同济大学计算机系

数字逻辑课程实验报告

学　　号 2350752

姓　　名 田思宇

专　　业 计算机科学与技术

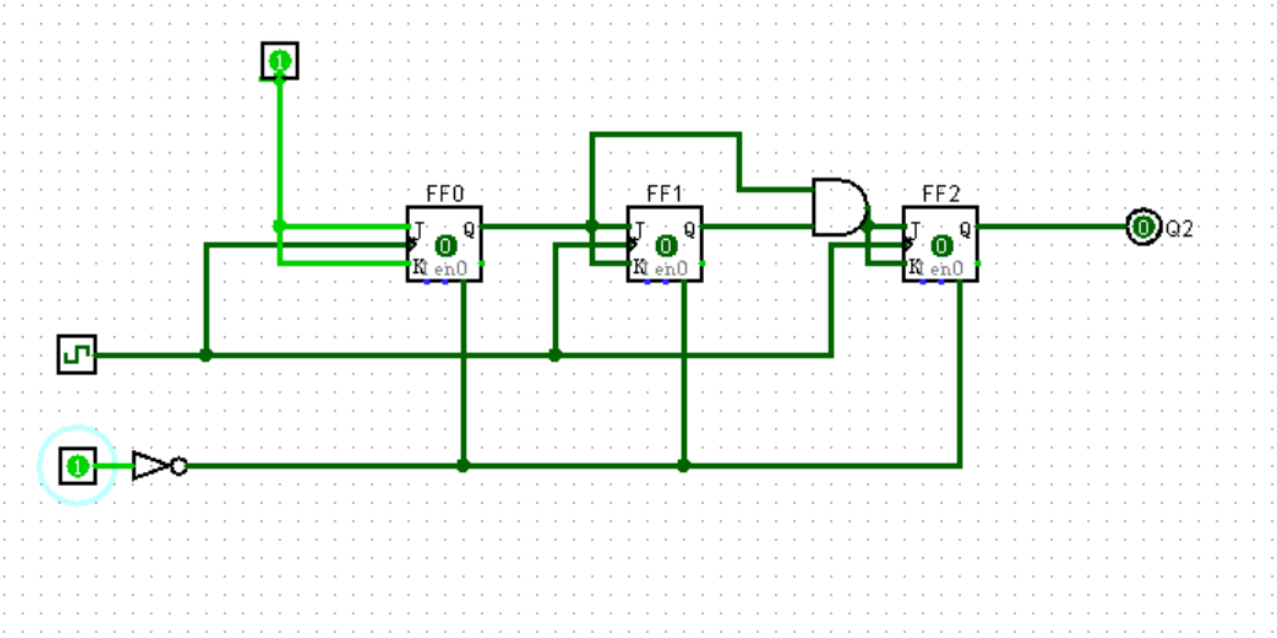
授课老师 张冬冬

一、实验内容

使用 Verilog HDL 语言实现计数器和分频器的设计和仿真
实验一：利用 jk 触发器实例化设计一个模 8 计数器，同时实例化数码管供人观察
实验二：利用计数器原理设计一个可调节参数的分频器

二、硬件逻辑图

（实验步骤中要求用 logisim 画图的实验，在该部分给出 logisim 原理图，否则该部分在实验报告中不用写）



三、模块建模

（该部分要求对实验中建模的所有模块进行功能描述，并列出各模块建模的 verilog 代码）

实验一：模 8 计数器

模块功能：本模块是一个模 8 的计数器，原理是实例化之前设计过的 jk 触发器，实现记录时钟上升沿个数的功能，复位信号是异步低电平有效，和之前的 jk 触发器相同，同时为了可以下板观察，这里还实例化了七位数码管和分频器，这样就可以在板上的数码管观察到当前的记录的次数。

```
module JK_FF(
input CLK,
input J,
input K,
input RST_n,
```

```
output reg Q1,
output reg Q2
);
always @(posedge CLK or negedge RST_n)
begin
if(!RST_n)
begin
Q1<=1'b0;
Q2<=1'b1;
end
else if(J==0&&K==0)
begin
Q1<=Q1;
Q2<=Q2;
end
else if(J==1&&K==0)
begin
Q1<=1'b1;
Q2<=1'b0;
end
else if(J==0&&K==1)
begin
Q1<=1'b0;
Q2<=1'b1;
end
else if(J==1&&K==1)
begin
Q1<=Q2;
Q2<=Q1;
end
end
endmodule
module display7(
input    [3:0] iData,
output    reg [6:0] oData
);
always@(*)
begin
case(iData)
4'b0000: oData=7'b1000000;//0
4'b0001: oData=7'b1111001;//1
4'b0010: oData=7'b0100100;//2
4'b0011: oData=7'b0110000;
4'b0100: oData=7'b0011001;
```

```
4'b0101: oData=7'b0010010;
4'b0110: oData=7'b0000010;
4'b0111: oData=7'b1111000;//7
4'b1000: oData=7'b0000000;//8
4'b1001: oData=7'b0010000;//9
default: oData=7'bxxxxxxx;
endcase
end
endmodule
module Divider (
input I_CLK,   //输入时钟信号，上升沿有效
input rst,
//复位信号，高电平有效
output reg O_CLK   //输出时钟
);
// 参数定义分频倍数
parameter DIVIDE_BY = 100000000;
// 内部计数器
reg [31:0] count = 0;
// 时钟上升沿和复位信号处理
always @(posedge I_CLK ) begin
    if (rst)
    begin
        // 同步复位，高电平有效
        count <= 0;
        O_CLK <= 1'b0; // 复位时输出低电平
    end
    else
    begin
        if (count == (DIVIDE_BY/2 - 1))
        begin
            // 计数器达到分频值减一时，切换输出时钟状态
            O_CLK <= ~O_CLK;
            count <= 0; // 重置计数器
        end
        else
        begin
            count <= count + 1; // 计数器加一
        end
    end
end
endmodule
module Counter8(
input CLK,
```

```
input rst_n,
output [2:0] oQ,
output [6:0] oDisplay
);
wire tand=oQ[0]&oQ[1]; //q0 和 q1 与运算
wire [3:0] temp1={1'b0,oQ};//最高位必须是零，级联一下成为三位的给 light 用
wire tclk;
Divider uut5(CLK,~rst_n,tclk);
JK_FF uut1(tclk,1'b1,1'b1,rst_n,oQ[0]);
JK_FF uut2(tclk,oQ[0],oQ[0],rst_n,oQ[1]);
JK_FF uut3(tclk,tand,tand,rst_n,oQ[2]);
display7 uut4(temp1,oDisplay);
endmodule
```

实验二：分频器

模块功能：本模块根据计数器的原理来实现对时钟的分频，具体采用的是行为级语言描述，首先@(posedge I_CLK)确保是时钟上升沿有效，如果此时复位信号为高电平，则复位（同步复位），DIVIDE_BY 为参数可以修改，其含义是分频倍数，时钟上升沿每来临一次寄存器就+1，如果寄存器此时的数值是 DIVIDE_BY/2 – 1，那么就翻转输出时钟，这样就实现了输出时钟的频率是输入时钟的 1/ DIVIDE_BY。

```
module Divider (
input I_CLK,   //输入时钟信号，上升沿有效
input rst,
//复位信号，高电平有效
output O_CLK   //输出时钟
);
//  参数定义分频倍数
parameter DIVIDE_BY = 20;

//  内部计数器
reg [31:0] count = 0;
reg temp=0;
assign O_CLK =temp;
//  时钟上升沿和复位信号处理
always @(posedge I_CLK ) begin
      if (rst)
      begin
            //  同步复位，高电平有效
            count <= 0;
            temp <= 1'b0; //  复位时输出低电平
      end
      else
```

```
    begin
        if (count == (DIVIDE_BY/2 - 1))
        begin
            // 计数器达到分频值减一时，切换输出时钟状态
            temp <= ~temp;
            count <= 0; // 重置计数器
        end
        else
        begin
            count <= count + 1; // 计数器加一
        end
    end
end
endmodule
```

四、测试模块建模

（要求列写各建模模块的 test bench 模块代码）

实验一：

```
`timescale 1ns / 1ps
module Counter8_tb;
// 输入和输出信号
reg CLK;
reg rst_n;
wire [2:0] oQ;
wire [6:0] oDisplay;
// 实例化 Counter8 模块
Counter8 uut (CLK,rst_n,oQ,oDisplay);
// 时钟信号生成
initial begin
    CLK = 0;
    forever #10 CLK = ~CLK; // 产生 50MHz 的时钟信号，周期为 20ns
end
// 测试序列
initial begin
    // 初始化
    rst_n = 0; // 复位信号低电平有效
    #100;       // 保持复位一段时间
    rst_n = 1; // 释放复位信号

    // 观察一段时间的输出
    #1000; // 观察 100 个周期的输出
```

```
    // 再次触发复位
    rst_n = 0;
    #100;
    rst_n = 1;
    // 结束仿真
    #1000;
end
endmodule
```

实验二：

```
`timescale 1ns / 1ps
module Divider_tb;
// 输入和输出信号
reg I_CLK;
reg Rst;
wire O_CLK;

// 实例化分频器模块
Divider uut (I_CLK,Rst,O_CLK);

// 时钟信号生成
initial begin
    I_CLK = 0;
    forever #10 I_CLK = ~I_CLK; // 产生 50MHz 的时钟信号，周期为 20ns
end

// 测试序列
initial begin
    // 初始化
    Rst = 1; // 复位信号高电平
    #40;      // 保持复位一段时间
    Rst = 0; // 释放复位信号

    // 观察一段时间的输出
    #500; // 观察 20 个周期的输出

    // 再次触发复位
    Rst = 1;
    #20;
    Rst = 0;
    // 结束仿真
    #100;
end
endmodule
```

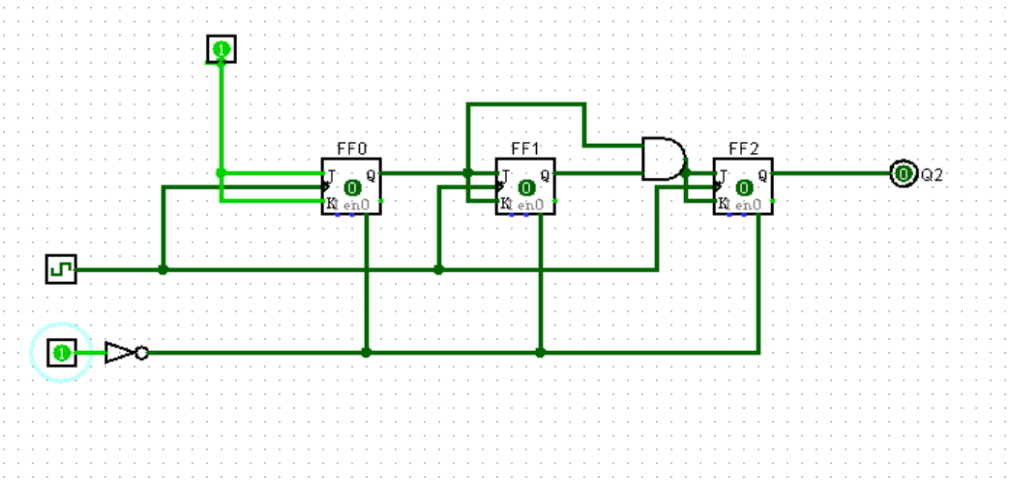
五、实验结果

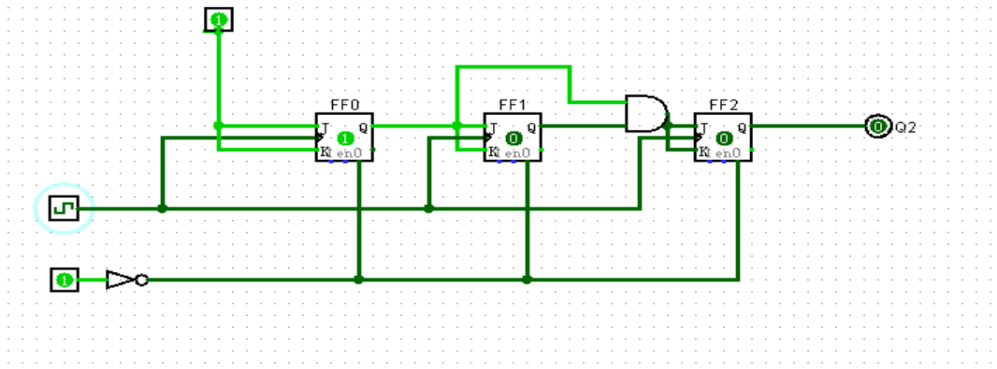
（该部分可截图说明，要求 logisim 逻辑验证图、modelsim 仿真波形图、以及下板后的实验结果贴图（实验步骤中没有下板要求的实验，不需要下板贴图））

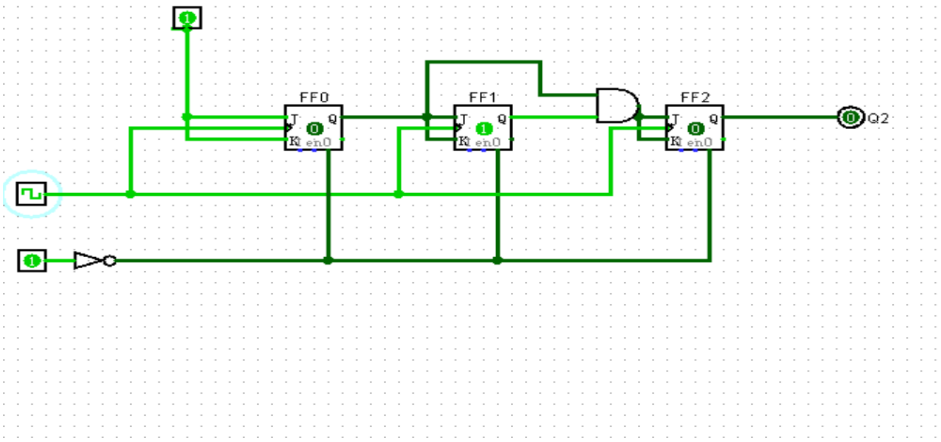
实验一：

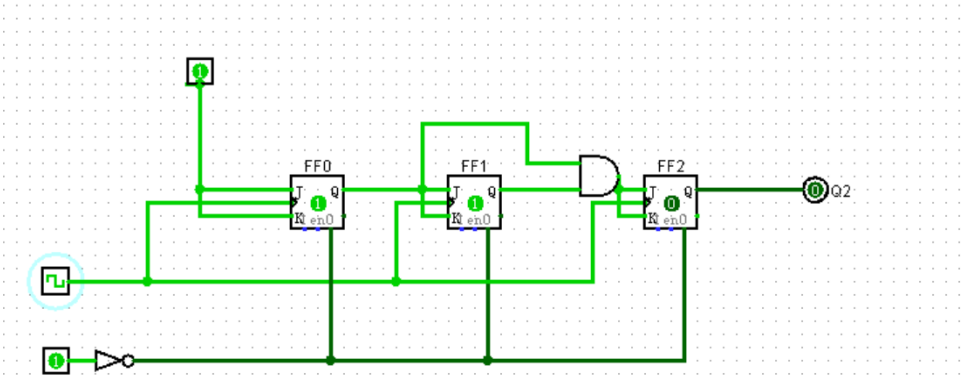
Logisim 逻辑验证图：

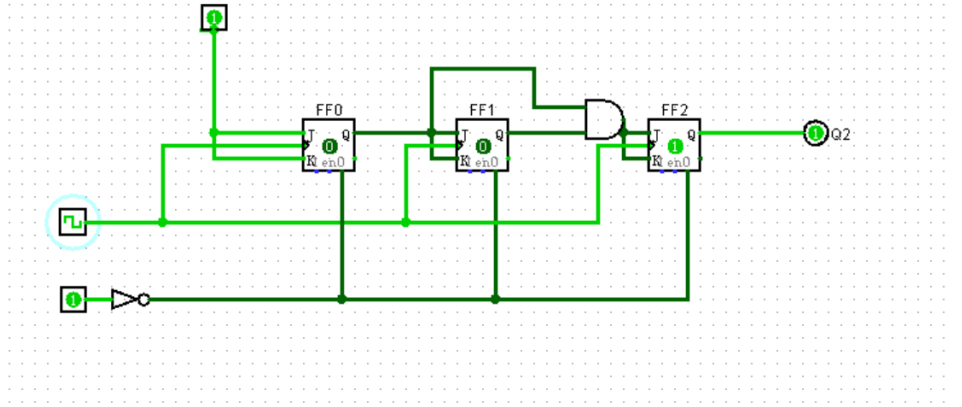
这里复位信号是高电平（无效状态），下图展示了时钟不断变化时计数器的存储数据的变化



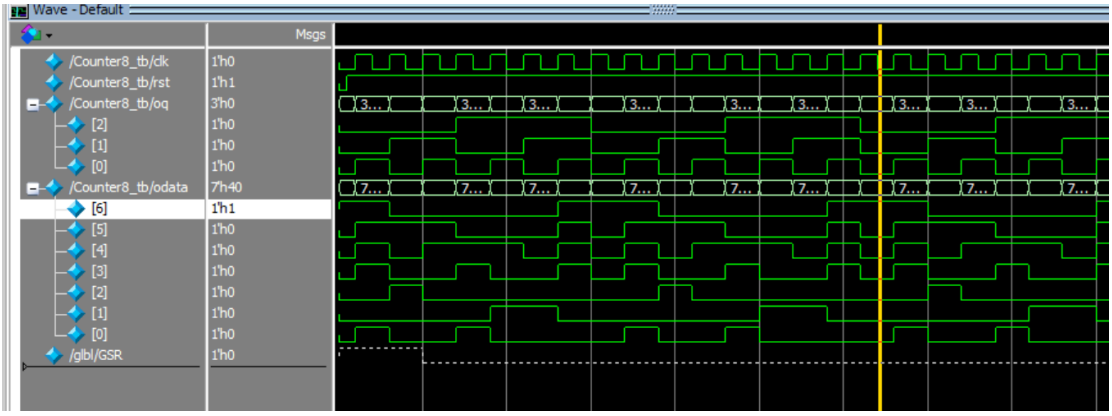






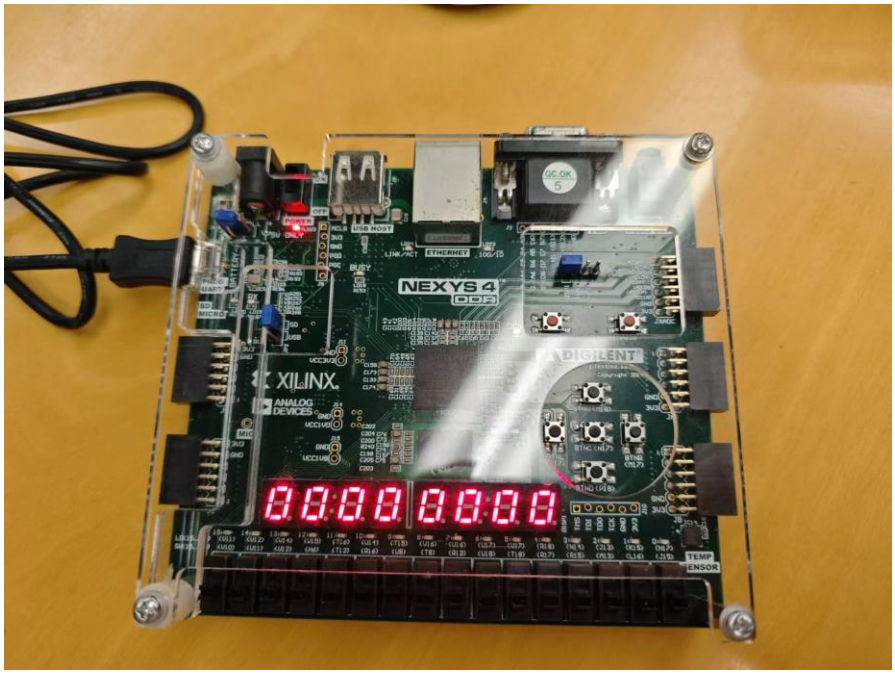


modelsim 仿真波形图：

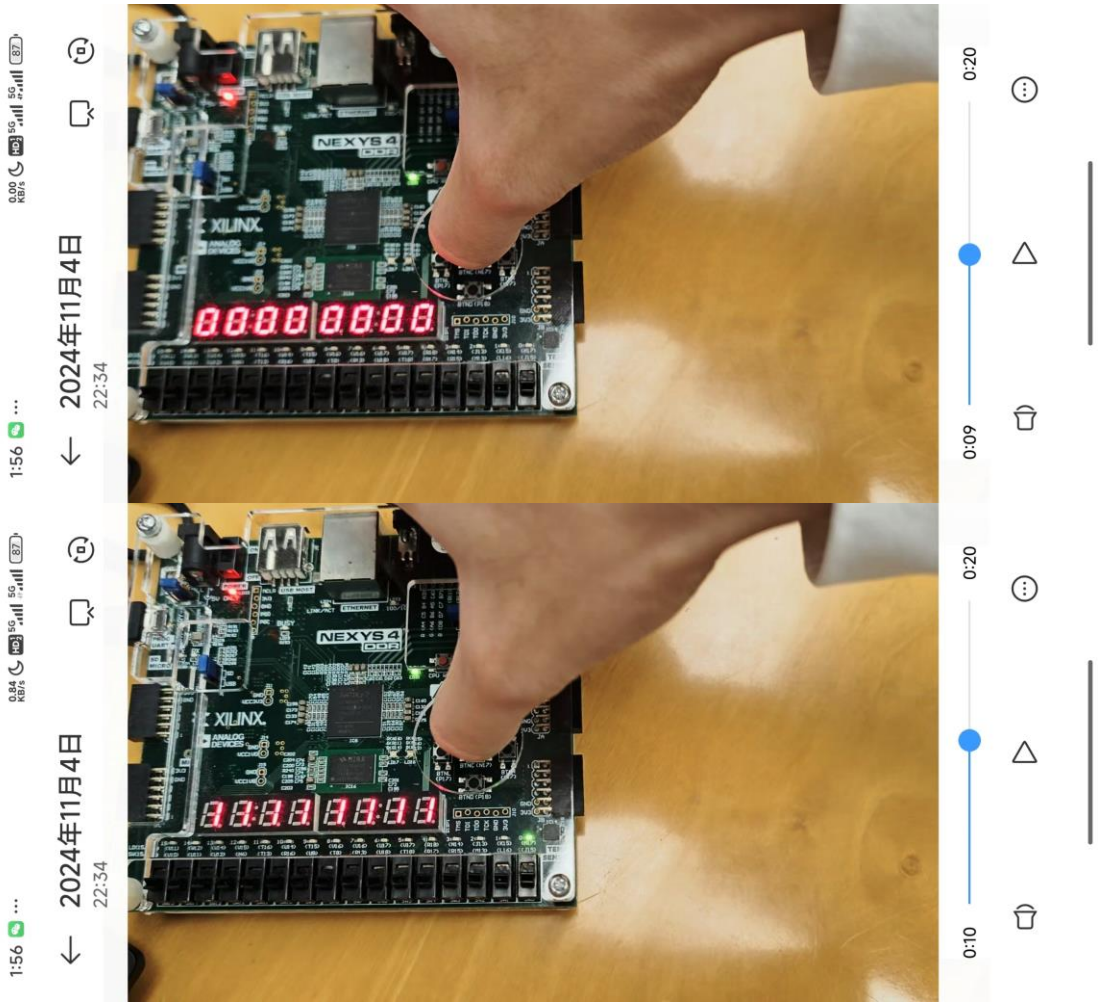


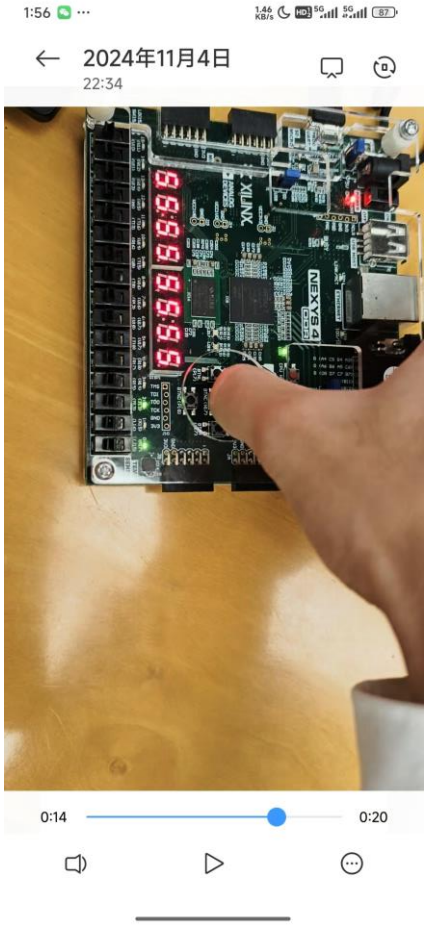
下板结果：

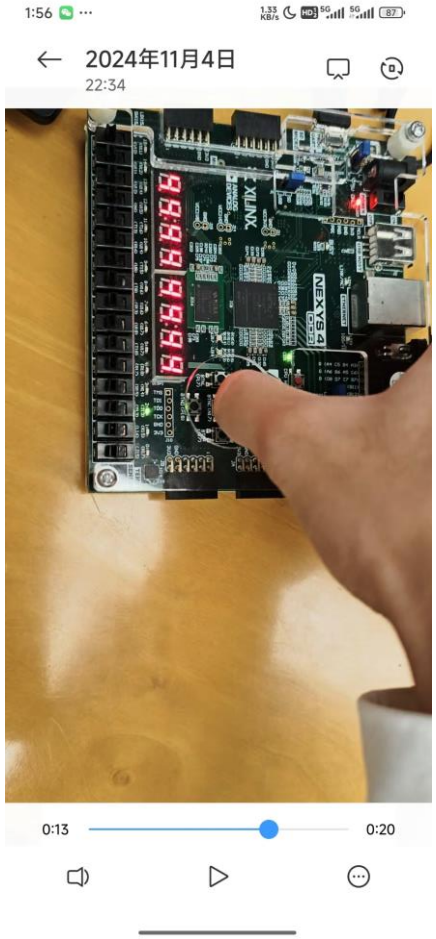
1、因为复位信号是异步低电平有效，所以不按住按钮是不会有显示的

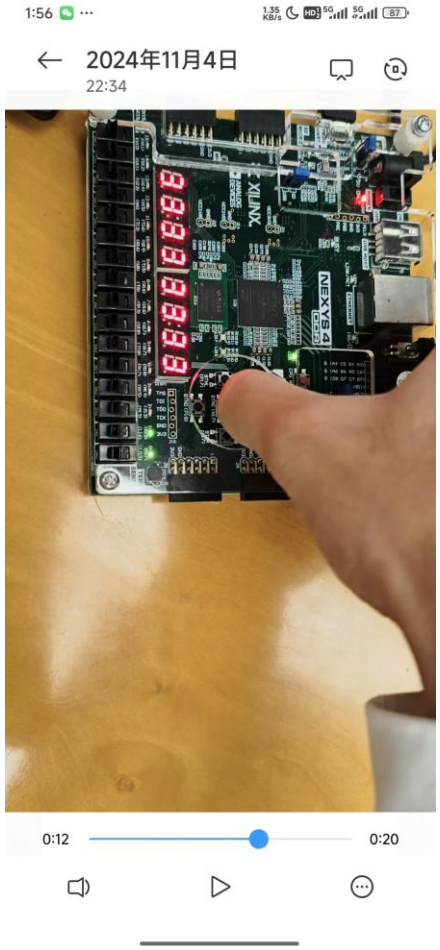


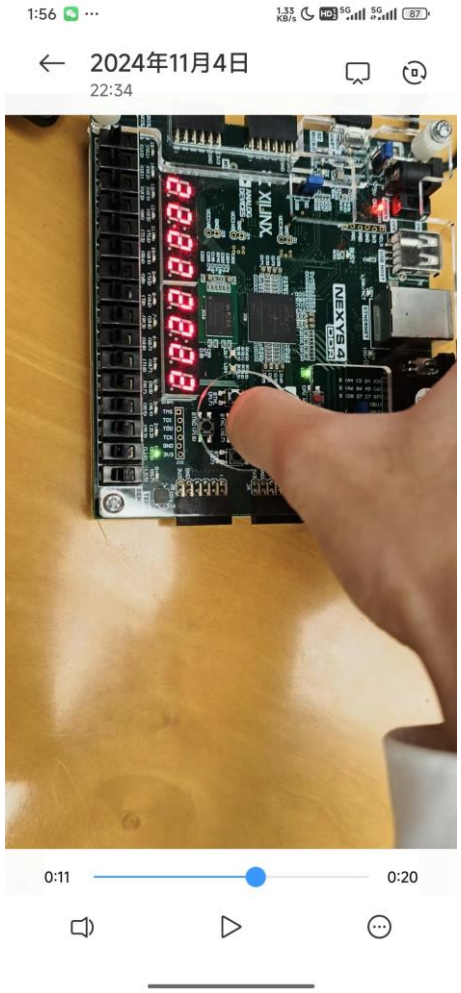
2、为了体现出板上时钟的连续变化，我录了视频然后对视频截屏，这样就能体现下板结果的有效性了

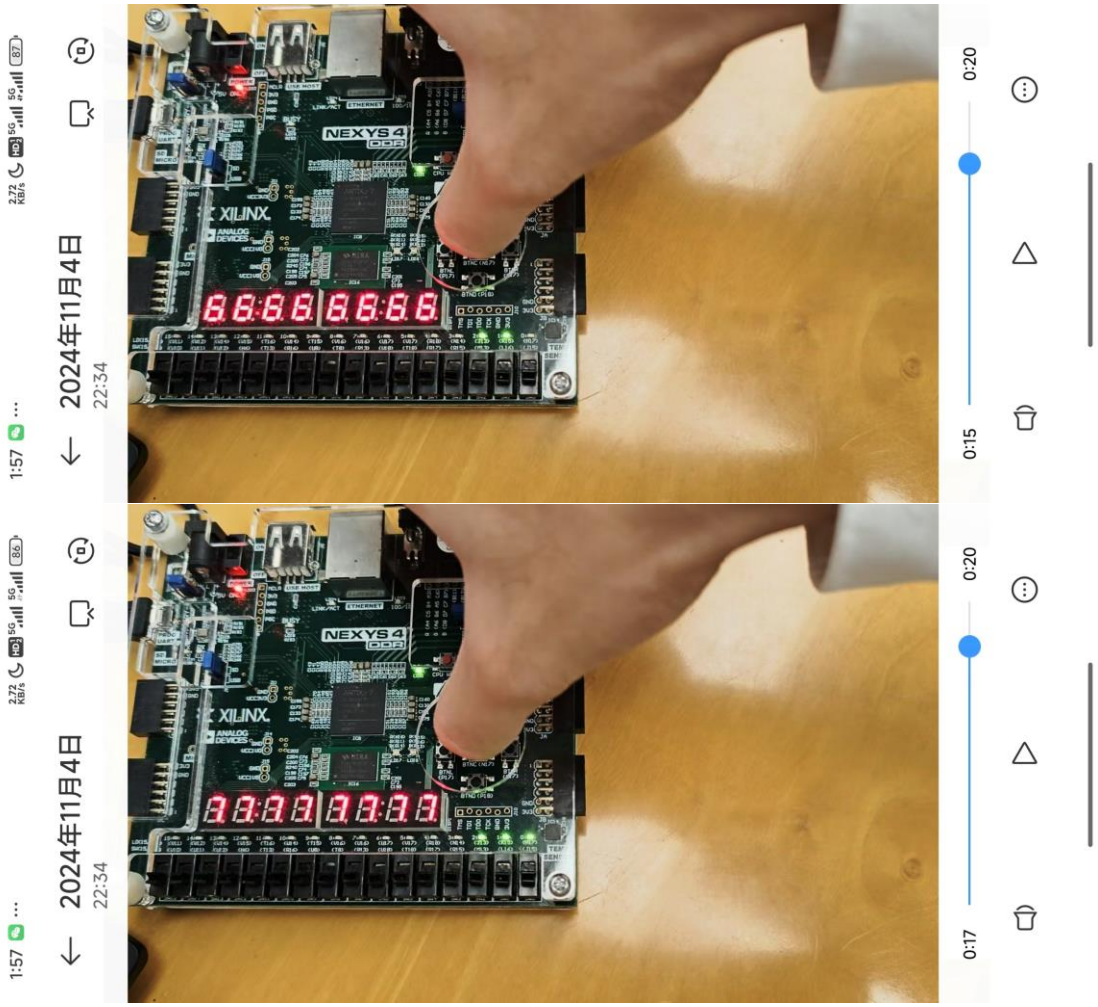



1:56
2024年11月4日
22:34
0:14
0:20


1:56
2024年11月4日
22:34
0:13
0:20


1:56
2024年11月4日
22:34
0:12
0:20

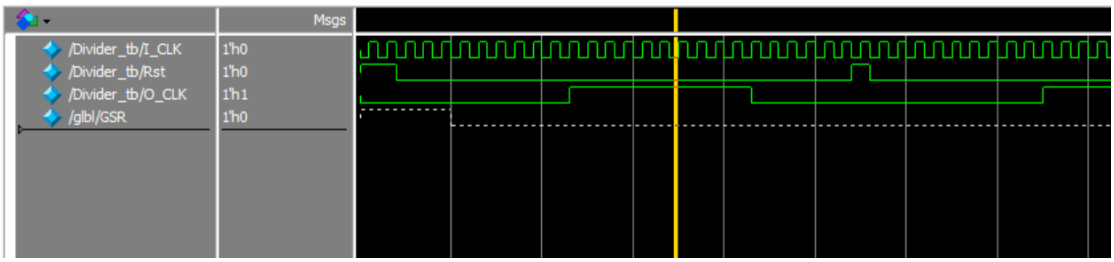

1:56
2024年11月4日
22:34
0:11
0:20



实验二：

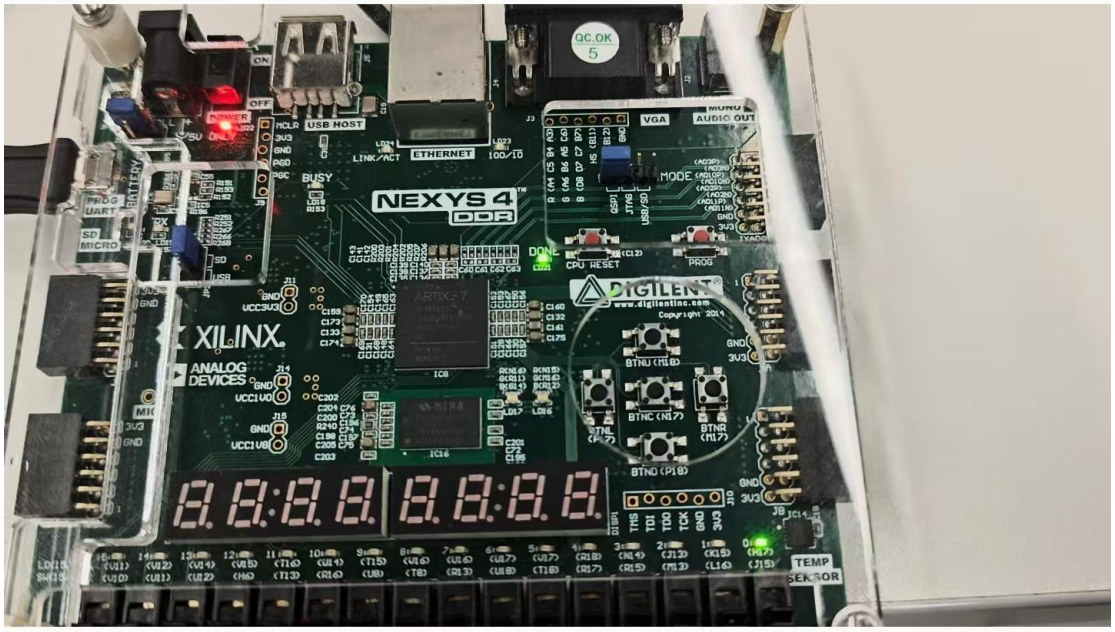
Modolsim 仿真波形图：

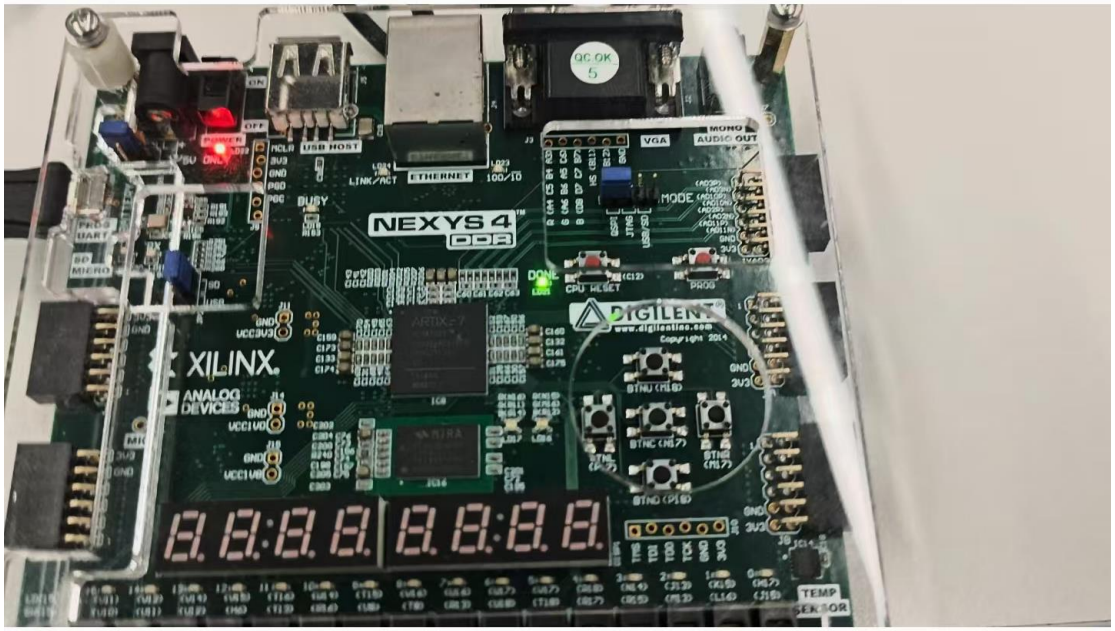
实验课上讲过输出时钟信号应该先是高电平，但是这样做提交了两次在 mips 网站上都显示错误，根据 mips 上面反馈的错误结果，应该输出时钟信号先是低电平，所以这里是修改成低电平的仿真波形图。



下板结果：

下面的小灯一秒闪一下


QC.OK
5
NEXYS 4
DDR
XILINX
DIGILENT
ANALOG DEVICES
ETHERNET
USB HOST
VGA
AUDIO OUT
CPU RESET
PROG
BTNU (M18)
BTNC (N17)
BTNL (P17)
BTNR (M17)
BTND (P18)
TEMP SENSOR


QC.OK
5
NEXYS 4
DDR
XILINX
DIGILENT
ANALOG DEVICES
ETHERNET
USB HOST
VGA
AUDIO OUT
CPU RESET
PROG
BTNU (M18)
BTNC (N17)
BTNL (P17)
BTNR (M17)
BTND (P18)
TEMP SENSOR