**中国科学技术大学计算机学院**

**《数字电路实验》报告**



实验题目：信号处理及有限状态机

学生姓名：\_\_\_\_ 黄瑞轩\_ \_\_\_\_

学生学号：\_\_ PB20111686 \_ \_\_

完成日期：\_\_ 2021.12.08 \_\_

计算机实验教学中心制

2020年09月

**实验题目**

信号处理及有限状态机

**实验目的**

* 进一步熟悉 FPGA 开发的整体流程
* 掌握几种常见的信号处理技巧
* 掌握有限状态机的设计方法
* 能够使用有限状态机设计功能电路

**实验环境**

* VLAB 平台： vlab.ustc.edu.cn
* FPGAOL 实验平台：fpgaol.ustc.edu.cn
* Logisim
* Vivado 工具

**实验练习**

**【题目1】**改成三段式后的代码如下。

module test(input clk, rst, output led);

parameter ZE = 2’b00;

parameter UN = 2’b01;

parameter DE = 2’b10;

parameter TR = 2’b11;

reg [1:0] CS = 2’b00; //实际电路中不需要，但是为了仿真需要给CS赋初值

reg [1:0] NS;

always@(\*) begin

case(CS)

ZE: NS = UN;

UN: NS = DE;

DE: NS = TR;

TR: NS = ZE;

endcase

end

always@(posedge clk or posedge rst) begin

if(rst)

CS <= ZE;

else

CS <= NS;

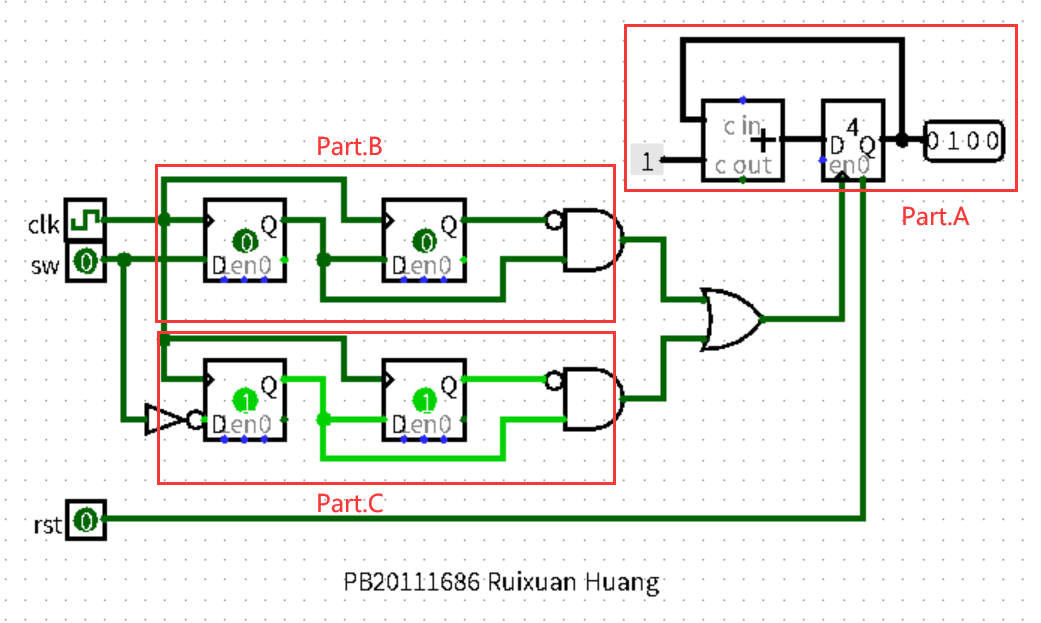
end

assign led = (CS == TR) ? 1’b1 : 1’b0;

endmodule

**【题目2】**

* 设计的电路图如下。



* 下面解释电路设计的思路：Part.A部分是一个寄存器，Part.B部分是捕获一个sw信号的上升沿，Part.C部分是捕获一个sw信号的下降沿，用或门获得sw信号电平的变动（p变n＋n变p），并以一个时钟周期上升沿的形式输出给寄存器。

**【题目3】**

* 十六进制计数器在前面的实验做过类似的，这里不再详细解释十六进制计数器的详细设计，只着重谈题目要求的部分是如何实现的。下面是设计文件，解释部分以注释的形式给出。

module test(

input CLK100MHZ,

input dir,

input rst,

input btn,

output reg [2:0] hexplay\_an,

output reg [3:0] hexplay\_data

);

reg [7:0] data;

reg [32:0] hexplay\_cnt;

always@(posedge CLK100MHZ) begin

if (hexplay\_cnt >= (2000000 / 8)) hexplay\_cnt <= 0;

else hexplay\_cnt <= hexplay\_cnt + 1;

end

always@(posedge CLK100MHZ) begin

if (hexplay\_cnt == 0) begin

if (hexplay\_an == 1) hexplay\_an <= 0;

else hexplay\_an <= hexplay\_an + 1;

end

end

always@(\*) begin

case(hexplay\_an)

0: hexplay\_data = data[3:0];

1: hexplay\_data = data[7:4];

endcase

end

reg [26:0] timer\_cnt;

always@(posedge CLK100MHZ) begin

if (timer\_cnt >= 100000000) timer\_cnt <= 0;

else timer\_cnt <= timer\_cnt + 1;

end

//给btn信号去毛刺

reg [3:0] cnt;

always@(posedge CLK100MHZ) begin

if (btn == 1'b0) cnt <= 4'h0;

else if (cnt < 4'h8) cnt <= cnt + 1;

end

wire btn\_c;

assign btn\_c = cnt[3];

//获得btn\_c信号的边沿

reg btn\_r1, btn\_r2;

wire btn\_edge;

always @(posedge CLK100MHZ) btn\_r1 <= btn\_c;

always @(posedge CLK100MHZ) btn\_r2 <= btn\_r1;

assign btn\_edge = btn\_r1 & (~btn\_r2);

//状态转移

always@(posedge CLK100MHZ) begin

if(rst) data <= 8'h1F;

else if (btn\_edge) begin

if (dir) begin

data <= data - 1;

end

else begin

data <= data + 1;

end

end

end

endmodule

* 约束文件test.xdc如下：

set\_property -dict { PACKAGE\_PIN E3 IOSTANDARD LVCMOS33 } [get\_ports { CLK100MHZ }];

create\_clock -add -name sys\_clk\_pin -period 10.00 -waveform {0 5} [get\_ports { CLK100MHZ }];

set\_property -dict { PACKAGE\_PIN D14 IOSTANDARD LVCMOS33 } [get\_ports { dir }];

set\_property -dict { PACKAGE\_PIN F16 IOSTANDARD LVCMOS33 } [get\_ports { rst }];

set\_property -dict { PACKAGE\_PIN B18 IOSTANDARD LVCMOS33 } [get\_ports { btn }];

set\_property -dict { PACKAGE\_PIN A14 IOSTANDARD LVCMOS33 } [get\_ports { hexplay\_data[0] }];

set\_property -dict { PACKAGE\_PIN A13 IOSTANDARD LVCMOS33 } [get\_ports { hexplay\_data[1] }];

set\_property -dict { PACKAGE\_PIN A16 IOSTANDARD LVCMOS33 } [get\_ports { hexplay\_data[2] }];

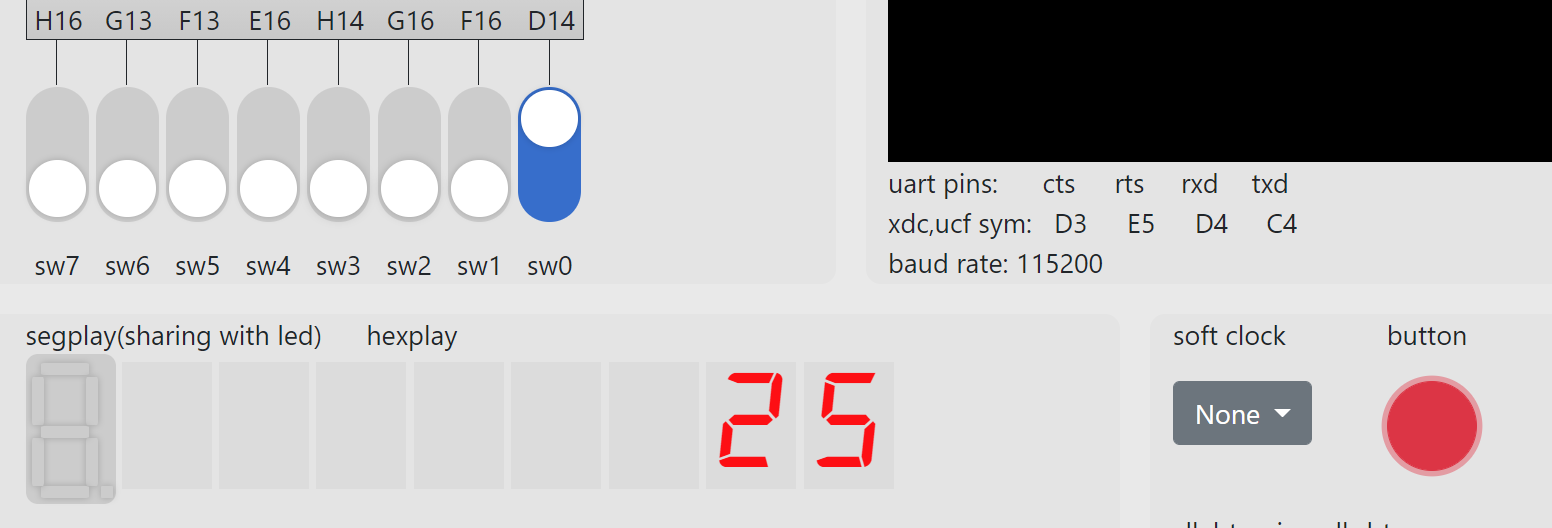
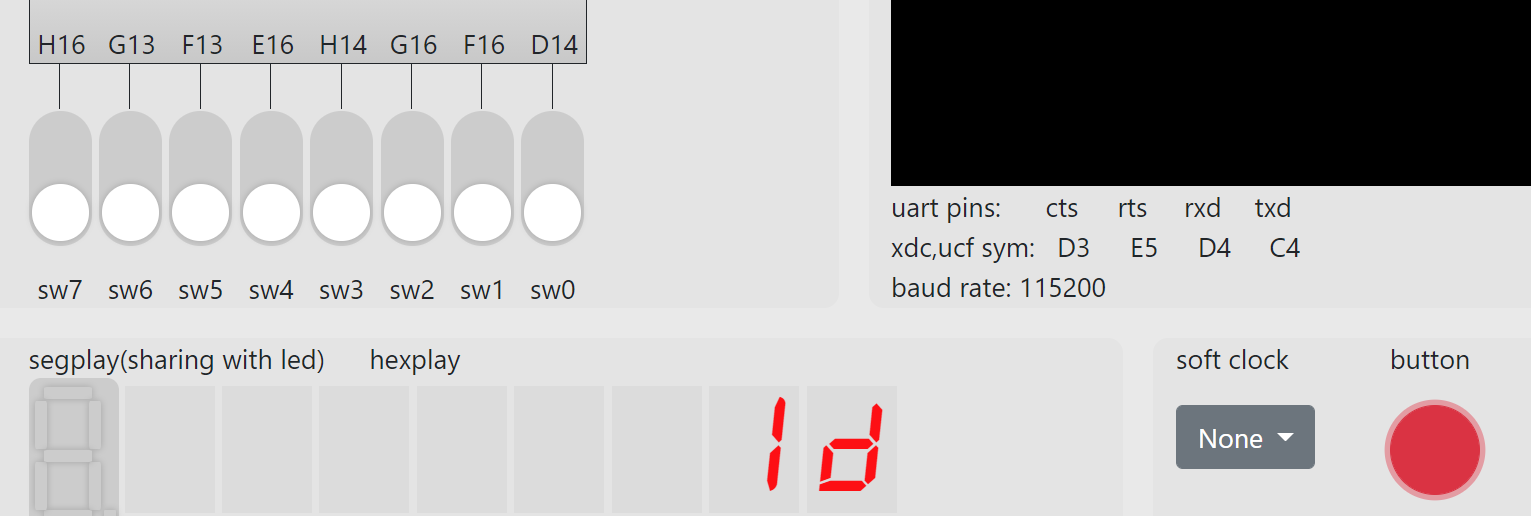
set\_property -dict { PACKAGE\_PIN A15 IOSTANDARD LVCMOS33 } [get\_ports { hexplay\_data[3] }];

set\_property -dict { PACKAGE\_PIN B17 IOSTANDARD LVCMOS33 } [get\_ports { hexplay\_an[0] }];

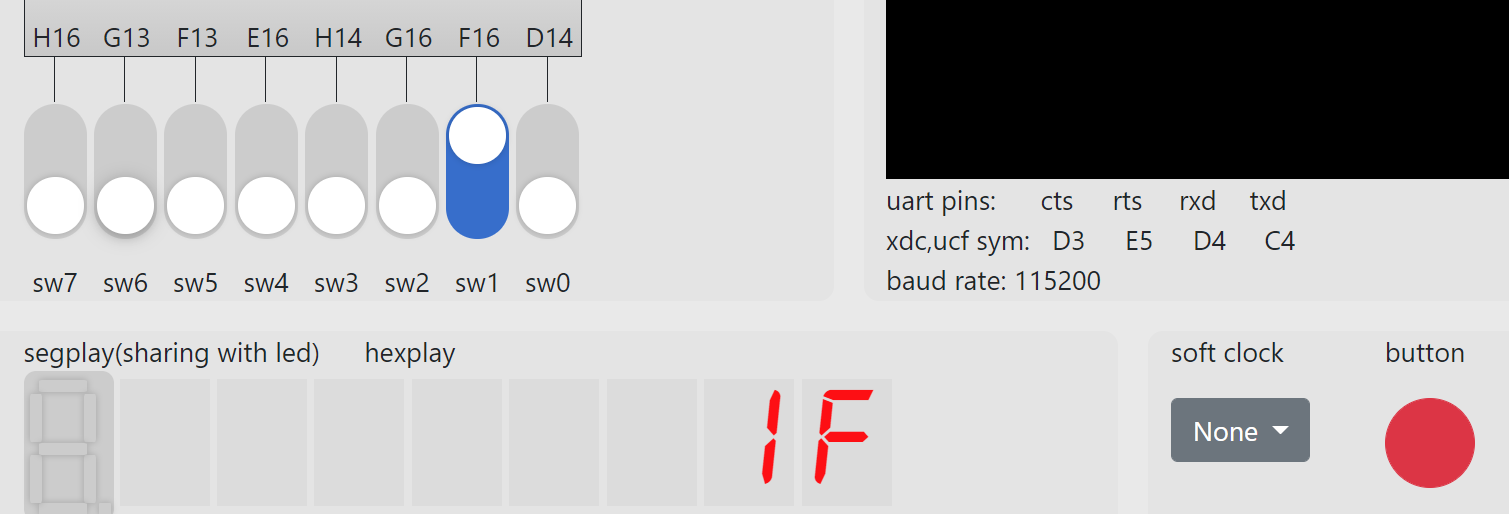
set\_property -dict { PACKAGE\_PIN B16 IOSTANDARD LVCMOS33 } [get\_ports { hexplay\_an[1] }];

set\_property -dict { PACKAGE\_PIN A18 IOSTANDARD LVCMOS33 } [get\_ports { hexplay\_an[2] }];

* 按实验指导手册上的实验过程生成.bit文件，上传到fpgaol平台烧写，结果如下所示。

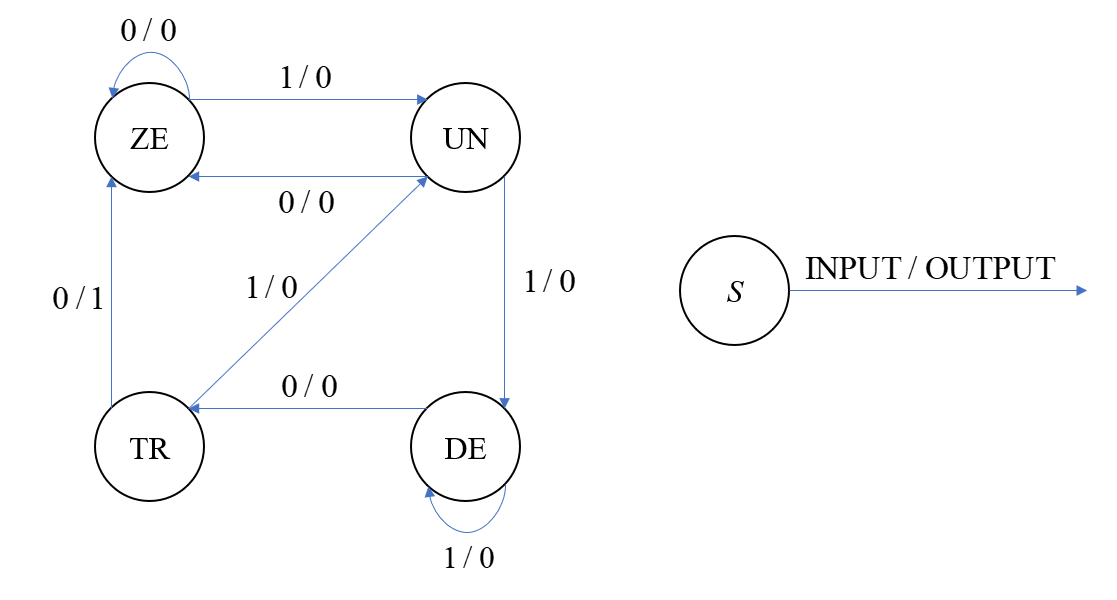
累加模式点击按钮0x25下 换到递减模式再点击按钮0x8下



复位

**【题目4】**

* 先画出状态图，如下图所示。



* 根据状态图设计电路，设计文件如下，思路解释部分以注释的形式给出。

module test(

input CLK100MHZ,

input dir,

input btn,

output reg [2:0] hexplay\_an,

output reg [3:0] hexplay\_data

);

parameter ZE = 4’h0;

parameter UN = 4’h1;

parameter DE = 4’h2;

parameter TR = 4’h3;

reg [3:0] CS = 4’h0;

reg [3:0] NS;

reg in = 0;

reg [3:0] num = 4’h0;

reg [15:0] history = 16’h0000;

//下面是给btn信号去毛刺

reg [3:0] cnt;

always@(posedge CLK100MHZ) begin

if(btn == 1’b0) cnt <= 4’h0;

else if(cnt < 4’h8) cnt <= cnt+1;

end

wire btn\_c;

assign btn\_c = cnt[3];

//下面是获得btn按下的边沿

reg btn\_r1,btn\_r2;

wire btn\_edge;

always@(posedge CLK100MHZ) begin

btn\_r1 <= btn\_c;

btn\_r2 <= btn\_r1;

end

assign btn\_edge = btn\_r1 & (~btn\_r2);

//下面是状态转移部分

always@(posedge CLK100MHZ) begin

if(btn\_edge) begin

history <= {history[11:0], 3’b0, dir};

if(dir == 0) begin

case(CS)

ZE: NS <= ZE;

UN: NS <= ZE;

DE: NS <= TR;

TR: begin

NS <= ZE;

num = num + 1;

end

endcase

end

else begin

case(CS)

ZE: NS <= UN;

UN: NS <= DE;

DE: NS <= DE;

TR: NS <= UN;

endcase

end

end

else NS <= NS;

end

//下面是状态赋值部分

always@(posedge CLK100MHZ) begin

CS <= NS;

end

reg [4:0] hexplay\_cnt;

always@(posedge CLK100MHZ) begin

if (hexplay\_cnt == 5’b10111)

hexplay\_cnt <= 0;

else

hexplay\_cnt <= hexplay\_cnt + 1;

end

always@(posedge CLK100MHZ) begin

hexplay\_an = hexplay\_cnt[4:2];

case(hexplay\_an)

3’b000: hexplay\_data = CS;

3’b001: hexplay\_data = num;

3’b010: hexplay\_data = history[3:0];

3’b011: hexplay\_data = history[7:4];

3’b100: hexplay\_data = history[11:8];

3’b101: hexplay\_data = history[15:12];

default: hexplay\_data = ZE;

endcase

end

endmodule

* 约束文件test.xdc如下：

set\_property -dict { PACKAGE\_PIN E3 IOSTANDARD LVCMOS33 } [get\_ports { CLK100MHZ }];

create\_clock -add -name sys\_clk\_pin -period 10.00 -waveform {0 5} [get\_ports { CLK100MHZ }];

set\_property -dict { PACKAGE\_PIN D14 IOSTANDARD LVCMOS33 } [get\_ports { dir }];

set\_property -dict { PACKAGE\_PIN B18 IOSTANDARD LVCMOS33 } [get\_ports { btn }];

set\_property -dict { PACKAGE\_PIN A14 IOSTANDARD LVCMOS33 } [get\_ports { hexplay\_data[0] }];

set\_property -dict { PACKAGE\_PIN A13 IOSTANDARD LVCMOS33 } [get\_ports { hexplay\_data[1] }];

set\_property -dict { PACKAGE\_PIN A16 IOSTANDARD LVCMOS33 } [get\_ports { hexplay\_data[2] }];

set\_property -dict { PACKAGE\_PIN A15 IOSTANDARD LVCMOS33 } [get\_ports { hexplay\_data[3] }];

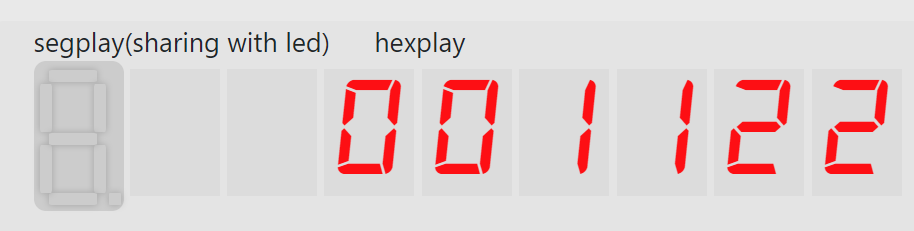
set\_property -dict { PACKAGE\_PIN B17 IOSTANDARD LVCMOS33 } [get\_ports { hexplay\_an[0] }];

set\_property -dict { PACKAGE\_PIN B16 IOSTANDARD LVCMOS33 } [get\_ports { hexplay\_an[1] }];

set\_property -dict { PACKAGE\_PIN A18 IOSTANDARD LVCMOS33 } [get\_ports { hexplay\_an[2] }];

* 按实验指导手册上的实验过程生成.bit文件，上传到fpgaol平台烧写，结果如下所示。





输入“0011001110011”后

**总结与思考**

* 本次实验中我学会了button的使用和相关的处理方法，除了sw[7:0]之外又学会了一种操控FPGA的方法，收获很大。
* 本次实验是在前面的实验基础上的综合练习，总体来说难度中等。
* 本次实验有4个小题，但是前两道小题比较简单，因此总体来说任务量不大；Vivado的Generate Bitstream每次需要消耗的时间较长，会导致多出很多调试时间。
* 改进建议：

给fpgaol平台的开关抖动情况创建一个直接可见波形的窗口，方便这门课程的同学进行直观理解和调试。