

**数字逻辑实验报告（3）**

**团队成员：**

|  |  |  |  |
| --- | --- | --- | --- |
| **姓名** | **班级** | **学号** | **贡献百分比** |
| **胡思勖** | **计卓1501** | **U201514898** | **50%** |
| **邓一夫** | **计卓1501** | **U201514629** | **50%** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **实验部分：实验完成结果、时间**  **（亮点、完成、基本完成、未完成）** | | | | **总分**  **（实验部分70% +报告30%）** |
|  | **第一个实验** | **第二个实验** | **第三个实验** |  |
| **检查结果** |  |  |  |
| **检查名次** |  |  |  |
| **检查老师** |  |  |  |

**报告人 ： 胡思勖**

**实验指导教师： 何云峰**

**报告批阅教师：**

**计算机科学与技术学院**

**2017年5月24日**

一、实验名称

Verilog HDL设计数字逻辑电路。

二、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。

三、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。

四、实验内容（在Xilinx NEXYS 4开发板完成）

1、组合、时序逻辑电路的“always”设计（必选）

**（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3-1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：

（1）验证程序3-1在“综合”时，是否会出现上述问题；

（2）如果存在上述问题，请更正程序3-1，帮这位同学完成设计。

程序3-1 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

//**电路输入输出绑定的FPGA引脚与开发板的新旧有关//标有“DDR”为新板//////////////////////以下为老板/**///////////////////

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3-1所示电路。



图3-1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（必选）

分析如图3-2所示的脉冲异步计数器电路，完成如下内容：

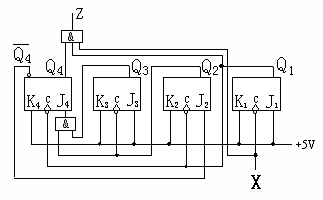


图3-2 脉冲异步计数器电路

（1）说明该计数器的模是多少？

（2）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（可选）

分析如图3-3所示的电路，完成如下内容：

（1）程序3-2是对图3-3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3-2分别进行仿真，如果有错误，请更正程序3-2，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

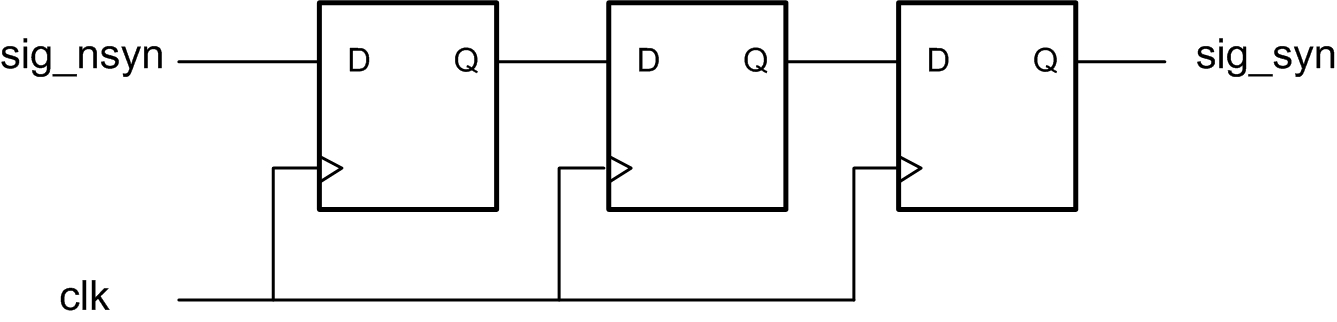


图3-3 3位移位寄存器电路

（2）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3-2中添加一个时钟分频部分，并将降频后的时钟信号接到图3-3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3-2 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

五、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3-1更正后的源程序。

更正后的程序3-1 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]; else valid\_data=0; end

2'b01 : begin if(flag) valid\_data = data[1]; else valid\_data=0;end

2'b10 : begin if(flag) valid\_data = data[2]; else valid\_data=0;end

2'b11 : begin if(flag) valid\_data = data[3]; else valid\_data=0;end

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

module lab3\_B(

input x,

input cp,

output reg y);

reg Q0,Q1;

initial begin

Q0<=0;

Q1<=0;

y<=0;

end

reg clk\_down = 0;

reg [31:0] cnt = 0;

parameter times = 10000000;

always@(posedge cp)

begin

if(cnt >= times)

begin

clk\_down <= ~clk\_down;

cnt <= 0;

end

else begin

cnt <= cnt + 1;

end

end

always @(negedge clk\_down)

begin

Q0<=~Q0;

Q1<=(x^Q0)^Q1;

y <=~(x&~Q1);

end

endmodule

（b）仿真程序

module lab3\_B\_sim;

wire y;

reg x;

reg cp;

wire Q0;

wire Q1;

initial begin

x <= 0;

cp <= 0;

end;

always #150 x=~x;

always #5 cp=~cp;

lab3\_B test(x,cp,y,Q0,Q1);

endmodule2、脉冲异步计数器的分析和设计方案

（A）源程序

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

input clk,j,k;

output q,qb;

reg q;

assign qb=~q;

initial begin

q=0;

end

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

module count(

input x,

output z,

output q1,

output q2,

output q3,

output q4

);

wire j1, k1, ck1, j2, k2, ck2, j3, k3, ck3, j4, k4, ck4, nq4;

assign j1 = 1;

assign k1 = 1;

assign ck1 = x;

assign j2 = nq4;

assign k2 = 1;

assign ck2 = q1;

assign j3 = 1;

assign k3 = 1;

assign ck3 = q2;

assign j4 = q2 & q3;

assign k4 = 1;

assign ck4 = q1;

assign z = x & q1 & q4;

J\_K jk1(j1, k1, ck1, q1, );

J\_K jk2(j2, k2, ck2, q2, );

J\_K jk3(j3, k3, ck3, q3, );

J\_K jk4(j4, k4, ck4, q4, nq4);

endmodule

（B）仿真程序

module td;

reg x;

wire q1,q2,q3,q4,z;

integer i;

count cou(

x,z,q1,q2,q3,q4

);

initial

begin

for(i=0;i<255;i=i+2)

begin

#5 x=1;

#5 x=0;

end

end

endmodule

（C）引脚约束（绑定）程序

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

set\_property -dict { PACKAGE\_PIN H17 IOSTANDARD LVCMOS33 } [get\_ports { q4 }]; #IO\_L18P\_T2\_A24\_15 Sch=led[0]

set\_property -dict { PACKAGE\_PIN K15 IOSTANDARD LVCMOS33 } [get\_ports { q3 }]; #IO\_L24P\_T3\_RS1\_15 Sch=led[1]

set\_property -dict { PACKAGE\_PIN J13 IOSTANDARD LVCMOS33 } [get\_ports { q2 }]; #IO\_L17N\_T2\_A25\_15 Sch=led[2]

set\_property -dict { PACKAGE\_PIN N14 IOSTANDARD LVCMOS33 } [get\_ports { q1 }]; #IO\_L8P\_T1\_D11\_14 Sch=led[3]

set\_property -dict { PACKAGE\_PIN R18 IOSTANDARD LVCMOS33 } [get\_ports { z }]; #IO\_L7P\_T1\_D09\_14 Sch=led[4]

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

**给出程序3-2更正后的程序。**

1. 源程序（前仿真和后仿真以及下载验证都正确）

`timescale 1ns/1ps

module pipe3b(

input sig\_nsyn,

input clk,

output reg q1,

output reg q2,

output reg sig\_syn

);

initial begin

q1 <= 0;

q2 <= 0;

sig\_syn <= 0;

end

always@(posedge clk) q2 <= q1;

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

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

endmodule

module divider(

input clk,

output reg ck

);

parameter t=10;

integer cnt;

initial begin

cnt <= 0;

ck <= 0;

end

always @(posedge clk)

begin

cnt = cnt+1;

if(cnt == t)

begin

ck = ~ck;

cnt = 0;

end

end

endmodule

（B）仿真程序

`timescale 1ns/1ps

module pipe3b\_sim(

output q1,

output q2,

output sig\_syn

);

reg sig\_nsyn;

reg CLK;

initial begin

sig\_nsyn <= 0;

CLK <= 0;

end

always #10 CLK <= ~CLK;

always #1000 sig\_nsyn <= ~sig\_nsyn;

divider tempdiv(CLK, ck);

pipe3b test(sig\_nsyn, ck, q1, q2, sig\_syn);

endmodule

六、实验结果记录

1、组合、时序逻辑电路“always”设计的记录结果

**（A）用“always块”设计纯组合逻辑电路**

­­**（a）程序3-1编译后产生的逻辑电路图（**RTL Analysis下的“Schematic”**）**

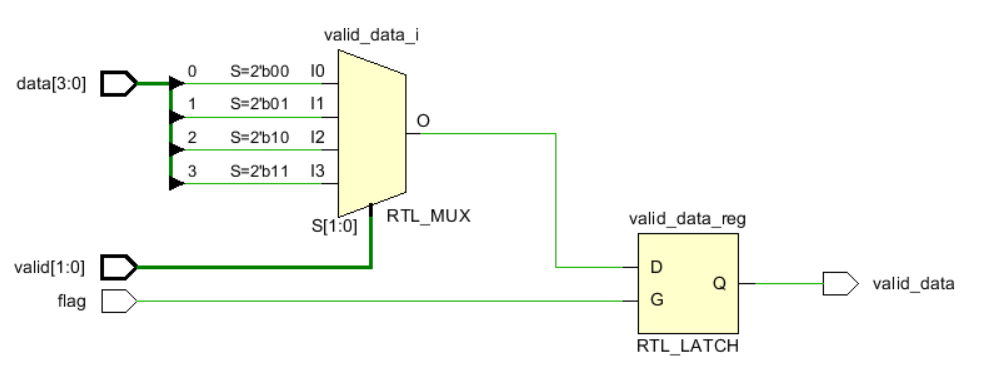


图3-4 4选1的多路选择器

**（b）程序3-1更正后产生的纯组合逻辑电路图（**RTL Analysis下的“Schematic”**）**

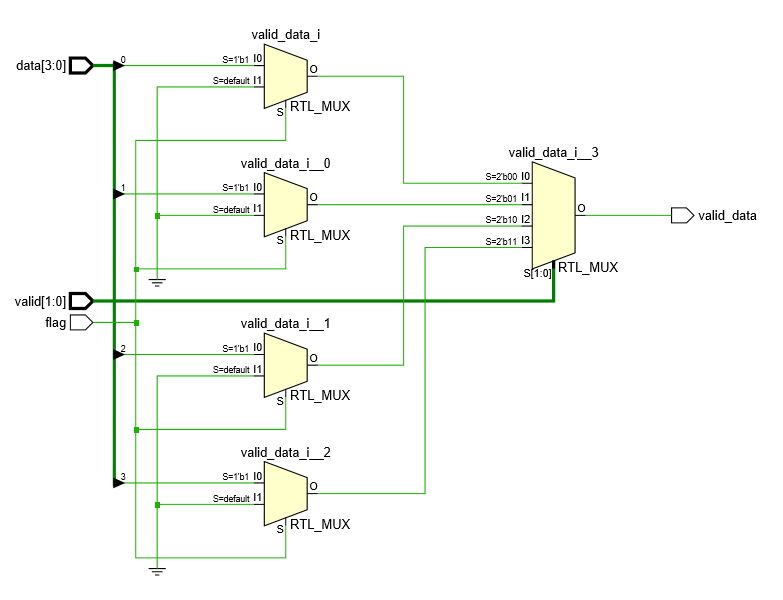


图3-5 4选1的多路选择器

**（B）用“always块”设计一个同步时序逻辑电路**

**（a）图3-1同步时序逻辑电路图（RTL Analysis下的“Schematic”）**

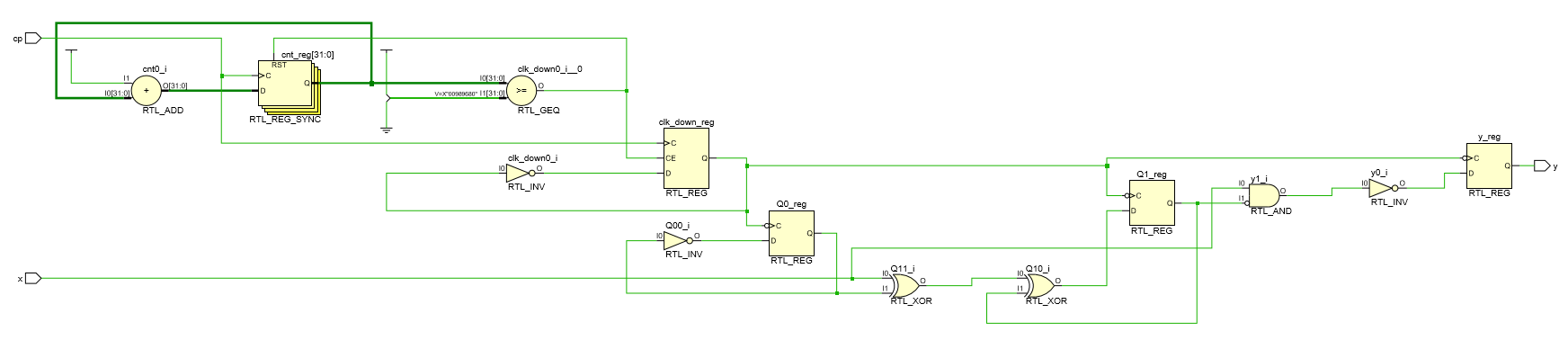


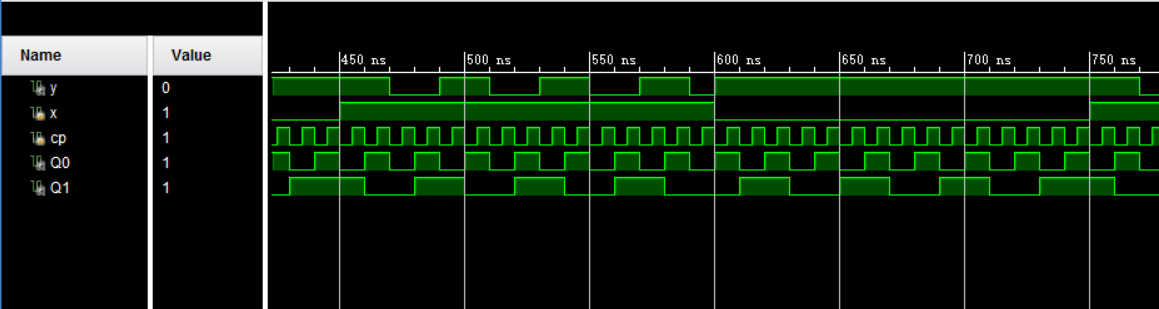
图3-6 同步时序逻辑电路

**（b）仿真**

*目的：验证电路功能的正确性*

*输入：共2个，分别为：cp（时钟）, x。*

*输出：共1个，为y。*



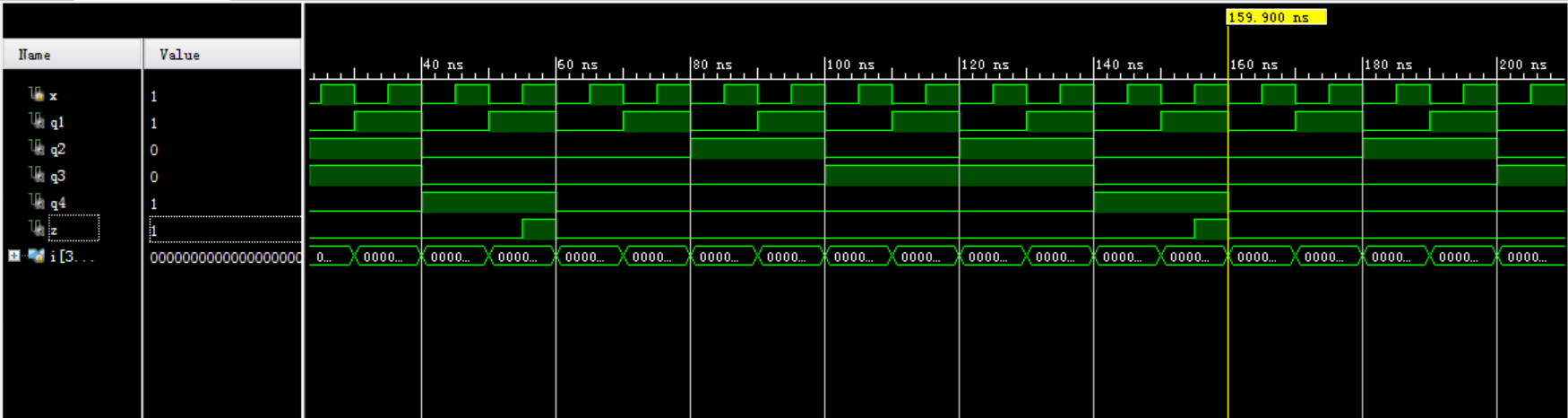
**当输入x为0时y的输入一直为1，当x为1时**

2、脉冲异步计数器的结果记录

**（1）该计数器的模是多少**

该计数器的模为10。

**（2）仿真**



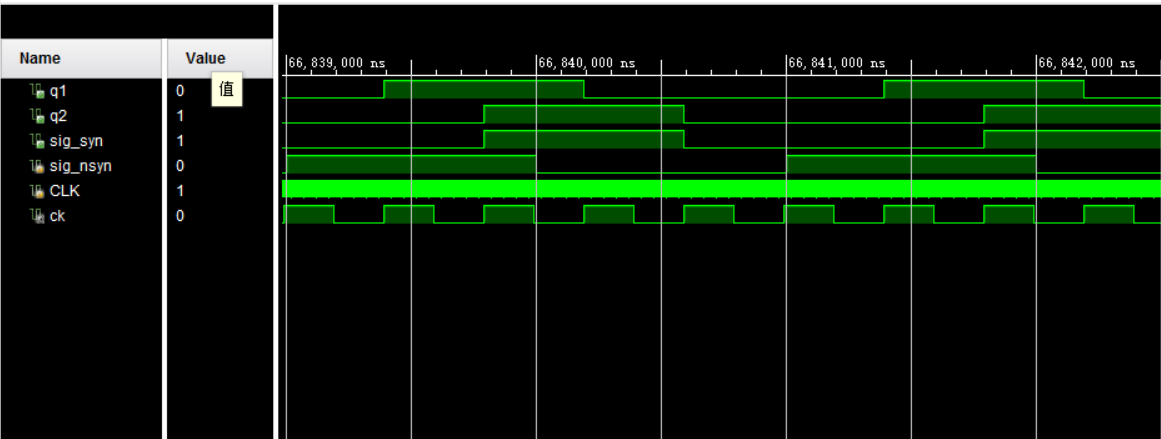
从60ns开始计数，每个上升沿记一次数，其中q1、q2、q3、q4为电路的状态，在第155ns时尾第十个上升沿，此时输出为1。第160ns回到0重新开始计数。

**（3）开发板上的验证情况**

对于程序直接进行测试，拨动程序输入所绑定的开关，每拨动一次，观察q1、q2、q3、q4所绑定的LED灯的变化情况，并与仿真中的变化情况作出比对。结果发现与仿真中的变化情况一致，状态总是在输入的下降沿更改，并以0000 – 0001 – 0010 – 0011 – 0100 – 0101 – 0110 – 0111 – 1000 – 0000 的状态循坏，每当计数达到10时输出为1，与预期结果一致。

3、Verilog设计中一致性问题的结果记录

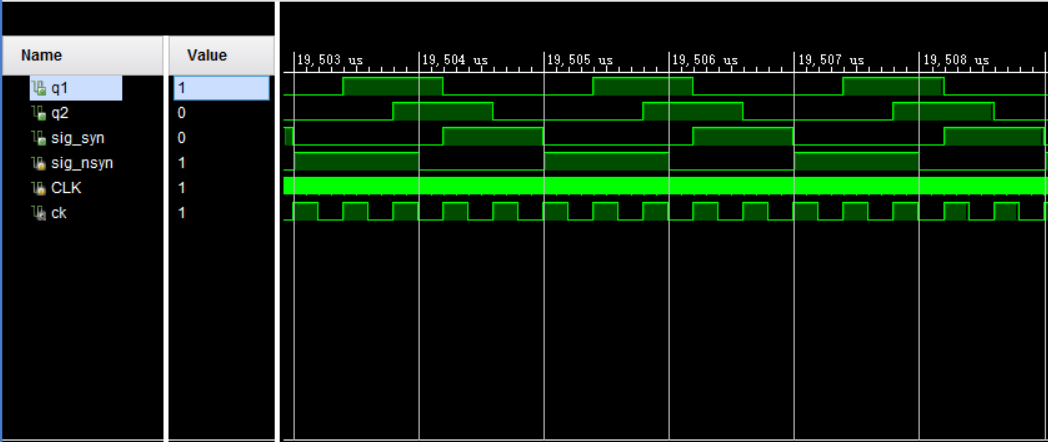
**（1）对程序3-2的仿真**



**如果仿真不正确，问题可以出在哪些方面？**

问题在于阻塞性赋值所导致的触发器状态的更新出现延迟。

**（2）程序3-2更正后的仿真**



**仿真结果正确吗？**

正确，更改为非阻塞性赋值后仿真结果变正确。

七、实验结果分析

对于实验一而言，其仿真的波形结果与理论推导的一致。而对于实验二而言，不仅仿真结果验证了其为十进制计数器，在开发板上验证的结果也与仿真结果一致。但是在开发板上验证时出现过由于按键抖动而引起的计数错误问题，这也使我们明白了现实世界与理论结果的区别。

对于实验三，由于对于Verilog语言不甚了解，导致未能在开发板上进行验证，但是前仿真和后方针的结果差异说明也说明了在设计电路时应该针对电路的功能和结构进行对应的设计，尤其是在赋值的顺序以及阻塞性和非阻塞性赋值的差异方面。

八、思考题

**1．****简述用“always块”设计纯组合逻辑电路的注意事项。**

在使用always块设计组合逻辑电路时，所有的参与赋值的信号都必须在always@中列出，always 中的if语句的判断表达式必须在敏感电平列表中列出，否将会产生锁存器。

**2. RTL Analysis下的“Schematic”与Synthesized Design下的“Schematic”的区别。**

RTL Analysis 下的Schemetic是寄存器传输图而不是电路的最终结果，主要是用于辅助设计者检验输入中存在的问题。而Synthesized Design中的Schemetic则是最终反映在FPGA中的效果，体现了实际的FPGA布线状况。

**3. 如果要保持“前仿真和后仿真以及下载验证都正确”，在Verilog设计电路时要注意哪些方面？**

在使用always块时可能会产生不必要的锁存器，导致险象的出现。因此在对于敏感电平及其他条件的描述应该完整，

九、心得体会、意见与建议

在本次实验中我收获颇丰。首先，是对于FPGA以及硬件编程语言有了初步的了解，并了解到使用这些编程语言对于大规模集成电路而言十分便捷。其二，我亲身实践了Verilog编程语言的使用，虽然对于这一门语言了解的不是十分深入，但在今后的学习过程中我必将不断的加深对其的了解，并将之应用于实际的学习生活之中。