

# 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： 潘翔**

**学 号： U201614898**

**班 级： IOT1601**

**指 导 教 师： 徐有青**

**计算机科学与技术学院**

**20 年 月 日**



**数字逻辑实验报告**

**Verilog HDL设计数字逻辑电路预习报告**

[数字逻辑实验报告（2） 1](#_Toc19686408)

[一、Verilog HDL设计数字逻辑电路 4](#_Toc618364369)

[1、实验名称 4](#_Toc2119704490)

[2、实验目的 4](#_Toc1702422668)

[3、实验所用设备 4](#_Toc1313148711)

[4、实验内容 4](#_Toc1349378183)

[5、实验方案设计 8](#_Toc733161145)

[二、Verilog HDL设计较复杂数字逻辑电路 20](#_Toc626744484)

[1、实验名称 20](#_Toc203528646)

[2、实验目的 20](#_Toc832757737)

[3、实验所用组件 20](#_Toc709947332)

[4、实验内容 20](#_Toc1503776830)

[5、实验方案设计 23](#_Toc1751122270)

[6、实验结果记录 31](#_Toc2074601493)

[7、实验后的思考 35](#_Toc1077614890)

# 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：

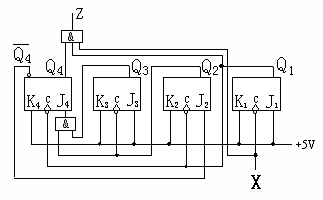


图1-2 脉冲异步计数器电路

（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

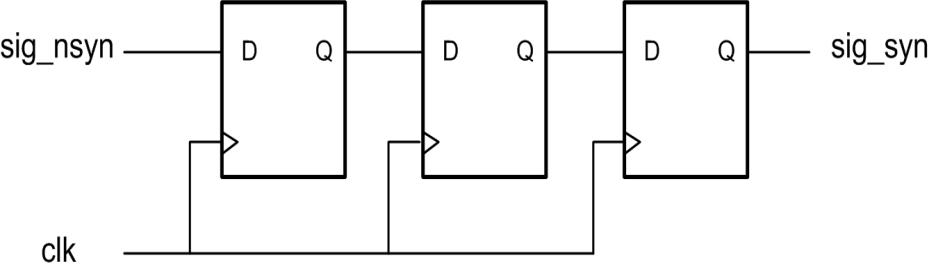


图1-3 3位移位寄存器电路

（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”。

**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**

**程序1-3 “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

valid\_data = 1'b0; // assign defualt value for valid\_data

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

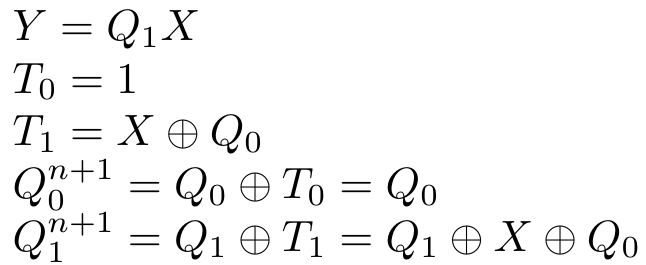
endcase

end

endmodule

（B）用“always块”设计实现一个同步时序逻辑电路

（a）函数表达式和激励函数表达式



（b）状态表

表3-1 同步时序逻辑电路状态表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 现态Q1Q0 | 次态 | | 输出Y | |
| X=0 | X=1 | X=0 | X=1 |
| 00 | 01 | 11 | 0 | 1 |
| 01 | 10 | 00 | 0 | 1 |
| 10 | 11 | 01 | 0 | 0 |
| 11 | 00 | 10 | 0 | 0 |

（c）源程序

**程序1-4 “always块”设计同步时序逻辑电路**

module lab3\_1B

(

input X,

input clk,

output reg Y,

output reg Q0,

output reg Q1,

output reg T1

);

reg Q0=1'b0;

reg Q1=1'b0;

reg T1=1'b0;

always @ (X,Q0,Q1)

begin

T1 <= X^Q0;

Y <= ~(X&~Q1);

end

always @ (negedge clk)

begin

Q0 <= 1^Q0;

Q1 <= T1^Q1;

end

endmodule

（d）仿真程序

**程序1-4 “always块”设计同步时序逻辑电路仿真程序**

module lab3\_1B\_sim

(

);

reg clk;

reg x;

initial begin

clk = 0;

x = 0;

end

always #10 clk <= ~clk;

always #30 x<=~x;

wire out;

wire Q0;

lab3\_1B lab3\_1B\_test(x,clk,out,Q0,Q1,T1);

endmodule

（e）RTL门电路图

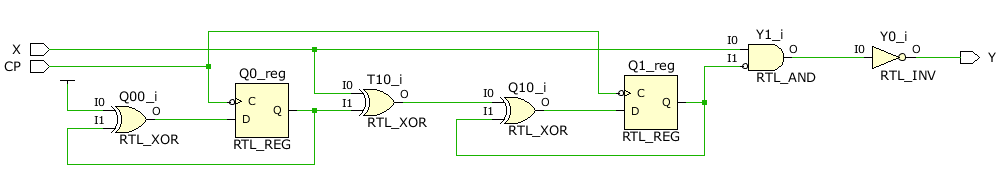


图1-1 同步时序逻辑电路RTL门电路图

（d）仿真结果

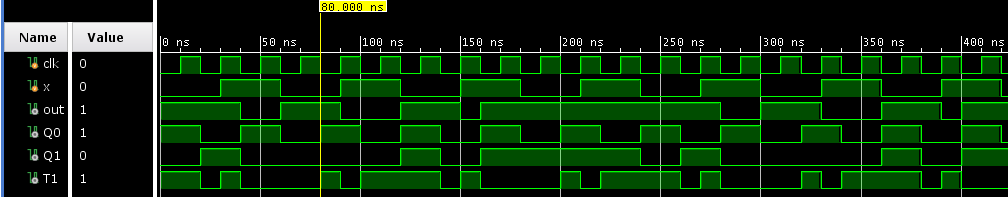


图1-2 同步时序逻辑电路仿真结果图

仿真结果分析：

在仿真中可以看到，利用clk进行时钟周期模拟。x进行翻转对触发器进行触发，满足如上状态表状态。

1. 脉冲异步计数器的分析和设计方案

（A）计数器说明

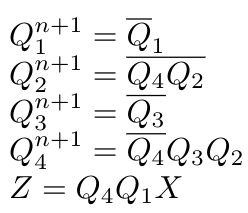
模10计数器；

X:输入脉冲

Z:输出进位

Q4Q3Q2Q1:计数量

激励函数表达式：



1. 源程序

**程序1-5 脉冲异步计数器电路**

module lab3\_2B

(

input X,

output reg Q1,

output reg Q2,

output reg Q3,

output reg Q4,

output reg Z

);

reg Q1=1'b0;

reg Q2=1'b0;

reg Q3=1'b0;

reg Q4=1'b0;

always @ (negedge X)

begin

Q1<= ~Q1;

end

always @ (negedge Q1)

begin

Q2<=~Q2&(~Q4);

Q4<=~Q4&Q2&Q3;

end

always @ (negedge Q2)

begin

Q3<=~Q3;

end

always @ (\*)

begin

Z=X&Q1&Q4;

end

endmodule

（C）仿真程序

**程序1-6 脉冲异步计数器电路仿真程序**

module lab3\_2B\_sim

(

);

reg X = 0;

always #10 X<=~X;

wire Z;

wire Q1;

wire Q2;

wire Q3;

wire Q4;

lab3\_2B lab3\_2B\_test(X,Q1,Q2,Q3,Q4,Z);

endmodule

1. 引脚约束（绑定）程序

**程序1-7 脉冲异步计数器引脚绑定**

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

# 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 {Z}]

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

#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}]

（E）RTL门电路图

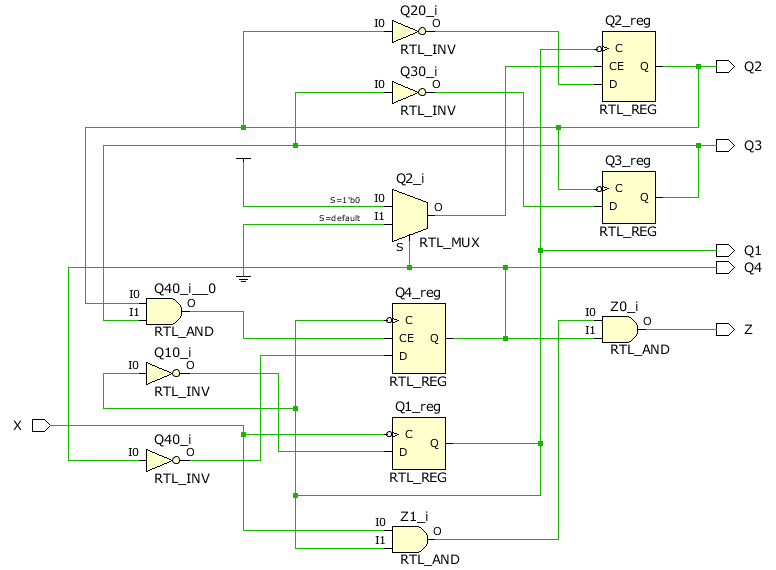


图1-3 脉冲异步计数器电路RTL门电路图

（F）仿真结果

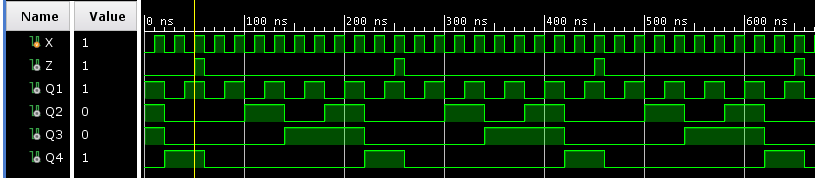


图1-4 脉冲异步计数器电路行为仿真结果图

仿真结果分析：从图中可以看出，在经过初始的稳定之后，进行模10技术Z正脉冲输出技术。Q4Q3Q2Q1为BCD码。

（G）开发板验证

在开发板上，利用led输出，利用开关模拟脉冲进行输出验证，满足结果。

（3）Verilog设计中一致性问题的解决方案

（A）源程序

**程序1-8 3位移位电路**

module lab3\_3\_divider(clk,sig\_nsyn,led); // top module

input clk;

input sig\_nsyn;

parameter delay=50\_000\_000;

output [2:0] led;

wire clk\_N;

divider sub\_divider

(

.clk(clk),

.clk\_need(clk\_N)

);

pipe3b sub\_pipe3b

(

sig\_nsyn,

clk\_N,

led[2],

led[1],

led[0]

);

endmodule

module pipe3b

(

input sig\_nsyn,

input clk,

output q1,

output q2,

output sig\_syn

);

reg q1;

reg q2;

reg sig\_syn;

initial begin

q1 = 0;

q2 = 0;

sig\_syn = 0;

end

always@(posedge clk) begin // use the same signal in different always and they will compete

q1<=sig\_nsyn;

q2<=q1;

sig\_syn<=q2;

end

endmodule

module divider

(

input clk,

//input [31:0] frequency\_need,

output reg clk\_need

);

parameter frequency\_before = 32'd100\_000\_000;

reg [31:0] counter;

parameter frequency\_need = 1;

wire [31:0] N;

assign N = (frequency\_before/frequency\_need) / 2;

initial counter = 0;

initial clk\_need = 0;

always @(posedge clk)

begin

counter <= counter+1;

if(counter == N-1)

begin

clk\_need <= ~clk\_need;

counter <= 0;

end

end

endmodule

1. 仿真程序

**程序1-9 3位移位寄存器电路仿真程序**

module lab3\_3\_divider\_sim

(

);

reg clk = 0;

always #10 clk <= ~clk;

reg sig\_nsyn = 0;

always #30 sig\_nsyn<=~sig\_nsyn;

reg [2:0] led;

//module lab3\_3\_divider(clk,sig\_nsyn,led); // top module

lab3\_3\_divider lab3\_3\_divider\_sim(clk,sig\_nsyn,led);

endmodule

1. 约束文件

**程序1-10 3位移位寄存器电路约束文件**

set\_property CLOCK\_DEDICATED\_ROUTE FALSE [get\_nets sig\_nsyn]

# 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}]

#CLOCK

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]

# LEDs

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

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

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

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

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

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

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

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

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

（D）RTL门电路图

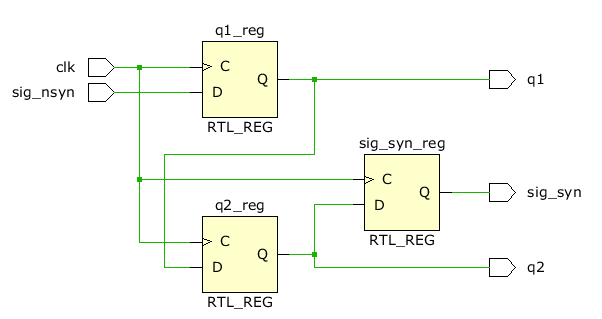


图1-5 3位移位寄存器RTL门电路图

（E）仿真结果

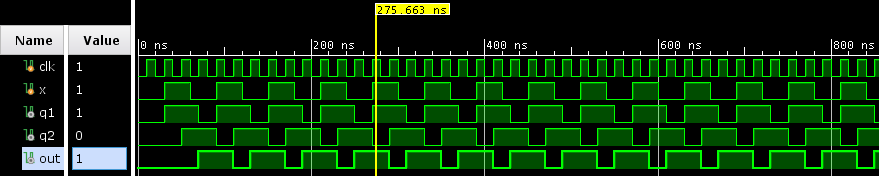


图1-6 3位移位寄存器行为仿真(Behavioral Simulation)图

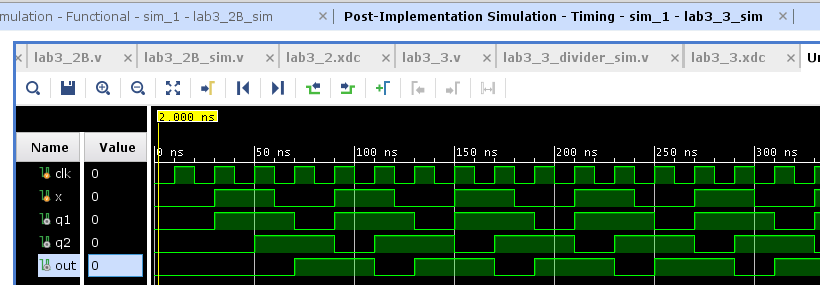


图3-1 3位移位寄存器后时序仿真图

（F）开发板验证

利用LED进行输出，在开发板上验证，拨动开关，依次亮起，熄灭，并可通过分屏器控制速度。



**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。

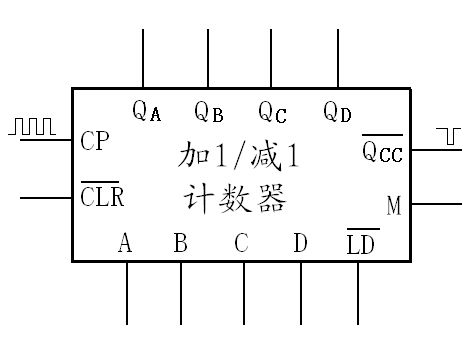


图2-1 4位二进制加法/减法计数器

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

**程序2-1 模16加1/减1计数器电路**

module lab4\_1\_divider

(

input [3:0]in,

input clk,

input CLR,

input LD,

input M,

output [3:0]Q,

output Qcc

);

divider sub\_divider(clk,CP);

lab4\_1 sub\_lab4\_1

(

.in(in),

.CP(CP),

.CLR(CLR),

.LD(LD),

.M(M),

.Q(Q),

.Qcc(Qcc)

);

endmodule

module lab4\_1

(

input [3:0]in,

input CP,

input CLR,

input LD,

input M,

output reg [3:0]Q,

output reg Qcc

);

initial begin

Q=4'b0000;

Qcc=1;

end

always@(posedge CP) begin

if(~CLR) begin //CLR=0

Q<=4'b0000;

Qcc<=0;

end

else

if(~LD) begin // output the input directly when LD=0

Q<=in;

Qcc<=0;

end

else

if(M==1) begin // add mode M=1

if(Q==4'b1111) begin

Qcc<=0;

//#10 Qcc<=1; // Qcc pluse

Q<=4'b0000;

end

else begin

Q<=Q+4'b0001;

Qcc<=1;

end

end

else

if(~M) begin //decrease mode m=0

if(Q==4'b0000)begin

Qcc<=0;

//#10 Qcc<=1; // Qcc pluse

Q<=4'b1111;

end

else begin

Q<=Q-4'b0001;

Qcc<=1;

end

end

end

endmodule

module divider

(

input clk,

//input [31:0] frequency\_need,

output reg clk\_need

);

parameter frequency\_before = 32'd100\_000\_000;

//parameter frequency\_before = 1;

reg [31:0] counter;

parameter frequency\_need = 1;

wire [31:0] N;

assign N = (frequency\_before/frequency\_need) / 2;

initial counter = 0;

initial clk\_need = 0;

always @(posedge clk)

begin

counter <= counter+1;

if(counter == N-1)

begin

clk\_need <= ~clk\_need;

counter <= 0;

end

end

endmodule

（b）仿真程序

**程序2-2 模16加1/减1计数器电路仿真程序**

module lab4\_1\_sim

(

);

reg [3:0] in;

wire [3:0] Q;

reg CP, M, CLR, LD;

wire Qcc; // out pluse

initial begin // set init val

CP = 0;

in<=4'b0010;

M<=1;

CLR<=1;

LD<=1;

end

initial begin // set triger

#25 LD = 0; // output directly

#45 LD = 1;

#65 CLR = 0;//set 0

#85 CLR = 1;

#360 M = 0;

end

always #10 CP = ~CP;

lab4\_1\_divider lab4\_1\_test\_divider

(

.in(in),

.clk(CP),

.CLR(CLR),

.LD(LD),

.M(M),

.Q(Q),

.Qcc(Qcc)

);

endmodule

（c）引脚约束（绑定）程序

**程序2-3 模16加1/减1计数器电路引脚约束**

#input A,

#input B,

#input C,

#input D,

#input LD, // prefabricated control

#input CLR, // set 0

#input clk,

#input M, //chose mode

#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 {A}]

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

set\_property PACKAGE\_PIN V6 [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 {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 {Qcc}]

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

（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

**程序2-4 初值为2的模8计数器**

module mod8\_counter

(

input [3:0]in,

input clk,

input CLR,

input LD,

input M,

output [3:0]Q,

output Qcc

);

initial begin

//\\in<=4'b0010;

end

wire CP;

wire LD16;

assign LD16 = (Q[3] & Q[0]) | (~LD)? 0 : 1;

lab4\_1\_divider sub\_lab4\_1\_divider

(

.in(in),

.clk(clk),

.CLR(CLR),

.LD(LD16),

.M(M),

.Q(Q),

.Qcc(Qcc)

);

endmodule

（b）仿真程序

**程序2-5 初值为2的模8计数器仿真程序**

module lab4\_1\_B\_sim

(

);

reg [3:0] in;

wire [3:0] Q;

reg CP, M, CLR, LD;

wire Qcc; // out pluse

initial begin // set init val

CP = 0;

in<=4'b0010;

M<=1;

CLR<=1;

LD<=1;

end

initial begin // set triger

#25 LD = 0; // output directly

#45 LD = 1;

#65 CLR = 0;//set 0

#85 CLR = 1;

end

always #10 CP = ~CP;

mod8\_counter mod8\_counter\_test

(

.in(in),

.clk(CP),

.CLR(CLR),

.LD(LD),

.M(M),

.Q(Q),

.Qcc(Qcc)

);

endmodule

（c）引脚约束（绑定）程序

**程序2-6 初值为2的模8计数器引脚约束**

#module mod8\_counter

#(

# input [3:0]in,

# input clk,

# input CLR,

# input LD,

# input M,

# output [3:0]Q,

# output Qcc

#);

# 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 {in[0]}]

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

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

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

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

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

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

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

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

set\_property PACKAGE\_PIN R6 [get\_ports {in[3]}]

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

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

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

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

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

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

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

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

set\_property PACKAGE\_PIN V6 [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 {Qcc}]

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

（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

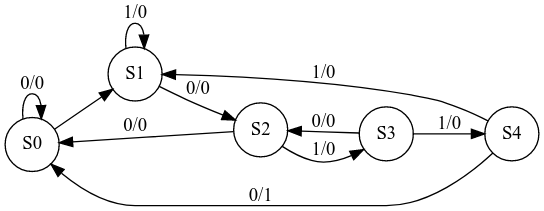


图2-1“10110”二进制序列检测器状态图

1. 源程序

**程序2-7 “10110”二进制序列检测器**

module FSM

(

input x,

input clk,

output reg y,

output reg [3:0] out

);

parameter S0=0,S1=1,S2=2,S3=3,S4=4;

reg [2:0] current\_state,next\_state;

initial begin

y<=0;

current\_state = S0;

out=4'b0000;

end

always @(negedge clk)begin

current\_state=next\_state;

if(current\_state==S0)

out<=4'b0000;

else if(current\_state==S1)

out<=4'b0001;

else if(current\_state==S2)

out<=4'b0010;

else if(current\_state==S3)

out<=4'b0100;

else if(current\_state==S4)

out<=4'b1000;

end

always @(current\_state or x) begin

case(current\_state)

S0:begin

y<=0;

if(x) begin

next\_state<=S1;

end

else begin

next\_state<=S0;

end

end

S1:begin

y<=0;

if(x) begin

next\_state<=S1;

end

else begin

next\_state<=S2;

end

end

S2:begin

y<=0;

if(x) begin

next\_state<=S3;

end

else begin

next\_state<=S0;

end

end

S3:begin

y<=0;

if(x) begin

next\_state<=S4;

end

else begin

next\_state<=S2;

end

next\_state<=x? S4:S2;

end

S4:begin

y<=x?0:1;

next\_state<=x? S1:S0;

end

default:begin

y<=0;

next\_state<=S0;

end

endcase

end

endmodule

1. 仿真程序

**程序2-8 “10110”二进制序列检测器仿真程序**

//module FSM

//(

// input x,

// input clk,

// output reg y,

// output reg [3:0] out

//);

module FSM\_sim

(

);

reg x;

reg clk;

wire y;

wire [2:0] current\_state;

FSM FSM\_test(x, clk, y, current\_state);

initial begin

x = 0;

clk = 0;

end

always #10 clk = ~clk;

initial begin //10110

x <= #45 1;

x <= #65 0;

x <= #85 1;

x <= #105 1;

x <= #125 0;

x <= #145 1;

x <= #165 0;

x <= #185 1;

x <= #205 1;

x <= #225 1;

//#25 $stop;

end

endmodule

（D）引脚约束（绑定）程序

**程序2-9 “10110”二进制序列检测器引脚约束**

set\_property CLOCK\_DEDICATED\_ROUTE FALSE [get\_nets 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}]

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

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

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

# 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 {out[0]}]

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

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

set\_property PACKAGE\_PIN R8 [get\_ports {out[1]}]

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

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

set\_property PACKAGE\_PIN T6 [get\_ports {out[2]}]

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

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

set\_property PACKAGE\_PIN T5 [get\_ports {out[3]}]

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

（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

**程序2-10 “**一位二进制数值比较器

module lab4\_3A

(

input A,

input B,

output reg [2:0] F

);

always@(A,B) begin

if(A>B)

begin

F<=3'b100;

end

else

if(A==B)

begin

F<=3'b010;

end

else

if(A<B) begin

F<=3'b001;

end

end

endmodule

（b）仿真程序

**程序2-11** 一位二进制数值比较器仿真程序

module lab4\_3A\_sim

(

);

reg A, B;

wire [2:0] F;

wire S;

initial begin

A = 1'b0;

B = 1'b0;

end

initial begin // set delay

#25 A = 1;

#45 B = 1;

#65 B = 0;

end

lab4\_3A lab4\_3A\_test(A,B,F);

endmodule

（c）引脚约束（绑定）程序

**程序2-12** 一位二进制数值比较器引脚约束

#module lab4\_3A

#(

# input A,

# input B,

# output reg [2:0] F

#);

# SWitches

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

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

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

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

set\_property PACKAGE\_PIN R6 [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 {F[0]}]

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

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

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

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

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

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

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

（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

程序说明：通过使用参数规定数据带宽进行空间循环生成电路

**程序2-13 3位二进制数值比较器**

module lab4\_3B

(

input [WIDTH-1:0] A,

input [WIDTH-1:0] B,

output [3\*WIDTH-1:0] F,

output [2:0] S

);

parameter WIDTH = 3;

integer i=0;

lab4\_3A lab4\_3A1(A[0],B[0],F[2:0]);

lab4\_3A lab4\_3A2(A[1],B[1],F[5:3]);

lab4\_3A lab4\_3A3(A[2],B[2],F[8:6]);

assign S[2] = F[8] | (F[7]&F[5]) | (F[7]&F[4]&F[2]);

assign S[1] = F[7] & F[4] & F[1];

assign S[0] = ~(S[2]|S[1]);

endmodule

1. 仿真程序

**程序2-14 3位二进制数值比较器仿真程序**

module lab4\_3\_sim

(

);

reg [2:0] A, B;

wire [2:0] F;

wire S;

initial begin

A = 3'b0;

B = 3'b0;

end

initial begin // set delay

#25 A[0] = 1;

#45 B[0] = 1;

#65 B[1] = 1;

#125 A[0] = 0;

#145 B[0] = 0;

#165 B[1] = 0;

end

lab4\_3 lab4\_3\_test(A, B,F,S);

endmodule

（c）引脚约束（绑定）程序

**程序2-15 3位二进制数值比较器引脚绑定程序**

#module lab4\_3B

#(

# input [WIDTH-1:0] A,

# input [WIDTH-1:0] B,

# output [3\*WIDTH-1:0] F,

# output [2:0] S

#);

# 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 {F[0]}]

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

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

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

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

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

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

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

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

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

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

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

set\_property PACKAGE\_PIN T5 [get\_ports {F[4]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {F[4]}]

#Bank = 34, Pin name = IO\_L12N\_T1\_MRCC\_34, Sch name = LED5

set\_property PACKAGE\_PIN T4 [get\_ports {F[5]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {F[5]}]

#Bank = 34, Pin name = IO\_L22P\_T3\_34, Sch name = LED6

set\_property PACKAGE\_PIN U7 [get\_ports {F[6]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {F[6]}]

#Bank = 34, Pin name = IO\_L22N\_T3\_34, Sch name = LED7

set\_property PACKAGE\_PIN U6 [get\_ports {F[7]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {F[7]}]

#Bank = 34, Pin name = IO\_L10N\_T1\_34, Sch name = LED8

set\_property PACKAGE\_PIN V4 [get\_ports {F[8]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {F[8]}]

#Bank = 34, Pin name = IO\_L8N\_T1\_34, Sch name = LED9

set\_property PACKAGE\_PIN U3 [get\_ports {S[0]}]

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

#Bank = 34, Pin name = IO\_L7N\_T1\_34, Sch name = LED10

set\_property PACKAGE\_PIN V1 [get\_ports {S[1]}]

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

#Bank = 34, Pin name = IO\_L17P\_T2\_34, Sch name = LED11

set\_property PACKAGE\_PIN R1 [get\_ports {S[2]}]

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

## 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2-2为4位二进制加法/减法计数器

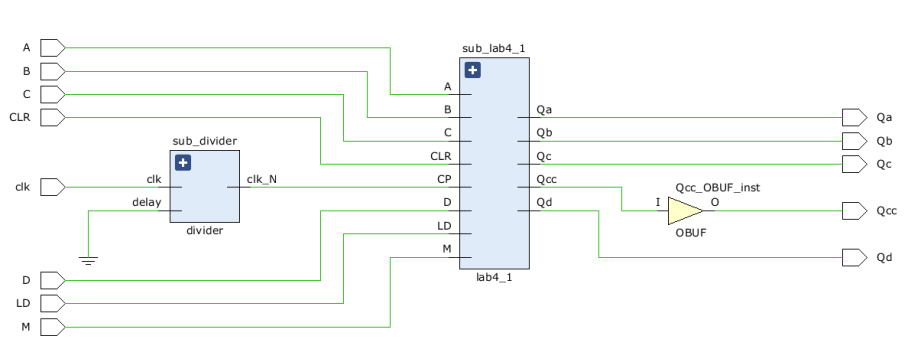


图2-2 模16加1/减1计数器整体电路图

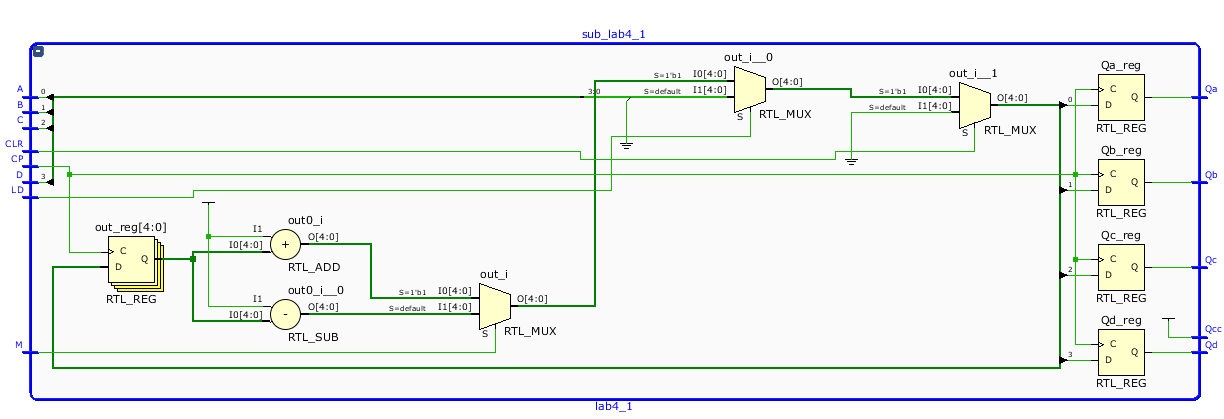


图2-3 模16加1/减1计数器模块电路图

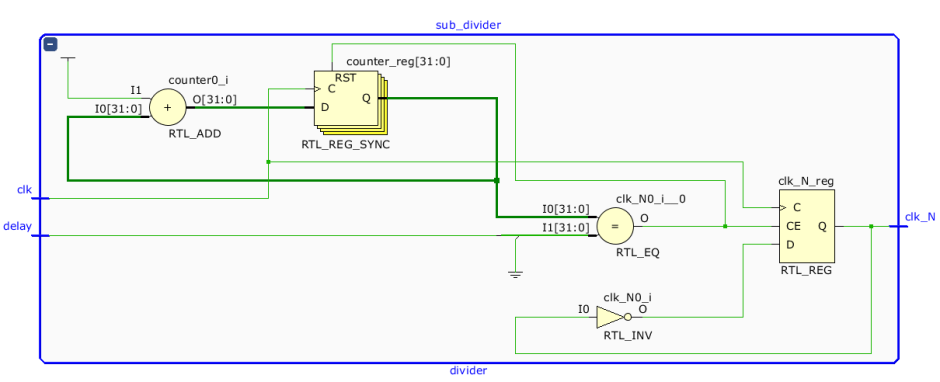


图2-4 模16加1/减1计数器分频器模块电路图

（B）模16计数器仿真结果截图（波形含CP，M，Qa，Qb，Qc，Qd，Q模8等）

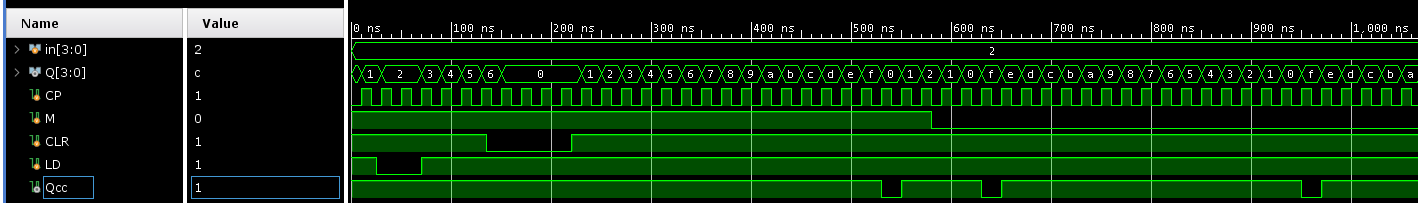


图2-5 模16加1/减1计数器行为仿真图

仿真结果分析：由图可看出LD低电平可直接输出预设值，Q为计数量，Qcc为进位、借位脉冲，M为功能选择。

（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为初值为2的模8计数器仿真

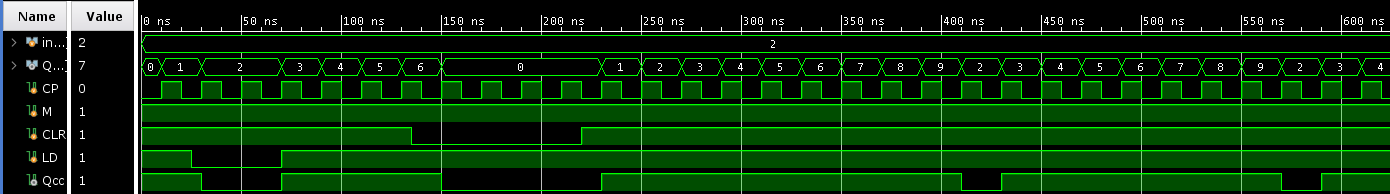


图2-6 初值为2的模8计数器仿真

仿真结果分析：经过初始的状态之后，进行模8计数，Qcc脉冲输出。

（C）开发板上的验证情况

验证结果：利用led输出，进行分频后控制开关完成预订功能，观察计数功能，达到设计。

（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序列检测器时序逻辑电路图

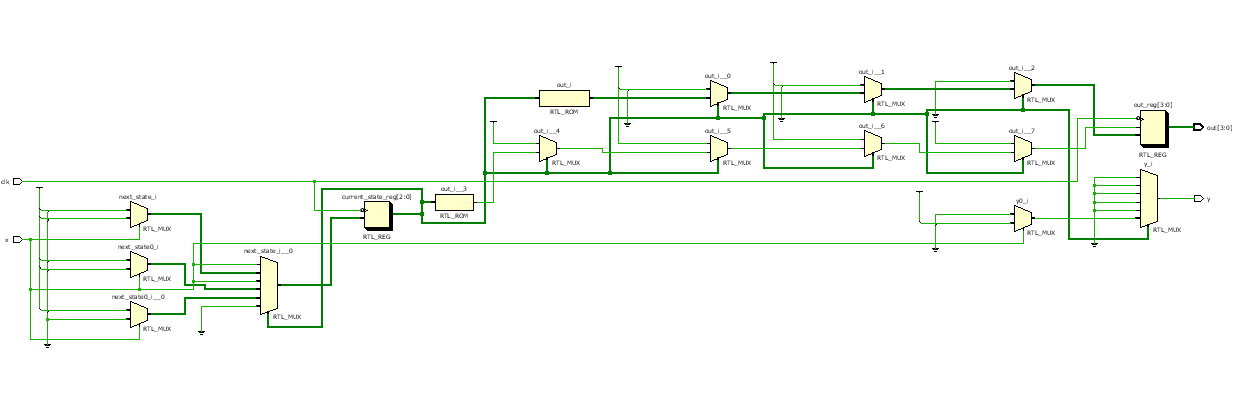


图2-7 10110序列检测器

（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为10110序列检测器仿真结果图

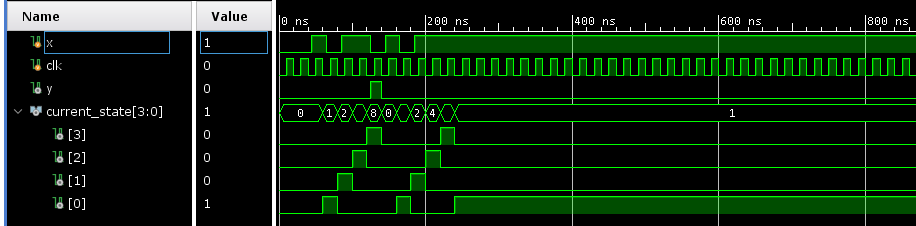


图2-8 10110序列检测器仿真

仿真结果分析：输出状态，进行状态转移仿真，当输入序列为”10110”时输出匹配脉冲y，而输出为”10111”时最后进入S1状态，未进行输出。

其中S0表示初始状态S1表示输入序列为1，S2表示输入序列为101，S3表示输入序列为101，S4表示输入序列为1011，S4根据输入状态决定输出为1或0.

根据如图所示状态变迁，满足要求。

（C）开发板上的验证情况（主要记录：验证过程和结论）

验证结果：利用四位LED输出状态，0代表S0，其余从右往左依次为S1S2S3S4进行结果输出,利用SW1模拟始终信号，SW0模拟输入脉冲，经过验证，结果满足设计。

**（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2-9为3位二进制数值比较器电路

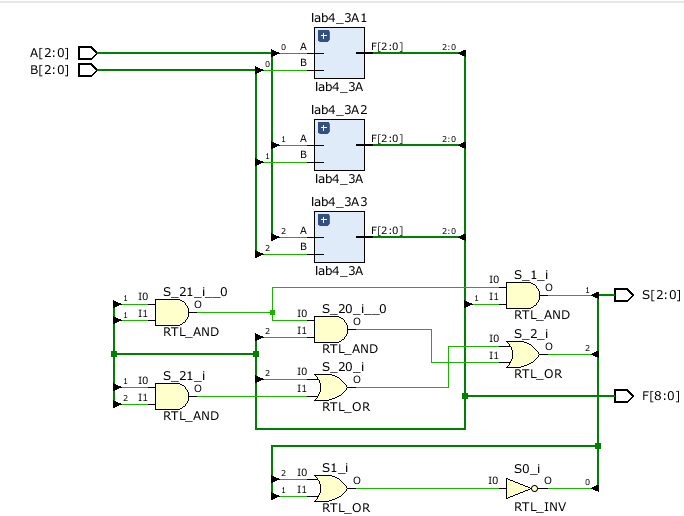


图2-9 3位二进制数值比较器电路

（B）仿真结果截图（含输入值，输出值等）

图2-10为3位二进制数值比较器仿真

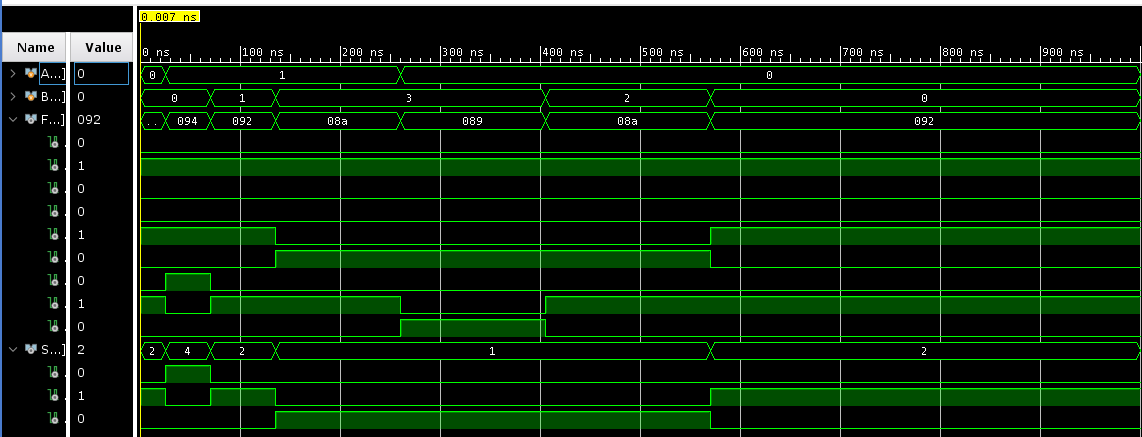


图2-7 3位二进制数值比较器仿真

仿真结果分析：利用F每三位进行输出位比较结果，S输出最终比较结果100表示A>B,010表示A=B，经过仿真图检验，结果正确。

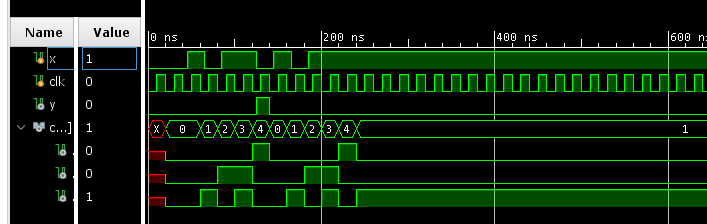
（C）开发板上的验证情况（主要记录：验证过程和结论）

验证情况：利用每三位led表示两个三位数之间的每一位大小差异，最终三位led表示最终的大小，经过验证，改变数值，结果改变，且满足设计。

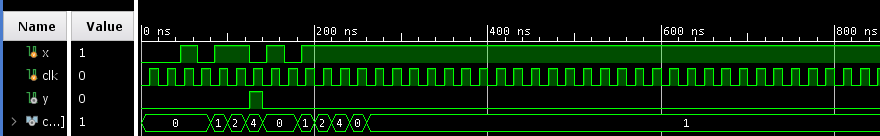
## 7、实验后的思考

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。

在进行状态机设计的时候，最初仿真出现了如下结果：



原因是直接对状态编码进行输出，而状态编码并未赋初值，故进行赋值处理，并进行状态输出：



通过底下的状态判断电路是否处于预订的状态，发现满足设想，档位10111时进入状态1.

仿真中出现的问题：

1. 仿真的频率设置过高，造成状态并未改变，而显示的状态不改变

解决：降低clk频率，进行仿真

1. 初始状态直接输出，造成存在不稳定状态，且无法对状态赋值

解决：利用另一个寄存器，进行，然后通过状态变量对寄存器赋值，那么输出的时候就不存在不稳定状态

1. 开始未阻塞赋值，造成输出的状态和实际的状态不一致

2．意见和建议

三四两次实验均是自己提前完成预习并完成书写，在实验课上仅仅是生成比特流并进行上板验证，但由于vivado2018速度很慢，导致稍有修改，就可能十几二十分钟的综合，而这段时间的利用率不高，考虑提前下发板子，进行设计。

实验中有一些设计要求较为严格，比如实验4.3三位比较器，可以采用for循环进行空间展开，代码简洁且自成模块，或者利用时序电路进行移位比较，使用较少的模块，而必须使用一位比较器，造成一定程度上的思路限制，考虑给与学生更大的设计空间。

整个实验中碰到了不少的问题，因为vivado缓存文件，造成一次失败，之后此次失败，因为未更新缓存，其中存在一些可以避免的坑，可以提前告知学生。

最后，谢谢老师的指导和帮助，让我们在不断的尝试中成长。