**实验报告**

姓名： 专业： 电子科学与技术 学号：

课程名称： 数字系统 任课老师： 叶德信

实验名称： Display a character string 实验日期： 2020.4.22

**1 实验目的和要求**

To learn how to implement displaying a character string circularly by using FSM.

学习如何利用状态机实现七段数码管滚动显示字符串。

**2 实验原理**

利用状态机实现七段数码管显示状态的变化。

**3 实验内容**

画状态图，写状态机代码，完成全部代码。

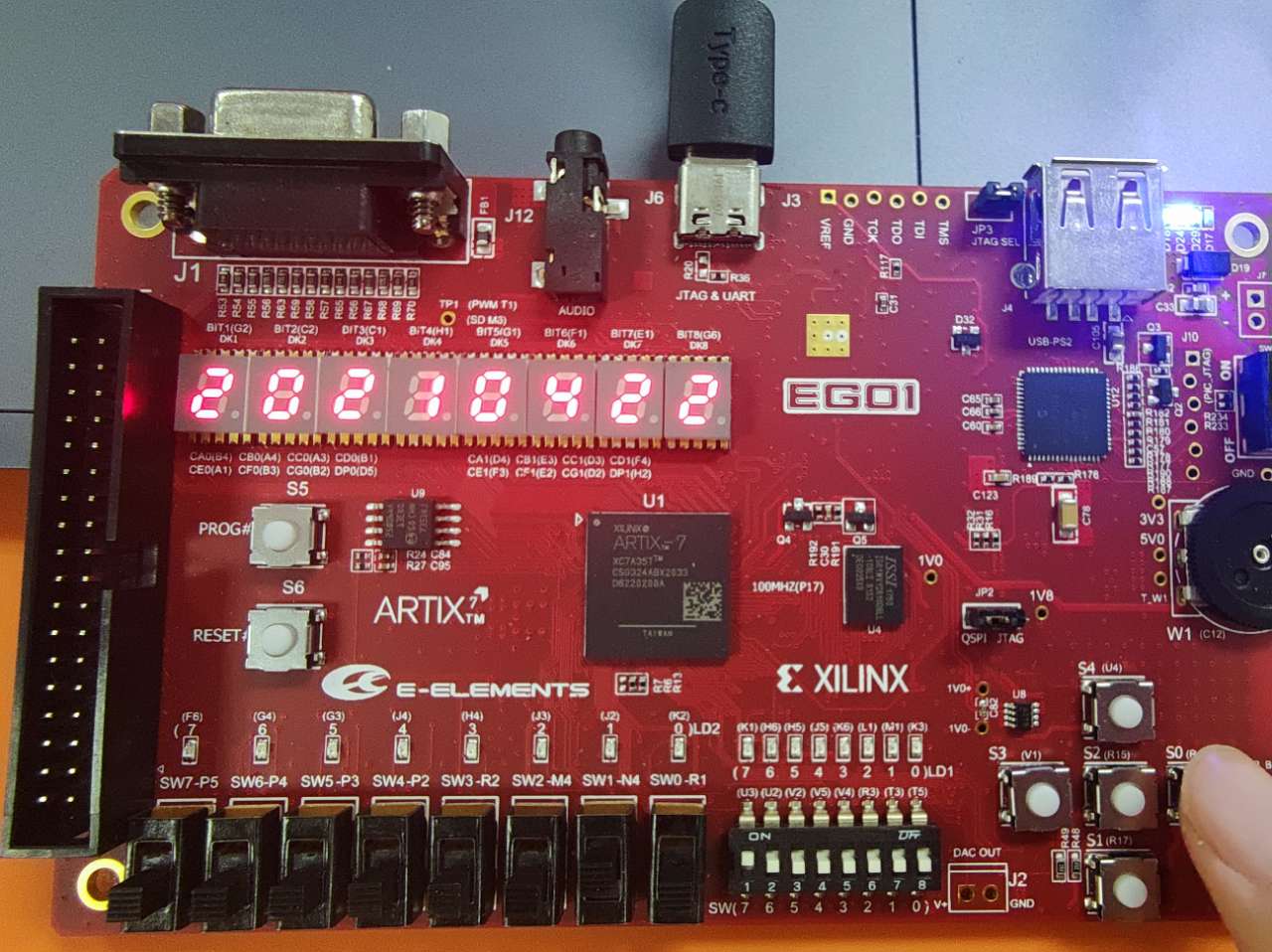
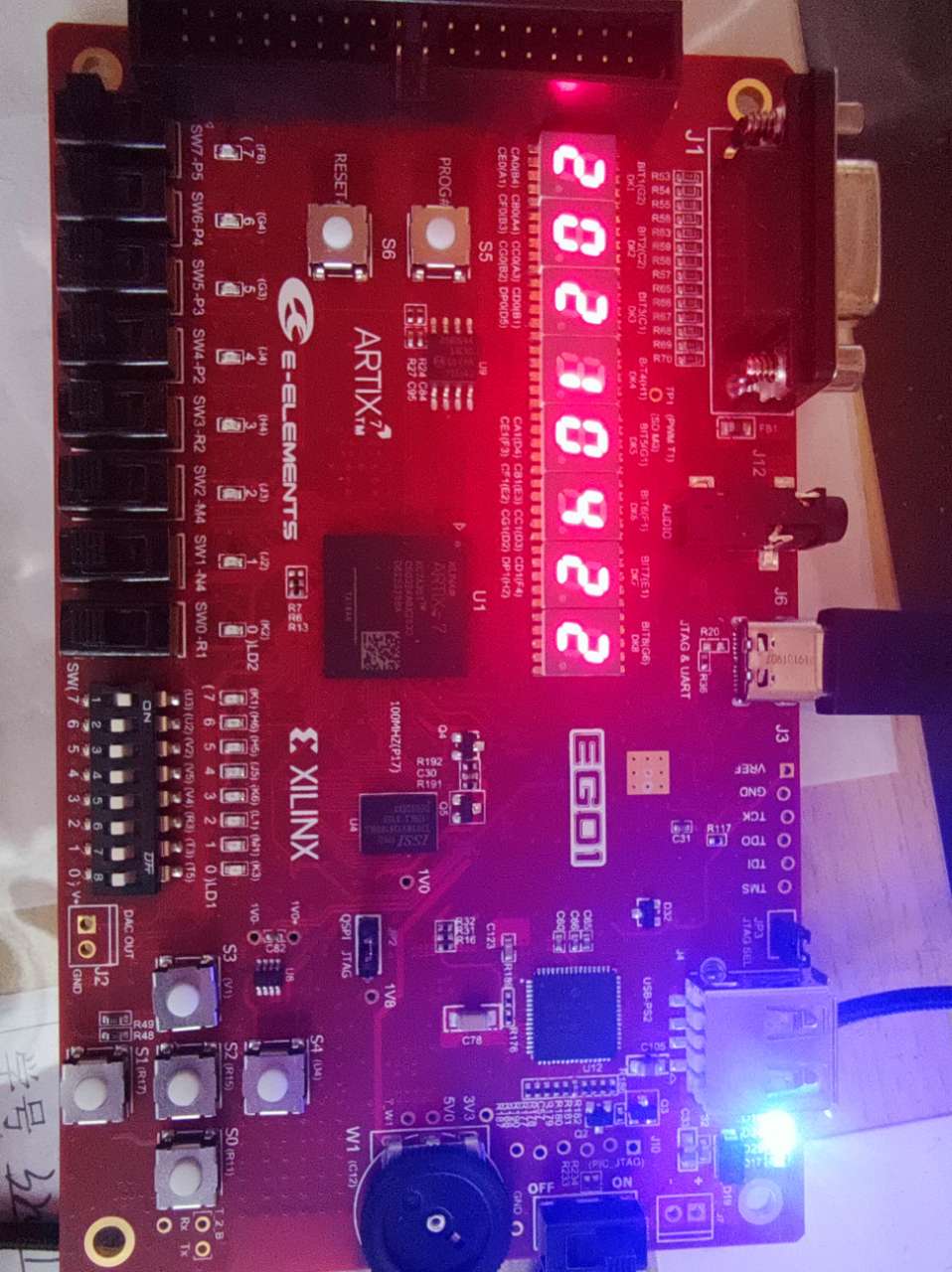
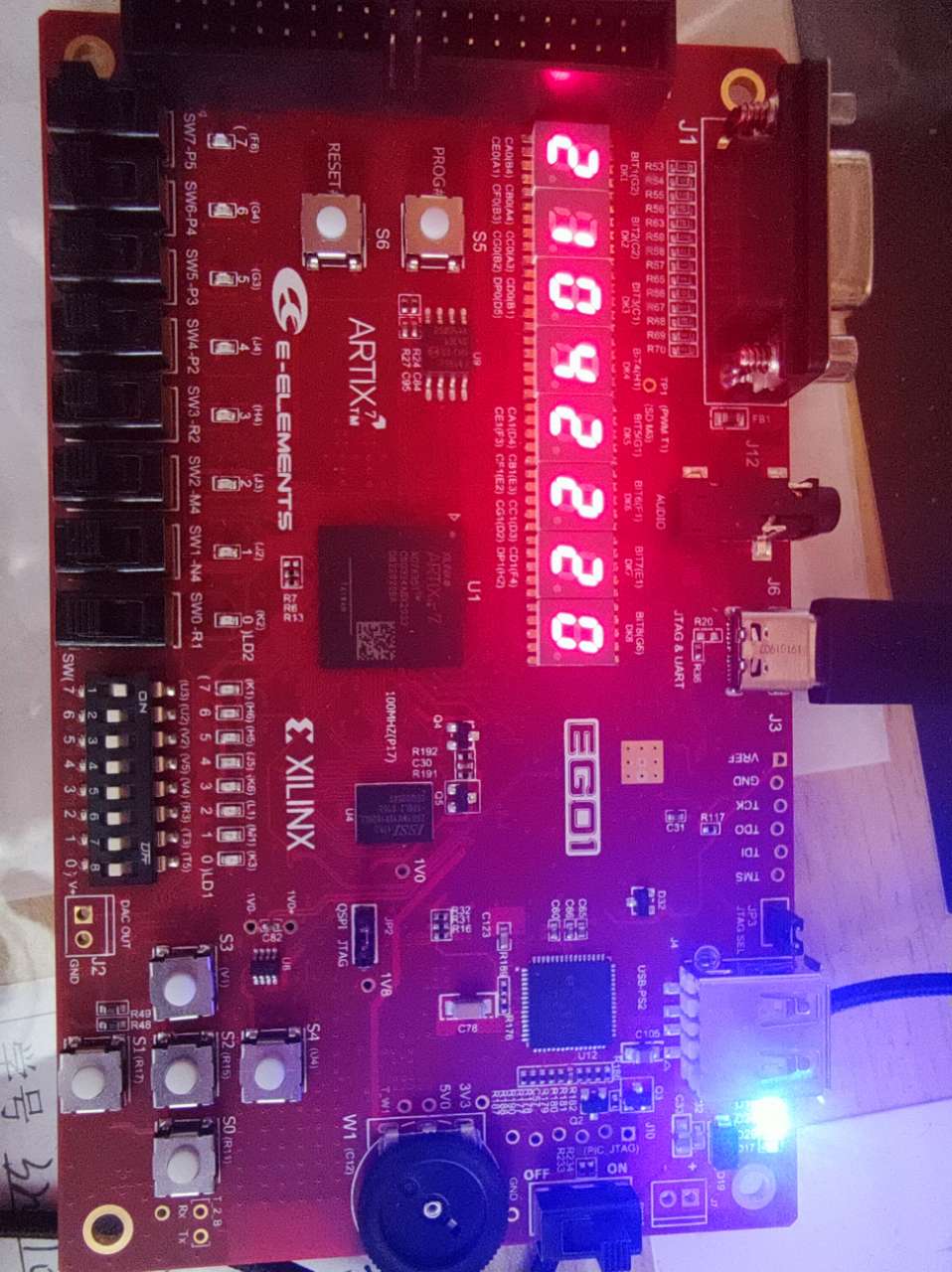
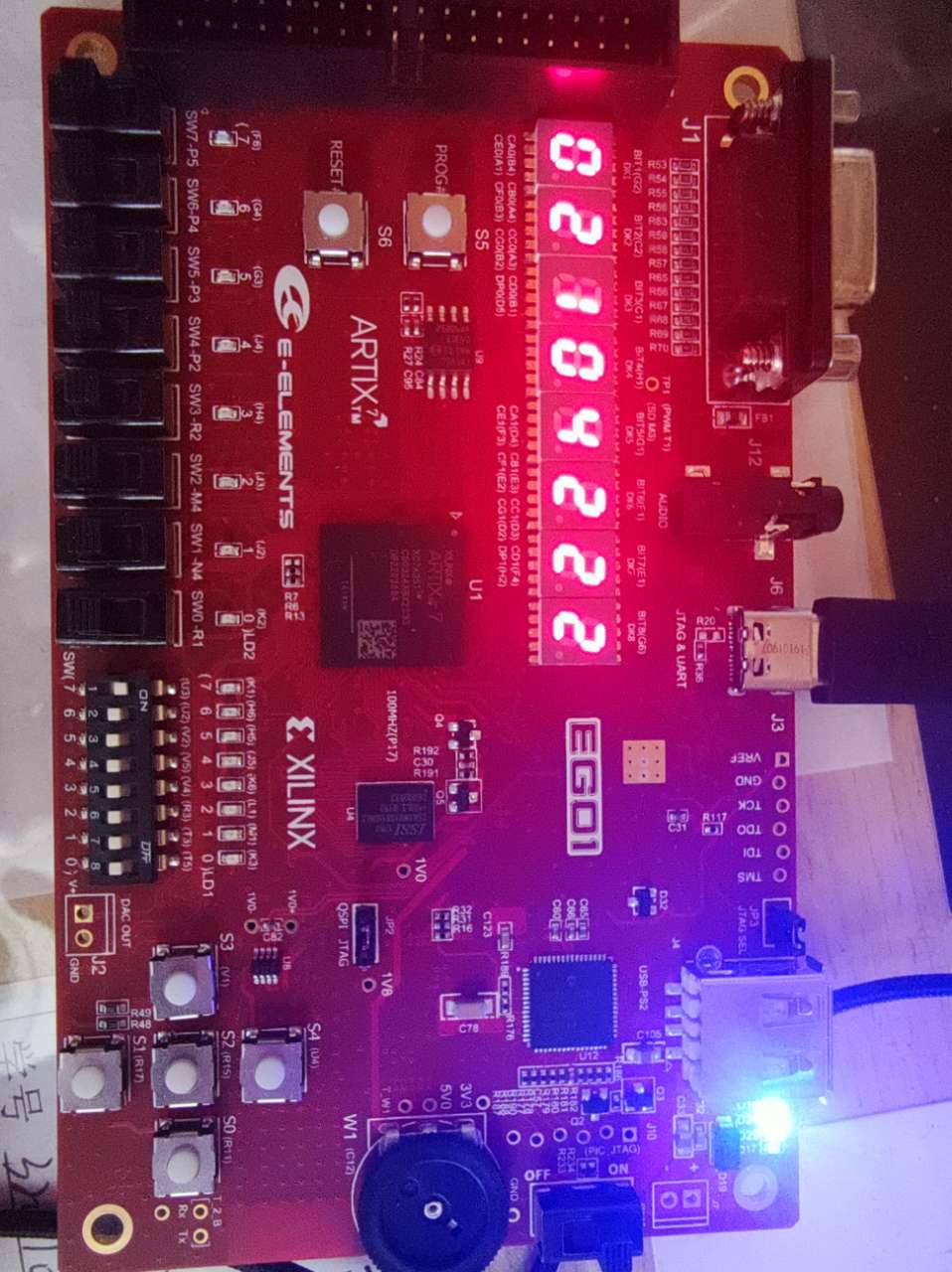
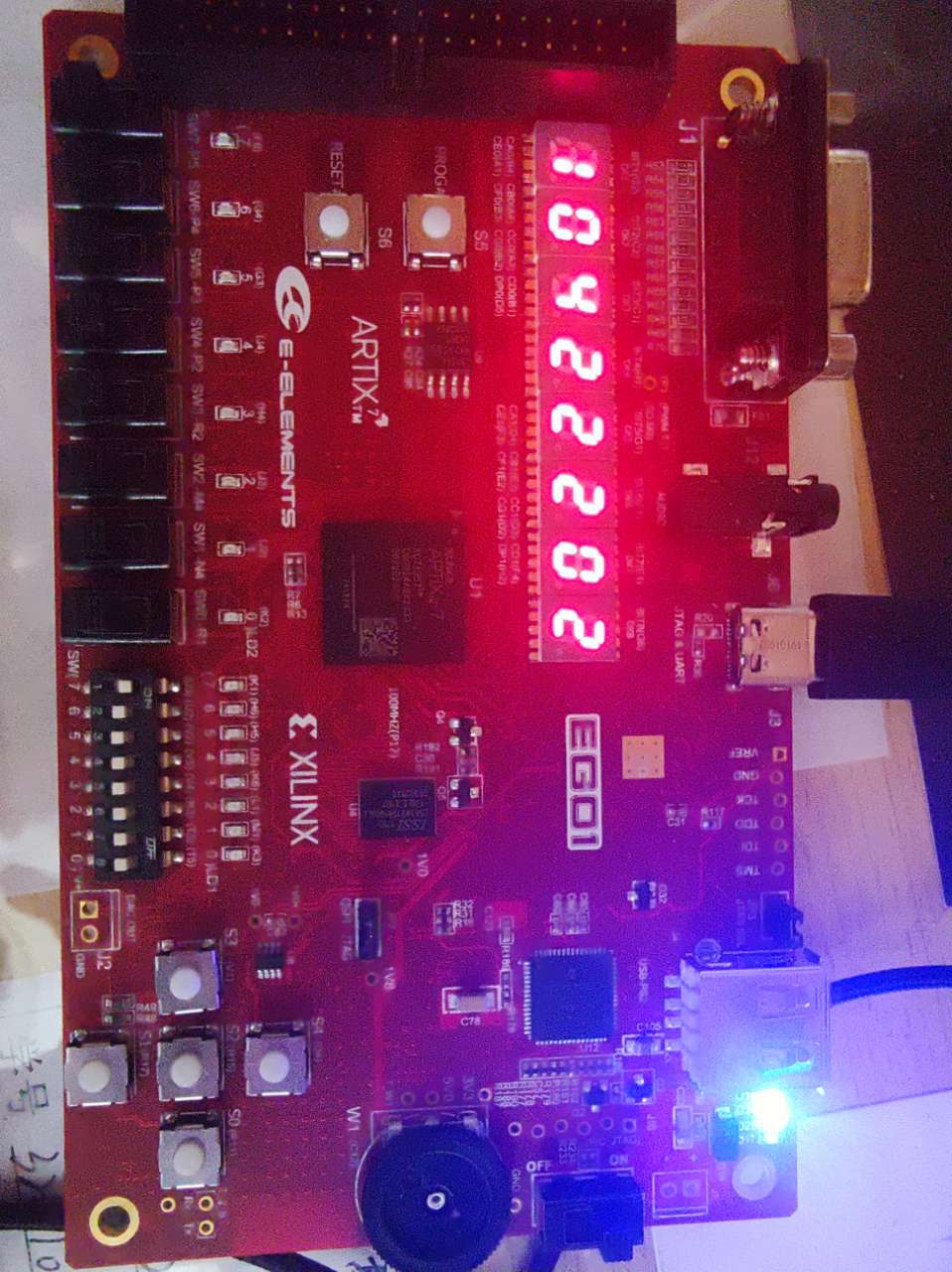
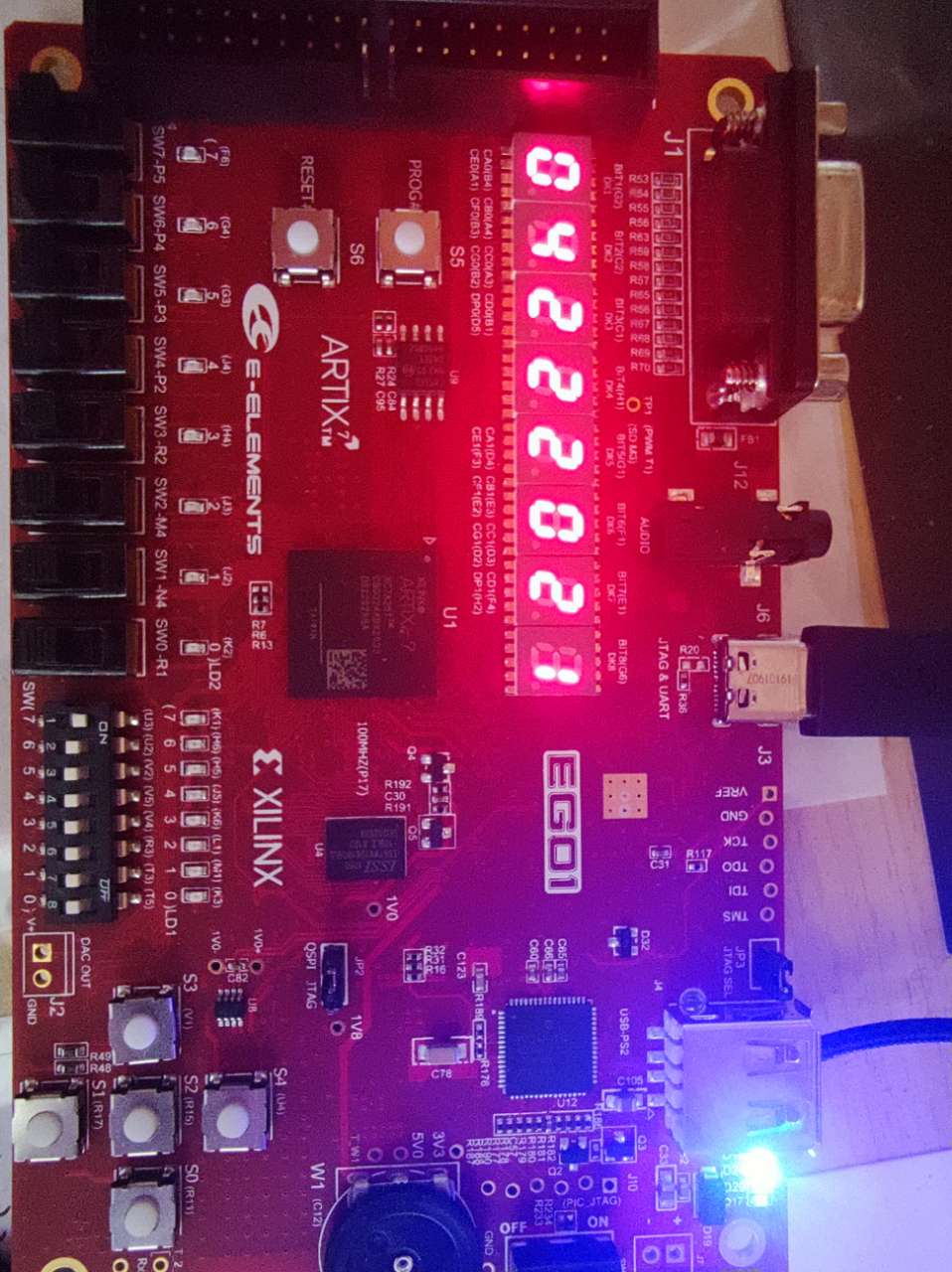
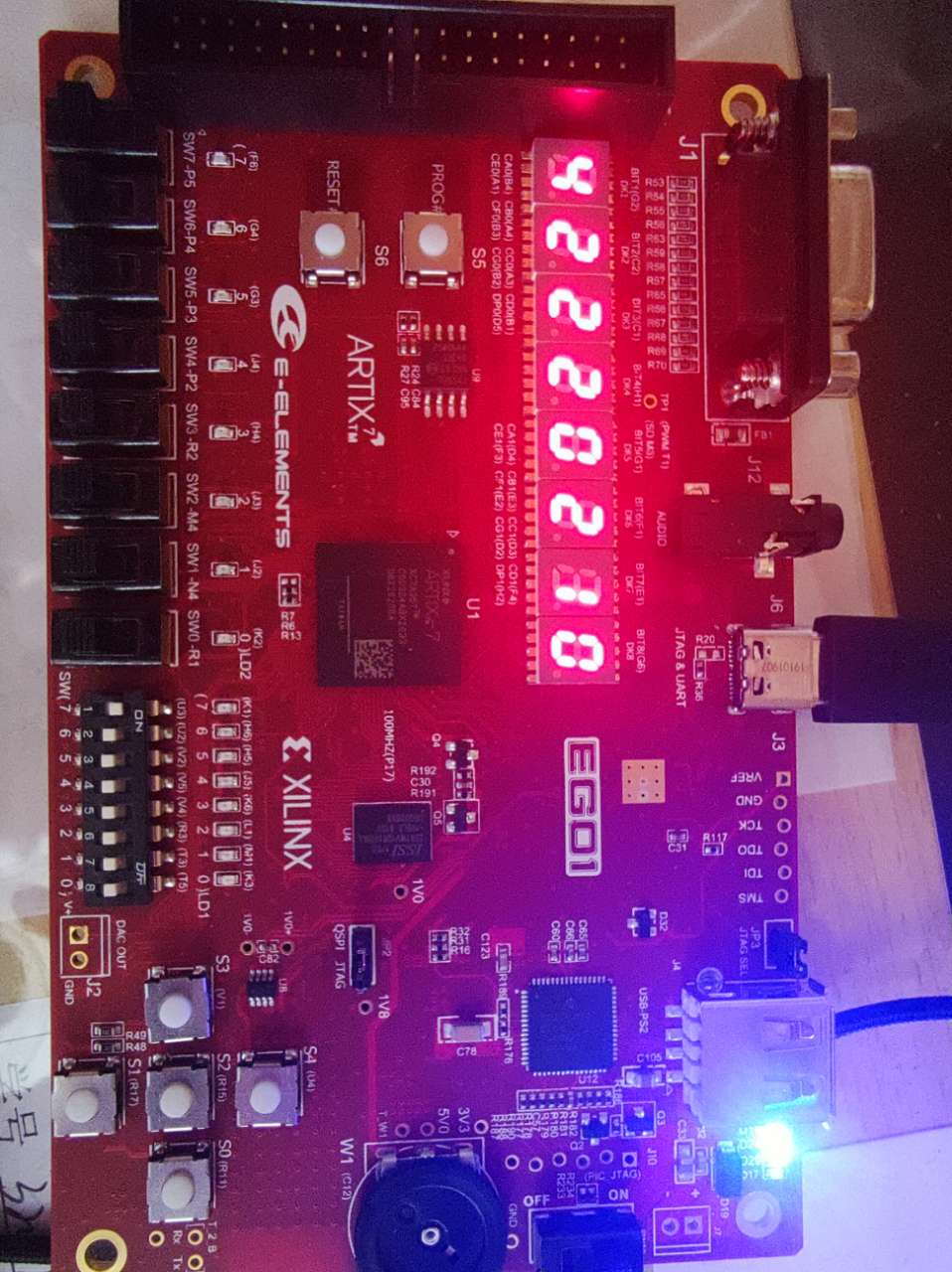
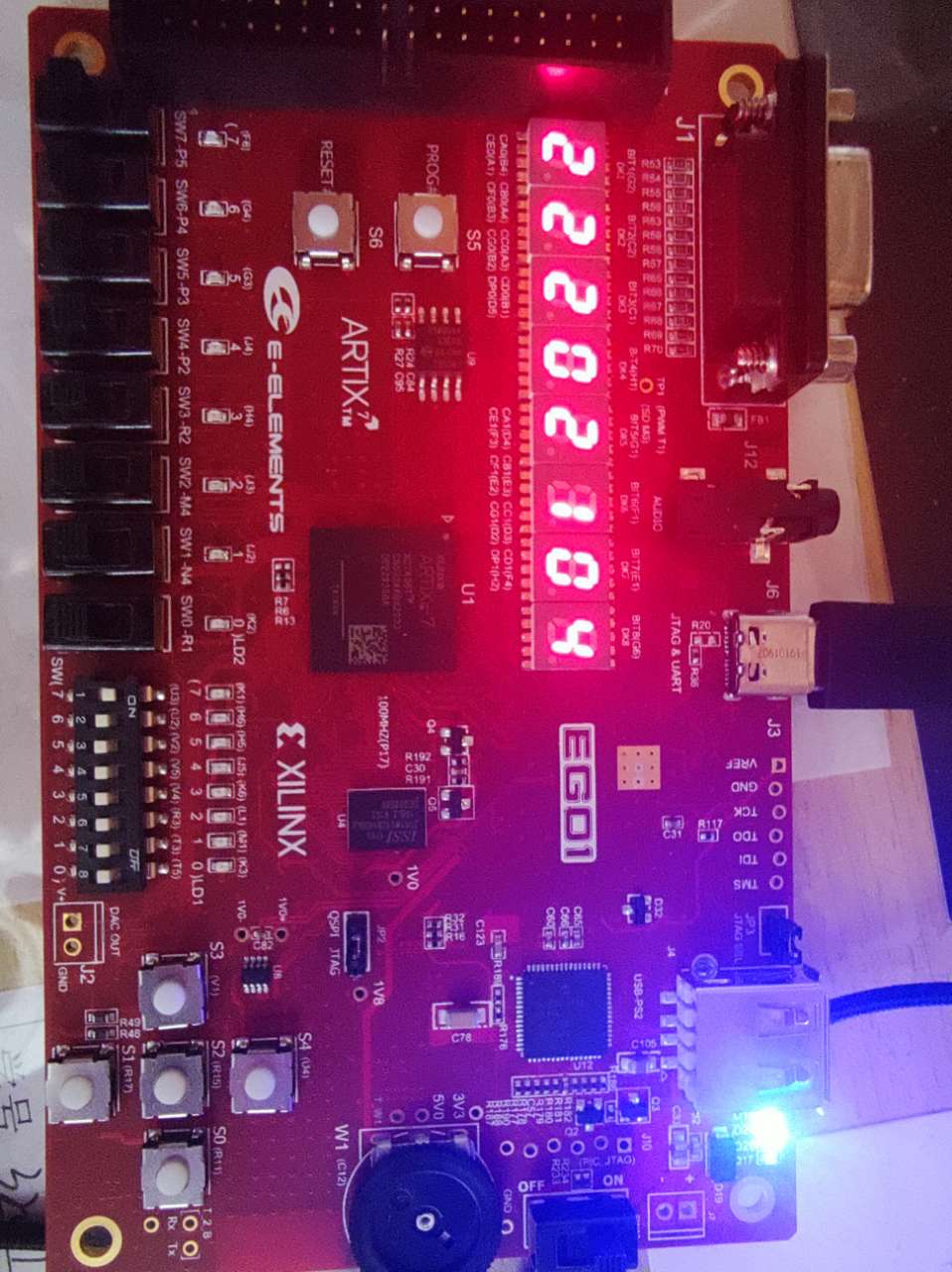
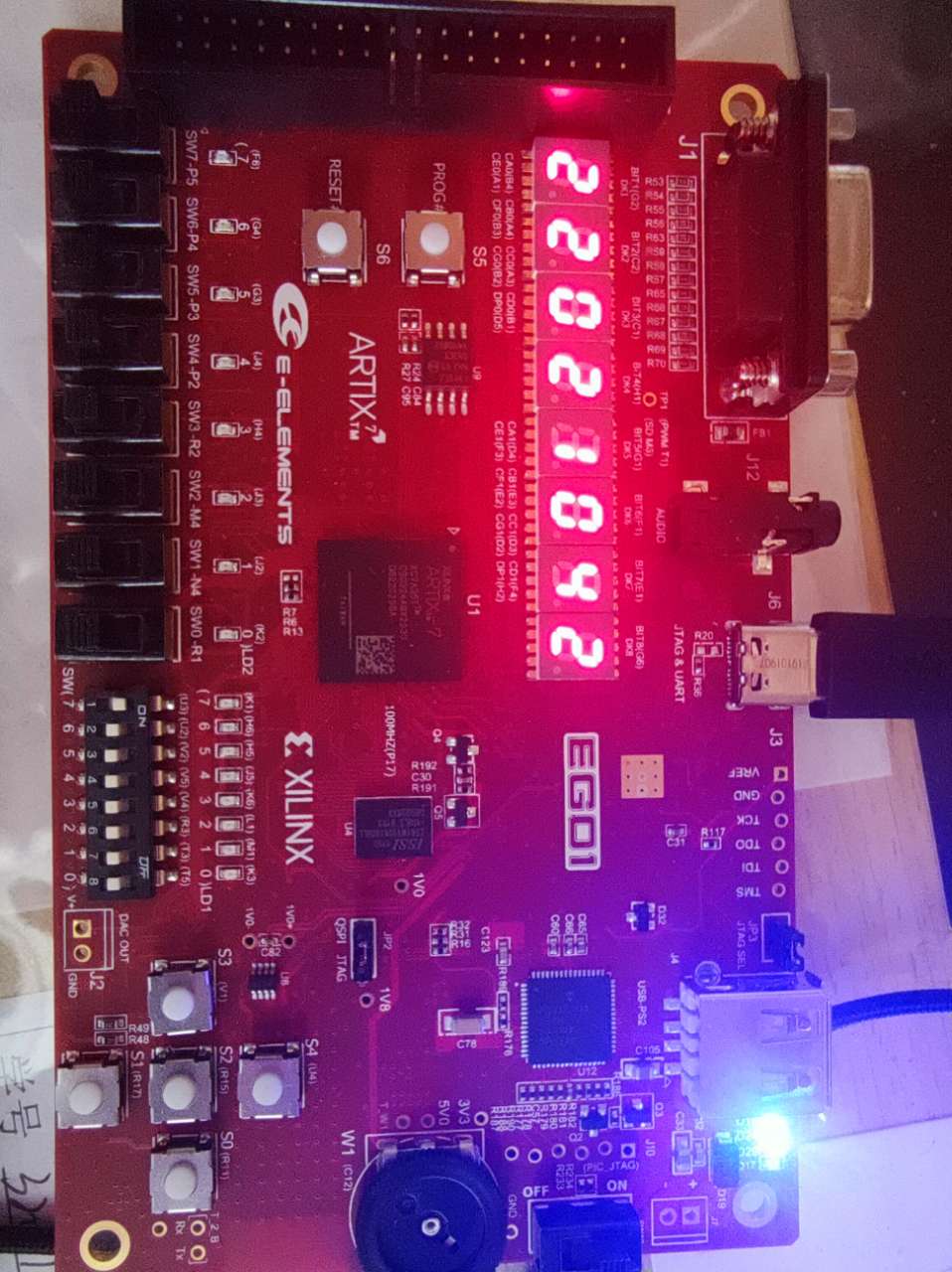
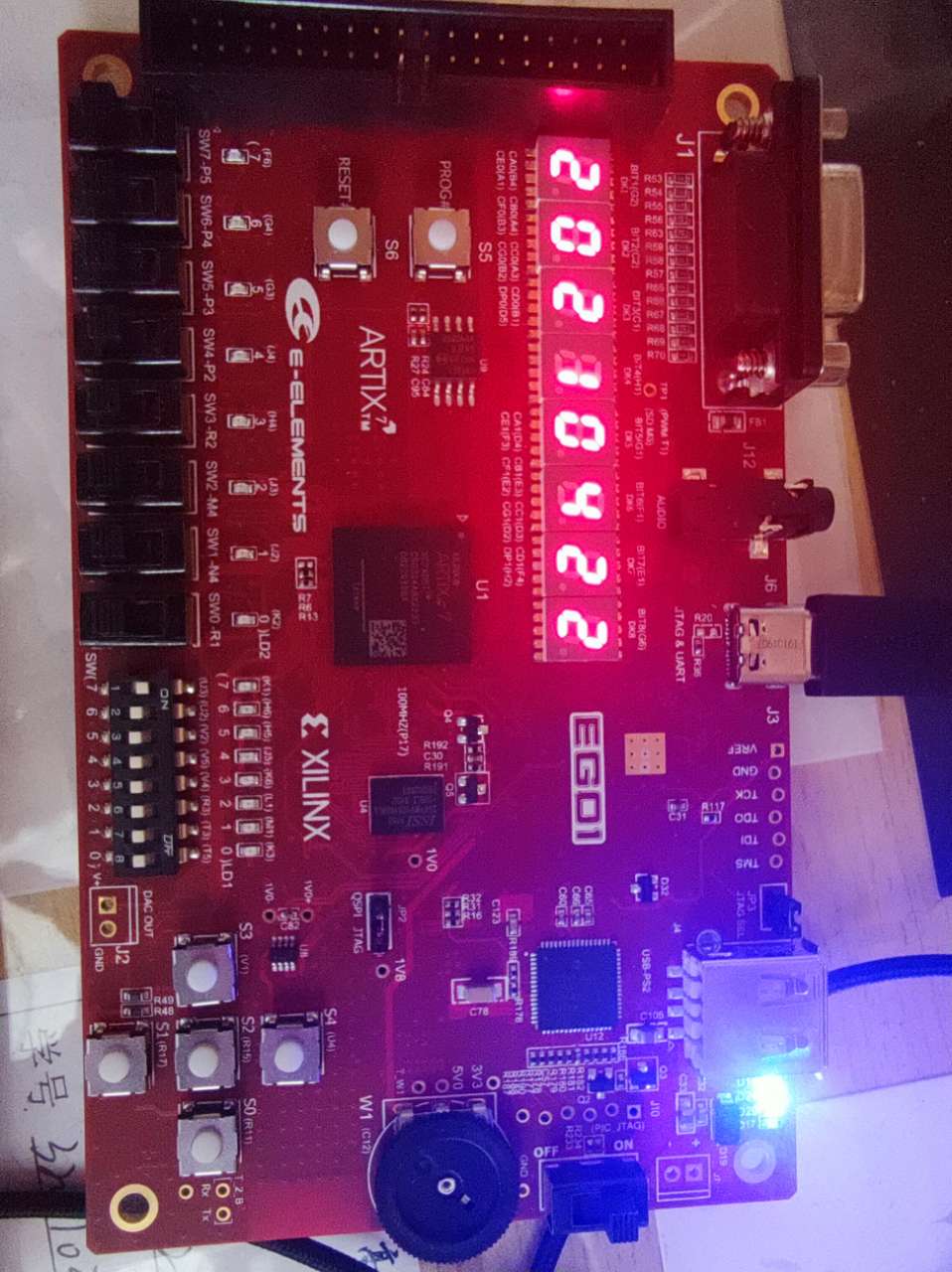
**4 实验结果和分析**

图 1按下reset

****

****

****

**5 实验结论和讨论**

实验结论：七位数码管可以滚动显示20210422的日期

遇到的问题和解决方案：

1. Q：七段数码管中四位是同步的，如何让其显示不同的数字？ A：以一种人眼无法识别的速度控制这四位循环显示需要的数字，看起来就是分别显示了不同的数字。
2. Q：显示的时钟和字符串移动的时钟信号？A：使用分频器，手动延长时钟信号，将分频后的信号作为时钟信号输入。
3. Q：最后reset按键作用恰好反了（没按时是按下的效果，只有按住才能滚动显示）？A：板子自带的reset按键是低电平有效。

**附件：源代码**

1、按钮

module debounce(

input clk, bounce,

output reg signal

);

parameter NDELAY = 650000;

reg [19:0] count;

reg xnew;

always@(posedge clk)

begin

if(bounce!= xnew) begin

xnew <= bounce; count <= 0;

end

else if(count == NDELAY) signal <= xnew;

else count <= count + 1;

end

module button(

input clk,button\_in,

output reg light

);

reg button,btemp;

always@(posedge clk)

{button,btemp} <= {btemp,button\_in};

wire bpressed;

debounce d1(.clk(clk),. bounce(button),. signal(bpressed));

reg old\_bpressed;

always@(posedge clk)

begin

if(old\_bpressed == 0 && bpressed == 1) light <= 1;

else light <= 0;

old\_bpressed <= bpressed;

end

endmodule

2、分频器

module M\_clk\_2ms(

input clk,

input reset,

output reg clk\_2ms

);

reg [16:0] count;

always@(posedge clk)

if(reset)

begin

count <= 0;

clk\_2ms <= 0;

end

else if(count >= 100000)

begin clk\_2ms <= ~clk\_2ms; count <= 0;end

else if(count < 100000)

count <= count + 1;

endmodule

module M\_clk\_2s(

input clk,

input reset,

output reg clk\_2s

);

reg [26:0] count;

always@(posedge clk)

if(reset)

begin

count <= 0;

clk\_2s <= 0;

end

else if(count == 100000000)

begin clk\_2s <= ~clk\_2s; count <= 0;end

else if(count <100000000)

count <= count + 1;

3、实现显示不同数字的状态机

module segment1(

input [2:0] state,

input clk\_2ms,

input reset,

output reg [11:0] seg1,seg2

);

reg [1:0] seg\_state,next\_seg\_state;

parameter n0 = 8'b1111\_1100;

parameter n1 = 8'b0110\_0000;

parameter n2 = 8'b1101\_1010;

parameter n4 = 8'b0110\_0110;

always@(posedge clk\_2ms)

if(reset)

seg\_state <= 0;

else

seg\_state <= next\_seg\_state;

always@(\*)

if(reset)

next\_seg\_state <= 0;

else

case(seg\_state)

0 : next\_seg\_state <= 1;

1 : next\_seg\_state <= 2;

2 : next\_seg\_state <= 3;

3 : next\_seg\_state <= 0;

default :next\_seg\_state <= 0;

endcase

always@(\*)

if(reset)

begin

seg1 <= 0;

seg2 <= 0;

end

else if(seg\_state == 0)

case(state)

0 : begin seg1 <= {4'b1000,n2}; seg2 <= {4'b1000,n0}; end

1 : begin seg1 <= {4'b1000,n0}; seg2 <= {4'b1000,n4}; end

2 : begin seg1 <= {4'b1000,n2}; seg2 <= {4'b1000,n2}; end

3 : begin seg1 <= {4'b1000,n1}; seg2 <= {4'b1000,n2}; end

4 : begin seg1 <= {4'b1000,n0}; seg2 <= {4'b1000,n2}; end

5 : begin seg1 <= {4'b1000,n4}; seg2 <= {4'b1000,n0}; end

6 : begin seg1 <= {4'b1000,n2}; seg2 <= {4'b1000,n2}; end

7 : begin seg1 <= {4'b1000,n2}; seg2 <= {4'b1000,n1}; end

default : begin seg1 <= {4'b1000,n2}; seg2 <= {4'b1000,n0}; end

endcase

else if(seg\_state == 1)

case(state)

0 : begin seg1 <= {4'b0100,n0}; seg2 <= {4'b0100,n4}; end

1 : begin seg1 <= {4'b0100,n2}; seg2 <= {4'b0100,n2}; end

2 : begin seg1 <= {4'b0100,n1}; seg2 <= {4'b0100,n2}; end

3 : begin seg1 <= {4'b0100,n0}; seg2 <= {4'b0100,n2}; end

4 : begin seg1 <= {4'b0100,n4}; seg2 <= {4'b0100,n0}; end

5 : begin seg1 <= {4'b0100,n2}; seg2 <= {4'b0100,n2}; end

6 : begin seg1 <= {4'b0100,n2}; seg2 <= {4'b0100,n1}; end

7 : begin seg1 <= {4'b0100,n2}; seg2 <= {4'b0100,n0}; end

default : begin seg1 <= {4'b0100,n0}; seg2 <= {4'b0100,n4}; end

endcase

else if(seg\_state == 2)

case(state)

0 : begin seg1 <= {4'b0010,n2}; seg2 <= {4'b0010,n2}; end

1 : begin seg1 <= {4'b0010,n1}; seg2 <= {4'b0010,n2}; end

2 : begin seg1 <= {4'b0010,n0}; seg2 <= {4'b0010,n2}; end

3 : begin seg1 <= {4'b0010,n4}; seg2 <= {4'b0010,n0}; end

4 : begin seg1 <= {4'b0010,n2}; seg2 <= {4'b0010,n2}; end

5 : begin seg1 <= {4'b0010,n2}; seg2 <= {4'b0010,n1}; end

6 : begin seg1 <= {4'b0010,n2}; seg2 <= {4'b0010,n0}; end

7 : begin seg1 <= {4'b0010,n0}; seg2 <= {4'b0010,n4}; end

default : begin seg1 <= {4'b0010,n2}; seg2 <= {4'b0010,n2}; end

endcase

else if(seg\_state == 3)

case(state)

0 : begin seg1 <= {4'b0001,n1}; seg2 <= {4'b0001,n2}; end

1 : begin seg1 <= {4'b0001,n0}; seg2 <= {4'b0001,n2}; end

2 : begin seg1 <= {4'b0001,n4}; seg2 <= {4'b0001,n0}; end

3 : begin seg1 <= {4'b0001,n2}; seg2 <= {4'b0001,n2}; end

4 : begin seg1 <= {4'b0001,n2}; seg2 <= {4'b0001,n1}; end

5 : begin seg1 <= {4'b0001,n2}; seg2 <= {4'b0001,n0}; end

6 : begin seg1 <= {4'b0001,n0}; seg2 <= {4'b0001,n4}; end

7 : begin seg1 <= {4'b0001,n2}; seg2 <= {4'b0001,n2}; end

default : begin seg1 <= {4'b0001,n1}; seg2 <= {4'b0001,n2}; end

endcase

endmodule

4、实现滚动显示的状态机（及顶层）

module display(

input clk,

input reset,

output [11:0] seg1,

output [11:0] seg2

);

reg [2:0] state , next\_state;

wire [2:0] s\_state;

wire clk\_2ms,clk\_2s,reset\_out;

button U1(clk,reset,reset\_out);

M\_clk\_2ms M1(.clk(clk),.reset(reset\_out),.clk\_2ms(clk\_2ms));

M\_clk\_2s M2(.clk(clk),.reset(reset\_out),.clk\_2s(clk\_2s));

segment1 S1(.clk\_2ms(clk\_2ms),.reset(reset\_out),.state(s\_state),.seg1(seg1),.seg2(seg2));

assign s\_state = state;

always@(posedge clk\_2s)

if(reset)

state <= 3'b000;

else

state <= next\_state;

always@(\*)

if(reset)

next\_state <= 3'b000;

else

case(state)

1 : next\_state <= 2;

2 : next\_state <= 3;

3 : next\_state <= 4;

4 : next\_state <= 5;

5 : next\_state <= 6;

6 : next\_state <= 7;