

**数字逻辑实验报告（4）**

**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** |  |  |  |
| **检查时间** |  |  |  |
| **检查老师** |  |  |  |

**报 告 人： 胡思勖**

**实验指导教师：**

**报告批阅教师：**

**计算机科学与技术学院**

**2017年6月1日**

一、实验名称

Verilog HDL设计较复杂数字逻辑电路。

二、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“电路实例化”、“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。

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、4位二进制加法/减法计数器的设计（必选）

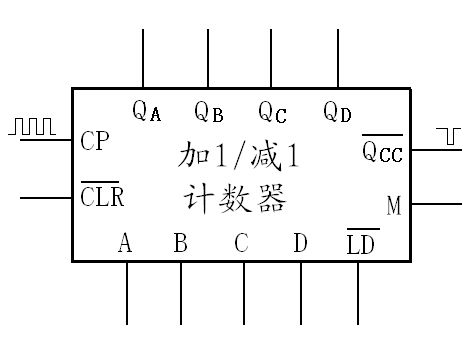
设计一个能清零、置数和进位/借位输出的加1/减1的4位二进制计数器，其结构框图如图4-1所示。

图4-1 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：

（1）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。

（2）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：**

（1）用Verilog HDL实现该计数器，将之下载到开发板中进行验证；

（2）用已实现了的“4位二进制计数器”，采用“电路实例化”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；

（3）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（必选）

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：

（1）给出不可重叠的“10110”二进制序列的状态转移图；

（2）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；

（3）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（可选）

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：

（1）用Verilog HDL设计一个一位二进制数值比较器；

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

（3）将所设计的电路下载到开发板上进行验证；

（4）给出设计占用FPGA芯片的资源情况（希望越少越好）。

五、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

module counter\_4bit(

input CP, A, B, C, D, CLR, LD, M,

output QA, QB, QC, QD,

output reg Qcc);

reg [3:0] counter = 4'b0000;

assign QA = counter[0];

assign QB = counter[1];

assign QC = counter[2];

assign QD = counter[3];

always @(posedge CP, negedge LD, negedge CLR) begin

if ( !CLR ) begin

counter <= 4'b0000;

Qcc <= 1;

end else if ( !LD ) begin

counter[0] <= A; counter[1] <= B;

counter[2] <= C; counter[3] <= D;

Qcc <= 1;

end else begin

// change counter

if (M) begin

counter <= counter + 1;

end else begin

counter <= counter - 1;

end

// change carry/borow flag

if (M && counter == 4'b1111) begin

Qcc <= 0;

end else if (!M && counter == 4'b0000) begin

Qcc <= 0;

end else begin

Qcc <= 1;

end

end

end

endmodule

（b）仿真程序

`timescale 1ns/100ps

module counter\_4bit\_test;

reg CP=0, A=0, B=0, C=0, D=0, LD=1, CLR=1, M=1;

wire Qa, Qb, Qc, Qd, Qc

counter\_4bit testCounter( CP, A, B, C, D, CLR, LD, M,

Qa, Qb, Qc, Qd, Qcc);

initial fork

#100 M <= 0;

#250 M <= 1;

#430 CLR <= 0;

#435 CLR <= 1;

#550 LD <= 0;

#446 A <= 1;

#446 B <= 1;

#446 C <= 0;

#446 D <= 0;

#555 LD <= 1;

join

always #10 CP <= ~CP;

endmodule

（c）引脚约束（绑定）程序

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 R17 IOSTANDARD LVCMOS33 } [get\_ports { CP }]; #IO\_L12N\_T1\_MRCC\_14 Sch=sw[4]

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

set\_property -dict { PACKAGE\_PIN U18 IOSTANDARD LVCMOS33 } [get\_ports { CLR }]; #IO\_L17N\_T2\_A13\_D29\_14 Sch=sw[6]

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

## LEDs

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

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

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

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

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

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

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

（a）源程序

module counter\_mod8(

input CP, A, B, C, CLR, LD, M,

output QA, QB, QC, Qcc);

// subcounter input

wire [2:0]input1;

wire subLD;

wire high;

// subcounter output

wire [3:0]output1;

wire \_Qcc1;

// input assignment

assign input1 ={CLR & C, (~CLR) | B, CLR & A};

assign subLD = CLR & LD;

counter\_4bit subCounter1(.CP(CP), .A(input1[0]), .B(input1[1]), .C(input1[2]), .D(1'b0), .CLR(1'b1), .LD(subLD), .M(M), .QA(output1[0]), .QB(output1[1]), .QC(output1[2]), .QD(output1[3]), .Qcc(\_Qcc1));

// output assignment

assign Qcc = (~subLD) | ( \_Qcc1 & (~(output1 == 4'b1000 && M)) & (~(output1 == 4'b0111 && !M)) );

assign {QC, QB, QA} = output1[2:0];

endmodule

（b）仿真程序

module counter\_mod8\_test;

reg CP=0, A=0, B=0, C=0, LD=1, CLR=1, M=1;

wire Qa, Qb, Qc, Qcc;

counter\_mod8 testCounter( CP, A, B, C, CLR, LD, M,

Qa, Qb, Qc, Qcc);

initial begin

#300 M <= 0;

#600 M <= 1;

#700 CLR <= 0;

#800 CLR <= 0;

#900 CLR <= 1;

end

always #10 CP <= ~CP;

endmodule

（c）引脚约束（绑定）程序

set\_property -dict {PACKAGE\_PIN J15 IOSTANDARD LVCMOS33} [get\_ports A];

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

set\_property -dict {PACKAGE\_PIN M13 IOSTANDARD LVCMOS33} [get\_ports C];

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

set\_property -dict {PACKAGE\_PIN R17 IOSTANDARD LVCMOS33} [get\_ports CP];

set\_property -dict {PACKAGE\_PIN T18 IOSTANDARD LVCMOS33} [get\_ports LD];

set\_property -dict {PACKAGE\_PIN U18 IOSTANDARD LVCMOS33} [get\_ports CLR];

set\_property -dict {PACKAGE\_PIN R13 IOSTANDARD LVCMOS33} [get\_ports M];

#set\_property CLOCK\_DEDICATED\_ROUTE FALSE [get\_nets {CLR\_IBUF}];

#et\_property CLOCK\_DEDICATED\_ROUTE FALSE [get\_nets {LD\_IBUF}];

## LEDs

set\_property -dict {PACKAGE\_PIN H17 IOSTANDARD LVCMOS33} [get\_ports QA];

set\_property -dict {PACKAGE\_PIN K15 IOSTANDARD LVCMOS33} [get\_ports QB];

set\_property -dict {PACKAGE\_PIN J13 IOSTANDARD LVCMOS33} [get\_ports QC];

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

set\_property -dict {PACKAGE\_PIN R18 IOSTANDARD LVCMOS33} [get\_ports Qcc];

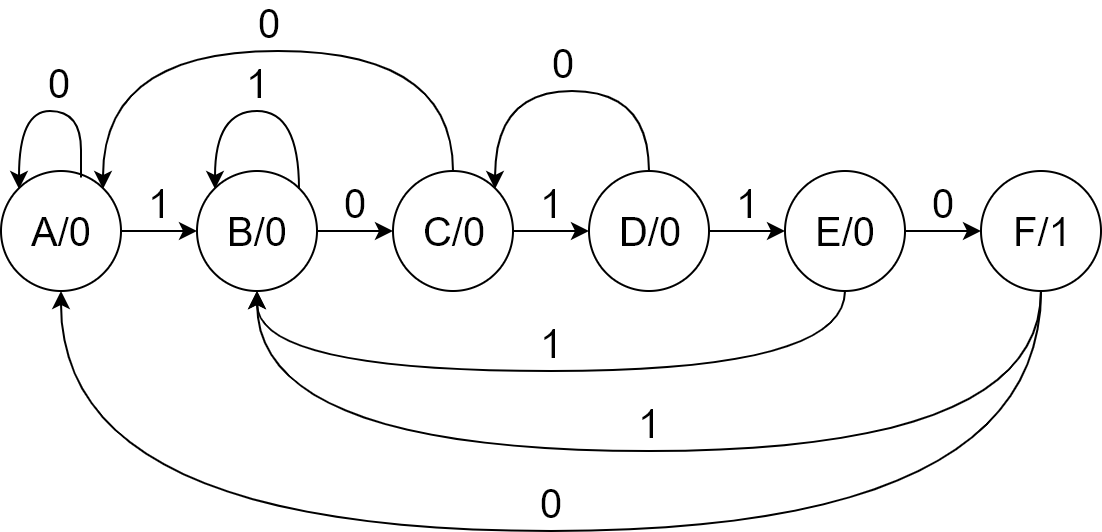
#set\_property CLOCK\_DEDICATED\_ROUTE FALSE [get\_nets LD\_IBUF];

set\_property CONFIG\_VOLTAGE 3.3 [current\_design]

set\_property CFGBVS VCCO [current\_design];

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



（B）源程序

module sequence\_checker(

input X, CP,

output reg FLAG);

reg [2:0] status = 3'b000;

always @(posedge CP) begin

case (status)

3'b000: begin //A

if (X == 1) begin

status <= status + 1;

end else begin

status <= 3'b000;

end

end

3'b001: begin //B

if (X == 0) begin

status <= status + 1;

end

end

3'b010: begin //C

if (X == 1) begin

status <= status + 1;

end else begin

status <= 3'b000;

end

end

3'b011: begin //D

if (X == 1) begin

status <= status + 1;

end else begin

status <= 3'b010;

end

end

3'b100: begin //E

if (X == 1) begin

status = 3'b001;

end else begin

status = status + 1;

end

end

3'b101: begin //F

if (X == 1) begin

status = 3'b001;

end else begin

status = 3'b000;

end

end

endcase

if ( status == 3'b101 ) begin

FLAG <= 1;

end else begin

FLAG <= 0;

end

end

endmodule

（C）仿真程序

`timescale 1ns / 100ps

module sequence\_checker\_test;

reg CP = 0;

reg in = 1;

wire out;

sequence\_checker testSequenceChecker(in, CP, out);

initial fork

#0 in <= 1;

#12 in <= 0;

#22 in <= 1;

#32 in <= 1;

#42 in <= 0;

join

always #5 CP <= ~CP;

endmodule

（D）引脚约束（绑定）程序

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

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

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

## LEDs

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

module comparator\_1bit (

input A, B,

output F1, F2, F3

);

assign F1 = A > B;

assign F2 = A == B;

assign F3 = A < B;

endmodule // comparator\_1bit

（b）仿真程序

module compartor\_1bit\_test;

reg A = 1'b0;

reg B = 1'b0;

wire [2:0] out;

comparator\_1bit testComparator(A, B, out[0], out[1], out[2]);

initial fork

#10 A <= 1'b1;

#20 A <= 1’b0

#20 B <= 1'b1;

#30 A <= 1’b0

#30 B <= 1'b0;

join

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 U12 IOSTANDARD LVCMOS33 } [get\_ports { A }]; #IO\_L20P\_T3\_A08\_D24\_14 Sch=sw[13]

set\_property -dict { PACKAGE\_PIN U11 IOSTANDARD LVCMOS33 }

## LEDs

set\_property -dict { PACKAGE\_PIN H17 IOSTANDARD LVCMOS33 } [get\_ports { F1 }]; #IO\_L18P\_T2\_A24\_15 Sch=led[0]

set\_property -dict { PACKAGE\_PIN K15 IOSTANDARD LVCMOS33 } [get\_ports { F2 }]; #IO\_L24P\_T3\_RS1\_15 Sch=led[1]

set\_property -dict { PACKAGE\_PIN J13 IOSTANDARD LVCMOS33 } [get\_ports { F3 }]; #IO\_L17N\_T2\_A25\_15 Sch=led[2]

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

（a）源程序

module comparator\_3bit (

input [2:0]a, [2:0]b,

output F1, F2, F3

);

wire [2:0] \_ABOVE; // A[x] > B[x]

wire [2:0] \_EQUAL; // A[x] = B[x]

wire [2:0] \_BELOW; // A[x] < B[x]

assign F1 = \_ABOVE[2] ||

(\_EQUAL[2] && \_ABOVE[1]) ||

(\_EQUAL[2] && \_EQUAL[1] && \_ABOVE[0]);

assign F2 = \_EQUAL[0] && \_EQUAL[1] && \_EQUAL[2]; // A = B

assign F3 = \_BELOW[2] ||

(\_EQUAL[2] && \_BELOW[1]) ||

(\_EQUAL[2] && \_EQUAL[1] && \_BELOW[0]);

comparator\_1bit comparator0(a[0], b[0], \_ABOVE[0], \_EQUAL[0], \_BELOW[0]);

comparator\_1bit comparator1(a[1], b[1], \_ABOVE[1], \_EQUAL[1], \_BELOW[1]);

comparator\_1bit comparator2(a[2], b[2], \_ABOVE[2], \_EQUAL[2], \_BELOW[2]);

endmodule // comparator\_3bit

（b）仿真程序

module compartor\_3bit\_test;

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

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

wire [2:0] out;

comparator\_3bit testComparator(A, B, out[0], out[1], out[2]);

initial fork

#10 A <= 3'd0;

#20 A <= 3'd1;

#30 A <= 3'd2;

#40 A <= 3'd3;

#50 B <= 3'd4;

#60 B <= 3'd5;

#70 B <= 3'd6;

#80 B <= 3'd7;

#90 A <= 3'd5;

#90 B <= 3'd5;

join

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 U12 IOSTANDARD LVCMOS33 } [get\_ports { a[0] }]; #IO\_L20P\_T3\_A08\_D24\_14 Sch=sw[13]

set\_property -dict { PACKAGE\_PIN U11 IOSTANDARD LVCMOS33 } [get\_ports { a[1] }]; #IO\_L19N\_T3\_A09\_D25\_VREF\_14 Sch=sw[14]

set\_property -dict { PACKAGE\_PIN V10 IOSTANDARD LVCMOS33 } [get\_ports { a[2] }]; #IO\_L21P\_T3\_DQS\_14 Sch=sw[15]

## LEDs

set\_property -dict { PACKAGE\_PIN H17 IOSTANDARD LVCMOS33 } [get\_ports { F1 }]; #IO\_L18P\_T2\_A24\_15 Sch=led[0]

set\_property -dict { PACKAGE\_PIN K15 IOSTANDARD LVCMOS33 } [get\_ports { F2 }]; #IO\_L24P\_T3\_RS1\_15 Sch=led[1]

set\_property -dict { PACKAGE\_PIN J13 IOSTANDARD LVCMOS33 } [get\_ports { F3 }]; #IO\_L17N\_T2\_A25\_15 Sch=led[2]

六、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

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

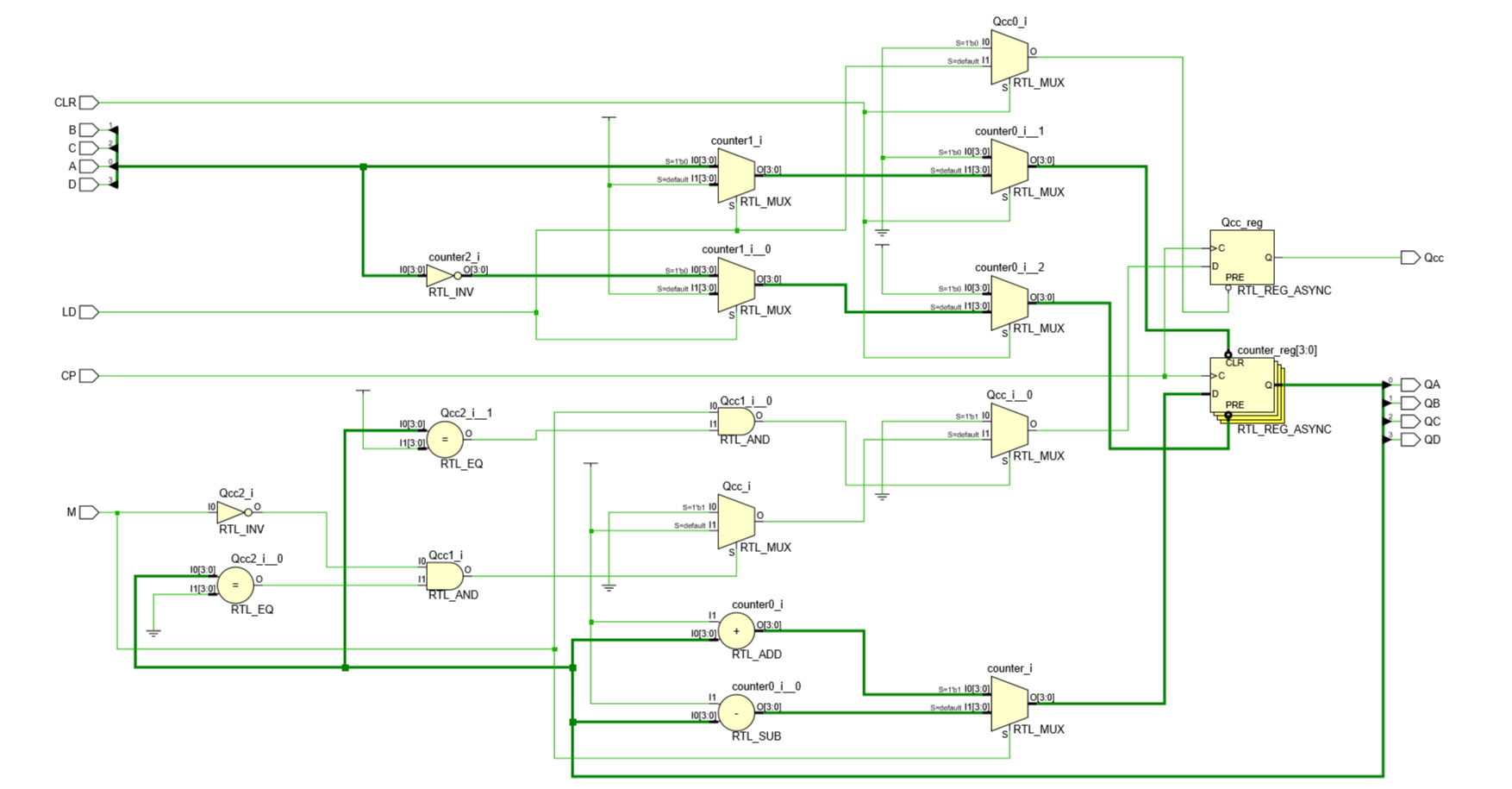
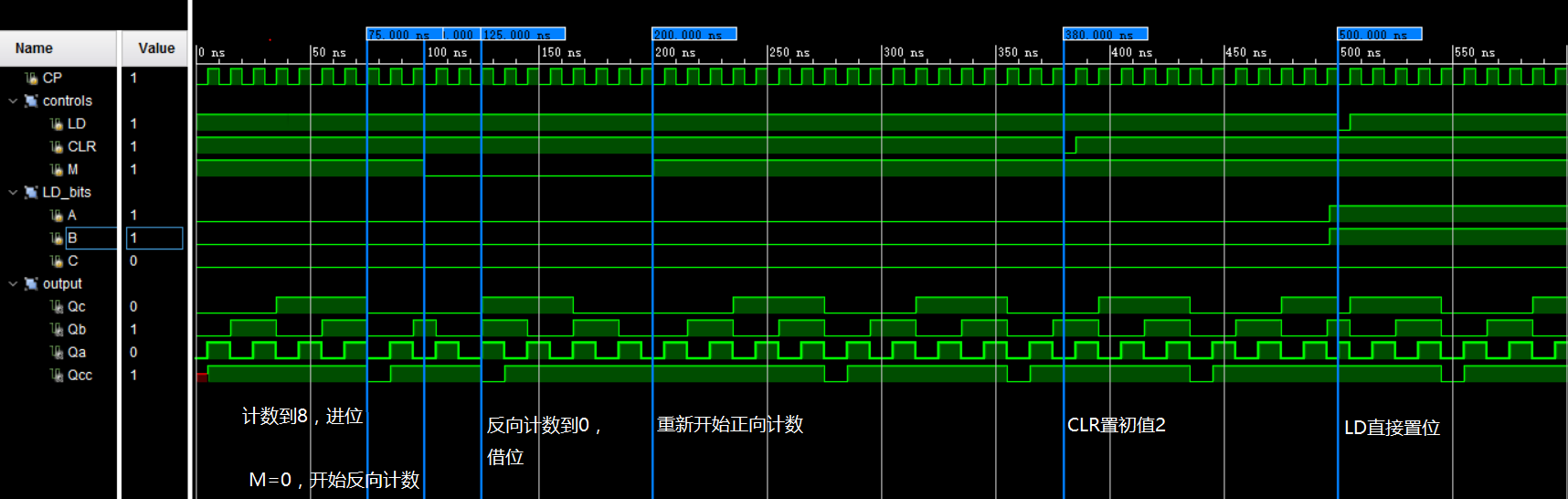


图4-2 4位二进制加法/减法计数器

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

（仿真结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4-3 **初值为2的模8计数器仿真**

**（3）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

**（1）给出Verilog设计的时序逻辑电路图（**RTL Analysis下的“Schematic”**）**

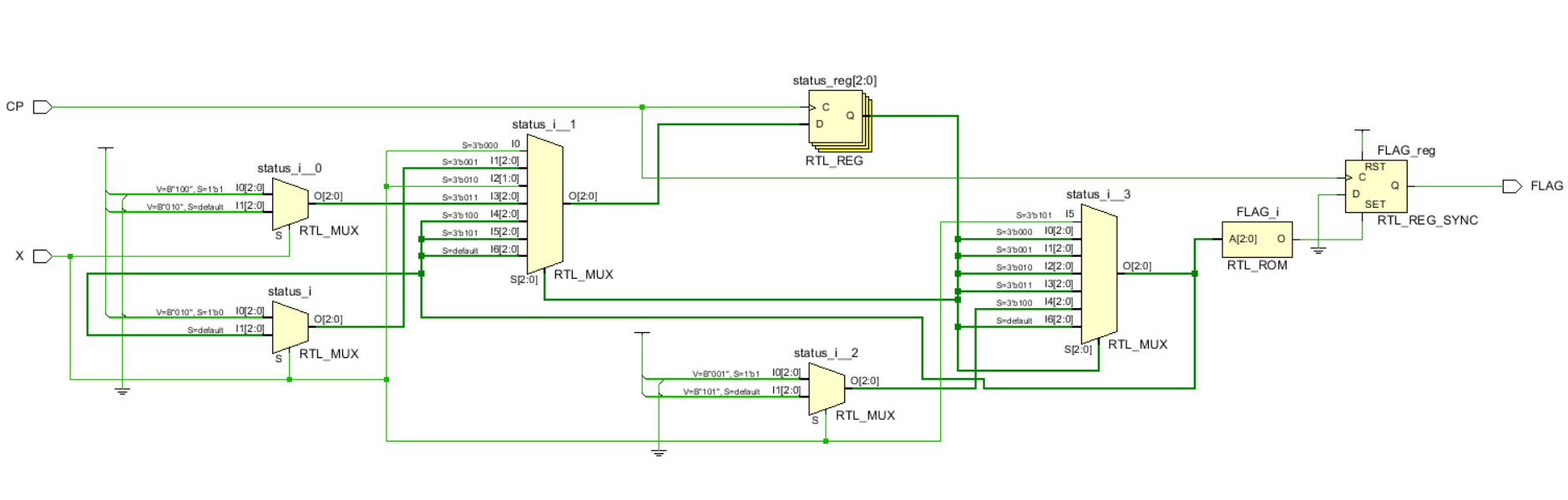
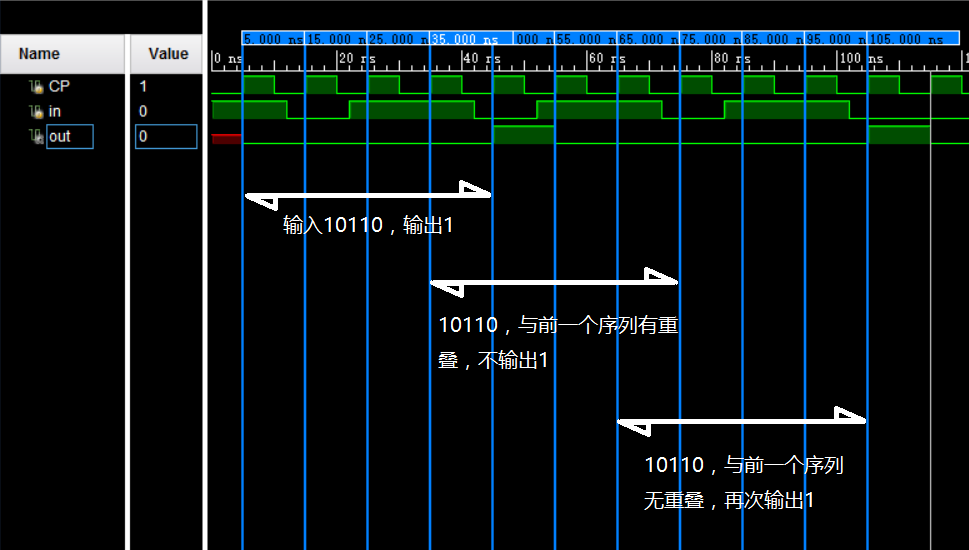


图4-4 10110序列检测器

**（2）仿真结果（波形含clk，输入值，输出值等）**

图4-5 10110序列检测器仿真

**（3）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

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

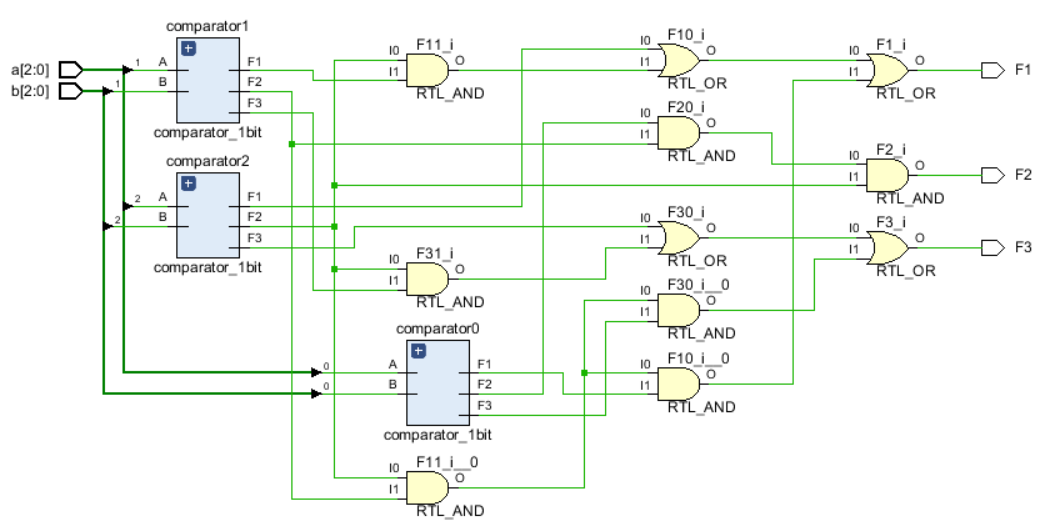
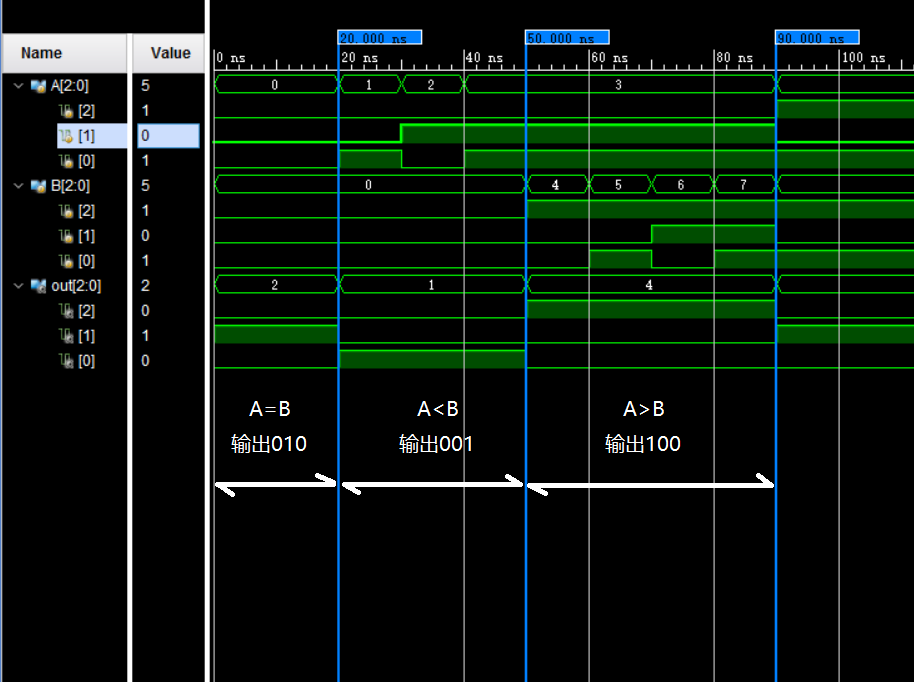


图4-6 3位二进制数值比较器

**（2）仿真结果（含输入值，输出值等）**

图4-7 3位二进制数值比较器仿真

**（3）开发板上的验证情况（主要记录：验证过程和结论）**

七、实验结果分析

八、思考题

1．简述一下“异步时序逻辑电路的同步化处理”的优缺点。

2．结构级描述和行为级描述各有什么特点？

九、心得体会、意见与建议