

# 数字逻辑实验报告（2）

|  |  |  |
| --- | --- | --- |
| **数字逻辑实验2** | | |
| **一、Verilog HDL设计数字逻辑电路50%** | **二、Verilog HDL设计较复杂数字逻辑电路50%** | **总成绩** |
|  |  |  |

评语：（包含：预习报告内容、实验过程、实验结果及分析）

教师签名：

**姓 名： 陈国轩**

**学 号： U201610489**

**班 级： 计算机（校交）1601**

**指 导 教 师： 徐有青**

**计算机科学与技术学院**

**2018 年 6 月 21 日**



**数字逻辑实验报告**

Verilog HDL设计数字逻辑电路预习报告

一、Verilog HDL设计数字逻辑电路

1、实验名称

Verilog HDL设计数字逻辑电路。

2、实验目的

要求同学用Verilog HDL设计数字逻辑电路，通过3个逻辑电路实验，并利用“Vivado 2015.2”软件进行“前、后”仿真检查电路设计，然后在“Xilinx NEXYS 4开发板”上操作、记录实验结果，最后验证设计是否达到要求。

通过以上设计、仿真、验证3个训练过程使同学们掌握Verilog HDL设计数字逻辑电路的基本方法，同时掌握如何避免锁存器的产生以及电路设计中的一致性问题的处理方法。

3、实验所用设备

Xilinx NEXYS 4开发板（芯片为XC7A100TCSG324-1，封装为CSG324，软件为Vivado 2015.2）1套。

4、实验内容

**（1）组合、时序逻辑电路的“always”设计**

**（A）用“always块”设计纯组合逻辑电路**

组合电路的一个特性是它的输出永远受输入变化的影响。也就是说组合电路绝不会保持它们以前的值，即输出不会出现锁存。

在使用always块中的case，if-else等语句设计纯组合逻辑电路时，要保证所有输入条件，其输出均有输出值，否则有可能会产生锁存器，导致“综合”出错。

**例如：**某题目要求使用Verilog设计实现一个纯组合逻辑电路的选择器，某同学设计了一个带“flag”标识的4选1的多路选择器，参见程序1-1所示，但是在“综合”时，报3个错误，即：“Place 30-574、Place 30-99、Common 17-69”。

具体要求：

（a）验证程序1-1在“综合”时，是否会出现上述问题；

（b）如果存在上述问题，请更正程序1-1，帮这位同学完成设计。

程序1-1 带“flag”标识的4选1的多路选择器

module mux\_latch(

input [3:0] data,

input [1:0] valid,

input flag,

output reg valid\_data);

initial begin

valid\_data=1'b0;

end

always @ (\*)

begin

case(valid)

2'b00 : begin if(flag) valid\_data = data[0];end

2'b01 : begin if(flag) valid\_data = data[1];end

2'b10 : begin if(flag) valid\_data = data[2];end

2'b11 : begin if(flag) valid\_data = data[3];end

endcase

end

endmodule

///////////////////////////\*.xdc文件//////////////////////////

set\_property PACKAGE\_PIN T16 [get\_ports flag]

set\_property IOSTANDARD LVCMOS33 [get\_ports flag]

set\_property PACKAGE\_PIN U8 [get\_ports {valid[0]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {valid[0]}]

set\_property PACKAGE\_PIN R7 [get\_ports {valid[1]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {valid[1]}]

set\_property PACKAGE\_PIN V7 [get\_ports {data[0]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {data[0]}]

set\_property PACKAGE\_PIN V6 [get\_ports {data[1]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {data[1]}]

set\_property PACKAGE\_PIN V5 [get\_ports {data[2]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {data[2]}]

set\_property PACKAGE\_PIN U4 [get\_ports {data[3]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {data[3]}]

set\_property PACKAGE\_PIN T8 [get\_ports valid\_data]

set\_property IOSTANDARD LVCMOS33 [get\_ports valid\_data]

**（B）用“always块”设计一个同步时序逻辑电路**

参见图1-1所示电路。



图1-1同步时序逻辑电路

具体要求：

利用Verilog HDL的“always块”设计该电路并进行仿真。

**（注意：阻塞与非阻塞赋值的不同，这种组合和时序的混合电路建议使用非阻塞赋值）**

**（2）脉冲异步计数器的设计**

分析如图1-2所示的脉冲异步计数器电路，完成如下内容：

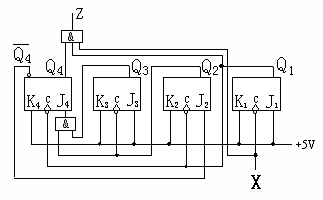
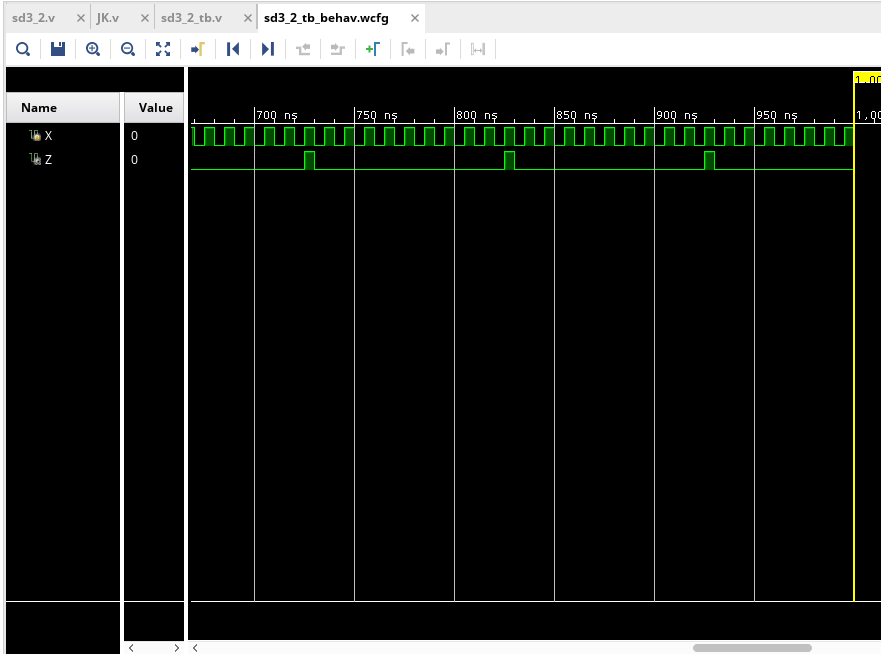


图1-2 脉冲异步计数器电路

（A）说明该计数器的模是多少？

（B）用Verilog DHL实现该电路，并通过仿真和在开发板上验证设计。

（A）如图所示 该计数器的模是10



**（3）Verilog数字电路设计中一致性问题**

分析如图1-3所示的电路，完成如下内容：

（A）程序1-2是对图1-3所示电路的描述，请用“**Behavior Simulation-它可以称为前仿真**”和“**非Behavior Simulation-它可以称为后仿真**”对程序1-2分别进行仿真，如果有错误，请更正程序1-2，并给出修正后的仿真结果。

（这就是：所谓前仿真和后仿真是否一致的问题）

q1

q2

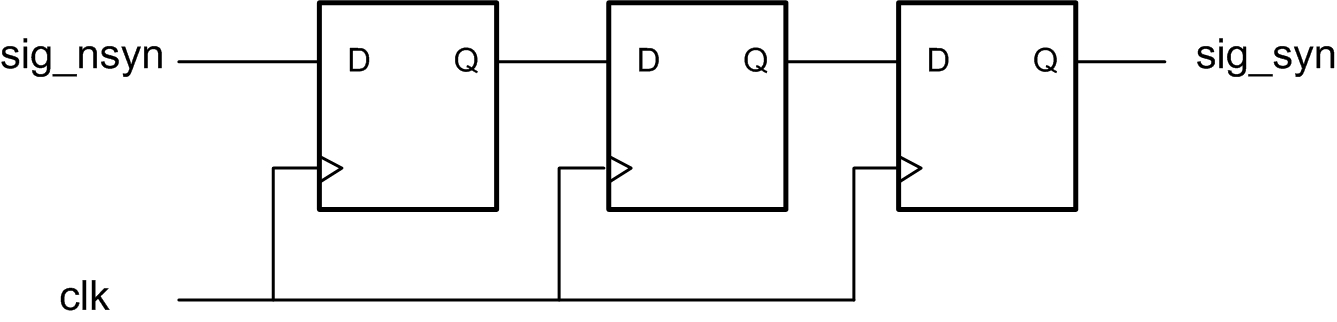


图1-3 3位移位寄存器电路

（B）NEXYS 4开发板提供了一个100Mhz的同步时钟，引脚绑定为“E3”，在更正后的程序1-2中添加一个时钟分频部分，并将降频后的时钟信号接到图1-3中的“clk”，编译成功后再下载到开发板上测试它。

结论：在今后的设计中要保持：“前仿真和后仿真以及下载验证都正确”哟。

程序1-2 3位移位寄存器电路

module pipe3b(

input sig\_nsyn,

input clk,

output q1,

output q2,

output sig\_syn

);

reg q1;

reg q2;

reg sig\_syn;

always@(posedge clk) q2=q1;

always@(posedge clk) sig\_syn=q2;

always@(posedge clk) q1=sig\_nsyn;

endmodule

5、实验方案设计

**（1）组合、时序逻辑电路的“always”设计方案**

（A）用“always块”设计纯组合逻辑电路

module mux\_latch(

input [3:0] data,

input [1:0] valid,

input flag,

output reg valid\_data);

initial begin

valid\_data=1'b0;

end

always @ (\*)

begin

case(valid)

2'b00 : begin

if(flag) valid\_data = data[0];

else valid\_data = 1'b0;

end

2'b01 : begin

if(flag) valid\_data = data[1];

else valid\_data = 1'b0;

end

2'b10 : begin

if(flag) valid\_data = data[2];

else valid\_data = 1'b0;

end

2'b11 : begin

if(flag) valid\_data = data[3];

else valid\_data = 1'b0;

end

endcase

end

endmodule

（B）用“always块”设计实现一个同步时序逻辑电路

1. 源程序

顶层模块：

module sd3\_1\_B(

input X,

input clk ,

output Y

);

wire Q0 ,Q1,nQ0,nQ1,t1,t;

//assign Y = ~(X & nQ1);

assign t1=1;

nand(Y,X,nQ1);

xor(t2,X,Q0);

T1 T\_trigger1(.t(t1) , .CP(clk) , .Q(Q0) , .nQ(nQ0));

T1 T\_trigger2(.t(t2) , .CP(clk) , .Q(Q1) , .nQ(nQ1));

endmodule

T触发器的实现

module T1(

input t,

input CP,

output reg Q,

output nQ

);

assign nQ = ~Q;

initial

begin

Q <=1'b0;

end

always @(posedge CP)

begin

if(t)

Q <= ~Q;

end

endmodule

（b）仿真程序

module sd3\_1\_B\_sim(

);

reg \_clk;

wire X,Y;

assign X=1;

sd3\_1\_B tb(.X(X) , .clk(\_clk) , .Y(Y));

// CLK shizhong(1,\_clk);

initial

\_clk <=1;

always #5

begin

\_clk = ~\_clk;

end

endmodule

**（2）脉冲异步计数器的分析和设计方案**

（A）源程序

顶层模块

module sd3\_2(

input X,

output Z

);

wire high5;

wire Q1,Q2,Q3,Q4,nQ4;

wire j4;

assign high5=1'b1;

JK JK1(.clk(X) , .j(high5) , .k(high5) , .q(Q1));

JK JK2(.clk(Q1) , .j(nQ4) , .k(high5) , .q(Q2));

JK JK3(.clk(Q2) , .j(high5) , .k(high5) , .q(Q3));

assign j4 = Q3 & Q2;

JK JK4(.clk(Q1) , .j(j4) , .k(high5) , .q(Q4) , .qb(nQ4));

and outZ(Z,Q4,Q1,X);

endmodule

JK触发器模块：

module JK( clk,j,k,q,qb );

input wire clk,j,k;

output wire qb;

output reg q;

assign qb = ~q;

initial

q <= 1'b0;

always@(posedge clk)begin

case({j,k})

2'b00:

q <= q;

2'b01:

q <= 0;

2'b10:

q <= 1;

2'b11:

q <= ~q;

endcase

end

endmodule

（B）仿真程序

module sd3\_2\_tb(

);

reg X=0;

wire Z;

sd3\_2 tb(.X(X) , .Z(Z));

always #5

begin

X = ~X;

end

endmodule

（C）引脚约束（绑定）程序

# Switches

#Bank = 34, Pin name = IO\_L21P\_T3\_DQS\_34, Sch name = SW0

set\_property PACKAGE\_PIN U9 [get\_ports {X]

set\_property IOSTANDARD LVCMOS33 [get\_ports {X}]

#Bank = 34, Pin name = IO\_25\_34, Sch name = SW1

# LEDs

#Bank = 34, Pin name = IO\_L24N\_T3\_34, Sch name = LED0

set\_property PACKAGE\_PIN T8 [get\_ports {Z}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {Z}]

**（3）Verilog设计中一致性问题的解决方案**

（A）源程序

module pipe3b(

input sig\_nsyn,

input clk,

output q1,

output q2,

output sig\_syn

);

reg q1;

reg q2;

reg sig\_syn;

always@(posedge clk) q2<=q1;

always@(posedge clk) sig\_syn<=q2;

always@(posedge clk) q1<=sig\_nsyn;

/\* always @(posedge clk)

begin

q1 <= sig\_nsyn;

q2 <= q1;

sig\_syn <= q2;

end

\*/

endmodule

（B）仿真程序

module pipe3b\_tb(

);

reg clk=0;

reg in=1;

wire out ,q1,q2;

pipe3b real\_pipe3b(.sig\_nsyn(in) , .clk(clk) , .q1(q1), .q2(q2) , .sig\_syn(out));

always #100

clk <= ~clk;

always #300

in = ~in;

endmodule



**数字逻辑实验报告**

Verilog HDL设计较复杂数字逻辑电路

二、Verilog HDL设计较复杂数字逻辑电路

1、实验名称

Verilog HDL设计较复杂数字逻辑电路。

2、实验目的

要求同学用Verilog HDL设计较复杂的数字逻辑电路，通过3个逻辑电路实验，并利用“Vivado 2015.2”软件进行“前、后”仿真检查电路设计，然后在“Xilinx NEXYS 4开发板”上操作、记录实验结果，最后验证设计是否达到要求。

通过以上设计、仿真、验证3个训练过程使同学们掌握Verilog HDL设计较复杂数字逻辑电路的基本方法，同时掌握“电路例化”、“模块化”的使用、异步时序逻辑电路的同步化处理以及用状态机设计控制电路。

3、实验所用组件

Xilinx NEXYS 4开发板（芯片为XC7A100TCSG324-1，封装为**CSG324**，软件为Vivado 2015.2）1套。

4、实验内容

**（1）4位二进制加法/减法计数器的设计**

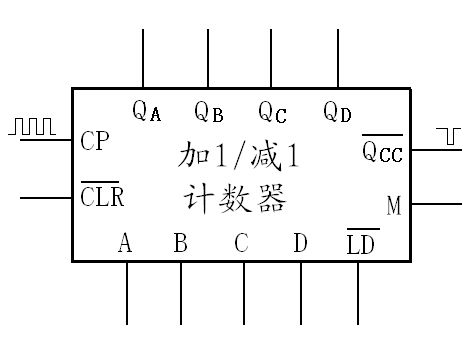
设计一个能清零、置数和进位/借位输出的加1/减1的4位二进制计数器，其结构框图如图2-1所示。

图2-1 4位二进制加法/减法计数器

电路输入为计数脉冲CP、工作模式选择M、预制初值D，C，B，A（其中D为高位，A为低位）和预制控制，清零端；



输出为计数值QD，QC，QB，QA（QD为高位，QA为低位）和进位/借位输出；



当为0时，电路输出清零；



预制控制=0时，将D、C、B、A的输入值送到计数器中，并立即在QD，QC，QB，QA中输出；



模式选择端M=1时加1计数；

当M=0时减1计数；

当CP端输入一个上升沿信号时进行一次计数；

计数有进位/借位时端输出一个负脉冲。



**注意：**用Verilog设计电路时，经常会遇到这样一些问题，例如：

（A）用两个always块对一个寄存器进行赋值，无论其中经过了怎样的条件判断，最终结果毫无疑问是将两个相独立的触发信号连在了寄存器的CLK端上，一个端口接入两信号，所以这样的语句是无法被综合成电路的。

（B）某电路如果有多个输入都可能引起输出值的改变，在使用“always”时，如果其触发条件为电路的“多个输入”时，如果语句的“并发性”处理不好，会造成系统编译成功、“行为仿真”也成功，但是系统生成不了“bit”文件。

解决方法：采用所谓“异步时序逻辑电路的同步化处理”，即：减少“always”的触发条件。

**具体要求：**

（A）用Verilog HDL实现该计数器，将之下载到开发板中进行验证；

（B）用已实现了的“4位二进制计数器”，采用“电路例化”或者“模块化”实现一个初值为2的模8计数器，并下载到开发板中进行验证；

（C）给出设计占用FPGA芯片的资源情况（希望越少越好）。

**（2）采用有限状态机(FSM) 实现序列检测器**

设计一个简单的状态机，其功能是检测一个串行的5位不可重叠的“10110”二进制序列检测器，当输入值出现“10110”时，给出输出标志。

具体要求如下：

（A）给出不可重叠的“10110”二进制序列的状态转移图；

（B）采用有限状态机“标准模板”来设计“10110”二进制序列检测器，在仿真正确后再下载到开发板中进行验证；

（C）给出设计占用FPGA芯片的资源情况（希望越少越好）。

**（所谓有限状态机“标准模板”请参考教材：夏雨闻。Verilog数字系统设计教程第3版。北京:北京航空航天大学出版社,2013.）**

**（3）3位二进制数值比较器的设计**

设计一个3位二进制数值比较器。当A>B时，F1=1，F2=F3=0；当A=B时， F2=1，F1=F3=0；当A<B时，F3=1，F1=F2=0。

具体要求：

（A）用Verilog HDL设计一个一位二进制数值比较器；

（B）用已实现了的一位二进制数值比较器，采用“电路例化”或者“模块化”实现一个3位二进制数值比较器；

（C）将所设计的电路下载到开发板上进行验证；

（D）给出设计占用FPGA芯片的资源情况（希望越少越好）。

5、实验方案设计

**（1）4位二进制加法/减法计数器的设计方案**

（A）模16加1/减1计数器

（a）源程序

`timescale 1ns / 1ps

module counter\_4bits(

input CP, //当CP端输入一个上升沿信号时进行一次计数；

input nCLR, //低电平有效 ， 为0时候清零

input M, //M为1时候加1计数，M为0时候减1计数

input nLD, //低电平有效 ， 预制控制=0时，将D、C、B、A的输入值送到计数器中，并立即在QD，QC，QB，QA中输出；

input A,

input B,

input C,

input D,

output reg QA=0,

output reg QB=0,

output reg QC=0,

output reg QD=0,

output reg nQCC=1//低电平有效 , 计数有进位/借位时端输出一个负脉冲

);

//assign {QD,QC,QB,QA} = {(~nLD)&A , (~nLD)&B , (~nLD)&C , (~nLD)&D};

always @(posedge CP)

begin

if(!nCLR) //如果清零

begin

{QD,QC,QB,QA} = 0;

nQCC = 1;

end

else //如果非清零

begin

if(!nLD)

begin

{QD,QC,QB,QA} = {A ,B,C,D};

nQCC = 1;

end

else

begin

if(M)//加1计数

begin

if({QD,QC,QB,QA} == 4'b1111)

nQCC = 0;

else

nQCC = 1;

{QD,QC,QB,QA} = {QD,QC,QB,QA} + 1;

end

else//M为0 减1计数

begin

if({QD,QC,QB,QA} == 4'b0000)

nQCC = 0;

else

nQCC = 1;

{QD,QC,QB,QA} = {QD,QC,QB,QA} - 1;

end

end

end

end

endmodule

（b）仿真程序

module counter\_4bits\_tb(

)

reg clk = 0;

//input CP, //当CP端输入一个上升沿信号时进行一次计数；

reg nCLR=1; //低电平有效 ， 为0时候清零

reg M = 1; //M为1时候加1计数，M为0时候减1计数

reg nLD=1; //低电平有效 ， 预制控制=0时，将D、C、B、A的输入值送到计数器中，并立即在QD，QC，QB，QA中输出；

reg A = 1;

reg B = 0;

reg C = 0;

reg D = 1;

wire QA ;

wire QB ;

wire QC ;

wire QD ;

wire nQCC;//低电平有效 , 计数有进位/借位时端输出一个负脉冲

counter\_4bits cnt(.CP(clk) , .nCLR(nCLR) , .M(M) , .nLD(nLD) ,

.A(A) , .B(B) , .C(C) , .D(D),

.QA(QA) , .QB(QB) , .QC(QC) , .QD(QD) ,

.nQCC(nQCC));

always #10

begin

clk <= ~clk;

end

always #160

begin

nCLR <= 0;

#10 nCLR <= 1;

end

always #320

begin

nLD <= 0;

#10 nLD <= 1;

end

always #500

begin

M <= ~M;

end

endmodule

（c）引脚约束（绑定）程序

#Bank = 34, Pin name = IO\_L21P\_T3\_DQS\_34, Sch name = SW0

set\_property PACKAGE\_PIN U9 [get\_ports {CP}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {CP}]

#Bank = 34, Pin name = IO\_25\_34, Sch name = SW1

set\_property PACKAGE\_PIN U8 [get\_ports {nCLR}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {nCLR}]

#Bank = 34, Pin name = IO\_L23P\_T3\_34, Sch name = SW2

set\_property PACKAGE\_PIN R7 [get\_ports {M}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {M}]

##Bank = 34, Pin name = IO\_L19P\_T3\_34, Sch name = SW3

set\_property PACKAGE\_PIN R6 [get\_ports {nLD}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {nLD}]

##Bank = 34, Pin name = IO\_L19N\_T3\_VREF\_34, Sch name = SW4

set\_property PACKAGE\_PIN R5 [get\_ports {A}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {A}]

##Bank = 34, Pin name = IO\_L20P\_T3\_34, Sch name = SW5

set\_property PACKAGE\_PIN V7 [get\_ports {B}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {B}]

##Bank = 34, Pin name = IO\_L20N\_T3\_34, Sch name = SW6

set\_property PACKAGE\_PIN V6 [get\_ports {C}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {C}]

##Bank = 34, Pin name = IO\_L10P\_T1\_34, Sch name = SW7

set\_property PACKAGE\_PIN V5 [get\_ports {D}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {D}]

# LEDs

#Bank = 34, Pin name = IO\_L24N\_T3\_34, Sch name = LED0

set\_property PACKAGE\_PIN T8 [get\_ports {QA}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {QA}]

#Bank = 34, Pin name = IO\_L21N\_T3\_DQS\_34, Sch name = LED1

set\_property PACKAGE\_PIN V9 [get\_ports {QB}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {QB}]

#Bank = 34, Pin name = IO\_L24P\_T3\_34, Sch name = LED2

set\_property PACKAGE\_PIN R8 [get\_ports {QC}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {QC}]

#Bank = 34, Pin name = IO\_L23N\_T3\_34, Sch name = LED3

set\_property PACKAGE\_PIN T6 [get\_ports {QD}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {QD}]

#Bank = 34, Pin name = IO\_L12P\_T1\_MRCC\_34, Sch name = LED4

set\_property PACKAGE\_PIN T5 [get\_ports {nQCC}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {nQCC}]

**（B）初值为2的模8计数器，要求：通过例化A【适当调整接口然后调用（A）实现的计数器， A中为了仿真方便，实现有些不同】来实现。**

（a）源程序

module counter\_mod8(

input CP,

input rst,

input M,

output [3:0]Q,

output nQCC

);

wire nLD , M , rst ,CP,nQCC;

wire [3:0]Q;

reg [3:0]DCBA;

counter\_4bits cnt\_mod8(.CP(CP),.CLR(1),.nLD(nLD),

.M(M),.DCBA(DCBA) , .Q(Q) );

assign nQCC = ~( (~Q[3] & ~Q[2] & Q[1] & ~Q[0] & ~M) | (Q[3] & ~Q[2] & ~Q[1] & Q[0] & M) );

assign nLD = ~( ~nQCC | ~rst);

always @(negedge CP)

begin

if(~rst)

begin

DCBA <= 4'b0010;

end

case(Q)

4'b0011:

if(~M) DCBA <= 4'b1001;

4'b1000:

if(M) DCBA <= 4'b0010;

default: DCBA <= 4'b0010;

endcase

end

endmodule

（b）仿真程序

module counter\_mod8\_tb(

);

reg M = 1;

reg CP = 1;

reg rst = 1;

wire [3:0]Q;

wire nQCC;

counter\_mod8 cnt\_mod8(.CP(CP) , .rst(rst) , .M(M) , .Q(Q) , .nQCC(nQCC));

always #500

begin

M <= ~M;

end

always #50

begin

CP <= ~CP;

end

endmodule

（c）引脚约束（绑定）程序

**# Switches**

**#Bank = 34, Pin name = IO\_L21P\_T3\_DQS\_34, Sch name = SW0**

**set\_property PACKAGE\_PIN U9 [get\_ports {CP}]**

**set\_property IOSTANDARD LVCMOS33 [get\_ports {CP}]**

**set\_property CLOCK\_DEDICATED\_ROUTE FALSE [get\_nets CP\_IBUF]**

**#Bank = 34, Pin name = IO\_25\_34, Sch name = SW1**

**set\_property PACKAGE\_PIN U8 [get\_ports {rst}]**

**set\_property IOSTANDARD LVCMOS33 [get\_ports {rst}]**

**#Bank = 34, Pin name = IO\_L23P\_T3\_34, Sch name = SW2**

**set\_property PACKAGE\_PIN R7 [get\_ports {M}]**

**set\_property IOSTANDARD LVCMOS33 [get\_ports {M}]**

**# LEDs**

**#Bank = 34, Pin name = IO\_L24N\_T3\_34, Sch name = LED0**

**set\_property PACKAGE\_PIN T8 [get\_ports {Q[0]}]**

**set\_property IOSTANDARD LVCMOS33 [get\_ports {Q[0]}]**

**#Bank = 34, Pin name = IO\_L21N\_T3\_DQS\_34, Sch name = LED1**

**set\_property PACKAGE\_PIN V9 [get\_ports {Q[1]}]**

**set\_property IOSTANDARD LVCMOS33 [get\_ports {Q[1]}]**

**#Bank = 34, Pin name = IO\_L24P\_T3\_34, Sch name = LED2**

**set\_property PACKAGE\_PIN R8 [get\_ports {Q[2]}]**

**set\_property IOSTANDARD LVCMOS33 [get\_ports {Q[2]}]**

**#Bank = 34, Pin name = IO\_L23N\_T3\_34, Sch name = LED3**

**set\_property PACKAGE\_PIN T6 [get\_ports {Q[3]}]**

**set\_property IOSTANDARD LVCMOS33 [get\_ports {Q[3]}]**

**#Bank = 34, Pin name = IO\_L12P\_T1\_MRCC\_34, Sch name = LED4**

**set\_property PACKAGE\_PIN T5 [get\_ports {nQCC}]**

**set\_property IOSTANDARD LVCMOS33 [get\_ports {nQCC}]**

**（2）采用有限状态机(FSM) 实现序列检测器的设计方案**

（A）串行5位不可重叠的“10110”二进制序列检测器的状态图

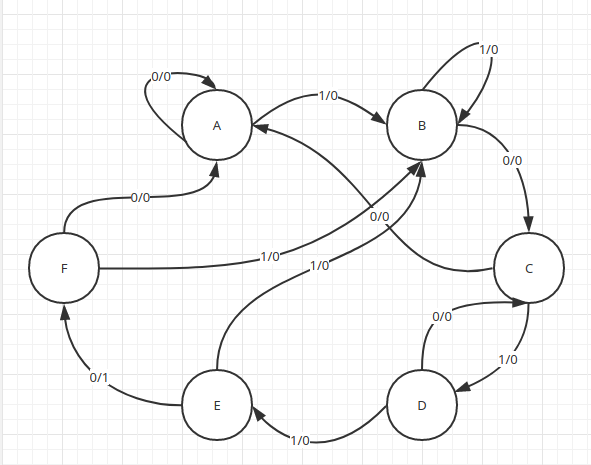


图 2-2 原始版本的状态转移图

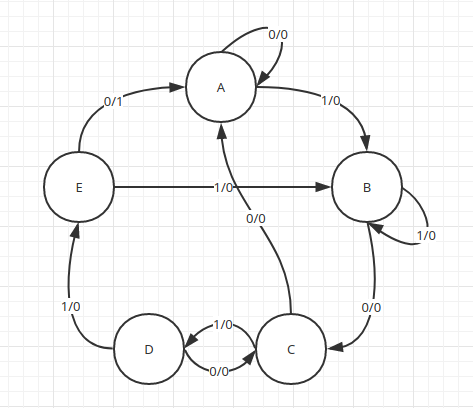


图 2-3 简化版状态转移图

（B）源程序

**`timescale 1ns / 1ps**

**module FSM(**

**input x,**

**input clk,**

**output reg y,**

**output [2:0] n\_state,**

**output clk\_N**

**);**

**parameter A=0,B=1,C=2,D=3,E=4,F=5;**

**reg [2:0] n\_state=0 , p\_state=0 ;**

**wire clk\_N;**

**divider div(.clk(clk) , .clk\_N(clk\_N));**

**initial**

**begin**

**p\_state <= A;**

**end**

**always @(negedge clk\_N)**

**begin**

**p\_state <= n\_state;**

**end**

**always @(p\_state or x)**

**begin:comb\_part**

**case(p\_state)**

**A: begin**

**y <= 0;**

**n\_state <= x ? B : A;**

**end**

**B: begin**

**y <= 0;**

**n\_state <= x ? B : C;**

**end**

**C: begin**

**y <= 0;**

**n\_state <= x ? D : A;**

**end**

**D: begin**

**y <= 0;**

**n\_state <= x ? E : C;**

**end**

**E: begin**

**y <= x ? 0 : 1;**

**n\_state <= x ? B : F;**

**end**

**F: begin**

**y <= 0;**

**n\_state <= x ? A : B;**

**end**

**endcase**

**end**

**endmodule**

**`timescale 1ns / 1ps**

**module divider(clk, clk\_N);**

**input clk; // 系统时钟**

**output reg clk\_N=0; // 分频后的时钟**

**parameter N = 100000000; // 1Hz的时钟,N=fclk/fclk\_N**

**reg [31:0] counter=0; /\* 计数器变量，通过计数实现分频。s**

**当计数器从0计数到(N/2-1)时，**

**输出时钟翻转，计数器清零 \*/**

**always @(posedge clk) begin // 时钟上升沿**

**// 功能实现**

**if( counter== (N/2 - 1) )**

**begin**

**clk\_N = ~clk\_N;**

**counter = 0;**

**end**

**counter = counter + 1;**

**end**

**endmodule**

（C）仿真程序

**module FSM\_tb(**

**);**

**wire y;**

**reg x;**

**reg clk=1;**

**wire[2:0] n\_state;**

**wire clk\_N;**

**FSM fsm( .x(x) , .clk(clk) , .y(y) , .n\_state(n\_state) , .clk\_N(clk\_N) );**

**always #5**

**begin**

**clk <= ~clk;**

**end**

**always**

**begin**

**x <= 1;**

**#10 x <= 0;**

**#10 x <= 1;**

**#10 x <= 1;**

**#10 x <= 0;**

**#10 x <= 1;**

**#10 x <= 0;**

**#10 x <= 1;**

**#10 x <= 0;**

**#10 x <= 1;**

**#10 x <= 1;**

**#10 x <= 0;**

**#10 x <= 1;**

**#10 x <= 1;**

**#10 x <= 0;**

**end**

**endmodule**

（D）引脚约束（绑定）程序

**set\_property PACKAGE\_PIN E3 [get\_ports clk]**

**set\_property IOSTANDARD LVCMOS33 [get\_ports clk]**

**create\_clock -add -name sys\_clk\_pin -period 10.00 -waveform {0 5} [get\_ports clk]**

**# Switches**

**#Bank = 34, Pin name = IO\_L21P\_T3\_DQS\_34, Sch name = SW0**

**set\_property PACKAGE\_PIN U9 [get\_ports {x}]**

**set\_property IOSTANDARD LVCMOS33 [get\_ports {x}]**

**# LEDs**

**#Bank = 34, Pin name = IO\_L24N\_T3\_34, Sch name = LED0**

**set\_property PACKAGE\_PIN T8 [get\_ports {y}]**

**set\_property IOSTANDARD LVCMOS33 [get\_ports {y}]**

**#Bank = 34, Pin name = IO\_L21N\_T3\_DQS\_34, Sch name = LED1**

**set\_property PACKAGE\_PIN V9 [get\_ports {n\_state[0]}]**

**set\_property IOSTANDARD LVCMOS33 [get\_ports {n\_state[0]}]**

**#Bank = 34, Pin name = IO\_L24P\_T3\_34, Sch name = LED2**

**set\_property PACKAGE\_PIN R8 [get\_ports {n\_state[1]}]**

**set\_property IOSTANDARD LVCMOS33 [get\_ports {n\_state[1]}]**

**#Bank = 34, Pin name = IO\_L23N\_T3\_34, Sch name = LED3**

**set\_property PACKAGE\_PIN T6 [get\_ports {n\_state[2]}]**

**set\_property IOSTANDARD LVCMOS33 [get\_ports {n\_state[2]}]**

**#Bank = 34, Pin name = IO\_L12P\_T1\_MRCC\_34, Sch name = LED4**

**set\_property PACKAGE\_PIN T5 [get\_ports {clk\_N}]**

**set\_property IOSTANDARD LVCMOS33 [get\_ports {clk\_N}]**

**（3）3位二进制数值比较器的设计方案**

**（A）一位二进制数值比较器**

（a）源程序

`timescale 1ns / 1ps

module cmp\_1bit(

input A,

input B,

output reg F1,

output reg F2,

output reg F3

);

always @(A or B)

begin

if(A > B)

{F1,F2,F3} <= 3'b100;

else if(A < B)

{F1,F2,F3} <= 3'b001;

else

{F1,F2,F3} <= 3'b010;

end

endmodule

（b）仿真程序

module cmp\_1bit\_tb(

);

wire F1,F2,F3;

reg A=0,B=0;

cmp\_1bit test( .A(A) , .B(B) , .F1(F1) , .F2(F2) , .F3(F3) );

always #5

begin

A <= ~A;

end

always #40

begin

B <= ~B;

end

endmodule

（c）引脚约束（绑定）程序

#Bank = 34, Pin name = IO\_25\_34, Sch name = SW1

set\_property PACKAGE\_PIN U8 [get\_ports {A}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {A}]

#Bank = 34, Pin name = IO\_L23P\_T3\_34, Sch name = SW2

set\_property PACKAGE\_PIN R7 [get\_ports {B}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {B}]

# LEDs

#Bank = 34, Pin name = IO\_L24N\_T3\_34, Sch name = LED0

set\_property PACKAGE\_PIN T8 [get\_ports {F1}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {F1}]

#Bank = 34, Pin name = IO\_L21N\_T3\_DQS\_34, Sch name = LED1

set\_property PACKAGE\_PIN V9 [get\_ports {F2}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {F2}]

#Bank = 34, Pin name = IO\_L24P\_T3\_34, Sch name = LED2

set\_property PACKAGE\_PIN R8 [get\_ports {F3}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {F3}]

**（B）3位二进制数值比较器，要求：通过例化A【调用（A）实现的一位二进制数值比较器】来实现。**

1. 源程序

`timescale 1ns / 1ps

module cmp\_3bits(

input A,

input B,

output reg F1,

output reg F2,

output reg F3

);

wire [2:0]A;

wire [2:0]B;

wire [2:0]f1;

wire [2:0]f2;

wire [2:0]f3;

cmp\_1bit \_0bit(.A(A[0]) , .B(B[0]) ,.F1(f1[0]) , .F2(f2[0]) , .F3(f3[0]));

cmp\_1bit \_1bit(.A(A[1]) , .B(B[1]) ,.F1(f1[1]) , .F2(f2[1]) , .F3(f3[1]));

cmp\_1bit \_2bit(.A(A[2]) , .B(B[2]) ,.F1(f1[2]) , .F2(f2[2]) , .F3(f3[2]));

always @(\*)

begin

if(f3[2] || f1[2])

{F1,F2,F3} <= f3[2] ? 3'B001 : 3'B100;

else if(f3[1] || f1[1])

{F1,F2,F3} <= f3[1] ? 3'B001 : 3'B100;

else if(f3[0] || f1[0])

{F1,F2,F3} <= f3[0] ? 3'B001 : 3'B100;

else

{F1,F2,F3} <= 3'b010;

end

endmodule

（b）仿真程序

`timescale 1ns / 1ps

module cmp\_3bits\_tb(

);

reg [2:0] A=3'b111,B=3'b000;

wire F1,F2,F3;

always #5

begin

A <= A-1;

end

always #40

begin

B <= B+1;

end

cmp\_3bits Test(.A(A),.B(B),.F1(F1) , .F2(F2) , .F3(F3));

endmodule

（c）引脚约束（绑定）程序

# Switches

#Bank = 34, Pin name = IO\_L21P\_T3\_DQS\_34, Sch name = SW0

set\_property PACKAGE\_PIN U9 [get\_ports {A[0]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {A[0]}]

#Bank = 34, Pin name = IO\_25\_34, Sch name = SW1

set\_property PACKAGE\_PIN U8 [get\_ports {A[1]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {A[1]}]

#Bank = 34, Pin name = IO\_L23P\_T3\_34, Sch name = SW2

set\_property PACKAGE\_PIN R7 [get\_ports {A[2]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {A[2]}]

##Bank = 34, Pin name = IO\_L19P\_T3\_34, Sch name = SW3

set\_property PACKAGE\_PIN R6 [get\_ports {B[0]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {B[0]}]

##Bank = 34, Pin name = IO\_L19N\_T3\_VREF\_34, Sch name = SW4

set\_property PACKAGE\_PIN R5 [get\_ports {B[1]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {B[1]}]

##Bank = 34, Pin name = IO\_L20P\_T3\_34, Sch name = SW5

set\_property PACKAGE\_PIN V7 [get\_ports {B[2]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {B[2]}]

# LEDs

#Bank = 34, Pin name = IO\_L24N\_T3\_34, Sch name = LED0

set\_property PACKAGE\_PIN T8 [get\_ports {F1}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {F1}]

#Bank = 34, Pin name = IO\_L21N\_T3\_DQS\_34, Sch name = LED1

set\_property PACKAGE\_PIN V9 [get\_ports {F2}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {F2}]

#Bank = 34, Pin name = IO\_L24P\_T3\_34, Sch name = LED2

set\_property PACKAGE\_PIN R8 [get\_ports {F3}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {F3}]

6、实验结果记录

**（1）4位二进制加法/减法计数器的实验结果记录**

**（A）给出Verilog设计的模16加1/减1计数器的电路图（**RTL Analysis下“Schematic”截图**）**

图2-4为4位二进制加法/减法计数器

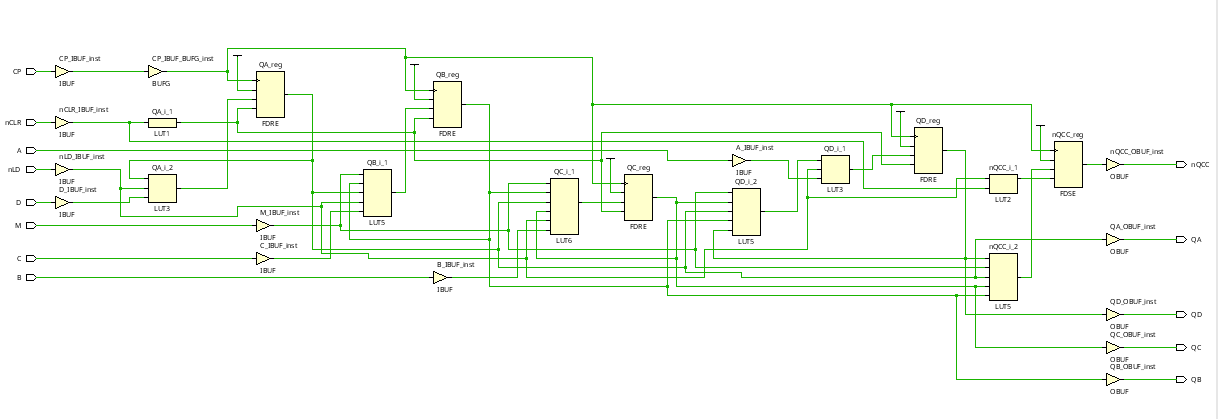


图2-4 4位二进制加法/减法计数器

**（B）初值为2的模8计数器仿真结果截图**

图2-5为初值为2的模8计数器仿真

M是计数器的模式，图中为1，加计数模式

CP是时钟沿，每个下降沿进行一次计数

Q是计数输出

rst是置位端

nQCC为进位退位端，低电平有效。

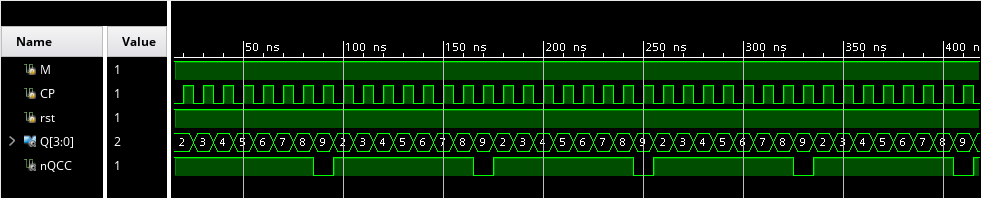


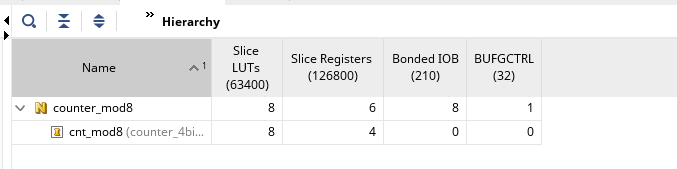
图2-5 初值为2的模8计数器仿真

**（C）开发板上的验证情况（主要记录：验证过程和结论）**

**每次拨动开关，也就是相当于时钟脉冲，当开关复位（下降沿）时候就会触发计数器。M为高电平的时候为为＋1计数，当M为低电平的时候为-1计数。**

**计数器为MOD8+2计数器，结果在2~9区间内。结果可以通过观察led灯来观察结果。**

**以下给出资源分析**



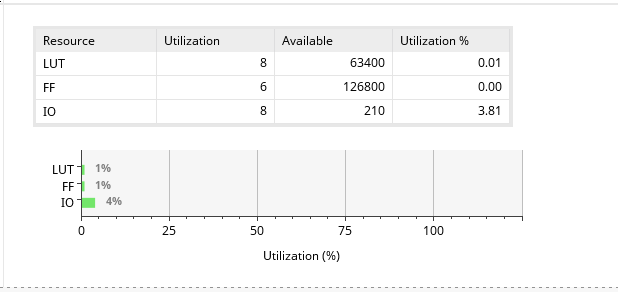


图2-6 资源利用情况

**（2）采用有限状态机(FSM) 实现序列检测器的实验结果记录**

**（A）给出Verilog设计的时序逻辑电路图（**RTL Analysis下“Schematic”截图**）**

图2-7为10110序列检测器

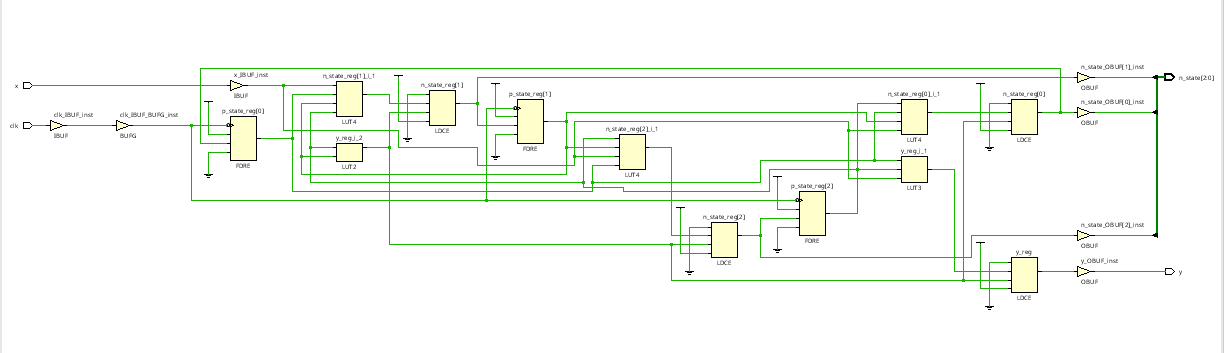


图2-7 10110序列检测器

**（B）仿真结果截图（波形含clk，输入值，输出值等）**

（要求对仿真进行如下说明：仿真对象、输入参数、输出参数、如果具有周期性的要在仿真图上将其“周期”加以标注和说明）

注意，仿真的时候将源代码中的分频器注释掉，并且把时钟改为clk（而不是原来的clk\_N）,因为仿真的时间只有1s，如果再分频就看不到波形了。

图2-8为随机截取的几个周期，其中：  
第一行：输出y，当探测到10110时候为高电平

第二行：输入x，输入0、1序列

第三行：时钟

第四行：状态

由图可见，当x输入的序列中被时钟锁入的部分出现连续的10110时候，y出现高电平脉冲，如图黄线所示

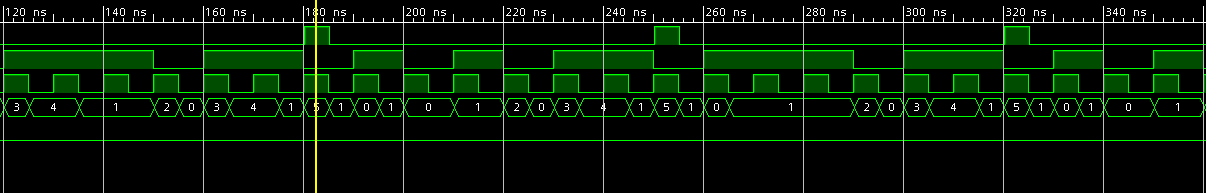


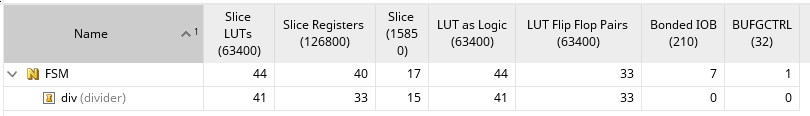
图2-8 10110序列检测器仿真

**（C）开发板上的验证情况**

在开发板上，在加入分频器之后，运行情况正常了（由于开发板上的时钟太快了，必须加入分频器）。**注意！在仿真的时候不要加入分频器！因为仿真只有1s，加入分频器之后无法看到输出。**

生成比特流，然后下载到板子上，可以观察到，用开关输入10110序列之后，y端（led灯）变亮。在输入的时候，注意观察时钟灯的闪烁，确保信号被锁入。同时也可以通过状态变量所绑定的led灯来观察状态的变化情况，每当输入一个变量时候状态会有按照状态转移图所示变化。

以下给出资源分析



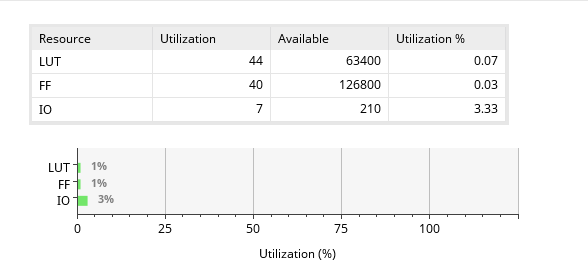


图2-9 资源利用情况

**（3）3位二进制数值比较器的实验结果记录**

**（A）给出Verilog设计的3位二进制数值比较器的电路图（**RTL Analysis下“Schematic”截图**）**

图2-10为 3位二进制数值比较器

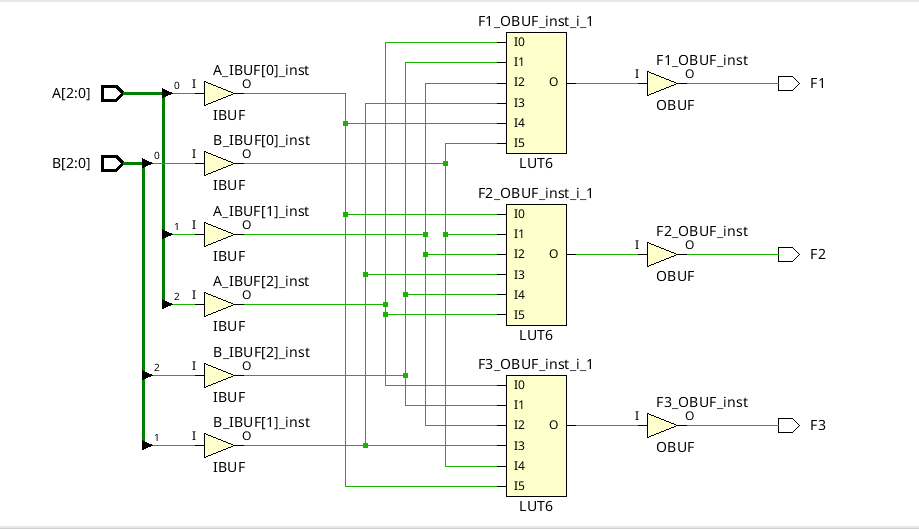


图2-10 3位二进制数值比较器

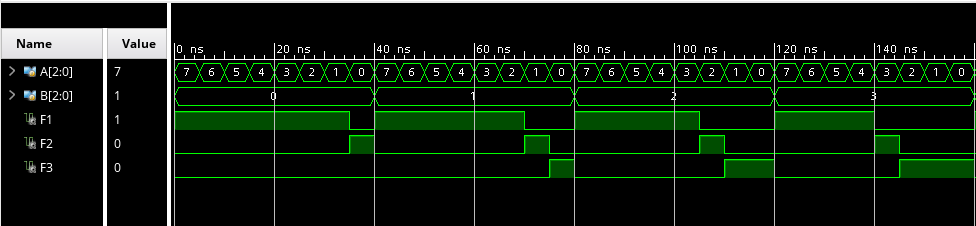
**（B）仿真结果截图**

图2-11为3位二进制数值比较器仿真

A,B分别为3位二进制输入

F1,F2,F3为比较结果的输出，当A>B时候，F1为高电平，当A==B的时候F2为高电平，当A<B时候，F3位高电平。

图中列举了若干不同的输入情况下F1~3的输出情况



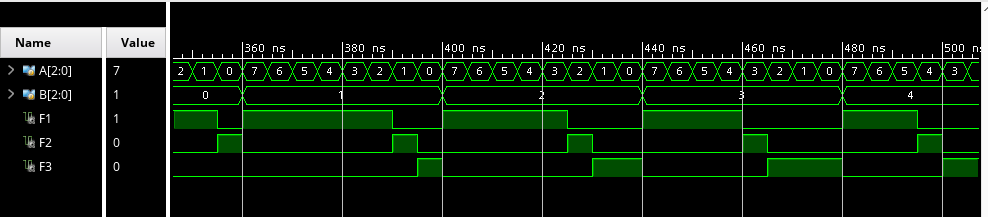
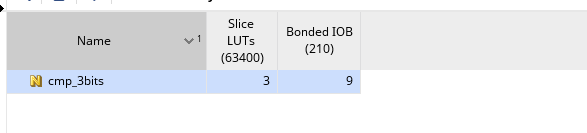


图2-11 3位二进制数值比较器仿真

**（C）开发板上的验证情况**

综合之后生成比特流，下载到开发板。通过输入不同的A和B，来观察F1,F2,F3所绑定的led灯那个亮来验证结果。

可以看到，输入的A,B完全相同的时候，F2灯亮，A大的时候F1灯亮，A小的时候F3灯亮。



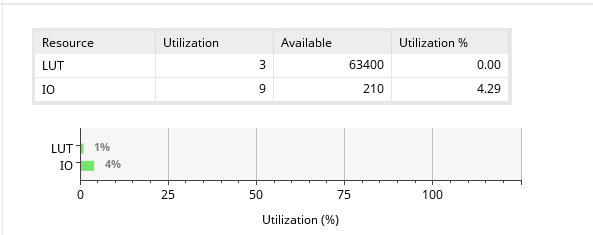


图2-12 资源领用情况

7、实验后的思考

**1．请通过一个具体的实例来说明你是如何用仿真来验证你电路设计的正确性。**

**在**3位二进制数比较器的实验中，我编写的仿真文件类似于一个二重循环，就是在一段比较长的时间中，保持A不变，枚举所有的B，观察输出。然后调整A＋1，再次枚举B，直到所有的 (A,B) 组合都被枚举完全。这样通过观察输出，就能得知所设计电路的正确性。

**2．意见和建议**

希望能将增加课时，给同学们更多的时间去学习、消化、完成所学的内容。