always @(EN,RST,D) begin

if (RST) begin

Q=0;

QN=1;

end

else if (EN) begin

Q<=D;

QN<=~D;

end

end

endmodule

仿真代码：

module sim\_D\_latch(

);

reg D,EN,RST;

wire Q,QN;

initial begin

fork

D=1;EN=0;RST=0;

#10 RST=~RST;

#50 RST=~RST;

forever #20 D=~D;

forever #30 EN=~EN;

join

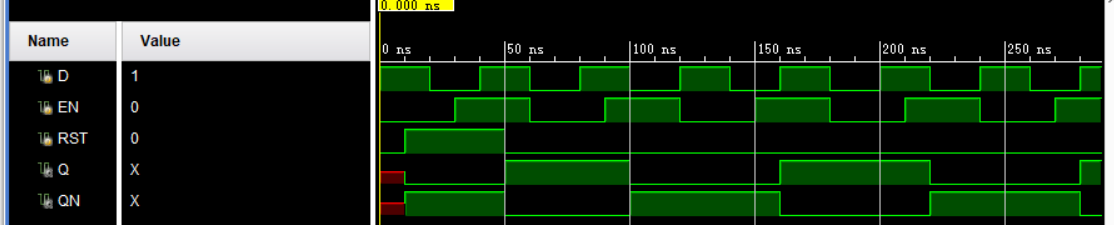
end

D\_latch test(Q,QN,D,EN,RST);

Endmodule

初始D=1，EN=0，RST=0.RST首先变为1，持续一段时间后变回0，这段时间Q应该为0。EN和D周期变化，因此在EN有效的时段输出Q与D一致，EN无效的时候Q应该和之前一段EN有效的时间内电平一致。QN始终与Q相反。

仿真结果

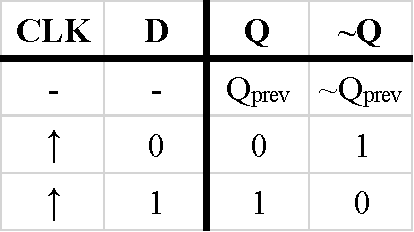


仿真波形是符合预期的，可以看出10-50nsRST有效时Q保持为0，此后只有当EN有效并且D与当前Q的状态不同时，Q输出翻转。比如50-60ns,Q=D=1，再保持30ns；90-100ns，Q=D=1；100-120nsEN还是有效但D翻转为0，Q跟着翻转；120-150nsEN无效，Q保持0；150-160nsEN有效但D=0，因此Q仍不变化；直到160-180ns，D为1，Q变为1；180-210nsEN为0，200-220nsD为1，因此在160-220nsQ都是1。

2 D触发器

相比D锁存器，D触发器的使能端由边沿触发，D触发器在时钟上升沿时，将D值复制到Q，在其他时间保持原有的状态。真值表：

-->



设计代码：

module D\_ff(

Q,QN,D,EN,RST,CLK

);

output reg Q,QN;

input D,EN,RST,CLK;

always @(posedge CLK) begin

//always @(posedge CLK,posedge RST) begin

if (RST) begin Q<=1'b0; QN<=1'b1;end

else if (EN) begin Q<=D;QN<=~D;end

end

endmodule

这里采用同步复位，清零或置位均在时钟CLK上升沿触发，输出Q和QN始终相反。在时钟上升沿时，若RST有效，则清零；RST无效时，若EN有效则执行置位，Q=D，QN=~Q，EN无效则保持状态。

仿真代码

module sim\_D\_ff(

);

reg D,EN,RST,CLK;

wire Q,QN;

initial begin

fork

D=1;CLK=0;RST=0;EN=0;

forever #30 CLK=~CLK;

forever #50 D=~D;

#40 EN=1;

#20 RST=1;

#60 RST=0;

#500 EN=0;

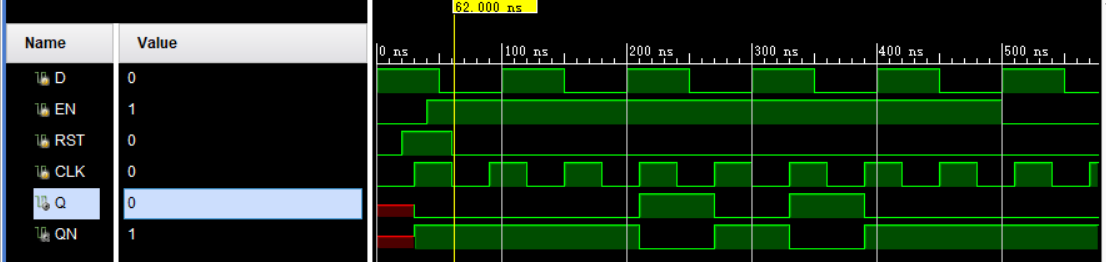
join

end

D\_ff test(Q,QN,D,EN,RST,CLK);

Endmodule

这里初始D=1，CLK=0，RST=0，EN=0，CLK和D都是周期变化的波形。RST在20-60ns的时间范围内有效，因此Q输出0；EN在40ns以后很长一段时间内都生效，但只有RST无效后，Q的输出与D信号波形一致。QN则始终与Q相反。



RST在20-60ns有效，EN在40-500ns有效，CLK上升沿在30ns，90ns，150ns……由于采用同步清零，第一个CLK上升沿到来之前Q，QN都是高阻态。第一个上升沿到来时RST已经有效，因此Q保持0。而90ns，150ns时D都是0，知道第四个CLK上升沿210ns时D为1，此时Q翻转为1，下一个上升沿来临时270ns，D为0，翻转；再下一个CLK上升沿在330ns，D为1，Q再变为1；390ns的上升沿D=0，Q翻转；此后的直到EN结束的上升沿时D都是0，因此Q保持0状态。这个波形符合设计预期。

3 寄存器

N位的寄存器由共享统一时钟的一排N个触发器组成，寄存器的所有位同时被更新。对于一个8位寄存器，由使能信号OE和时钟信号CLK共同控制。当三态输出控制OE为1时，控制寄存器输出8位高阻态；OE为0时寄存器正常工作，在CLK的上升沿将输入D读入到输出Q；OE为0，CLK不触发上升沿时，保存当前寄存器的8位二进制数状态。

设计代码：

module register(

Q,D,OE,CLK

);

parameter N=8;

output reg[N-1:0] Q;

input [N:1] D;

input OE,CLK;

always @(posedge CLK or posedge OE)

if (OE) Q<=8'bzzzz\_zzzz;

else Q<=D;

endmodule

仿真代码：

module sim\_register(

);

wire [7:0] Q;

reg [7:0] D;

reg CLK,OE;

integer i;

initial begin

fork

OE=1;CLK=0;D=8'b0000\_0001;

#20 OE=0;

forever #10 CLK=~CLK;

for (i=0;i<7;i=i+1) #10 D=D<<1;

join

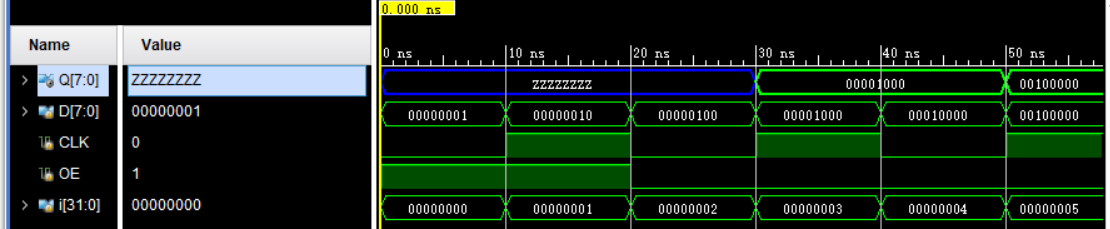
end

register test(Q,D,OE,CLK);

endmodule

这里OE初值为1，20ns后变位l0，因此前20nsQ输出高阻态。CLK初值为0，每隔10ns翻转一次，同时D的值随之左移一位（也即1的位置左移一位），共7次，CLK上升沿每20ns来临一次。因此，输出Q在结束高阻态后，每20ns跟随时钟上升沿更新一次，每次Q会左移2位。

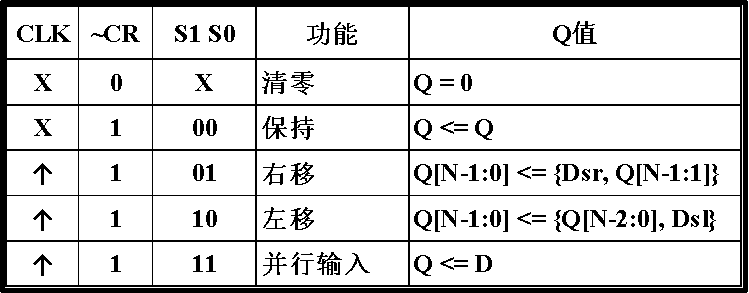
仿真结果：



仿真波形与预期结果相符。

移位寄存器在此基础上，除了可以并行置位、清零、存储之外，还可以根据输入的控制信号选择将当前存储的值左移或右移一位，空出的位置由特定的输入端置入。

-->



设计代码

module shift\_register(

S1,S0,D,Dsl,Dsr,Q,CLK,CR

);

parameter N=4;

input S1,S0;//control input

input Dsl,Dsr;//控制输入

input CLK,CR;//时钟、异步清零

input [N-1:0] D;//并行置入端

output reg [N-1:0] Q;

always @(posedge CLK or posedge CR)

if (CR)

Q<=0;

else

case ({S1,S0})

2'b00:Q<=Q;

2'b01:Q<={Dsr,Q[N-1:1]};//右移

2'b10:Q<={Q[N-2:0],Dsl};

2'b11:Q<=D;

endcase

endmodule

这里用Dsl，Dsr分别用来置入左移和右移后空出的位置，实现的功能与上表描述相同。

仿真代码：

module sim\_shift\_register(

);

reg S1,S0,Dsl,Dsr,CLK,CR;

reg[3:0] D;

wire[3:0] Q;

initial begin

fork

D=4'b0101;Dsl=0;Dsr=1;S1=1;S0=1;CLK=0;CR=0;

#10 CR=1;

#30 CR=0;

forever #10 CLK=~CLK;

forever #30 S0=~S0;

forever #60 S1=~S1;

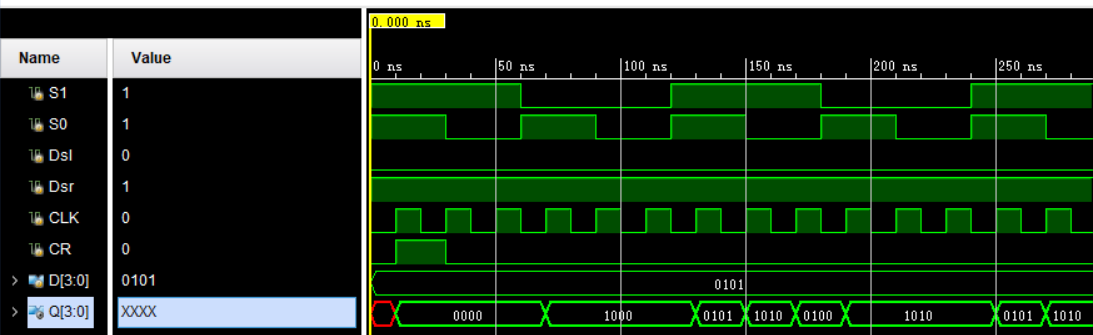
join

end

shift\_register test(S1,S0,D,Dsl,Dsr,Q,CLK,CR);

endmodule

仿真波形：

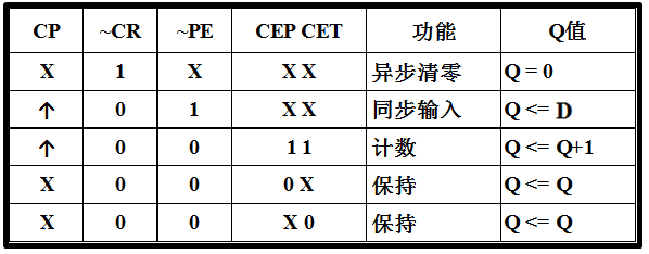


起初10nsCLK和CR都无效，Q为不确定态。10-30nsCR有效，Q置零。30-60ns左移并低位置零，所以Q仍是0000。60-90ns右移并高位置1，且只有70ns一个时钟上升沿，因此60-70ns，Q=0000，70-90ns，Q=1000；90-120ns，保持；120-150ns，并行置入，在130ns的上升沿触发，Q=D=0101，150ns时再次触发左移置零，Q=1010.仿真结果符合设计预期。

4 计数器

计数器由控制信号CEP，CET，PE，CLK和CR控制实现并行输入、清零、计数累加等功能，并行输入数据信号D，输出和D相同位数的二进制信号Q以及进位CT，具体实现为：

-->



设计代码：

module counter74x161(

CEP,CET,PE,CLK,CR,D,TC,Q

);

parameter N=8;

parameter M=34;

input CEP,CET,PE,CLK,CR;//控制信号

input [N-1:0] D;//并行输入

output reg TC;//进位输出

output reg [N-1:0] Q;//数据输出

wire CE;

assign CE=CEP&CET;//CE=1，计数器计数

always @(posedge CLK,negedge CR)

if (~CR) begin Q<=0;TC=0;end//异步清零

else if (~PE) Q<=D;//PE=0，同步装入输入数据

else if (CE) begin//计数

if (Q==M-1) begin

TC<=1;//进位

Q<=0;//计数归零

end

else Q<=Q+1;//计数

end

else Q<=Q;//保持

endmodule

这是一个模34计数器，实现功能与上表一致，在计数累加时如果达到模值则清零计数值，并把进位置1。

仿真代码：

module sim\_counter(

);

reg CEP,CET,PE,CLK,CR;

reg [7:0] D;

wire TC;

wire [7:0] Q;

initial begin

fork

D=32;

CR=1;#20 CR=0;#30 CR=1;

CLK=0;forever #10 CLK=~CLK;

CEP=0;#30 CEP=1;#200 CEP=0;

CET=0;#30 CET=1;#160 CET=0;

PE=1;#50 PE=0;#60 PE=1;

join

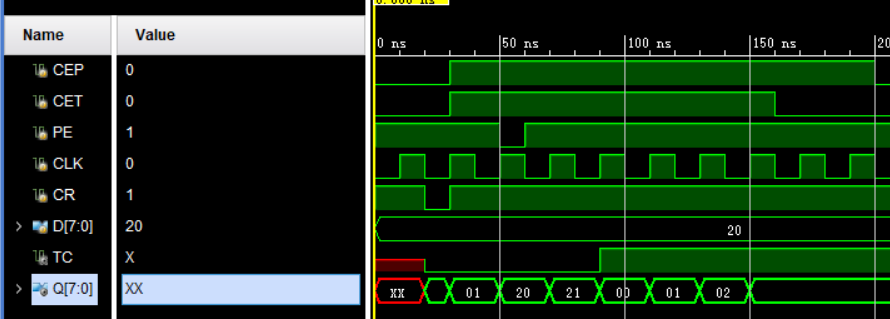
end

counter74x161 text(CEP,CET,PE,CLK,CR,D,TC,Q);

endmodule

仿真波形：

-->

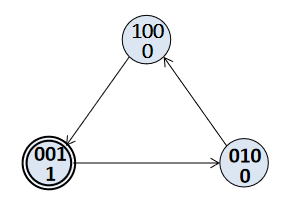


20ns前Q和TC的状态不确定。20ns时CR变为低有效，Q置零。第二个CLK上升沿到来时，CEP，CET都有效，计数加一。第三个上升沿到来时PE低有效，因此并行置入20H。再过2个CLK上升沿，计数器达到模值34，因此清零之后重新累加，并且从此开始进位TC从0变为1。仿真波形符合设计预期。

5 状态机

状态机由状态寄存器和组合逻辑电路构成，能够根据控制信号按照预先设定的状态进行状态转移，是协调相关信号动作、完成特定操作的控制中心。一个3分频计数器状态转换图如下：

-->



状态机在001->010->100三个状态之间按时间信号循环切换，并且在001状态时输出信号为1。

设计代码：

module FSM\_case(

input clk,input reset,output y

);

reg [2:0] state,nextstate;

always @(posedge clk,posedge reset)

if (reset) state=3'b001;

else state=nextstate;

always @(posedge clk)

case (state)

'b001:nextstate='b010;

'b010:nextstate='b100;

'b100:nextstate='b001;

default:nextstate='b001;

endcase

assign y=(state=='b001);

endmodule

reset在上升沿有效，将状态置为001。不进行置位时，会在时钟上升沿将状态state更新为缓存值nextstate，同时更新缓存的nextstate。状态间的切换关系与状态图一致。

仿真代码：

module sim\_FSM\_case(

);

reg clk,reset;

wire y;

initial begin

fork

clk=0;forever #10 clk=~clk;

reset=0;#10 reset=1;#20 reset=0;;

join

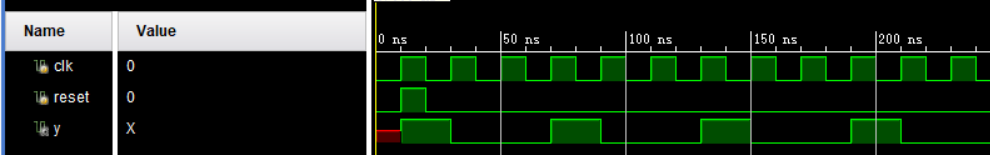
end

FSM\_case test(clk,reset,y);

Endmodule

仿真波形：

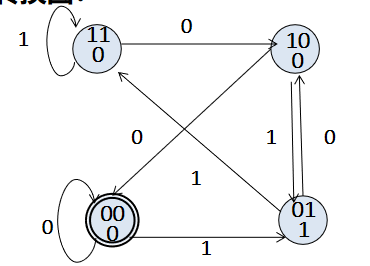
-



10ns，时钟信号clk上升沿和reset一同到来，state直接置001，因此输出y变为1.此后reset不再生效，每经过3个时钟上升沿状态又来到001，因此输出y每经过3个时钟上升沿就输出一次1。这个波形是符合设计期望的。

用状态机实现检测2进制数序列(如10011101)中的01子序列，检测到01后输出1。原始状态图：

-->



设计代码：

module FSM\_caseif(

input clk,input reset,input a,output y

);

reg [1:0] state,nextstate;

always @(posedge clk,posedge reset)

if (reset) state=2'b00;

else state=nextstate;

always @ (posedge clk)

case(state)

'b00:if(a) nextstate='b01;

else nextstate='b00;

'b01:if(a) nextstate='b11;

else nextstate='b10;

'b10:if(a) nextstate='b01;

else nextstate='b00;

'b11:if(a) nextstate='b11;

else nextstate='b10;

default:nextstate='b00;

endcase

assign y=(state=='b01);

endmodule

在reset的上升沿将状态置位00。在reset无效时，每个时钟上升沿到来时根据当前的状态和输入值确定下一状态，并会在下个时钟上升沿到来时更新当前状态。当state=01时，y输出为1。现态、次态和输入之间的转化关系与原始状态图相符。

仿真代码：

module sim\_FSM\_cf(

);

reg clk,reset,a;

wire y;

reg [7:0] string;

integer i;

initial begin

fork

i=7;

string=8'b1001\_1101;

clk=0;forever #10 clk=~clk;

reset=0; #10 reset=1;#20 reset=0;

join

end

always @(posedge clk)

begin

if (i>=0)

begin

a=string[i];

i=i-1;

end

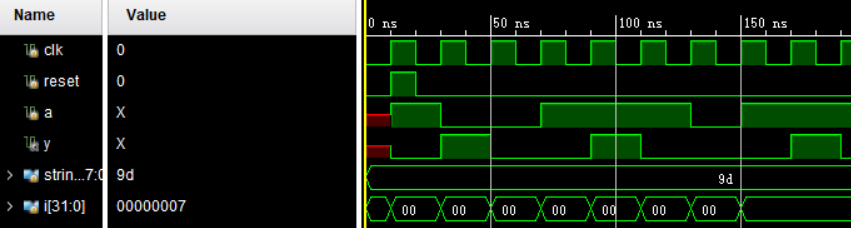
end

FSM\_caseif test(clk,reset,a,y);

Endmodule

仿真波形：

-->



用string存储检验的二进制串，并且在每个时钟上升沿逐个输入。10ns第一个时钟上升沿到来，同时reset也生效，a写入第一个字符1。此时state=00，neststate=01，所以y=0。直到下一个时钟上升沿到来，a的第一个字符写入完毕，要写入第二个字符0，state更新为之前缓存的nextstate（01），y就输出1。此后就可以看到，a的写入波形只要在两个时钟周期内出现01，y就会在延迟1个时钟周期后输出为1，比如在90ns，170ns时y的输出变为1。这些结果都符合设计预期。

**五、调试和心得体会**

1 在对设计中一些复位、清零的信号做仿真时，仿真代码的设计要在使这些信号有效一段时间从而能看到效果之后，再将这些信号置位无效。否则，在一些设计中这些信号比时钟信号优先触发，会一直生效，从而观察不到输出随时钟信号的变化波形。

2 状态机的各个状态间的切换可以在原始状态图的基础上简化，也就是合并等价的状态。