

# 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：

教师签名

**姓 名： 樊俊超**

**学 号： U201614702**

**班 级： 计科1607**

**指 导 教 师： 赵贻竹**

**计算机科学与技术学院**

**2018年6月5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 P17 [get\_ports flag]

set\_property IOSTANDARD LVCMOS33 [get\_ports flag]

set\_property PACKAGE\_PIN J15 [get\_ports {valid[0]}]

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

set\_property PACKAGE\_PIN L16 [get\_ports {valid[1]}]

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

set\_property PACKAGE\_PIN T18 [get\_ports {data[0]}]

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

set\_property PACKAGE\_PIN U18 [get\_ports {data[1]}]

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

set\_property PACKAGE\_PIN R13 [get\_ports {data[2]}]

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

set\_property PACKAGE\_PIN T8 [get\_ports {data[3]}]

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

set\_property PACKAGE\_PIN H17 [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：

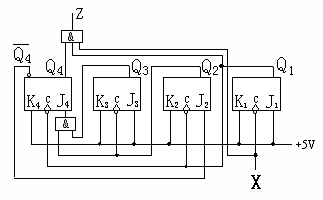


图1-2 脉冲异步计数器电路

（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

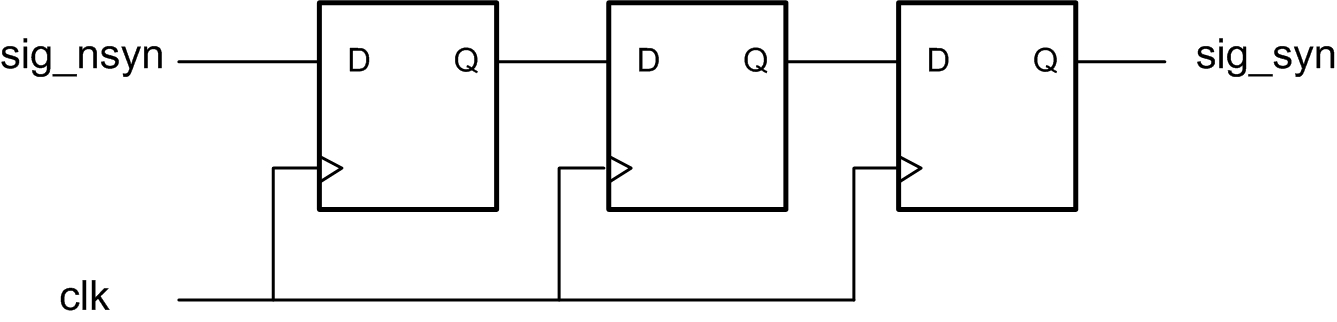


图1-3 3位移位寄存器电路

（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 LAB\_1**

（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

default:valid\_data=1'b0;

endcase

end

endmodule

（B）用“always块”设计实现一个同步时序逻辑电路

（a）源程序

`timescale 1ns / 1ps

module lab1\_b(x,clk,q0,q1,y,led);

input x,clk;

output q0,q1,y,led;

wire t0,t1;

assign t0=1;

assign t1=q0^x;

assign y=(~x) | q1;

assign led=x;

T test1(clk,t0,q0);

T test2(clk,t1,q1);

endmodule

module T(

input clk,t,

output q

);

reg q;

initial

q=1'b0;

always @(negedge clk) begin

if(t)

q<=~q;

else

q<=q;

end

endmodule

（b）仿真程序

`timescale 1ns / 1ps

module lab1\_b\_tb();

wire y,led;

reg x,clk;

lab1\_b test(x,clk,q0,q1,y,led);

initial begin

x=1'b1;

clk=1'b0;

end

always begin

#50 x = ~x;

#20 clk=~clk;

end

endmodule

**（2）脉冲异步计数器的分析和设计方案 LAB\_2**

（A）源程序

`timescale 1ns / 1ps

module lab2(

input x,

output z,q1,q2,q3,q4

);

wire j1, k1, c1, j2, k2, c2, j3, k3, c3, j4, k4, c4;

assign j1=1'b1;

assign k1=1'b1;

assign c1=x;

assign j2=~q4;

assign k2=1'b1;

assign c2=q1;

assign j3=1'b1;

assign k3=1'b1;

assign c3=q2;

assign j4=q2 & q3;

assign k4=1'b1;

assign c4=q1;

assign z=q4 & q1 & x;

choose jk1(j1,k1,c1,q1);

choose jk2(j2,k2,c2,q2);

choose jk3(j3,k3,c3,q3);

choose jk4(j4,k4,c4,q4);

endmodule

module choose(j,k,c,q);

input j,k,c;

output reg q;

initial

q=1'b0;

always @(negedge c) begin

case ({j,k})

2'b00: q<=q;

2'b01: q<=1'b0;

2'b11: q<=~q;

2'b10: q<=1'b1;

default: q<=q;

endcase

end

endmodule

（B）仿真程序

`timescale 1ns / 1ps

module lab2\_tb;

reg x;

wire z,q1,q2,q3,q4;

lab2 test(.x(x),.z(z),.q1(q1),.q2(q2),.q3(q3),.q4(q4));

initial begin

x=1'b0;

end

always begin

#20 x=~x;

end

endmodule

（C）引脚约束（绑定）程序

set\_property -dict { PACKAGE\_PIN U3 IOSTANDARD LVCMOS33 } [get\_ports { z }];

set\_property -dict { PACKAGE\_PIN U9 IOSTANDARD LVCMOS33 } [get\_ports { x }];

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

set\_property -dict { PACKAGE\_PIN P5 IOSTANDARD LVCMOS33 } [get\_ports { q1 }];

set\_property -dict { PACKAGE\_PIN U1 IOSTANDARD LVCMOS33 } [get\_ports { q2 }];

set\_property -dict { PACKAGE\_PIN R2 IOSTANDARD LVCMOS33 } [get\_ports { q3 }];

set\_property -dict { PACKAGE\_PIN P2 IOSTANDARD LVCMOS33 } [get\_ports { q4 }];

**（3）Verilog设计中一致性问题的解决方案 LAB\_3**

（A）源程序

`timescale 1ns/1ps

module lab3(clk,sig\_nsyn,q1,q2,sig\_syn,out);

input clk,sig\_nsyn;

output q1,q2,sig\_syn,out;

wire clk\_N;

divider #(75\_000\_000) testa(clk,clk\_N);

pipe3b testb(sig\_nsyn,clk\_N,q1,q2,sig\_syn,out);

// pipe3b testb(sig\_nsyn,clk,q1,q2,sig\_syn,out);

endmodule

module pipe3b(

input sig\_nsyn,clk,

output reg q1,q2,sig\_syn,

output out

);

initial begin

q1<=0;

q2<=0;

sig\_syn<=0;

end

assign out=sig\_nsyn;

always @(posedge clk) begin

q2<=q1;

sig\_syn<=q2;

q1<=sig\_nsyn;

end

endmodule

module divider (clk, clk\_N);

input clk; // 系统时钟

output reg clk\_N; // 分频后的时钟

parameter N = 1000\_000\_000; // 1Hz的时钿,N=fclk/fclk\_N

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

当计数器仿0计数刿(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)

begin

clk\_N<=~clk\_N;

counter<=0;

end

else

counter<=counter+1;

end

endmodule

（B）仿真程序

`timescale 1ns/1ps

module lab3\_tb();

reg sig\_nsyn,clk;

wire q1,q2,sig\_syn,out;

lab3 test(clk,sig\_nsyn,q1,q2,sig\_syn,out);

initial begin

sig\_nsyn<=1'b1;

clk<=1'b0;

end

always begin

#20 clk=~clk;

#60 sig\_nsyn=~sig\_nsyn;

end

endmodule

（C）约束文件

set\_property PACKAGE\_PIN U9 [get\_ports sig\_nsyn]

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

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

set\_property IOSTANDARD LVCMOS33 [get\_ports clk]

set\_property PACKAGE\_PIN T8 [get\_ports out]

set\_property IOSTANDARD LVCMOS33 [get\_ports out]

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

set\_property IOSTANDARD LVCMOS33 [get\_ports q1]

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

set\_property IOSTANDARD LVCMOS33 [get\_ports q2]

set\_property PACKAGE\_PIN T6 [get\_ports sig\_syn]

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



**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**

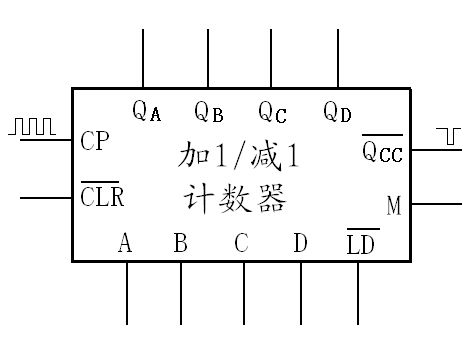
设计一个能清零、置数和进位/借位输出的加1/减1的4位二进制计数器，其结构框图如图2-1所示。

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 LAB\_4**

（A）模16加1/减1计数器

（a）源程序

`timescale 1ns/1ps

module lab4(sin,cp,m,ld,clr,qout,qcc,led);

parameter min=4'b0000;

parameter max=4'b1111;

input [3:0]sin;

input cp,clr,ld,m;

output reg [3:0] qout;

output reg qcc;

output [6:0]led;

initial begin

qout<=min;

qcc<=1'b1;

end

assign led[3:0]=sin[3:0];

assign led[4]=m;

assign led[5]=ld;

assign led[6]=clr;

always@(posedge cp) begin

if(~clr) begin

qout<=min;

qcc<=1'b1;

end

else if(~ld) begin

qout<=sin;

qcc<=1'b1;

end

else if(m) begin

if(qout==max) begin

qout<=min;

qcc<=1'b0;

end

else begin

qout<=qout+1;

qcc<=1'b1;

end

end

else begin

if(qout==min) begin

qout<=max;

qcc<=1'b0;

end

else begin

qout<=qout-1;

qcc<=1'b1;

end

end

end

endmodule

（b）仿真程序

module lab4tb();

integer i;

reg [3:0] sin;

reg cp,m,ld,clr;

wire [3:0] qout;

wire qcc;

wire [6:0]led;

initial begin

sin=4'b0100;

m=1'b1;

ld=1'b1;

clr=1'b1;

cp=1'b0;

ld=1'b0;

#80

ld=1'b1;

#80

ld=1'b0;

#80

ld=1'b1;

clr=1'b0;

#30

clr=1'b1;

end

lab4 test(.sin(sin),.cp(cp),.m(m),.ld(ld),.clr(clr),.qout(qout),.qcc(qcc),.led(led));

always begin

for (i=0; i<100;i=i+1) begin

#20 cp=~cp;

if (i==50)

m=~m;

end

end

endmodule

（c）引脚约束（绑定）程序

set\_property -dict { PACKAGE\_PIN E3 IOSTANDARD LVCMOS33 } [get\_ports { cp }];

set\_property -dict { PACKAGE\_PIN U9 IOSTANDARD LVCMOS33 } [get\_ports { sin[0] }];

set\_property -dict { PACKAGE\_PIN U8 IOSTANDARD LVCMOS33 } [get\_ports { sin[1] }];

set\_property -dict { PACKAGE\_PIN R7 IOSTANDARD LVCMOS33 } [get\_ports { sin[2] }];

set\_property -dict { PACKAGE\_PIN R6 IOSTANDARD LVCMOS33 } [get\_ports { sin[3] }];

set\_property -dict { PACKAGE\_PIN R5 IOSTANDARD LVCMOS33 } [get\_ports { m }];

set\_property -dict { PACKAGE\_PIN V7 IOSTANDARD LVCMOS33 } [get\_ports { ld }];

set\_property -dict { PACKAGE\_PIN V6 IOSTANDARD LVCMOS33 } [get\_ports { clr }];

set\_property -dict { PACKAGE\_PIN T8 IOSTANDARD LVCMOS33 } [get\_ports { led[0] }];

set\_property -dict { PACKAGE\_PIN V9 IOSTANDARD LVCMOS33 } [get\_ports { led[1] }];

set\_property -dict { PACKAGE\_PIN R8 IOSTANDARD LVCMOS33 } [get\_ports { led[2] }];

set\_property -dict { PACKAGE\_PIN T6 IOSTANDARD LVCMOS33 } [get\_ports { led[3] }];

set\_property -dict { PACKAGE\_PIN T5 IOSTANDARD LVCMOS33 } [get\_ports { led[4] }];

set\_property -dict { PACKAGE\_PIN T4 IOSTANDARD LVCMOS33 } [get\_ports { led[5] }];

set\_property -dict { PACKAGE\_PIN U7 IOSTANDARD LVCMOS33 } [get\_ports { led[6] }];

set\_property -dict { PACKAGE\_PIN U6 IOSTANDARD LVCMOS33 } [get\_ports { led[7] }];

set\_property -dict { PACKAGE\_PIN R1 IOSTANDARD LVCMOS33 } [get\_ports{ qout[0] }];

set\_property -dict { PACKAGE\_PIN P5 IOSTANDARD LVCMOS33 } [get\_ports { qout[1] }];

set\_property -dict { PACKAGE\_PIN U1 IOSTANDARD LVCMOS33 } [get\_ports { qout[2] }];

set\_property -dict { PACKAGE\_PIN R2 IOSTANDARD LVCMOS33 } [get\_ports { qout[3] }];

set\_property -dict { PACKAGE\_PIN P2 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。 LAB\_5**

（a）源程序

`timescale 1ns/1ps

module lab5(sin,cp,m,ld,clr,qout,qcc,led);

input [3:0]sin;

input cp,clr,ld,m;

output [3:0] qout;

output qcc;

output [6:0]led;

//wire clk\_N;

//divider #(25\_000\_000) divider1(.clk(cp),.clk\_N(clk\_N));

//counter #(4'b0010,4'b1001) fun(sin,clk\_N,m,ld,clr,qout,qcc,led);

counter #(4'b0010,4'b1001) fun(sin,cp,m,ld,clr,qout,qcc,led);

endmodule

counter模块

module counter(sin,cp,m,ld,clr,qout,qcc,led);

parameter min=4'b0000;

parameter max=4'b1111;

input [3:0]sin;

input cp,clr,ld,m;

output reg [3:0] qout;

output reg qcc;

output [6:0]led;

initial begin

qout<=min;

qcc<=1'b1;

end

assign led[3:0]=sin[3:0];

assign led[4]=m;

assign led[5]=ld;

assign led[6]=clr;

always@(posedge cp) begin

if(~clr) begin

qout<=min;

qcc<=1'b1;

end

else if(~ld) begin

qout<=sin;

qcc<=1'b1;

end

else if(m) begin

if(qout==max) begin

qout<=min;

qcc<=1'b0;

end

else begin

qout<=qout+1;

qcc<=1'b1;

end

end

else begin

if(qout==min) begin

qout<=max;

qcc<=1'b0;

end

else begin

qout<=qout-1;

qcc<=1'b1;

end

end

end

endmodule

divider模块

module divider (clk, clk\_N);

input clk;

output reg clk\_N;

parameter N = 100\_000\_000;

reg [31:0] counter;

always @(posedge clk) begin

if(counter>=N/2)

begin

clk\_N<=~clk\_N;

counter<=0;

end

else

counter<=counter+1;

end

endmodule

（b）仿真程序

module lab5tb();

integer i;

reg [3:0] sin;

reg cp,m,ld,clr;

wire [3:0] qout;

wire qcc;

wire [6:0]led;

initial begin

sin=4'b0100;

m=1'b1;

ld=1'b1;

clr=1'b1;

cp=1'b0;

end

lab5 test(.sin(sin),.cp(cp),.m(m),.ld(ld),.clr(clr),.qout(qout),.qcc(qcc),.led(led));

always begin

for (i=0; i<60;i=i+1) begin

#20 cp=~cp;

if (i==30)

m=~m;

end

end

endmodule

（c）引脚约束（绑定）程序

set\_property -dict { PACKAGE\_PIN E3 IOSTANDARD LVCMOS33 } [get\_ports { cp }];

set\_property -dict { PACKAGE\_PIN U9 IOSTANDARD LVCMOS33 } [get\_ports { sin[0] }];

set\_property -dict { PACKAGE\_PIN U8 IOSTANDARD LVCMOS33 } [get\_ports { sin[1] }];

set\_property -dict { PACKAGE\_PIN R7 IOSTANDARD LVCMOS33 } [get\_ports { sin[2] }];

set\_property -dict { PACKAGE\_PIN R6 IOSTANDARD LVCMOS33 } [get\_ports { sin[3] }];

set\_property -dict { PACKAGE\_PIN R5 IOSTANDARD LVCMOS33 } [get\_ports { m }];

set\_property -dict { PACKAGE\_PIN V7 IOSTANDARD LVCMOS33 } [get\_ports { ld }];

set\_property -dict { PACKAGE\_PIN V6 IOSTANDARD LVCMOS33 } [get\_ports { clr }];

set\_property -dict { PACKAGE\_PIN T8 IOSTANDARD LVCMOS33 } [get\_ports { led[0] }];

set\_property -dict { PACKAGE\_PIN V9 IOSTANDARD LVCMOS33 } [get\_ports { led[1] }];

set\_property -dict { PACKAGE\_PIN R8 IOSTANDARD LVCMOS33 } [get\_ports { led[2] }];

set\_property -dict { PACKAGE\_PIN T6 IOSTANDARD LVCMOS33 } [get\_ports { led[3] }];

set\_property -dict { PACKAGE\_PIN T5 IOSTANDARD LVCMOS33 } [get\_ports { led[4] }];

set\_property -dict { PACKAGE\_PIN T4 IOSTANDARD LVCMOS33 } [get\_ports { led[5] }];

set\_property -dict { PACKAGE\_PIN U7 IOSTANDARD LVCMOS33 } [get\_ports { led[6] }];

set\_property -dict { PACKAGE\_PIN R1 IOSTANDARD LVCMOS33 } [get\_ports { qout[0] }];

set\_property -dict { PACKAGE\_PIN P5 IOSTANDARD LVCMOS33 } [get\_ports { qout[1] }];

set\_property -dict { PACKAGE\_PIN U1 IOSTANDARD LVCMOS33 } [get\_ports { qout[2] }];

set\_property -dict { PACKAGE\_PIN R2 IOSTANDARD LVCMOS33 } [get\_ports { qout[3] }];

set\_property -dict { PACKAGE\_PIN P2 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 LAB\_6**

（A）串行5位不可重叠的“10110”二进制序列检测器的状态图

A:初始状态为0

B:1； C:10; D:101; E:1011; F:10110

状态图如图2-2

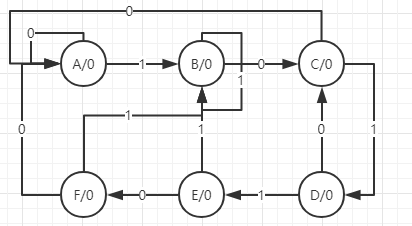


图2-2

（B）源程序

`timescale 1ns/1ps

module lab6(clk,x,z,led);

input clk,x;

output reg z;

output [3:0]led;

reg [2:0]state;

assign led[2:0]=state[2:0];

assign led[3]=x;

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

initial begin

state<=A;

z<=1'b0;

end

always @(posedge clk) begin

case(state)

A:begin

z<=1'b0;

if (x)

state<=B;

else

state<=A;

end

B:begin

z<=1'b0;

if (x)

state<=B;

else

state<=C;

end

C:begin

z<=1'b0;

if (x)

state<=D;

else

state<=A;

end

D:begin

z<=1'b0;

if (x)

state<=E;

else

state<=C;

end

E:begin

z<=1'b1;

if (x)

state<=B;

else

state<=F;

end

F:begin

z<=1'b0;

if (x)

state<=B;

else

state<=A;

end

default:begin

state<=A;

end

endcase

end

endmodule

（C）仿真程序

`timescale 1ns/1ps

module lab6tb();

reg x,clk;

wire z,led;

integer count=109;

lab6 test(clk,x,z,led);

initial begin

x=1'b0;

clk=1'b0;

end

always begin

#20 clk=~clk;

#45 x=count&1;

count=count>>1;

#20 clk=~clk;

end

endmodule

（D）引脚约束（绑定）程序

set\_property -dict { PACKAGE\_PIN U9 IOSTANDARD LVCMOS33 } [get\_ports { x }]

## Clock signal

set\_property -dict { PACKAGE\_PIN U8 IOSTANDARD LVCMOS33 } [get\_ports { clk }]

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

set\_property -dict { PACKAGE\_PIN U7 IOSTANDARD LVCMOS33 } [get\_ports { led[0] }]

set\_property -dict { PACKAGE\_PIN U6 IOSTANDARD LVCMOS33 } [get\_ports { led[1] }]

set\_property -dict { PACKAGE\_PIN V4 IOSTANDARD LVCMOS33 } [get\_ports { led[2] }]

set\_property -dict { PACKAGE\_PIN T8 IOSTANDARD LVCMOS33 } [get\_ports { led[3] }]

set\_property -dict { PACKAGE\_PIN P2 IOSTANDARD LVCMOS33 } [get\_ports { z }]

**（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 LAB\_7**

（a）源程序

`timescale 1ns/1ps

module lab7(A,B,F1,F2,F3);

input A,B;

output F1,F2,F3;

assign F1=(A>B)? 1'b1:1'b0;

assign F2=(A==B)? 1'b1:1'b0;

assign F3=(A<B)? 1'b1:1'b0;

endmodule

（b）仿真程序

`timescale 1ns/1ps

module lab7tb();

reg A,B;

wire F1,F2,F3;

lab7 test(A,B,F1,F2,F3);

integer countA=12;

integer countB=5;

integer i;

initial begin

A=1'b0;

B=1'b0;

end

always begin

for (i=0; i<4; i=i+1) begin

#20 A=countA&1;

B=countB&1;

countA=countA>>1;

countB=countB>>1;

end

end

endmodule

（c）引脚约束（绑定）程序

##SW[0]:A SW[1]:B

set\_property PACKAGE\_PIN J15 [get\_ports A]

set\_property IOSTANDARD LVCMOS33 [get\_ports A]

set\_property PACKAGE\_PIN L16 [get\_ports B]

set\_property IOSTANDARD LVCMOS33 [get\_ports B]

##LED[2:0]:F3,F2,F1

set\_property PACKAGE\_PIN H17 [get\_ports F1]

set\_property IOSTANDARD LVCMOS33 [get\_ports F1]

set\_property PACKAGE\_PIN K15 [get\_ports F2]

set\_property IOSTANDARD LVCMOS33 [get\_ports F2]

set\_property PACKAGE\_PIN J13 [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。 LAB\_8**

（a）源程序

`timescale 1ns/1ps

module lab8(A,B,F1,F2,F3);

input [2:0]A,B;

output F1,F2,F3;

wire [2:0]FA,FB,FC,TA,TB;

lab7 TEST1(A[2],B[2],FA[2],FB[2],FC[2]);

lab7 TEST2(A[1],B[1],FA[1],FB[1],FC[1]);

lab7 TEST3(A[0],B[0],FA[0],FB[0],FC[0]);

assign F1=FA[2] | (FA[1] & FB[2]) | (FB[2] & FB[1] & FA[0]);

assign F2=FB[2] & FB[1] & FB[0];

assign F3=FC[1] | (FB[2] & FC[1]) | (FB[2] & FB[1] & FC[0]);

endmodule

（b）仿真程序

`timescale 1ns/1ps

module lab7tb();

reg [2:0]A,B;

wire F1,F2,F3;

lab8 test(A,B,F1,F2,F3);

integer countA=12;

integer countB=5;

integer i;

initial begin

A=3'b000;

B=3'b000;

end

always begin

for (i=0; i<4; i=i+1) begin

#20 A=countA&7;

B=countB&7;

countA=countA>>1;

countB=countB>>1;

end

end

endmodule

（c）引脚约束（绑定）程序

##SW[2:0]:A SW[5:3]:B

set\_property PACKAGE\_PIN J15 [get\_ports A[0]]

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

set\_property PACKAGE\_PIN L16 [get\_ports A[1]]

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

set\_property PACKAGE\_PIN M13 [get\_ports A[2]]

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

set\_property PACKAGE\_PIN R15 [get\_ports B[0]]

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

set\_property PACKAGE\_PIN R17 [get\_ports B[1]]

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

set\_property PACKAGE\_PIN T18 [get\_ports B[2]]

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

##LED[2:0]:F3,F2,F1

set\_property PACKAGE\_PIN H17 [get\_ports F1]

set\_property IOSTANDARD LVCMOS33 [get\_ports F1]

set\_property PACKAGE\_PIN K15 [get\_ports F2]

set\_property IOSTANDARD LVCMOS33 [get\_ports F2]

set\_property PACKAGE\_PIN J13 [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**）**

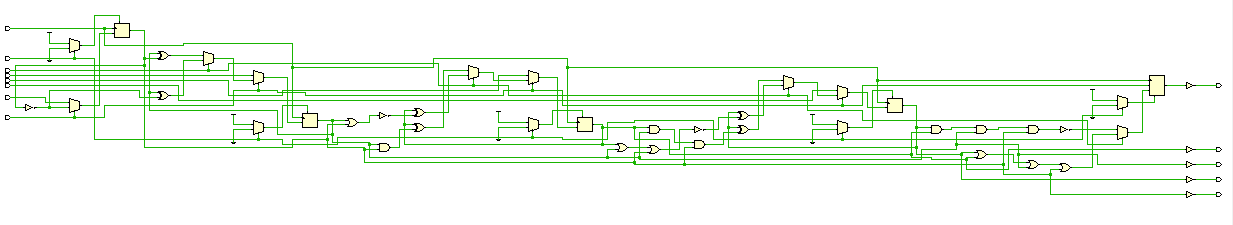


图2-3 4位二进制加法/减法计数器

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

输入参数：工作模式m，输入sin[3:0]，预置ld，清零端clr，时钟控制信号cp

输出参数：结果qout[3:0]，借位进位信号qcc，led灯

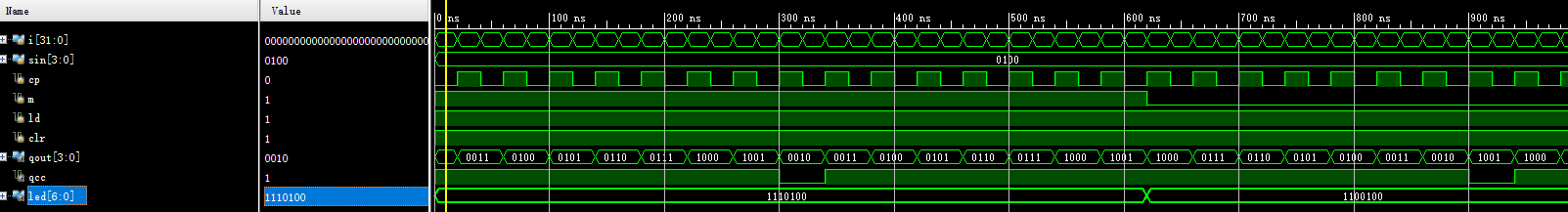


图2-4 初值为2的模8计数器仿真

**（C）开发板上的验证情况（主要记录：验证过程和结论）**

**（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**）**

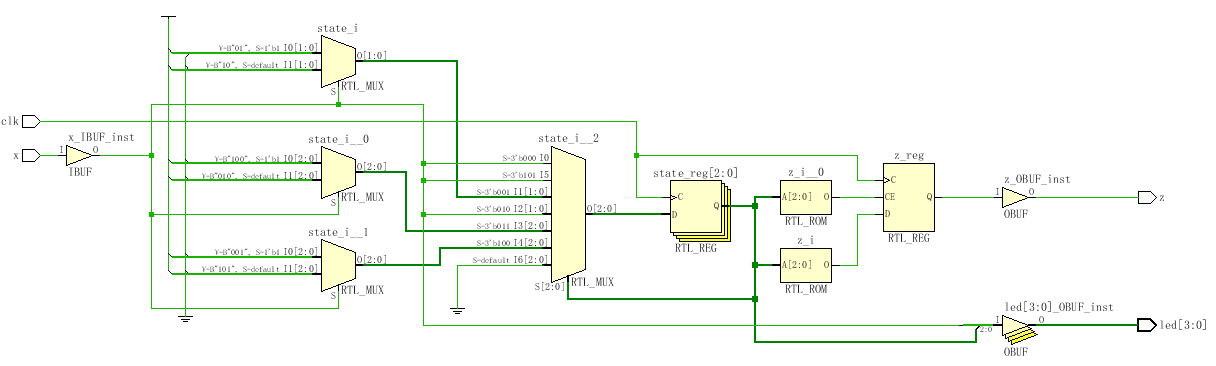


图2-5 10110序列检测器

**（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）

输入参数：脉冲x，时钟信号clk；输出参数：结果z，状态及输入显示led

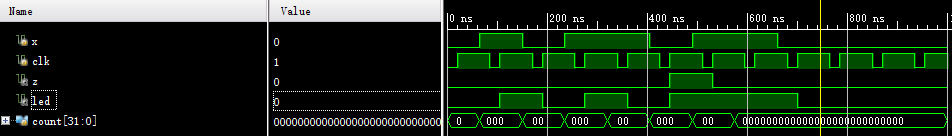


图2-6 10110序列检测器仿真

**（C）开发板上的验证情况（主要记录：验证过程和结论）**

**（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**）**

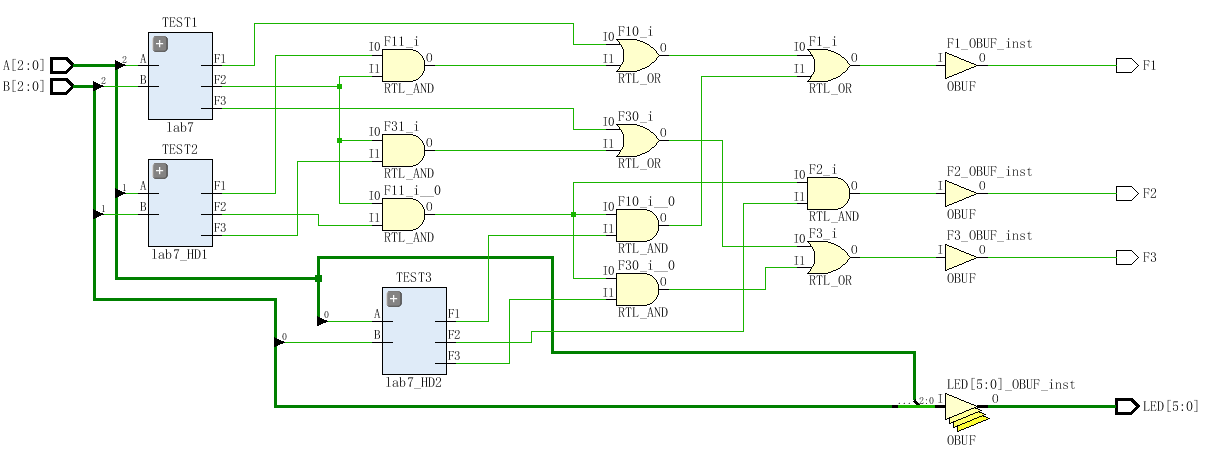


图2-7 3位二进制数值比较器

**（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）

输入参数：A[2:0],B[2:0]；输出参数：F1,F2,F3,输入状态显示led[5:0]

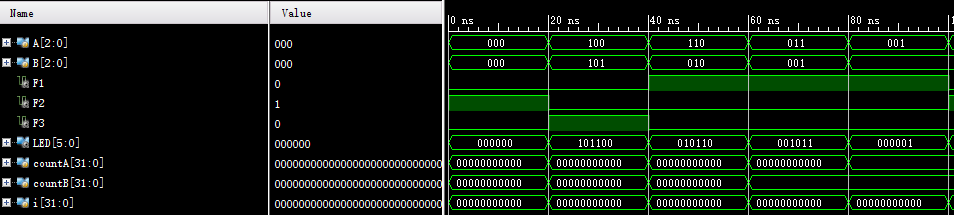


图2-8 3位二进制数值比较器仿真

**（C）开发板上的验证情况（主要记录：验证过程和结论）**

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。**

拿模16来说，老师要求仿真时要同时表示出累加累减，预置初值，清零的作用，经过思考为表示预置和清零的区别，在工作一段时间后我令ld发挥作用，清零端不干预，接着令清零端发挥作用，这些都在仿真中完美表示出来。接下来，我又利用#设置使工作模式m在一段时间后反转，仿真成功。

**2．意见和建议**

这次实验主要是很繁琐，最开始又不明白模8那个初值为2的意思，浪费了很多时间，导致预习报告写完也没有当场检查几个。在后面去其他班蹭检查真的是身心俱疲一点收获都没有，但这次实验我一点一点的把我认为的问题揪出来解决还是很满足的。