实验名称：　　姓名：　　学号：

浙江大学

本科实验报告

课程名称：　数字系统

姓　　名：　郭奇然

院　　系：　求是学院

专　　业：　电子科学与技术

学　　号：　3220105852

指导老师：　叶德信

选课时间：

2023 年　6 月　12 日

实验名称：　　姓名：　　学号：

专业：电子科学与技术
姓名：郭奇然
学号：3220105852
日期：
地点：

浙江大学实验报告

课程名称：数字系统　　指导老师：叶德信　　成绩：_____

实验名称：通用微处理器 EC-1 设计

一、实验目的和要求

学习如何通过合理设计状态机来实现一个通用微处理器

二、实验内容和原理（必填）

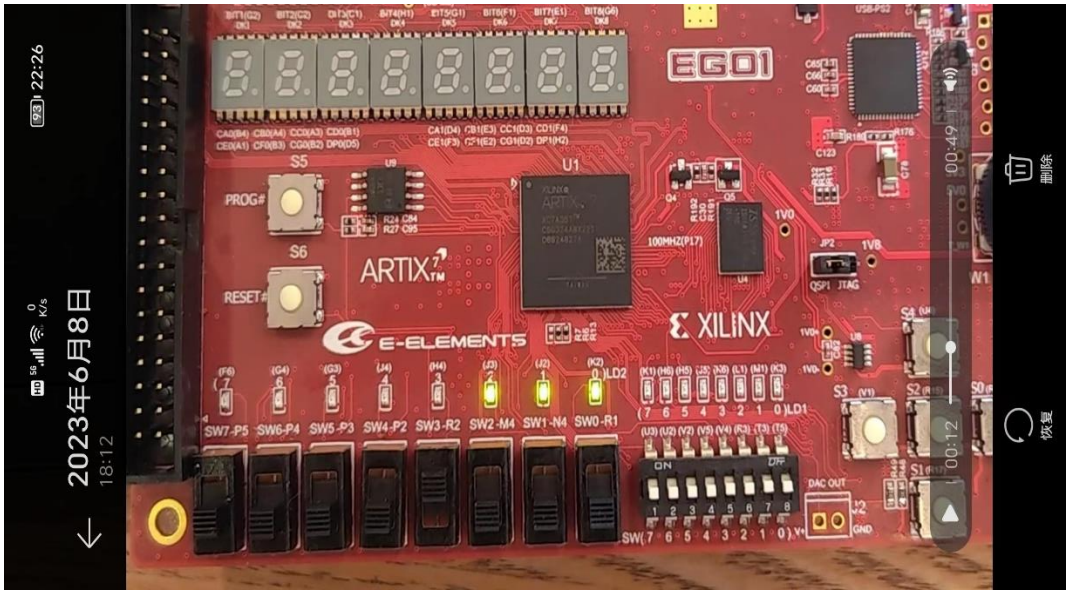
实验内容：

对于 EC-1 微处理器，在 Verilog 模块中分别实现数据路径电路和 FSM 电路，使用顶层模块将它们连接在一起，然后在 Basys3 板中实现。注意，你需要把编译好的汇编代码。即，在 ROM 中的二进制可执行代码在你的数据通路电路中运行的程序在微处理器。将板上的开关连接到输入引脚，并将输出引脚连接到 LED

三、实验结果与分析

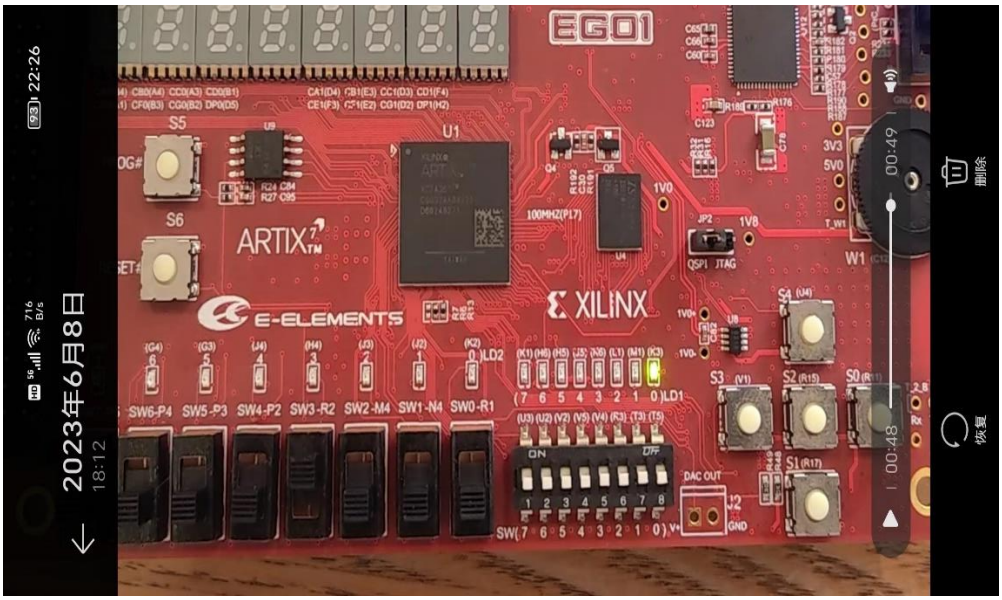
烧录结果如图

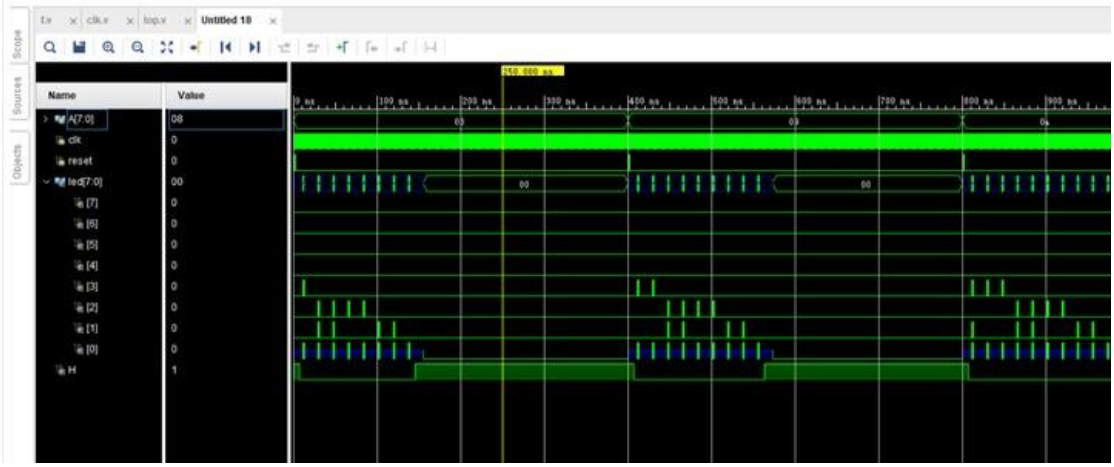
实验名称：　　姓名：　　学号：


EGO1
ARTIX
E-ELEMENTS
XILINX
PROG#
RESET#
SW7-P5 SW6-P4 SW5-P3 SW4-P2 SW3-R2 SW2-M4 SW1-N4 SW0-R1
2023年6月8日
18:12
22:26
00:49
00:12
删除
恢复


EGO1
ARTIX
E-ELEMENTS
XILINX
PROG#
RESET#
SW7-P5 SW6-P4 SW5-P3 SW4-P2 SW3-R2 SW2-M4 SW1-N4 SW0-R1
2023年6月8日
18:12
22:26
00:49
00:18
删除
恢复

实验名称：　　姓名：　　学号：





LED 灯代表的二进制数逐次减一，最后减至 0 时 K3 灯稳定亮起。

四、实验中的问题及解决方案

1.第一次烧录完成后功能不稳定，后来发现是因为 reset 按钮未消除抖动，因此加上一个 button 模块再烧录，成功稳定实现预期功能。

五、实验心得体会

这次实验我学会了如何设计通用微处理器 EC-1，加深了对使用 verilog 语言设计微处理器的理解，并且认识到消抖对实现按钮功能的重要性。

附：源代码、仿真代码

源代码：

顶层：

```
module top(
input [7:0] A,
input clk, res,
output [7:0] led,
output H
);
wire clk1,reset;
button b1(.clk(clk),.button_in(res),.button_out(reset));
clk_div div1(.clk(clk),.reset(reset),.clk1(clk1));
wire Irload, Jnzmux,Pcload,Inmux, Aload, Oute;
wire Anotzero;
wire [2:0]Ir_opword;
datapath d1(.clk(clk1),.reset(reset),.Irload(Irload),.Jnzmux(Jnzmux),.Pcload(Pcload),
.Inmux(Inmux),.Aload(Aload),.Oute(Oute),.A(A),.Ir_opword(Ir_opword),
.Anotzero(Anotzero),.led(led),.H(H));
control c1(.clk(clk1),.reset(reset),.Ir_opword(Ir_opword),.Anotzero(Anotzero),
.Irload(Irload),.Jnzmux(Jnzmux),.Pcload(Pcload),.Inmux(Inmux),.Aload(Aload),
.Oute(Oute));
endmodule
```

数据通路：

```
module datapath(
input clk, reset, Irload, Jnzmux, Pcload, Inmux, Aload, Oute,
input [7:0]A,
output reg [2:0] Ir_opword,
output Anotzero,
output [7:0] led,
output H
);
wire [3:0]Increment_o;
reg [3:0]Ir_o, Pc_o, Mux24_o;
wire [7:0]Rom_o,Decrement_o;
reg [7:0]A_o, Mux28_o;
always@(posedge clk or posedge reset) begin
if(reset)begin
Ir_opword<=3'b000;
```

```
Ir_o<=4'b0000;
end
else if(Irload)begin
Ir_opword<=Rom_o[7:5];
Ir_o<=Rom_o[3:0];
end
end
always@(posedge clk or posedge reset) begin
if(reset)Pc_o<=4'b0000;
else if(Pcload)Pc_o<=Mux24_o;
end
always@(posedge clk or posedge reset) begin
if(reset)A_o<=8'b00000000;
else if(Aload)A_o<=Mux28_o;
end
assign Increment_o = Pc_o+1;
assign Decrement_o = A_o-1;

always@(*)begin
if(Jnzmux)Mux24_o=Ir_o;
else Mux24_o=Increment_o;
end
always@(*)begin
if(Inmux)Mux28_o=A;
else Mux28_o=Decrement_o;

end
ROM_register r1(.ina(Pc_o),.out(Rom_o));
assign Anotzero = A_o!=0?1:0;
assign H = A_o == 0?1:0;
assign led =(Oute == 1'b1) ? A_o:1'bz;
endmodule
```

控制电路：

```
module control(
input clk,reset,
input [2:0] Ir_opword,
input Anotzero,
output reg Irload, Jnzmux, Pcload, Inmux, Aload, Oute
);
parameter FETCH=0;parameter DECODE=1;parameter INPUT=2;
parameter OUTPUT=3;parameter DEC=4;parameter JNZ=5;
parameter HALT=6;
reg [2:0] state, next_state;
```

```
always@(posedge clk or posedge reset) begin
if(reset)begin
state<=FETCH;
end
else
state<=next_state;
end
always @(*) begin
case (state)
FETCH: next_state = DECODE;
DECODE: if(Ir_opword == 3'b011) next_state = INPUT;
else if(Ir_opword == 3'b100) next_state = OUTPUT;
else if(Ir_opword == 3'b101) next_state = DEC;
else if(Ir_opword == 3'b110) next_state = JNZ;
else if(Ir_opword == 3'b111) next_state = HALT;
else next_state = FETCH;
HALT: next_state = HALT;
default: next_state <= FETCH;
endcase
end

always @(*)begin
case (state)
FETCH:begin Irload=1;Pcload=1;Inmux=0;Aload=0;Jnzmux=0;Oute=0;end
DECODE:begin Irload=0;Pcload=0;Inmux=0;Aload=0;Jnzmux=0;Oute=0;end
INPUT:begin Irload=0;Pcload=0;Inmux=1;Aload=1;Jnzmux=0;Oute=0;end
OUTPUT:begin Irload=0;Pcload=0;Inmux=0;Aload=0;Jnzmux=0;Oute=1;end
DEC:begin Irload=0;Pcload=0;Inmux=0;Aload=1;Jnzmux=0;Oute=0;end
JNZ:begin Irload=0;Pcload=Anotzero==1?1:0;Inmux=0;Aload=0;Jnzmux=1;Oute=0;end
HALT:begin Irload=0;Pcload=0;Inmux=0;Aload=0;Jnzmux=0;Oute=1;end
default:begin Irload=1'bz;Pcload=1'bz;Inmux=1'bz;Aload=1'bz;Jnzmux=1'bz;Oute=1'bz;end
endcase
end
endmodule
```

分频：

```
module clk_div(
input clk,
input reset,
output reg clk1
);
reg [31:0]count;
always@(posedge clk or posedge reset )begin
if(reset) begin
```

```
count <=0; clk1 <=0;
end
else if(count == 15000000) begin
count <=0; clk1 <=~clk1;
end
else count = count+1;
end
endmodule
```

仿真代码：

```
module lab7_sim(

);
reg [7:0] A;
reg clk, reset;
wire [7:0] led;
wire H;

top mm1(.A(A),.clk(clk),.res(reset),.led(led),.H(H));

initial
begin
A=8'b00001000;
clk=0;
reset=1;
end

always
begin
#400
A=A+1;
end

always
begin
#1
clk=~clk;
end

always
begin
#2
reset=0;
#398
```

```
reset=1;
end
endmodule
```