**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：Ouedraogo Ezekiel B.

学生学号：PL19215001

完成日期：12/16/2020

计算机实验教学中心制

2020年10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：vlab.ustc.eud.cn

FPGAOL: fpgaol.ustc.edu.cn

Logisim

Vivado

【实验过程】

1. 信号整形及去毛刺

Verilog 代码

**module** jitter\_clr**(**

**input** clk**,**

**input** button**,**

**output** button\_clean

**);**

**reg** **[**3**:**0**]** cnt**;**

**always@(posedge** clk**)**

**begin**

**if(**button**==**1'b0**)**

cnt **<=** 4'h0**;**

**else** **if(**cnt**<**4'h8**)**

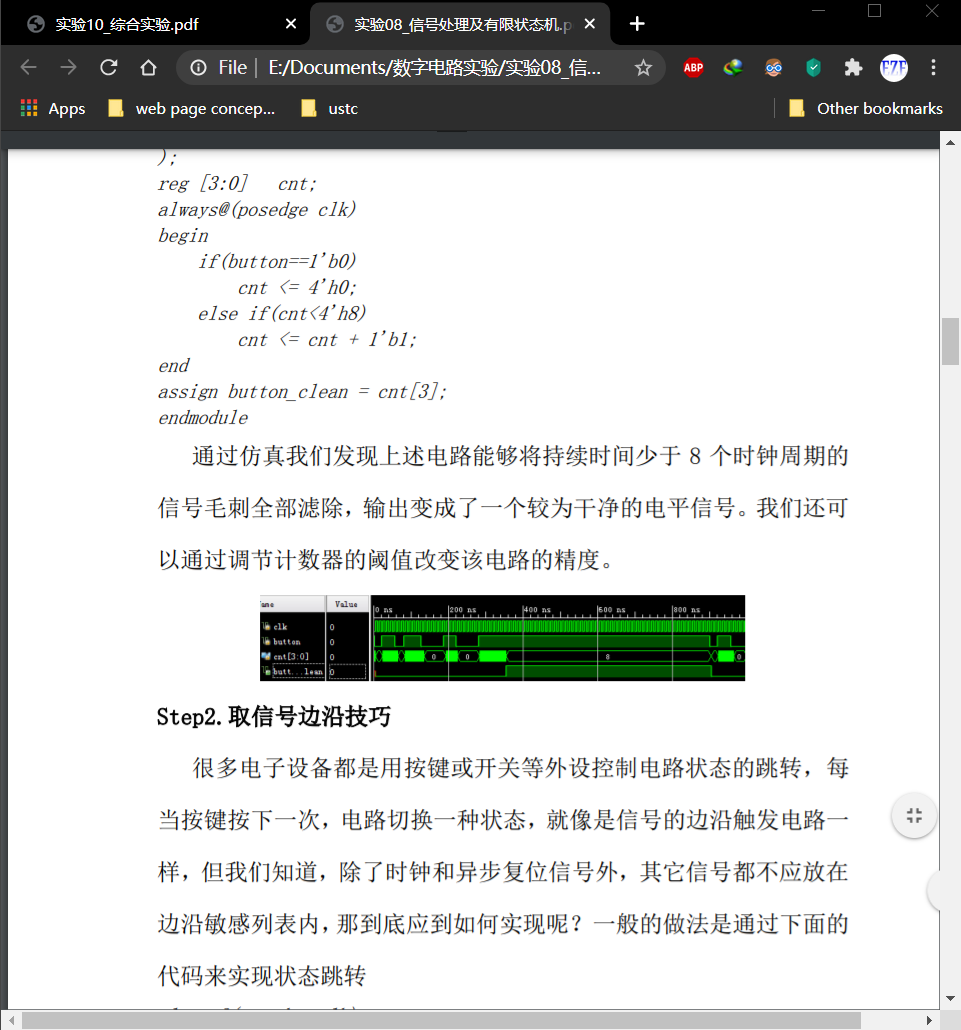
cnt **<=** cnt **+** 1'b1**;**

**end**

**assign** button\_clean **=** cnt**[**3**];**

**endmodule**

仿真结果



1. 取信号边沿技巧

Verilog代码

**module** signal\_edge**(**

**input** clk**,**

**input** button**,**

**output** button\_edge**);**

**reg** button\_r1**,**button\_r2**;**

**always@(posedge** clk**)** button\_r1 **<=** button**;**

**always@(posedge** clk**)** button\_r2 **<=** button\_r1**;**

**assign** button\_edge **=** button\_r1 **&** **(~**button\_r2**);**

**endmodule**

仿真文件

**module** tb**();**

**reg** clk**,**button**;**

**wire** button\_edge**;**

signal\_edge s**(.**clk**(**clk**),.**button**(**button**),.**button\_edge**(**button\_edge**));**

**initial** clk **<=** 0**;**

**always** **#**1 clk**<=** **~**clk**;**

**initial**

**begin**

button **=** 0**;**

**#**5 button **=** 1**;** **#**5 button **=** 0**;** **#**5 button **=** 1**; #**5 button **=** 0**;**

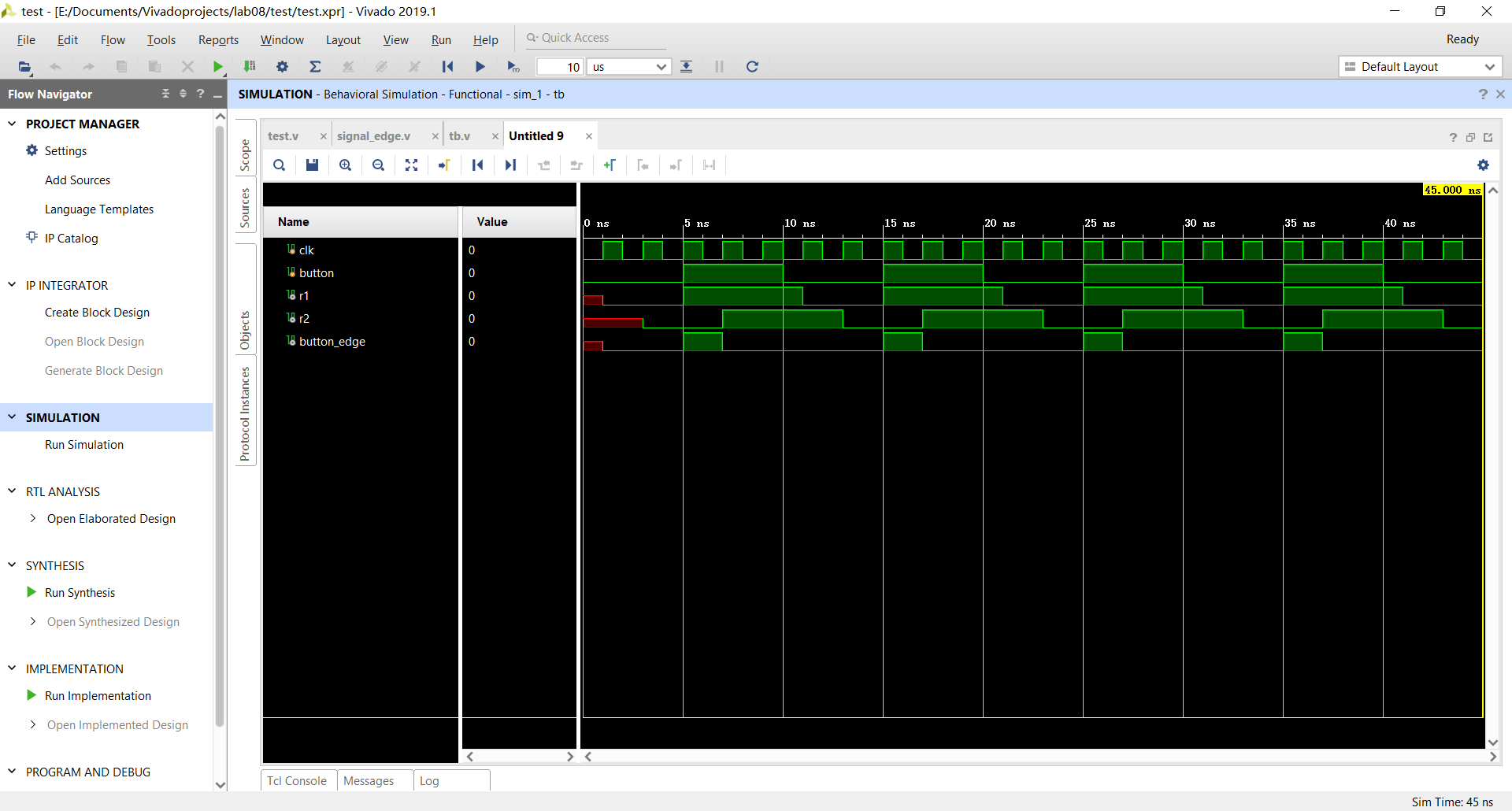
**#**5 button **=** 1**;** **#**5 button **=** 0**;** **#**5 button **=** 1**; #**5 button **=** 0**;**

**#**5 $finish**;**

**end**

**endmodule**

结果

【实验练习】

1. 有限状态机

将代码改写成三段式有限状态机的形式

**module** test**(**

**input** clk**,**rst**,**

**output** led**);**

**parameter** STATE\_0 **=** 2'b00**;**

**parameter** STATE\_1 **=** 2'b01**;**

**parameter** STATE\_2 **=** 2'b10**;**

**parameter** STATE\_3 **=** 2'b11**;**

**reg** **[**1**:**0**]** curr\_state**;**

**reg** **[**1**:**0**]** next\_state**;**

//有限状态机第一部分

**always@(\*)**

**begin**

**case(**curr\_state**)**

STATE\_0**:** next\_state **=** STATE\_0**;**

STATE\_1**:** next\_state **=** STATE\_2**;**

STATE\_2**:** next\_state **=** STATE\_3**;**

STATE\_3**:** next\_state **=** STATE\_0**;**

**endcase**

**end**

//有限状态机第二部分

**always@(posedge** clk **or** **posedge** rst**)**

**begin**

**if(**rst**)**

curr\_state **<=** STATE\_0**;**

**else**

curr\_state **<=** next\_state**;**

**end**

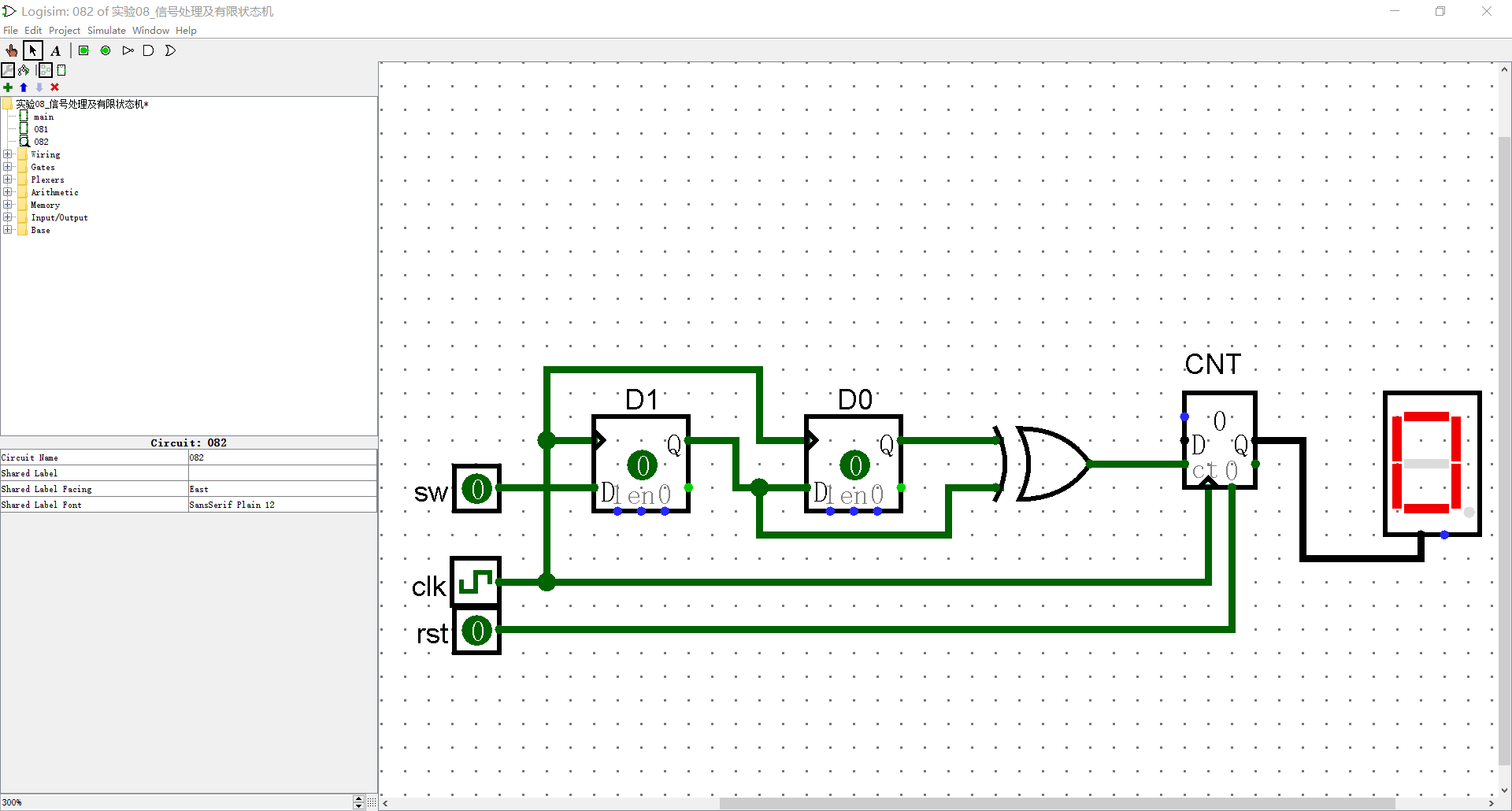
//有限状态机第三部分

**assign** led **=** **(**curr\_state**==**STATE\_3**);**

**endmodule**

1. 计数器

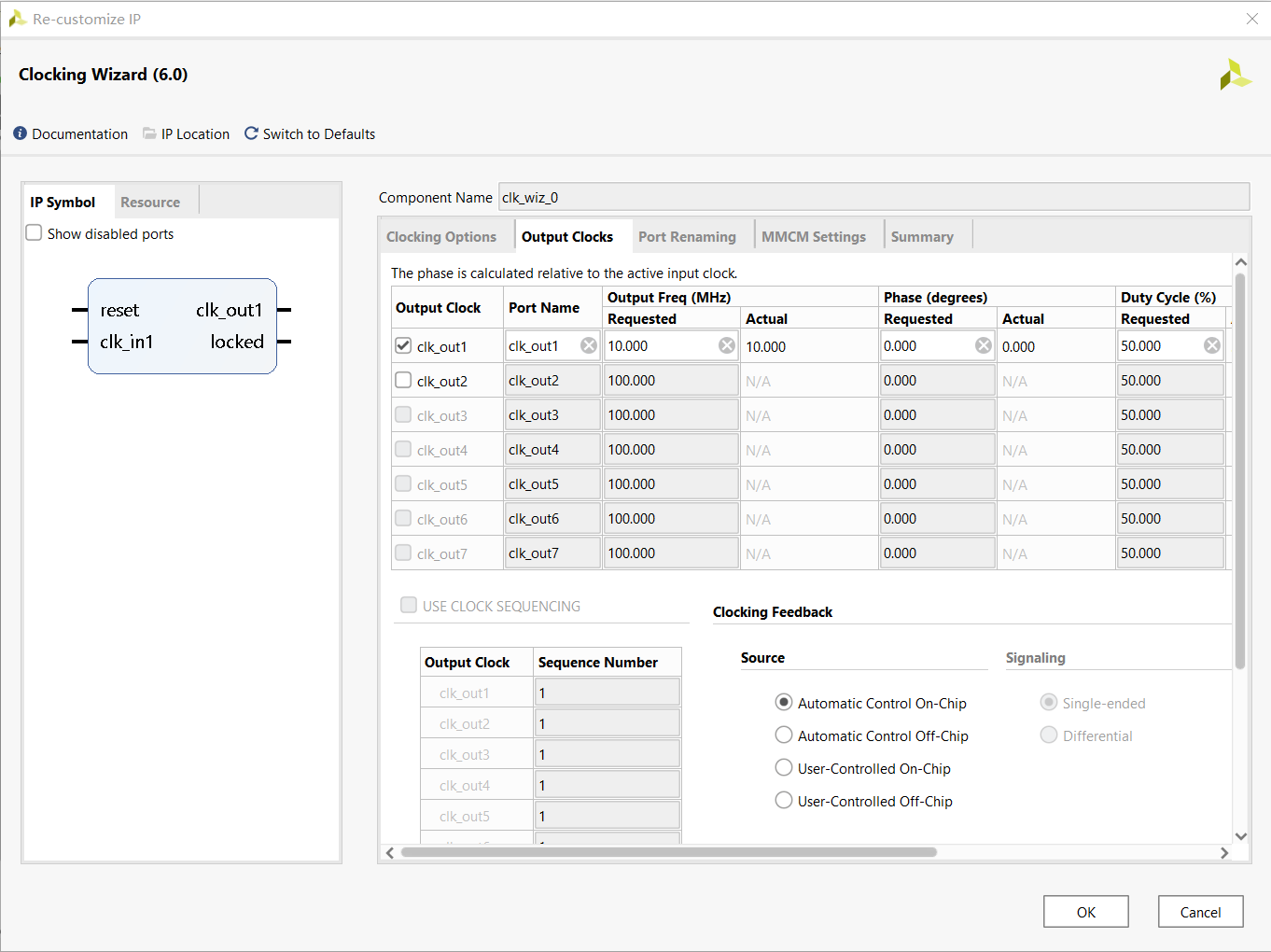
在 Logisim 中设计一个 4bit 位宽的计数器电路，在输入信号 sw 电平发生变化时，计数值 cnt 加 1。



图中D0,D1为D触发器，CNT为4位宽的计数器。

1. 八位的十六进制计数器

例化一个10Mhz的时钟



Verilog代码

**module** signal\_edge**(**

**input** clk**,**

**input** button**,**

**output** button\_edge**);**

**reg** button\_r1**,**button\_r2**;**

**always@(posedge** clk**)** button\_r1 **<=** button**;**

**always@(posedge** clk**)** button\_r2 **<=** button\_r1**;**

**assign** button\_edge **=** button\_r1 **&** **(~**button\_r2**);**

**endmodule**

**module** count**(**

**input** clk**,** start**,**

**input** **[**1**:**0**]** sw**,**

**output** **reg** **[**3**:**0**]**out**,**

**output** **reg** **[**2**:**0**]**an

**);**

**wire** start\_edge**;**

signal\_edge signal\_edge**(.**clk**(**clk**),.**button**(**start**),.**button\_edge**(**start\_edge**));**

**reg** **[**7**:**0**]**tmp**;**

**wire** clk\_10mz**;**

clk\_wiz\_0 clk\_wiz**(.**clk\_in1**(**clk**),.**reset**(**0**),.**clk\_out1**(**clk\_10mz**));**

**always@(posedge** clk**)**

**begin**

**if(**sw**[**1**])** tmp **<=** 8'h1f**;**

**else** **if(**start\_edge**)**

**begin**

**if(**sw**)** tmp **<=** tmp **+** 8'h1**;**

**else** tmp **<=** tmp **-** 8'h1**;**

**end**

**end**

**always@(\*)**

**begin**

**if(**clk\_10mz**)**

**begin**

an **<=** 3'b0**;**

out **<=** tmp**[**3**:**0**];**

**end**

**else**

**begin**

an **<=** 3'b1**;**

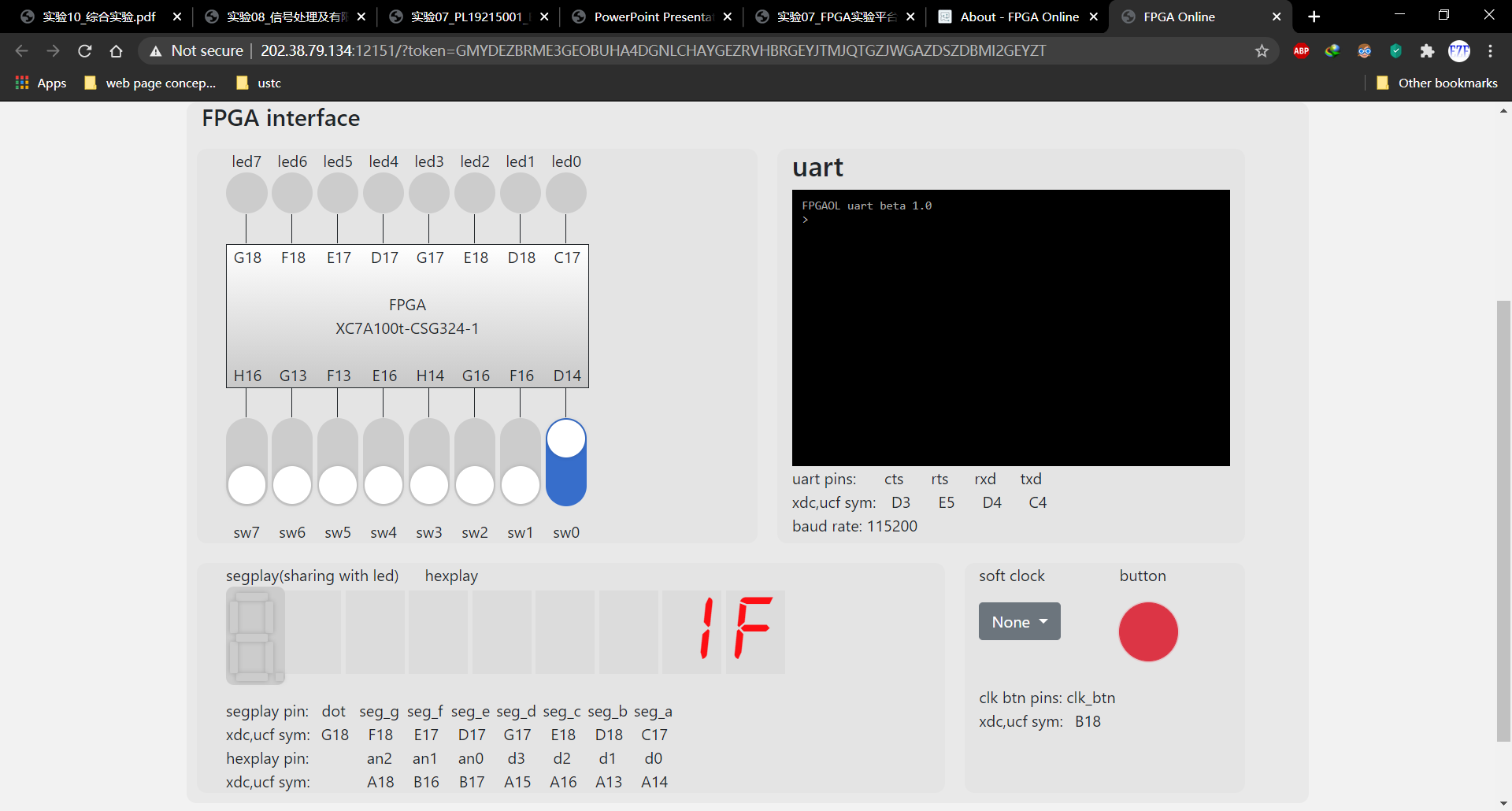
out **<=** tmp**[**7**:**4**];**

**end**

**end**

**endmodule**

结果



1. 序列检测电路
   1. 状态跳转图



* 1. 电路实现

代码

**module** counter**(**

**input** clk**,** rst**,** start**,** sw**,**

**output** **reg** **[**3**:**0**]** curr\_state**,** count**,** last

**);**

**parameter** **[**3**:**0**]** STATE\_0 **=** 4'h0**,**

STATE\_1 **=** 4'h1**,**

STATE\_2 **=** 4'h2**,**

STATE\_3 **=** 4'h3**,**

STATE\_4 **=** 4'h4**;**

**reg** **[**3**:**0**]** next\_state**,** last\_tmp**;**

**wire** start\_edge**;**

signal\_edge s**(.**clk**(**clk**),.**button**(**start**),.**button\_edge**(**start\_edge**));**

**always@(\*)**

**begin**

**if(**start\_edge**)**

**begin**

**case(**curr\_state**)**

STATE\_0**:** **begin**

**if(**sw**)** next\_state **<=** STATE\_1**;**

**else** next\_state **<=** curr\_state**;**

**end**

STATE\_1**:** **begin**

**if(**sw**)** next\_state **<=** STATE\_2**;**

**else** next\_state **<=** STATE\_0**;**

**end**

STATE\_2**:** **begin**

**if(**sw**)** next\_state **<=** curr\_state**;**

**else** next\_state **<=** STATE\_3**;**

**end**

STATE\_3**:** **begin**

**if(**sw**)** next\_state **<=** STATE\_1**;**

**else** next\_state **<=** STATE\_4**;**

**end**

STATE\_4**:** **begin**

**if(**sw**)** next\_state **<=** STATE\_1**;**

**else** next\_state **<=** STATE\_0**;**

**end**

**default:** next\_state **<=** STATE\_0**;**

**endcase**

**end**//if start\_edge

**else** next\_state **<=** curr\_state**;**

**end**

**reg** once1**;** //use once to make sure a task just run only once

**always@(posedge** clk **or** **posedge** rst**)**

**begin**

**if(**rst**)** last **<=** 4'h0**;**

**else** **if(**start\_edge**)**

**begin**

**if(**once1**)**

**begin**

//save last inputs

last **<=** **{**sw**,**last\_tmp**[**3**:**1**]};**

once1 **<=** 1'b0**;**

**end**

**end**

**else**

**begin**

once1 **<=** 1'b1**;**

last\_tmp **<=** last**;**

**end**

**end**

**always@(posedge** clk **or** **posedge** rst**)**

**begin**

**if(**rst**)** curr\_state **<=** STATE\_0**;**

**else** curr\_state **<=** next\_state**;**

**end**

**reg** once2**;**

**always@(posedge** clk**)**

**begin**

**if(**rst**)** count **<=** 4'h0**;**

**else** **if(**curr\_state**==**STATE\_4**)**

**begin**

**if(**once2 **)**

**begin**

count **<=** count **+** 4'b1**;**

once2 **<=** 1'b0**;**

**end**

**end**

**else** once2 **<=** 1'b1**;**

**end**

**endmodule**

**module** display**(**

**input** clk**,** rst**,** start**,** sw**,**

**output** **reg** **[**3**:**0**]** out**,**

**output** **reg** **[**2**:**0**]** an

**);**

**wire** **[**3**:**0**]** curr\_state**,** count**,** last**;**

counter c**(.**clk**(**clk**),.**rst**(**rst**),.**start**(**start**),.**sw**(**sw**),.**curr\_state**(**curr\_state**),.**count**(**count**),.**last**(**last**));**

**wire** clk\_out1**;**

clk\_wiz\_0 clk\_wiz\_0**(.**clk\_in1**(**clk**),.**reset**(**0**),.**clk\_out1**(**clk\_out1**));**

**reg** **[**2**:**0**]** tmp**;**

**always@(posedge** clk\_out1**)**

**begin**

**if(**tmp **>=** 3'h5**)** tmp **<=** 3'h0**;**

**else** tmp **<=** tmp **+** 3'h1**;**

**end**

**always@(\*)**

**begin**

**case(**tmp**)**

3'h0**:** **begin**

an **<=** 3'h0**;**

out **<=** curr\_state**;**

**end**

3'h1**:** **begin**

an **<=** 3'h2**;**

out **<=** count**;**

**end**

3'h2**:** **begin**

an **<=** 3'h4**;**

out **<=** **{**3'h0**,**last**[**0**]};**

**end**

3'h3**:** **begin**

an **<=** 3'h5**;**

out **<=** **{**3'h0**,**last**[**1**]};**

**end**

3'h4**:** **begin**

an **<=** 3'h6**;**

out **<=** **{**3'h0**,**last**[**2**]};**

**end**

3'h5**:** **begin**

an **<=** 3'h7**;**

out **<=** **{**3'h0**,**last**[**3**]};**

**end**

**endcase**

**end**

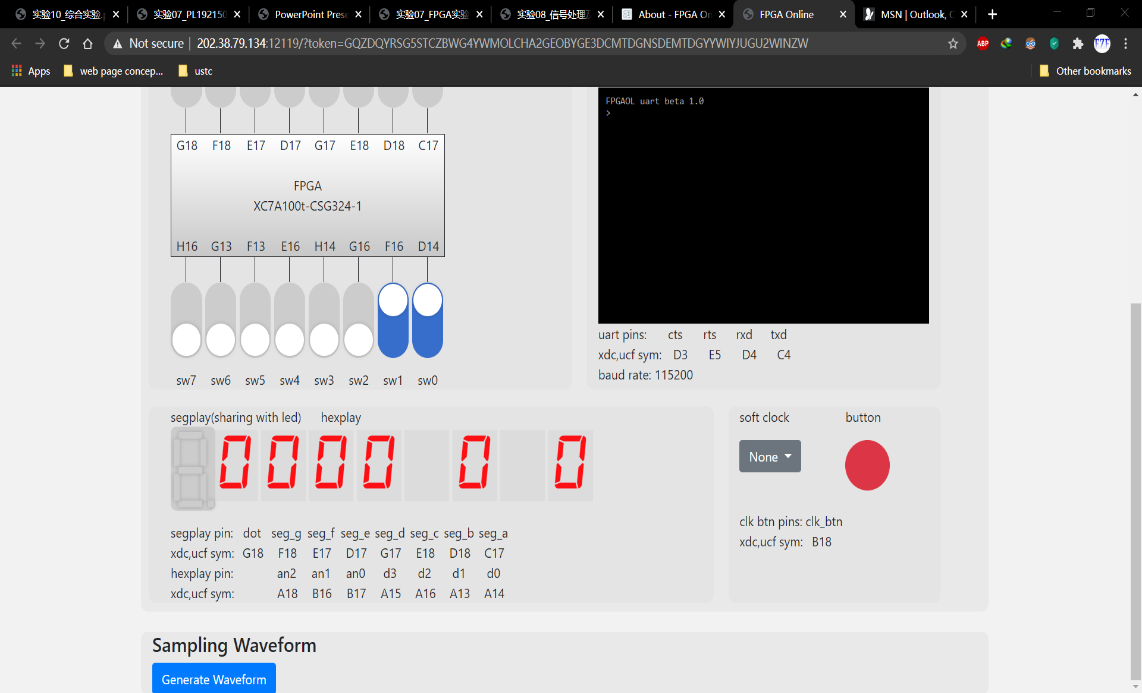
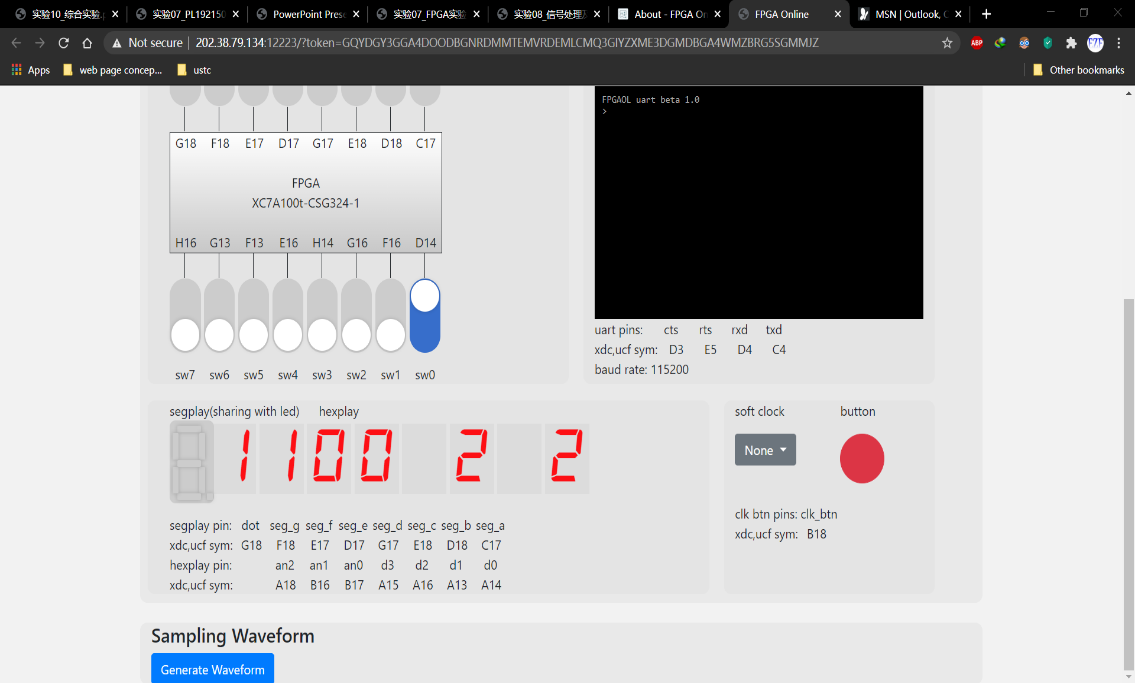
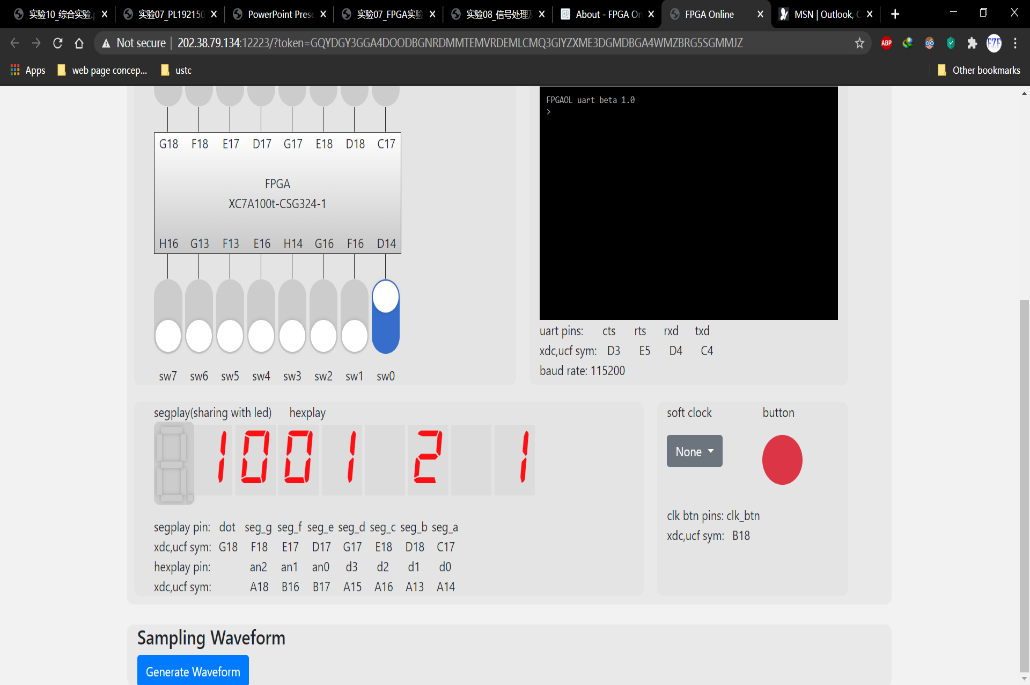
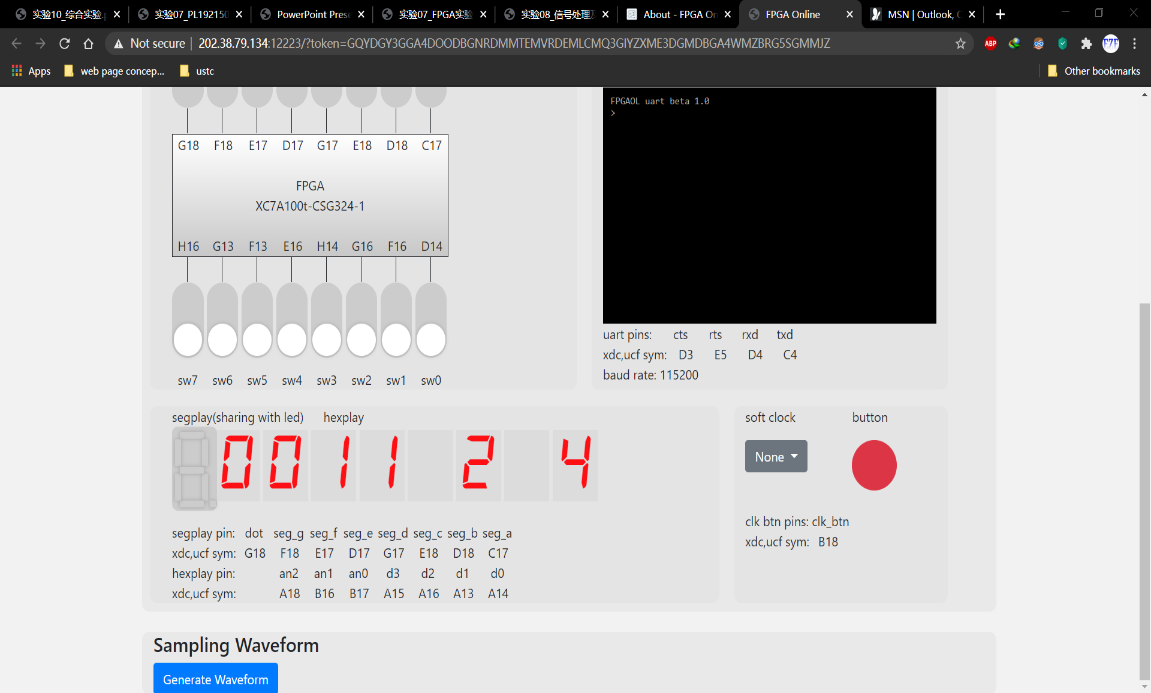
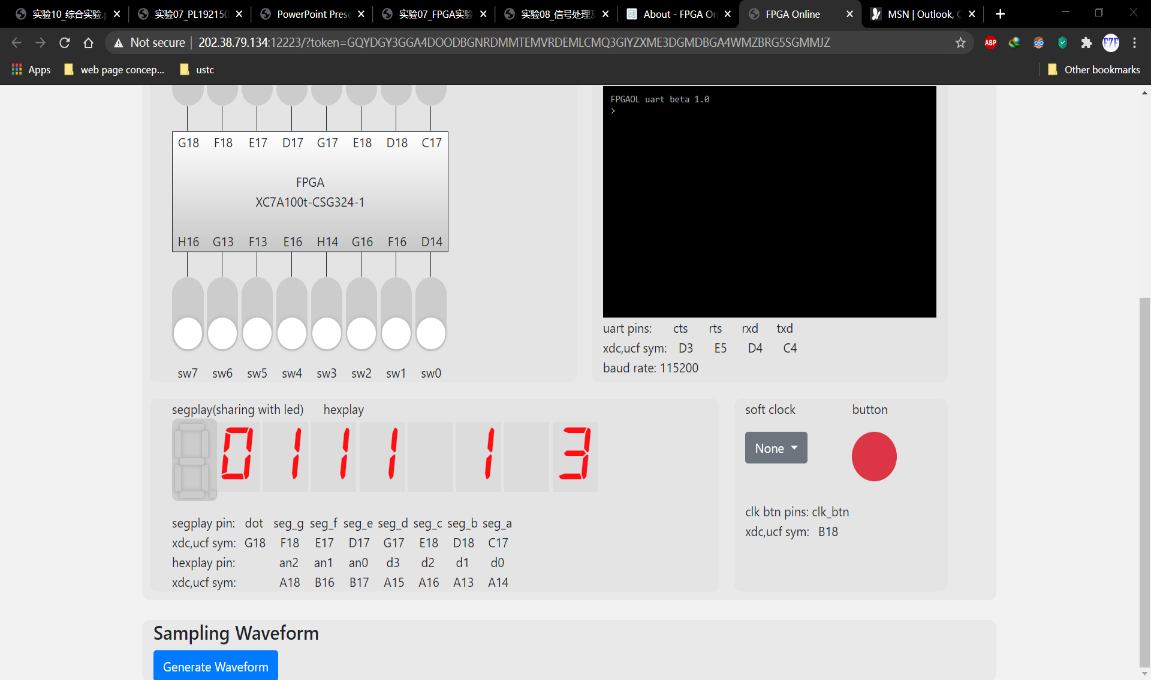
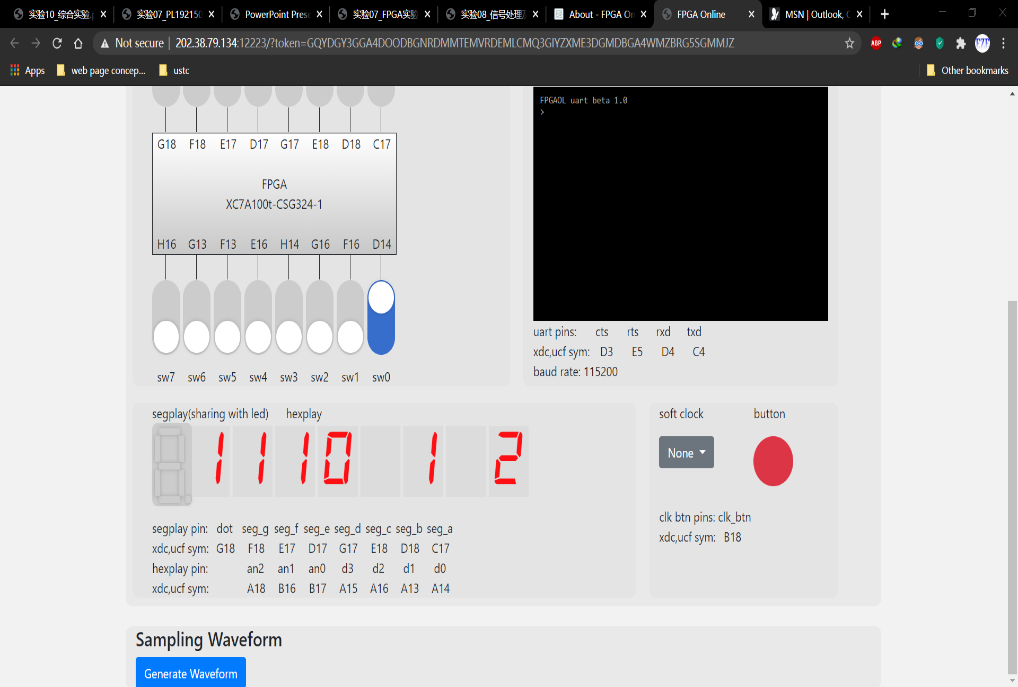
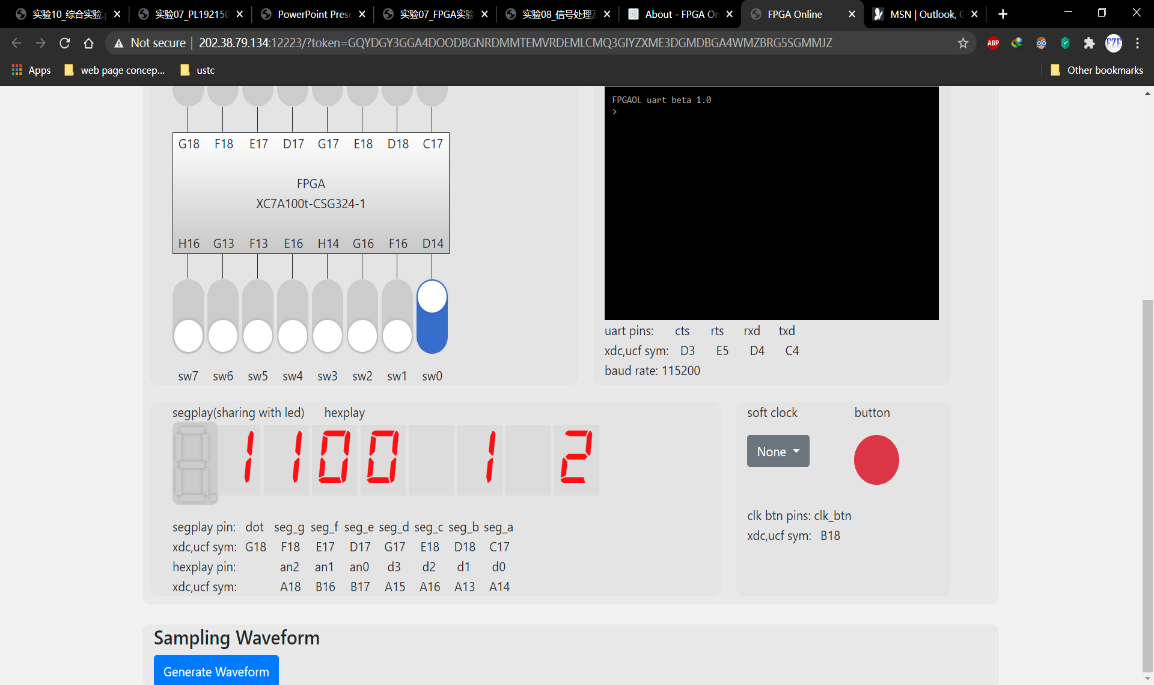
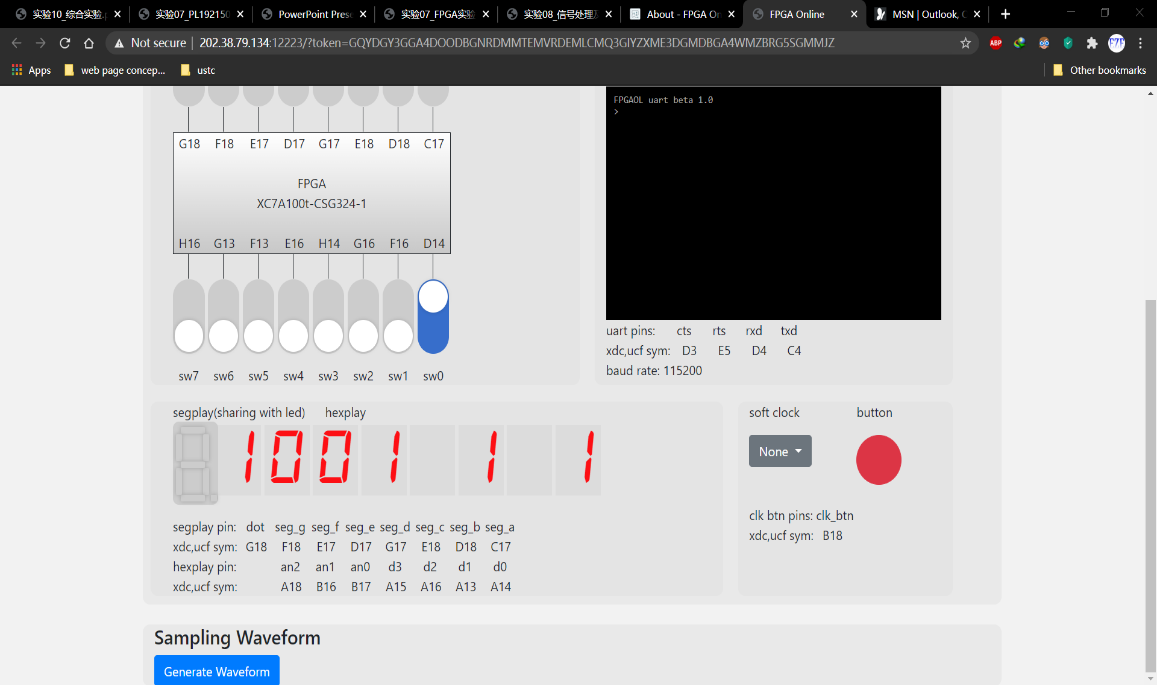
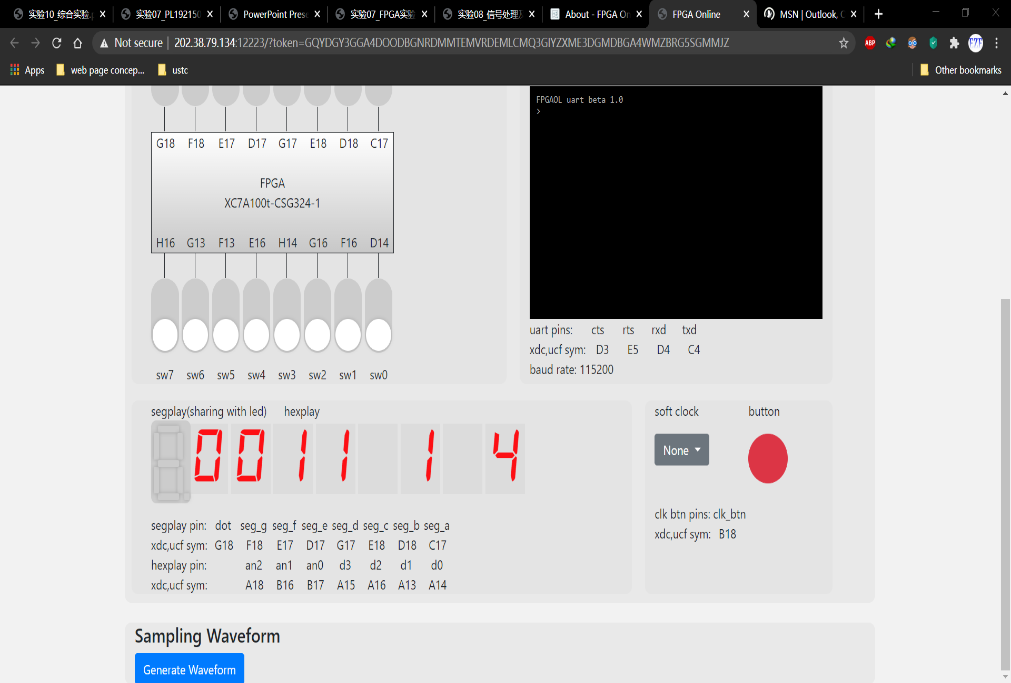
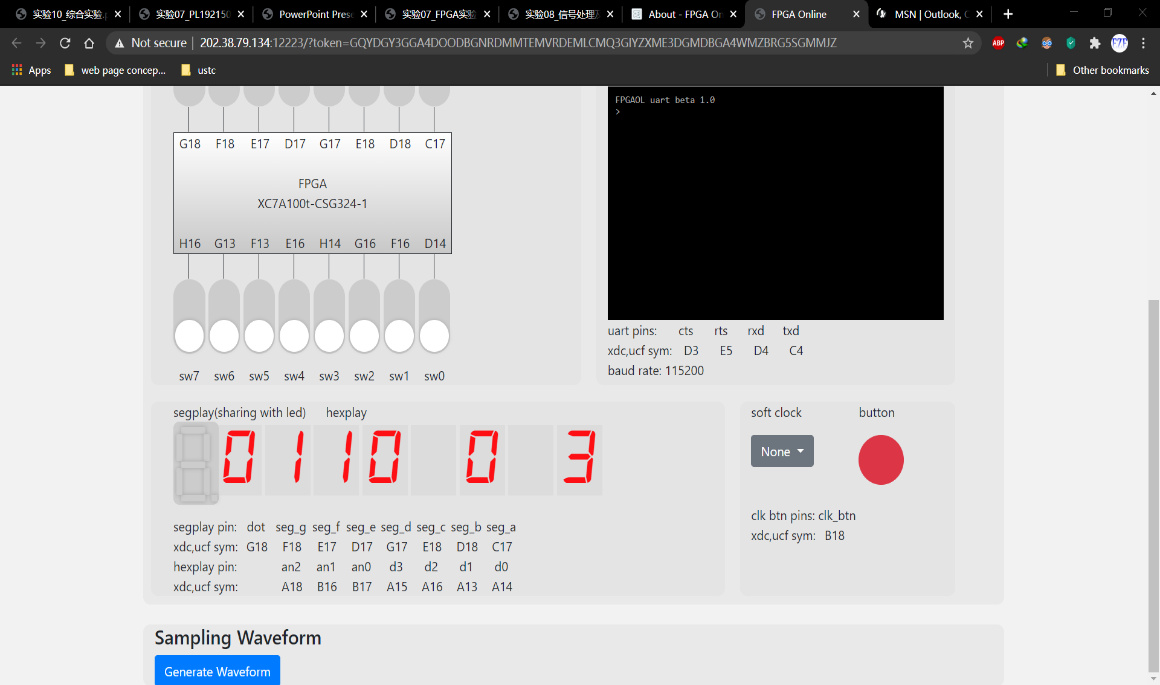
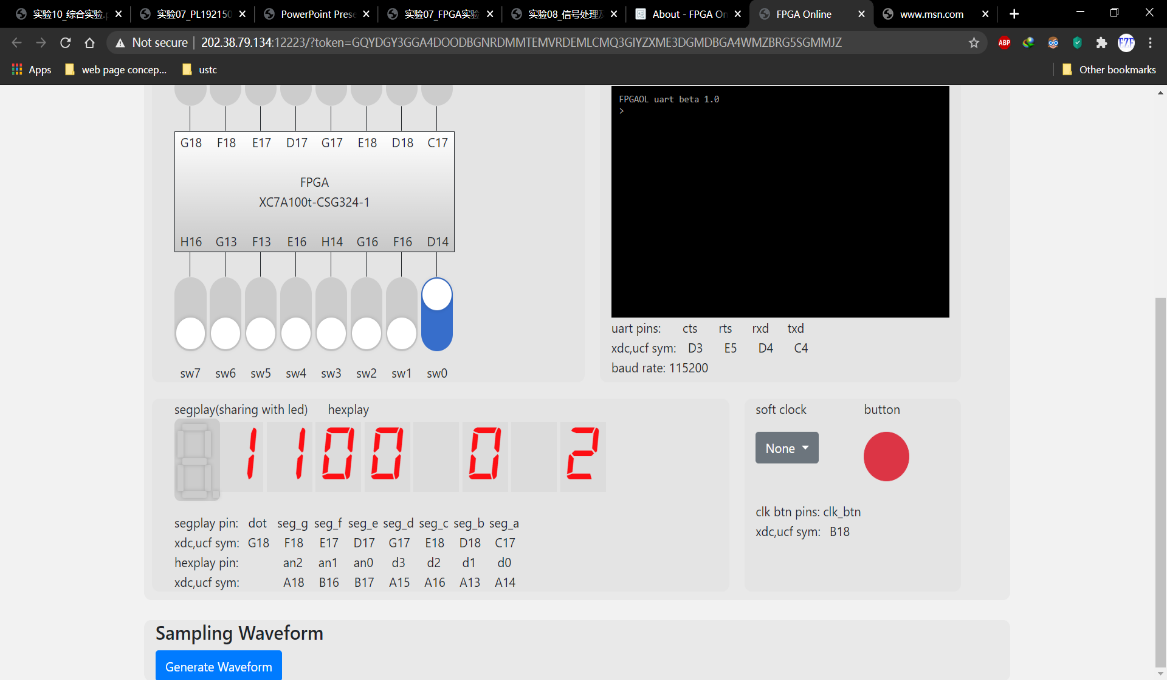
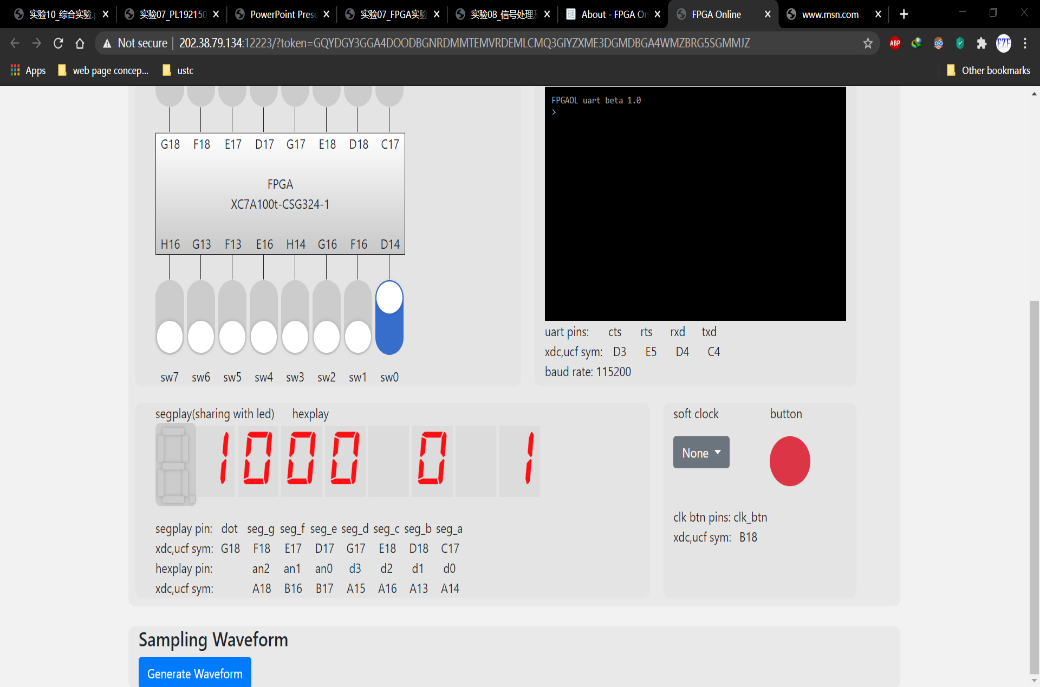
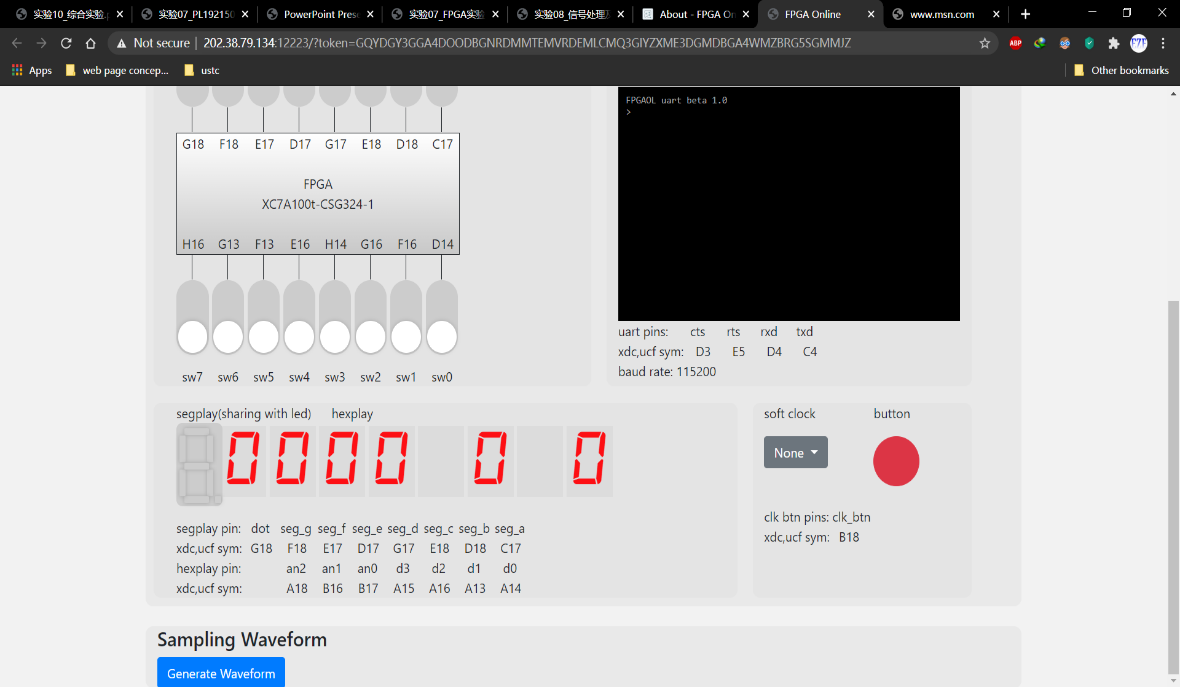
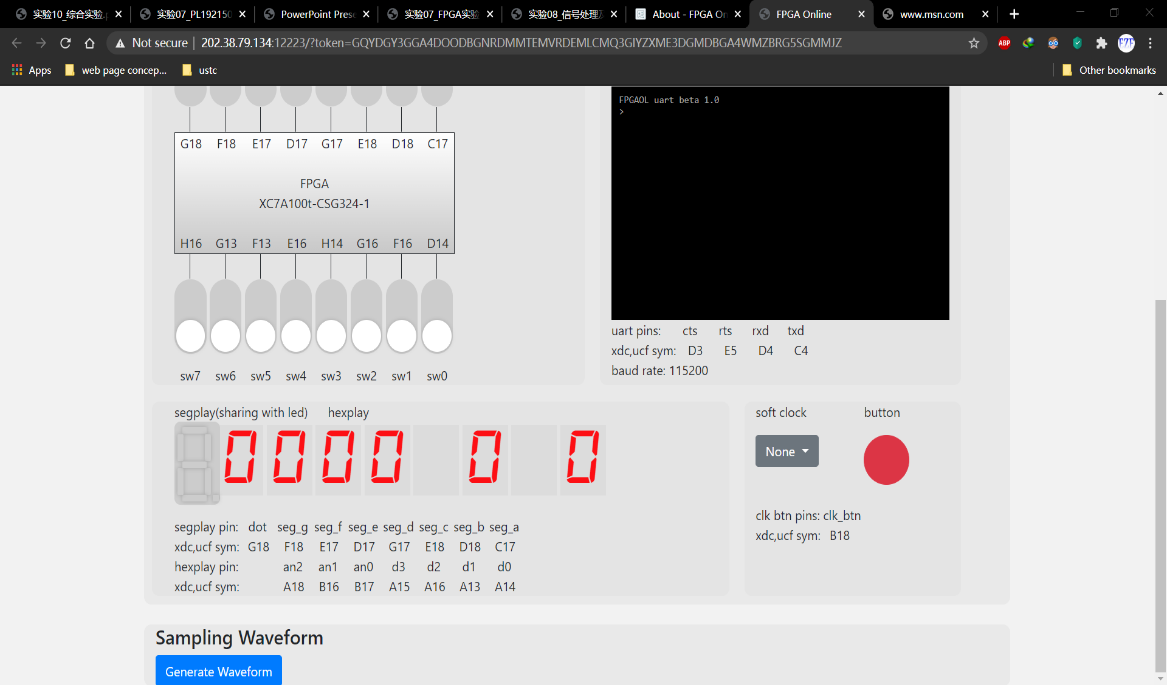
**endmodule**

XDC文件

## 

结果

下面输入为 “0011001110011” ,最后清零（rst）



【总结与思考】

在本次实验中咱们掌握几种常见的信号处理技巧如取信号边沿，也掌握有限状态机的设计方法。