

# 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：

**计算机科学与技术学院**

**2018 年 06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：

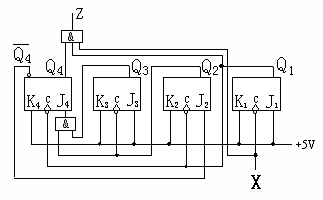


图1-2 脉冲异步计数器电路

（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。

**（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

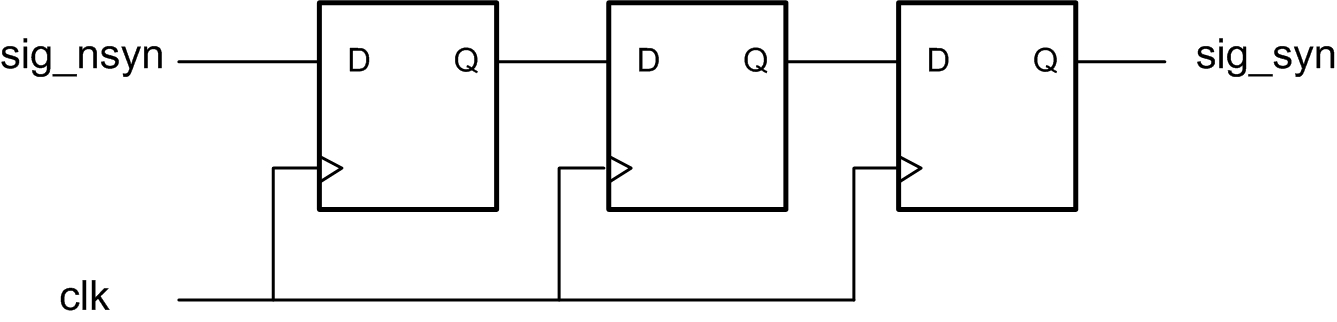


图1-3 3位移位寄存器电路

（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

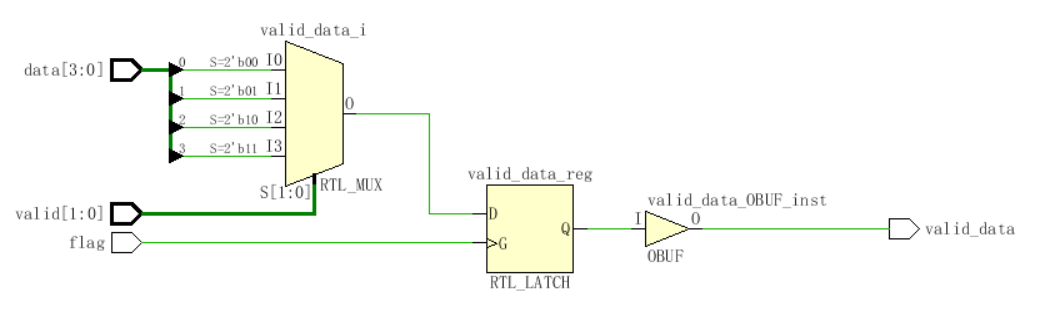


图1-4 未经修改时的电路图

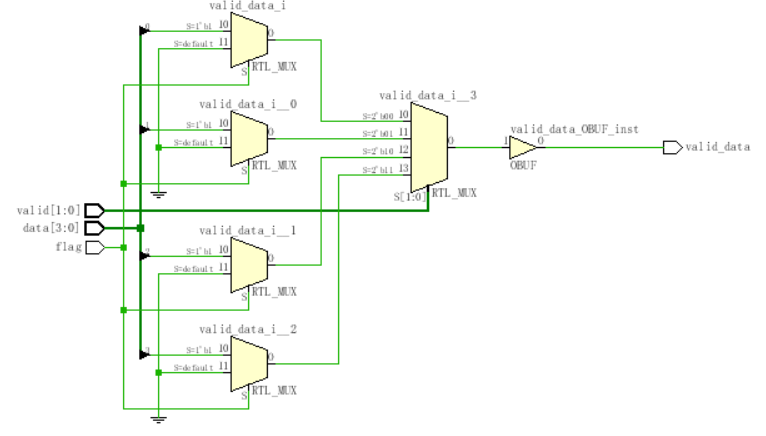


图1-5 修改后的电路图

修改程序如下：

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

根据T触发器的功能，将其作为一个模块实现，之后根据电路图可以写出T触发器的输入参数，以此实现该同步时序逻辑电路。

其中：

T0=1;C0=CP;

T1=X^Q0;C2=CP;

Y=X&/Q1;

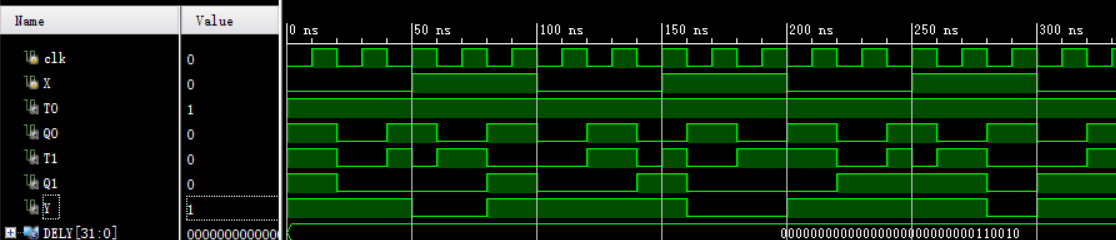


图1-6 前仿真波形

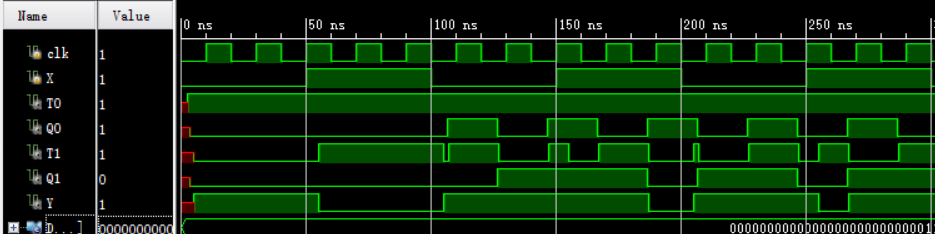


图1-7 后仿真波形

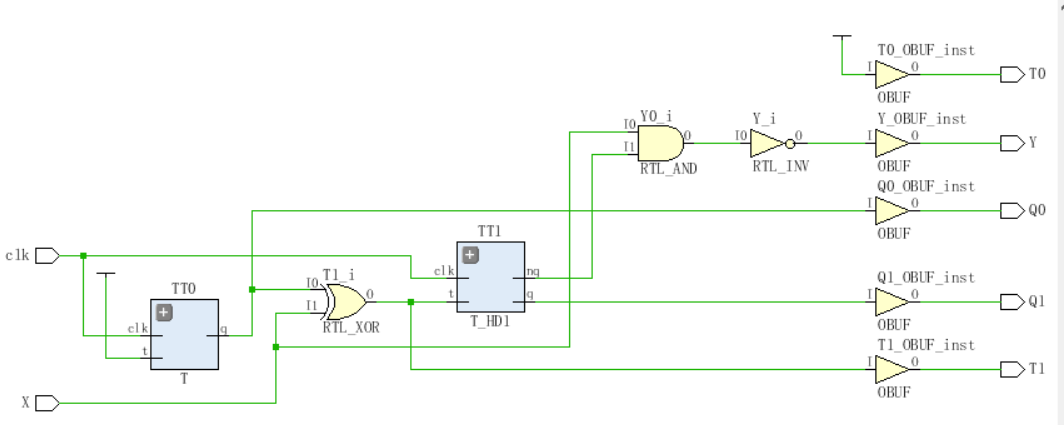


图1-8 RTL电路图

（a）源程序

module main(

input X,

input clk,

output reg Y,

output reg T0,

output Q0,

output reg T1,

output Q1

);

wire NQ0,NQ1;

T TT0(clk,1,Q0,NQ0);

T TT1(clk,Q0^X,Q1,NQ1);

always @(\*)begin

T0<=1;

T1<=Q0^X;

Y<=~( X & NQ1 );

end

endmodule

module T(clk,t,q,nq);

input clk;

input t;

output reg q;

output nq;

initial begin

q <= 0;

end

assign nq=~q;

always @(negedge clk)begin

if(t == 0)

q <= q;

else

q <= ~q;

end

endmodule

（b）仿真程序

module sim();

reg clk,X;

wire Y,Q0,Q1,T0,T1;

parameter DELY=50;

main mymain(X,clk,Y,T0,Q0,T1,Q1);

initial begin

clk <= 0;

X <= 0;

end

always #(DELY/5) clk = ~clk;

always #(DELY) X = ~X;

endmodule

**（2）脉冲异步计数器的分析和设计方案**

根据电路图可分析电路中的逻辑表达式如下所示：

J1=K1=K2=J3=K3=K4=1；

J2=/Q4；J4=Q2&Q3；

C1=X；C2=Q1；C3=Q2；C4=Q1；

Z=Q4&Q1&X；

该电路为模10计数器。

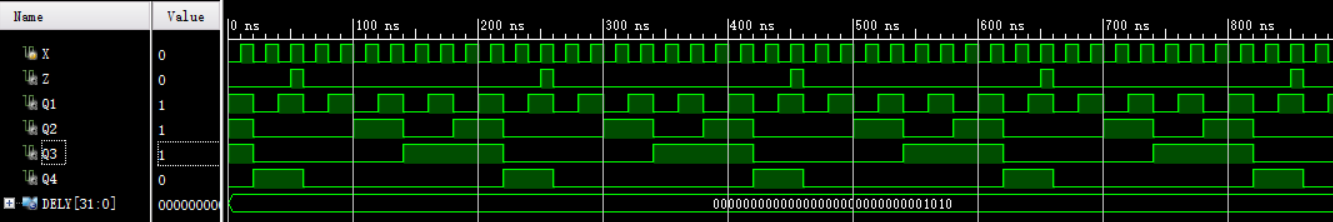


图1-9 前仿真波形

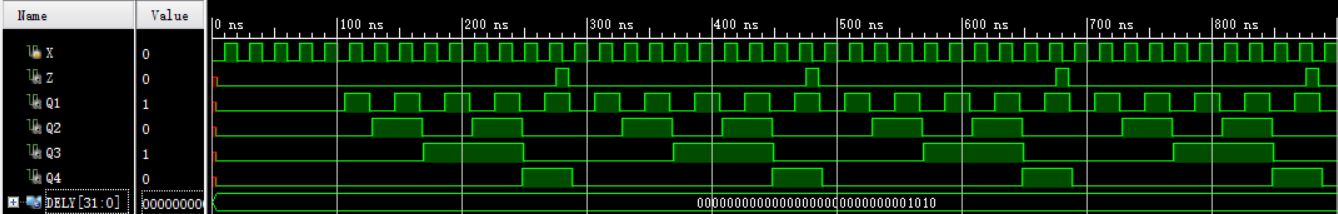


图1-10 后仿真波形

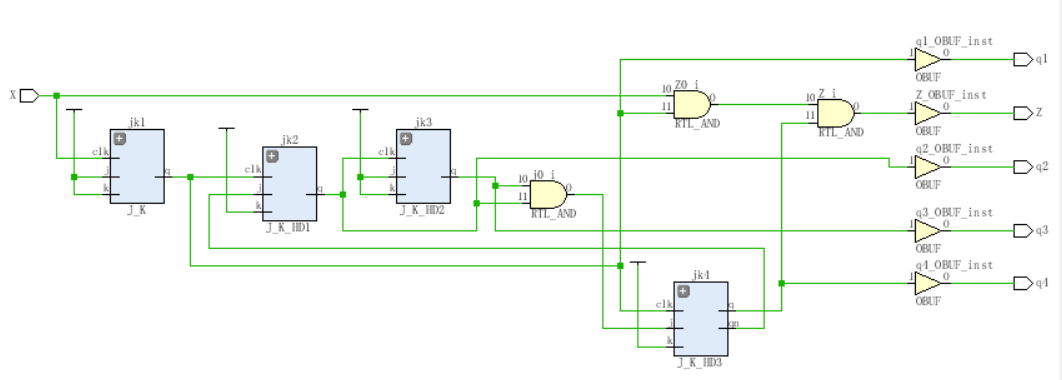


图1-11 RTL电路图

（A）源程序

/////////////////////////////lab.v//////

module lab(

input X,

output Z,

output q1,

output q2,

output q3,

output q4

);

wire qn1,qn2,qn3,qn4;

J\_K jk1(1'b1,1'b1,X,q1,qn1);

J\_K jk2(qn4,1'b1,q1,q2,qn2);

J\_K jk3(1'b1,1'b1,q2,q3,qn3);

J\_K jk4(q3&q2,1'b1,q1,q4,qn4);

assign Z=X&q1&q4;

endmodule

//////////////////////////////J\_K.v

module J\_K(j,k,clk,q,qn);

input j,k,clk;

output reg q;

output qn;

initial begin

q<=0;

end

assign qn = ~q;

always@(negedge 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 lab2tb();

reg X;

wire Z,Q1,Q2,Q3,Q4;

parameter DELY=10;

lab mylab2(X,Z,Q1,Q2,Q3,Q4);

initial begin

X<=0;

end

always #(DELY) X = ~X;

endmodule

（C）引脚约束（绑定）程序

//////////////////Nexys4\_Master.xdc

## 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\_L21N\_T3\_DQS\_34, Sch name = LED1

set\_property PACKAGE\_PIN V9 [get\_ports {q1}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {q1}]

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

set\_property PACKAGE\_PIN R8 [get\_ports {q2}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {q2}]

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

set\_property PACKAGE\_PIN T6 [get\_ports {q3}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {q3}]

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

set\_property PACKAGE\_PIN T5 [get\_ports {q4}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {q4}]

##Bank = 34, Pin name = IO\_L15P\_T2\_DQS\_34, Sch name = LED15

set\_property PACKAGE\_PIN P2 [get\_ports {Z}]

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

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

**（3）Verilog设计中一致性问题的解决方案**

（A）修改程序

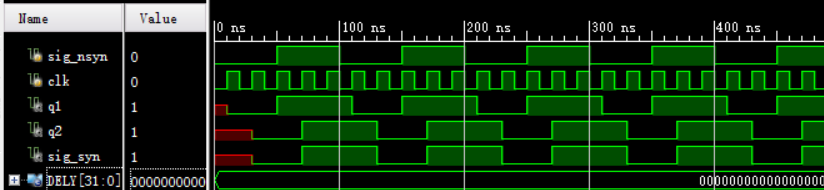
****

图1-12 未修改前的前仿真波形

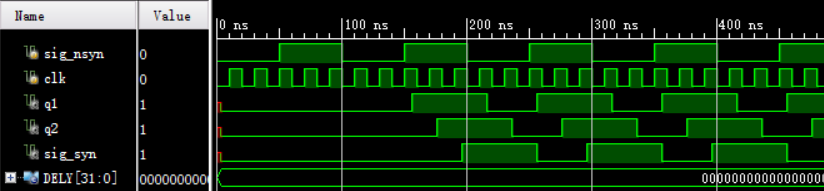
****

图1-13 未修改前的后仿真波形

可以明显看出前后仿真差异较大，在前仿真中q2和sig\_syn是同步变化的，分析原因可能为程序采用阻塞式赋值，在赋值过程中区别顺序，即以此执行 q2=q1;sig\_syn=q2;q1=sig\_nsyn;即sig\_syn得到的其实是q1的值。将程序中的赋值过程修改为非阻塞式赋值后，sig\_syn得到的是q2的原始值，即并不是q1的值，进行前后仿真，观察到如下所示现象。

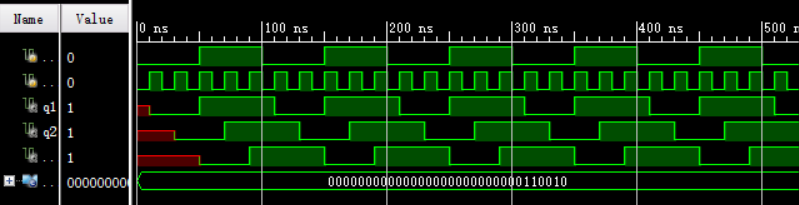


图1-14 修改后的前仿真波形

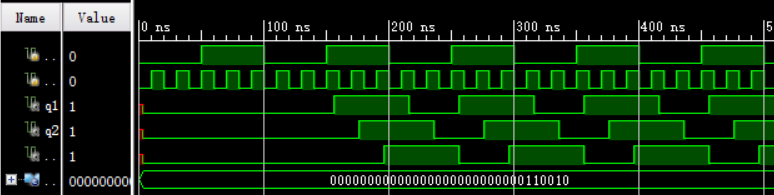


图1-15 修改后的后仿真波形

（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;

endmodule

（b）仿真程序

module pipe3b\_td();

reg sig\_nsyn,clk;

wire q1,q2,sig\_syn;

parameter DELY=50;

pipe3b mypipe3b(sig\_nsyn,clk,q1,q2,sig\_syn);

initial begin

clk <= 0;

sig\_nsyn <= 0;

end

always #(DELY/5) clk = ~clk;

always #(DELY) sig\_nsyn = ~sig\_nsyn;

endmodule

（B）添加分频

1. 源程序

module pipe3b(

input sig\_nsyn,

input clk,

output wire clk\_n,

output reg q1,

output reg q2,

output reg sig\_syn

);

divider mydivider(clk,clk\_n);

initial begin

q1<=0;

q2<=0;

sig\_syn<=0;

end

always@(posedge clk\_n) begin

q2<=q1;

sig\_syn<=q2;

q1<=sig\_nsyn;

end

endmodule

module divider(clk,clk\_n);

input clk;

output reg clk\_n;

parameter n=100\_000\_000;

reg [31:0] counter;

initial begin

clk\_n<=0;

counter<=0;

end

always @(posedge clk)begin

counter<=counter+1;

if(counter==n/2-1)begin

clk\_n<=~clk\_n;

counter<=0;

end

end

endmodule

1. 仿真程序

module td();

reg sig\_nsyn,clk;

wire clk\_n,q1,q2,sig\_syn;

parameter DELY=50;

pipe3b mypipe3b(sig\_nsyn,clk,clk\_n,q1,q2,sig\_syn);

initial begin

clk <= 0;

sig\_nsyn <= 0;

end

always #(DELY/10) clk = ~clk;

always #(DELY) sig\_nsyn = ~sig\_nsyn;

endmodule

1. 约束文件

## Clock signal

##Bank = 35, Pin name = IO\_L12P\_T1\_MRCC\_35, Sch name = CLK100MHZ

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 {sig\_nsyn}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {sig\_nsyn}]

# LEDs

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

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

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

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

set\_property PACKAGE\_PIN R8 [get\_ports {q1}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {q1}]

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

set\_property PACKAGE\_PIN T6 [get\_ports {q2}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {q2}]

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

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

set\_property IOSTANDARD LVCMOS33 [get\_ports {sig\_syn}]

get\_property C\_USER\_SCAN\_CHAIN [get\_debug\_cores dbg\_hub]



**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**

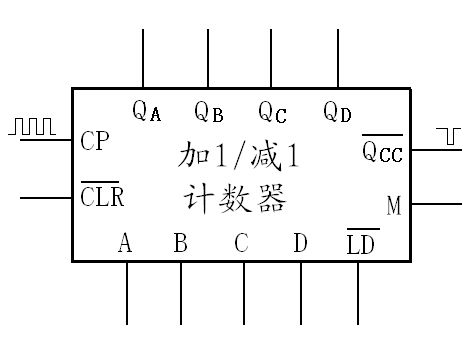
设计一个能清零、置数和进位/借位输出的加1/减1的4位二进制计数器，其结构框图如图2-1所示。

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(

input D,C,B,A,

input clk,

input M,

input LD,

input CLR,

output reg QD,QC,QB,QA,

output reg Qcc

);

initial begin

{QD,QC,QB,QA}<=4'b0000;

Qcc<=1'b1;

end

always @(posedge clk or negedge LD or negedge CLR)begin

Qcc<=1;

if(CLR==0)

{QD,QC,QB,QA}<=4'b0000;

else if(LD==0)begin

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

end

else if(M==1&&{QD,QC,QB,QA}==4'b1111)begin

{QD,QC,QB,QA}<=4'b0000;

Qcc<=0;

end

else if(M==0&&{QD,QC,QB,QA}==4'b0000)begin

{QD,QC,QB,QA}<=4'b1111;

Qcc<=0;

end

else if(M==1)

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

else if(M==0)

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

end

endmodule

（b）仿真程序

`timescale 1ns / 1ps

module sim();

reg CP;

reg M;

wire D,C,B,A;

reg LD;

reg CLR;

wire QD,QC,QB,QA;

wire Qcc;

initial begin

CP<=0;

M<=1;

LD<=1;

CLR<=1;

#50 LD<=0;

#50 LD<=1;

#50 LD<=0;

#50 LD<=1;

#500 M<=0;

end

assign {D,C,B,A}=4'b0100;

always #10 CP<=~CP;

counter my\_counter(D,C,B,A,CP,M,LD,CLR,QD,QC,QB,QA,Qcc);

endmodule

（c）引脚约束（绑定）程序

## Clock signal

set\_property -dict { PACKAGE\_PIN E3 IOSTANDARD LVCMOS33 } [get\_ports { clk }]; #IO\_L12P\_T1\_MRCC\_35 Sch=clk100mhz

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

##Switches

set\_property -dict { PACKAGE\_PIN J15 IOSTANDARD LVCMOS33 } [get\_ports { A }]; #IO\_L24N\_T3\_RS0\_15 Sch=sw[0]

set\_property -dict { PACKAGE\_PIN L16 IOSTANDARD LVCMOS33 } [get\_ports { B }]; #IO\_L3N\_T0\_DQS\_EMCCLK\_14 Sch=sw[1]

set\_property -dict { PACKAGE\_PIN M13 IOSTANDARD LVCMOS33 } [get\_ports { C }]; #IO\_L6N\_T0\_D08\_VREF\_14 Sch=sw[2]

set\_property -dict { PACKAGE\_PIN R15 IOSTANDARD LVCMOS33 } [get\_ports { D }]; #IO\_L13N\_T2\_MRCC\_14 Sch=sw[3]

set\_property -dict { PACKAGE\_PIN R13 IOSTANDARD LVCMOS33 } [get\_ports { M }]; #IO\_L5N\_T0\_D07\_14 Sch=sw[7]

set\_property -dict { PACKAGE\_PIN T8 IOSTANDARD LVCMOS18 } [get\_ports { LD }]; #IO\_L24N\_T3\_34 Sch=sw[8]

set\_property -dict { PACKAGE\_PIN U8 IOSTANDARD LVCMOS18 } [get\_ports { CLR }]; #IO\_25\_34 Sch=sw[9]

set\_property -dict { PACKAGE\_PIN T16 IOSTANDARD LVCMOS33 } [get\_ports { QA }]; #IO\_L15N\_T2\_DQS\_DOUT\_CSO\_B\_14 Sch=led[11]

set\_property -dict { PACKAGE\_PIN V15 IOSTANDARD LVCMOS33 } [get\_ports { QB }]; #IO\_L16P\_T2\_CSI\_B\_14 Sch=led[12]

set\_property -dict { PACKAGE\_PIN V14 IOSTANDARD LVCMOS33 } [get\_ports { QC }]; #IO\_L22N\_T3\_A04\_D20\_14 Sch=led[13]

set\_property -dict { PACKAGE\_PIN V12 IOSTANDARD LVCMOS33 } [get\_ports { QD }]; #IO\_L20N\_T3\_A07\_D23\_14 Sch=led[14]

set\_property -dict { PACKAGE\_PIN V11 IOSTANDARD LVCMOS33 } [get\_ports { Qcc }]; #IO\_L21N\_T3\_DQS\_A06\_D22\_14 Sch=led[15]

**（B）初值为2的模8计数器，要求：通过例化A【调用（A）实现的计数器】来实现。**

（a）源程序

`timescale 1ns / 1ps

module counter\_8(

input clk,

input M,

input CLR,

output wire QD,QC,QB,QA,

output wire Qcc

);

wire LD;

wire [3:0]in;

reg Q;

assign LD=({QD,QC,QB,QA}==10&&M==1)|({QD,QC,QB,QA}==1&&M==0)|({QD,QC,QB,QA}==0)?0:1;

assign in=(M==1)?4'b0010:4'b1001;

assign Qcc=(Q==0)?0:1;

counter my\_counter(in[3],in[2],in[1],in[0],clk,M,LD,CLR,QD,QC,QB,QA,);

always @(posedge clk)begin

Q<=1;

if({QD,QC,QB,QA}==9&&M==1)begin

Q<=0;

end

else if({QD,QC,QB,QA}==2&&M==0)begin

Q<=0;

end

end

endmodule

module counter(

input D,C,B,A,

input clk,

input M,

input LD,

input CLR,

output reg QD,QC,QB,QA,

output reg Qcc

);

initial begin

{QD,QC,QB,QA}<=4'b0000;

Qcc<=1'b1;

end

always @(posedge clk or negedge LD or negedge CLR)begin

Qcc<=1;

if(CLR==0)

{QD,QC,QB,QA}<=4'b0000;

else if(LD==0)begin

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

end

else if(M==1&&{QD,QC,QB,QA}==4'b1111)begin

{QD,QC,QB,QA}<=4'b0000;

Qcc<=0;

end

else if(M==0&&{QD,QC,QB,QA}==4'b0000)begin

{QD,QC,QB,QA}<=4'b1111;

Qcc<=0;

end

else if(M==1)

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

else if(M==0)

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

end

endmodule

（b）仿真程序

`timescale 1ns / 1ps

module sim();

reg clk;

reg M;

reg CLR;

wire QD,QC,QB,QA;

wire Qcc;

initial begin

M<=1;

CLR<=1;

clk<=0;

end

always #20 clk=~clk;

always #600 M=~M;

counter\_8 my\_counter\_8(clk,M,CLR,QD,QC,QB,QA,Qcc);

endmodule

（c）引脚约束（绑定）程序

# Clock signal

set\_property -dict { PACKAGE\_PIN E3 IOSTANDARD LVCMOS33 } [get\_ports { clk }]; #IO\_L12P\_T1\_MRCC\_35 Sch=clk100mhz

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

set\_property -dict { PACKAGE\_PIN J15 IOSTANDARD LVCMOS33 } [get\_ports { M }]; #IO\_L24N\_T3\_RS0\_15 Sch=sw[0]

set\_property -dict { PACKAGE\_PIN L16 IOSTANDARD LVCMOS33 } [get\_ports { CLR }];

set\_property -dict { PACKAGE\_PIN T16 IOSTANDARD LVCMOS33 } [get\_ports { QA }]; #IO\_L15N\_T2\_DQS\_DOUT\_CSO\_B\_14 Sch=led[11]

set\_property -dict { PACKAGE\_PIN V15 IOSTANDARD LVCMOS33 } [get\_ports { QB }]; #IO\_L16P\_T2\_CSI\_B\_14 Sch=led[12]

set\_property -dict { PACKAGE\_PIN V14 IOSTANDARD LVCMOS33 } [get\_ports { QC }]; #IO\_L22N\_T3\_A04\_D20\_14 Sch=led[13]

set\_property -dict { PACKAGE\_PIN V12 IOSTANDARD LVCMOS33 } [get\_ports { QD }]; #IO\_L20N\_T3\_A07\_D23\_14 Sch=led[14]

set\_property -dict { PACKAGE\_PIN V11 IOSTANDARD LVCMOS33 } [get\_ports { Qcc }]; #IO\_L21N\_T3\_DQS\_A06\_D22\_14 Sch=led[15]

**（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

表2-1 “10110”二进制序列检测器的真值表

|  |  |  |
| --- | --- | --- |
| 现态 | 次态/输出 | |
| X=0 | X=1 |
| A | A/0 | B/0 |
| B | C/0 | B/0 |
| C | A/0 | D/0 |
| D | C/0 | E/0 |
| E | A/1 | B/0 |

状态图如下所示：

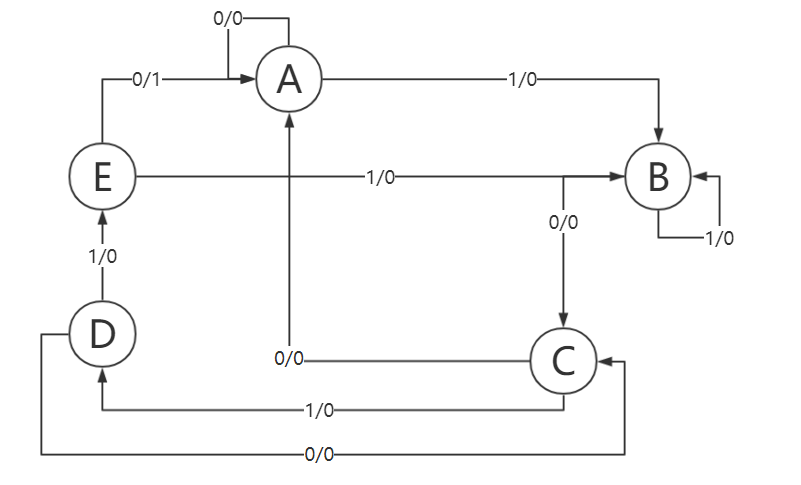


图2-2 “10110”二进制序列检测器的状态图

（B）源程序

`timescale 1ns / 1ps

module mealy(

input clk,

input in,

output reg [4:0] state,

output reg out

);

parameter STA=5'b00001,STB=5'b00010,STC=5'b00100,STD=5'b01000,STE=5'b10000;

initial state<=STA;

always@(posedge clk)begin

case(state)

STA:

if(in==1'b0)begin state<=STA;out<=1'b0;end

else begin state<=STB;out<=1'b0;end

STB:

if(in==1'b0)begin state<=STC;out<=1'b0;end

else begin state<=STB;out<=1'b0;end

STC:

if(in==1'b0)begin state<=STA;out<=1'b0;end

else begin state<=STD;out<=1'b0;end

STD:

if(in==1'b0)begin state<=STC;out<=1'b0;end

else begin state<=STE;out<=1'b0;end

STE:

if(in==1'b0)begin state<=STA;out<=1'b1;end

else begin state<=STB;out<=1'b0;end

default:begin state<=STA;out<=1'b0;end

endcase

end

endmodule

（C）仿真程序

`timescale 1ns / 1ps

module sim();

reg clk;

reg in;

wire out;

wire [4:0]state;

always #5 clk=~clk;

initial begin

in<=0;

clk<=0;

#3 in<=1;

#10 in<=0;#10 in<=1;

#10 in<=0;#10 in<=1;

#10 in<=0;#10 in<=1;

#10 in<=1;#10 in<=0;

#10 in<=1;#10 in<=1;

#10 in<=0;#10 in<=1;

#10 in<=1;#10 in<=0;

#10 in<=1;#10 in<=1;

#10 in<=0;#10 in<=1;

#10 in<=0;#10 in<=1;

#10 in<=1;#10 in<=0;

#10 in<=1;#10 in<=1;

#10 in<=0;#10 in<=1;

#10 in<=1;#10 in<=0;

#10 in<=1;#10 in<=0;

end

mealy my\_mealy(clk,in,state,out);

endmodule

（D）引脚约束（绑定）程序

# Clock signal

set\_property -dict { PACKAGE\_PIN E3 IOSTANDARD LVCMOS33 } [get\_ports { clk }]; #IO\_L12P\_T1\_MRCC\_35 Sch=clk100mhz

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

set\_property -dict { PACKAGE\_PIN T15 IOSTANDARD LVCMOS33 } [get\_ports { state[0] }]; #IO\_L14N\_T2\_SRCC\_14 Sch=led[9]

set\_property -dict { PACKAGE\_PIN U14 IOSTANDARD LVCMOS33 } [get\_ports { state[1] }]; #IO\_L22P\_T3\_A05\_D21\_14 Sch=led[10]

set\_property -dict { PACKAGE\_PIN T16 IOSTANDARD LVCMOS33 } [get\_ports { state[2] }]; #IO\_L15N\_T2\_DQS\_DOUT\_CSO\_B\_14 Sch=led[11]

set\_property -dict { PACKAGE\_PIN V15 IOSTANDARD LVCMOS33 } [get\_ports { state[3] }]; #IO\_L16P\_T2\_CSI\_B\_14 Sch=led[12]

set\_property -dict { PACKAGE\_PIN V14 IOSTANDARD LVCMOS33 } [get\_ports { state[4] }]; #IO\_L22N\_T3\_A04\_D20\_14 Sch=led[13]

set\_property -dict { PACKAGE\_PIN V12 IOSTANDARD LVCMOS33 } [get\_ports { out }]; #IO\_L20N\_T3\_A07\_D23\_14 Sch=led[14]

set\_property -dict { PACKAGE\_PIN V11 IOSTANDARD LVCMOS33 } [get\_ports { clk\_n }]; #IO\_L21N\_T3\_DQS\_A06\_D22\_14 Sch=led[15]

set\_property -dict { PACKAGE\_PIN J15 IOSTANDARD LVCMOS33 } [get\_ports { in }]; #IO\_L24N\_T3\_RS0\_15 Sch=sw[0]

**（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 compare(

input A,

input B,

output F1,

output F2,

output F3

);

assign F1=A&(~B);

assign F2=(A&B)|((~A)&(~B));

assign F3=(~A)&B;

endmodule

（b）仿真程序

`timescale 1ns / 1ps

module sim();

reg A;

reg B;

wire F1,F2,F3;

initial begin

#5 begin A<=1;B<=0;end

#5 begin A<=0;B<=0;end

#5 begin A<=1;B<=1;end

#5 begin A<=0;B<=1;end

#5 begin A<=1;B<=0;end

#5 begin A<=0;B<=0;end

#5 begin A<=1;B<=1;end

#5 begin A<=0;B<=1;end

end

compare my\_compare(A,B,F1,F2,F3);

endmodule

（c）引脚约束（绑定）程序

set\_property -dict { PACKAGE\_PIN J15 IOSTANDARD LVCMOS33 } [get\_ports { B }]; #IO\_L24N\_T3\_RS0\_15 Sch=sw[0]

set\_property -dict { PACKAGE\_PIN L16 IOSTANDARD LVCMOS33 } [get\_ports { A }]; #IO\_L3N\_T0\_DQS\_EMCCLK\_14 Sch=sw[1]

set\_property -dict { PACKAGE\_PIN V14 IOSTANDARD LVCMOS33 } [get\_ports { F3 }]; #IO\_L22N\_T3\_A04\_D20\_14 Sch=led[13]

set\_property -dict { PACKAGE\_PIN V12 IOSTANDARD LVCMOS33 } [get\_ports { F2 }]; #IO\_L20N\_T3\_A07\_D23\_14 Sch=led[14]

set\_property -dict { PACKAGE\_PIN V11 IOSTANDARD LVCMOS33 } [get\_ports { F1 }]; #IO\_L21N\_T3\_DQS\_A06\_D22\_14 Sch=led[15]

**（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。**

（a）源程序

`timescale 1ns / 1ps

module compare\_3(

input [2:0]A,

input [2:0]B,

output F1,

output F2,

output F3

);

wire F10,F11,F12,F20,F21,F22,F30,F31,F32;

compare my\_compare0(A[0],B[0],F10,F20,F30);

compare my\_compare1(A[1],B[1],F11,F21,F31);

compare my\_compare2(A[2],B[2],F12,F22,F32);

assign F1=F12|(F22&F11)|(F22&F21&F10);

assign F2=F22&F21&F20;

assign F3=(~F1)&(~F2);

endmodule

module compare(

input A,

input B,

output F1,

output F2,

output F3

);

assign F1=A&(~B);

assign F2=(A&B)|((~A)&(~B));

assign F3=(~A)&B;

endmodule

（b）仿真程序

`timescale 1ns / 1ps

module sim();

reg [2:0]A,B;

wire F1,F2,F3;

parameter DELY=10;

compare\_3 com(A,B,F1,F2,F3);

initial begin

A=3'b000;

B=3'b111;

end

always begin #(DELY/10) A=A+1;

#(DELY/10) B=B-1;

end

endmodule

（c）引脚约束（绑定）程序

set\_property -dict { PACKAGE\_PIN J15 IOSTANDARD LVCMOS33 } [get\_ports { B[0] }]; #IO\_L24N\_T3\_RS0\_15 Sch=sw[0]

set\_property -dict { PACKAGE\_PIN L16 IOSTANDARD LVCMOS33 } [get\_ports { B[1] }]; #IO\_L3N\_T0\_DQS\_EMCCLK\_14 Sch=sw[1]

set\_property -dict { PACKAGE\_PIN M13 IOSTANDARD LVCMOS33 } [get\_ports { B[2] }]; #IO\_L6N\_T0\_D08\_VREF\_14 Sch=sw[2]

set\_property -dict { PACKAGE\_PIN R15 IOSTANDARD LVCMOS33 } [get\_ports { A[0] }]; #IO\_L13N\_T2\_MRCC\_14 Sch=sw[3]

set\_property -dict { PACKAGE\_PIN R17 IOSTANDARD LVCMOS33 } [get\_ports { A[1] }]; #IO\_L12N\_T1\_MRCC\_14 Sch=sw[4]

set\_property -dict { PACKAGE\_PIN T18 IOSTANDARD LVCMOS33 } [get\_ports { A[2] }]; #IO\_L7N\_T1\_D10\_14 Sch=sw[5]

set\_property -dict { PACKAGE\_PIN V14 IOSTANDARD LVCMOS33 } [get\_ports { F3 }]; #IO\_L22N\_T3\_A04\_D20\_14 Sch=led[13]

set\_property -dict { PACKAGE\_PIN V12 IOSTANDARD LVCMOS33 } [get\_ports { F2 }]; #IO\_L20N\_T3\_A07\_D23\_14 Sch=led[14]

set\_property -dict { PACKAGE\_PIN V11 IOSTANDARD LVCMOS33 } [get\_ports { F1 }]; #IO\_L21N\_T3\_DQS\_A06\_D22\_14 Sch=led[15]

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**）**

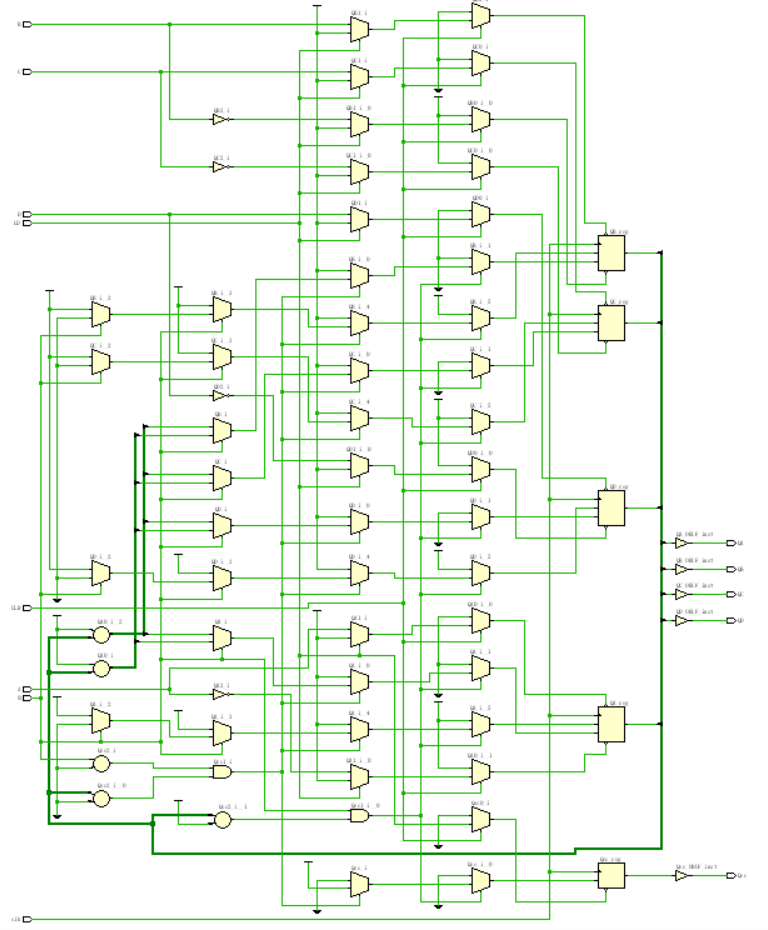


图2-3 4位二进制加法/减法计数器

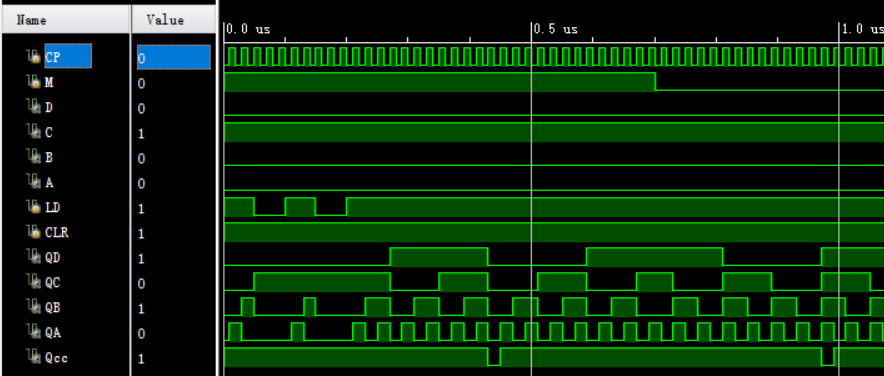
****

图2-4 4位二进制加法/减法计数器前仿真波形

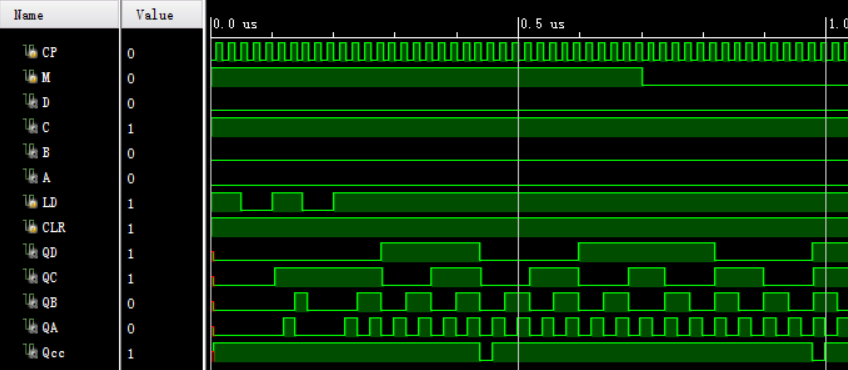
****

图2-5 4位二进制加法/减法计数器前仿真波形

**（B）初值为2的模8计数器仿真结果截图（波形含CP，M，Qa，Qb，Qc，Qd，Q模8等）**

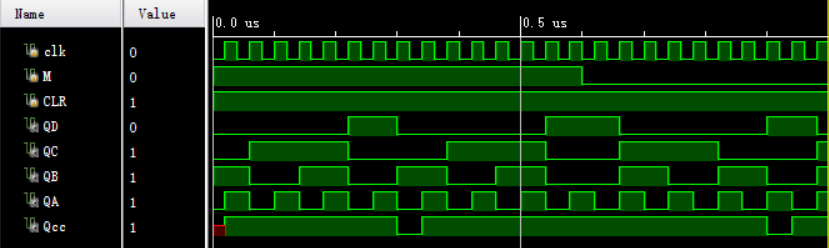
****

图2-6 初值为2的模8计数器前仿真波形

**（C）开发板上的验证情况（主要记录：验证过程和结论）**

添加分频后连接开发板，先将CLR开关和M开关置1，观察到Qcc灯一直在亮，信号由2增加到9，之后又跳转至2，同时Qcc灯灭，在加到3时Qcc再次亮起。

在为3的时候将M拨为0，进行减法的验证，发现结果由2直接跳转至9且Qcc灯灭，之后由9开始进行减法且Qcc灯亮，但是减法时存在问题的，当减至4时数据直接跳转至9，分析原因可能是4-3-2-9的这个过程太快以至于观察不到。之后重新阅读代码修改代码，猜测原因可能和阻塞赋值和非阻塞赋值有关系，但最后依旧没有修改正确。

**（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**）**

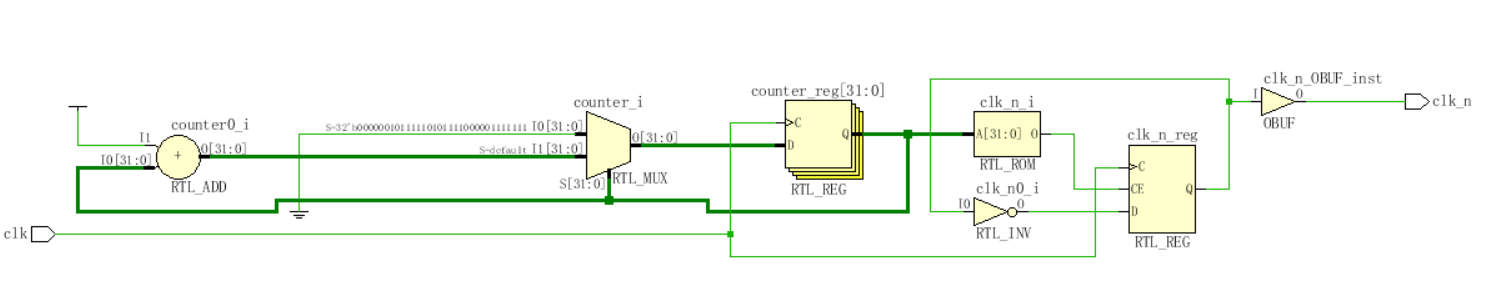


图2-7 10110序列检测器

**（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）

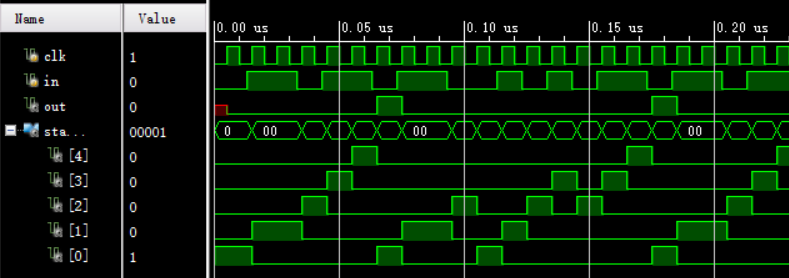


图2-8 10110序列检测器前仿真波形

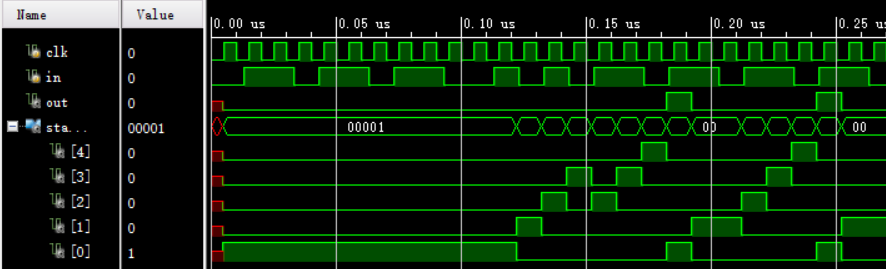


图2-9 10110序列检测器后仿真波形

**（C）开发板上的验证情况（主要记录：验证过程和结论）**

Led[15]一直在闪，其显示的是分频后的时钟信号，验证时须在灯闪动的间隙快速拨动开关，在下次灯闪动时即为输入。依次输入“10110”，观察到lesd[9]、led[10]、led[11]、led[12]、led[13]、led[14]依次闪动，前五个灯表示五个状态，最后一个灯亮表示检测到“10110”的序列，满足题意，验证正确。

**（D）给出设计占用FPGA芯片的资源情况**

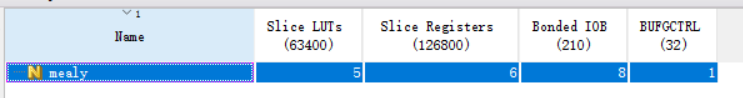


图2-10 占用FPGA芯片资源情况

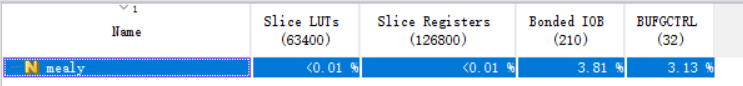


图2-11 占用FPGA芯片资源情况

**（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**）**

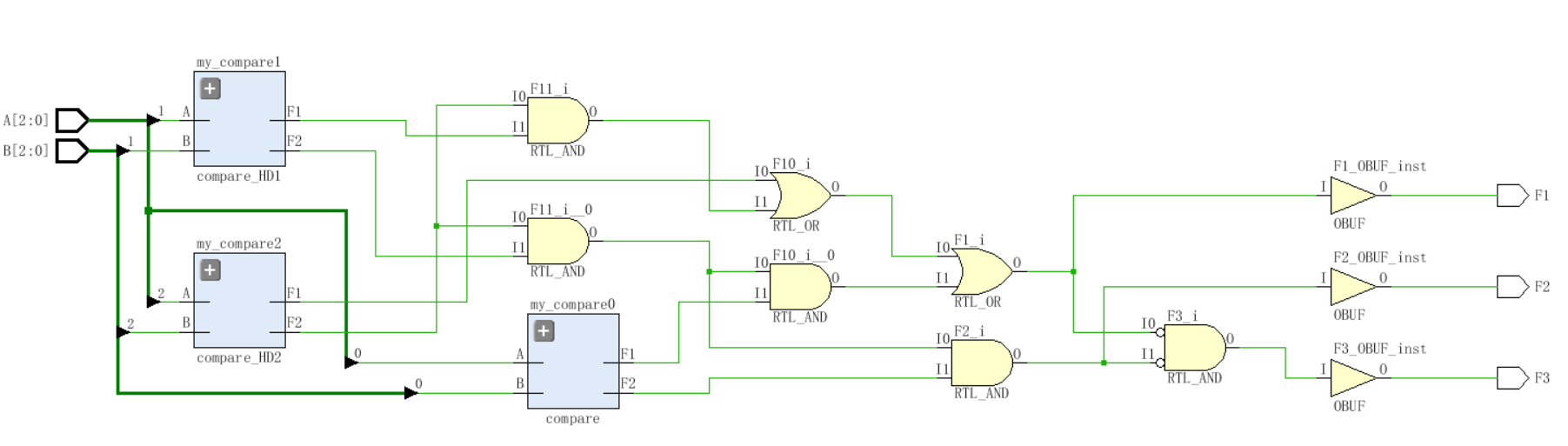


图2-12 3位二进制数值比较器

**（B）仿真结果截图（含输入值，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）

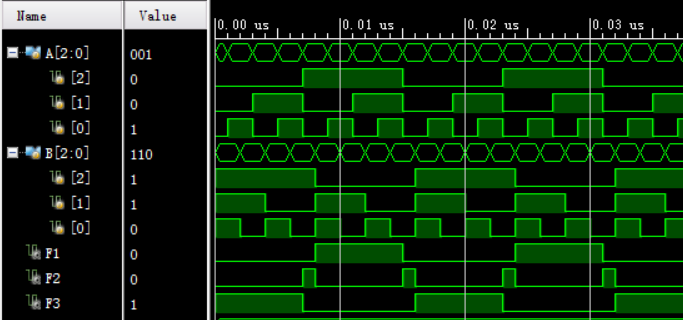


图2-13 3位二进制数值比较器前仿真波形

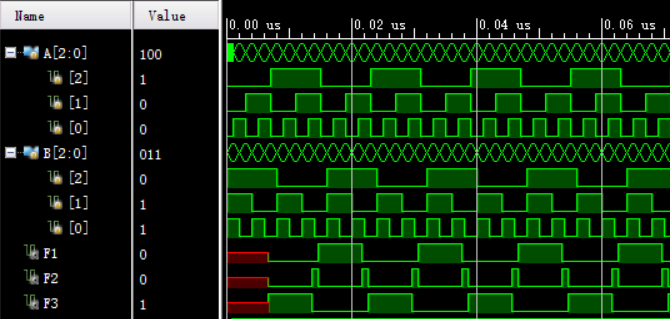


图2-14 3位二进制数值比较器后仿真波形

**（C）开发板上的验证情况（主要记录：验证过程和结论）**

首先AB都为0时仅F2灯亮，先将AB都拨动到7，观察仅F2灯亮，拨动开关使A>B观察到仅F1灯亮，A<B时仅F3灯亮，满足题意，验证正确。

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。**

比如最后一个实验中，先进行前仿真，通过一一对应可以判断出F1、F2、F3的结果都是正确的。之后再进行后仿真，由于F1、F2、F3三个数值并没有赋予初值，而后仿真存在一定的延时，所以仿真前一段为红线，即数值未知。之后的绿线也通过一一对应的方式可以判断出F1、F2、F3的结果是正确的。在判断前后仿真都正确之后，电路的设计就一般不会再有问题了，而且下板子也会是正确地运行。

**2．意见和建议**

无。