同济大学计算机系

数字逻辑课程综合实验报告

学　　号 2352595

姓　　名 张嘉麟

专　　业 计算机科学与技术

授课老师 郭玉臣

一、实验内容

本次实验包括四个模块，分别是同步 D 触发器、异步 D 触发器、JK 触发器和 PC 寄存器。

Synchronous D Flip-Flop (同步 D 触发器)
该模块实现一个同步 D 触发器，触发器的输出状态在时钟上升沿发生变化。RST_n 为低电平复位信号，当复位有效时，Q1 设为 0，Q2 设为 1；否则 Q1 和 Q2 分别等于输入 D 和 ~D。仅在时钟上升沿触发变化，且复位信号的响应也是同步的。通过时钟信号控制，模拟同步电路的行为。

Asynchronous D Flip-Flop (异步 D 触发器)
该模块实现一个异步 D 触发器，输出状态在时钟上升沿或复位信号的下降沿发生变化。RST_n 为低电平复位信号，复位有效时，Q1 被设为 0，Q2 被设为 1；否则 Q1 和 Q2 根据输入 D 和 ~D 的值更新。触发器的输出在时钟上升沿或复位信号下降沿更新。与同步 D 触发器不同，异步 D 触发器在时钟或复位信号发生变化时立即响应，因此适用于对复位要求较高的电路。

JK Flip-Flop (JK 触发器)
该模块实现一个 JK 触发器，输出状态在时钟上升沿或复位信号下降沿发生变化。J 和 K 控制触发器的状态转换，当 J 和 K 为不同值时，分别执行设置或清除操作；当两者相同为 1 时，触发器的输出取反。通过 J 和 K 的组合控制触发器的行为，支持保持状态、设定、清除以及翻转等操作。该模块适合实现复杂的状态转换电路。

PC Register (PC 寄存器)
该模块实现一个 32 位的 PC 寄存器，输出状态在时钟上升沿或复位信号上升沿发生变化。复位信号有效时，将 data_out 设为 0；在有效使能信号 ena 控制下，data_out 根据输入 data_in 的值更新。PC 寄存器用于保存当前指令地址，便于程序顺序执行。在使能信号的控制下，实现 PC 寄存器的加载或保持，适合用于处理器指令控制流。

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）

（1）Synchronous_D_FF

```
module Synchronous_D_FF(
    input wire CLK,
    input wire D,
```

```
    input wire RST_n,
    output reg Q1,
    output reg Q2
);
    always @ (posedge CLK) begin
        if (!RST_n) begin
            Q1 <= 0;
            Q2 <= 1;
        end else begin
            Q1 <= D;
            Q2 <= ~D;
        end
    end
endmodule
```

（2）Asynchronous_D_FF

```
module Asynchronous_D_FF(
    input CLK,
    input D,
    input RST_n,
    output reg Q1,
    output reg Q2
);
    always@ (posedge CLK or negedge RST_n)
    begin
        if(!RST_n)
        begin
            Q1 = 0;
            Q2 = 1;
        end
        else
        begin
            Q1 = D;
            Q2 = !D;
        end
    end
endmodule
```

（3）JK_FF

```
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 <= 0;
            Q2 <= 1;
        end
        else
        begin
            case({J, K})
            2'b00:
            begin
                Q1 <= Q1;
                Q2 <= Q2;
            end
            2'b01:
            begin
                Q1 <= 0;
                Q2 <= 1;
            end
            2'b10:
            begin
                Q1 <= 1;
                Q2 <= 0;
            end
            2'b11:
            begin
                Q1 <= !Q1;
                Q2 <= !Q2;
            end
            endcase
        end
    end
endmodule
```

（4）pcreg

```
module pcreg(
    input clk,
    input rst,
    input ena,
    input [31:0] data_in,
    output reg [31:0] data_out
```

```
    );

    always @ (posedge clk or posedge rst) begin
        if (rst)
            data_out <= 0;
        else if (ena)
            data_out <= data_in;
    end
endmodule

module top();//实例化
    reg clk;
    reg rst;
    reg ena;
    reg [31:0] data_in;
    wire [31:0] data_out;
    // Instantiate the pcreg module within the top module
    pcreg pcreg_inst (
        .clk(clk),
        .rst(rst),
        .ena(ena),
        .data_in(data_in),
        .data_out(data_out)
    );
    // Initialize signals and simulate clock pulse
    initial begin
        data_in = 32'b11111111_00000000_11110000_00001111;
        rst = 0;
        ena = 1;
        clk = 1;
        #5 clk = ~clk; // Generate clock pulse
        #5 clk = ~clk;
        data_in = 32'b11111111_11111111_11111111_11111111;
        #5 clk = ~clk;
        #5 clk = ~clk;
        rst = 1;
        #5 clk = ~clk;
        rst = 0;
        ena = 0;
        data_in = 32'b11111111_00000000_11110000_00001111;
        #5 clk = ~clk;
        ena = 1;
        #5 clk = ~clk;
    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）

（1）Synchronous_D_FF

```
module Synchronous_D_FF_tb();
    reg CLK;
    reg D;
    reg RST_n;
    wire Q1;
    wire Q2;
    // 时钟生成
    always #5 CLK = ~CLK;
    // 初始设置
    initial begin
        // 初始化信号
        CLK = 0;
        D = 0;
        RST_n = 1;
        // 等待一个时钟周期
        #10;
        // 测试不同输入
        D = 1;
        #10;
        // 复位信号
        RST_n = 0;
        #10;
        RST_n = 1;
        #10;
        // 结束仿真
        $stop;
    end
    // 实例化被测模块
    Synchronous_D_FF Synchronous_D_FF_inst (
        .CLK(CLK),
        .D(D),
        .RST_n(RST_n),
        .Q1(Q1),
        .Q2(Q2)
    );
endmodule
```

（2）Asynchronous_D_FF

```
module Asynchronous_D_FF_tb();
    reg CLK;
    reg D;
    reg RST_n;
    wire Q1;
    wire Q2;

    initial
    begin
        RST_n = 1;
        D = 0;
        #5;
        CLK = 1;
        #5;
        CLK = 0;
        #5;
        D = 1;
        #5;
        CLK = 1;
        #5;
        CLK = 0;
        #5;
        RST_n = 0;
        #5;
        CLK = 1;
        #5;
        RST_n = 1;
        CLK = 0;
        #5;
        CLK = 1;
    end

    Asynchronous_D_FF Asynchronous_D_FF_inst(CLK, D, RST_n, Q1, Q2);

endmodule
```

（3）JK_FF

```
module JK_FF_tb();
    reg CLK;
    reg J;
    reg K;
    reg RST_n;
    wire Q1;
```

```
    wire Q2;

    // Instantiate the JK_FF module
    JK_FF uut (
        .CLK(CLK),
        .J(J),
        .K(K),
        .RST_n(RST_n),
        .Q1(Q1),
        .Q2(Q2)
    );

    // Clock generation
    initial begin
        CLK = 0; // Initialize clock
        forever #5 CLK = ~CLK; // Toggle clock every 5 time units
    end

    // Test sequence
    initial begin
        // Initialize inputs
        J = 0;
        K = 0;
        RST_n = 0; // Activate reset
        #10; // Wait for 10 time units
        // Release reset
        RST_n = 1;
        #10; // Wait for 10 time units
        // Test case 1: Set (J=1, K=0)
        J = 1; K = 0;
        #10; // Wait for 10 time units
        // Test case 2: Reset (J=0, K=1)
        J = 0; K = 1;
        #10; // Wait for 10 time units
        // Test case 3: Toggle (J=1, K=1)
        J = 1; K = 1;
        #10; // Wait for 10 time units
        // Additional tests can be added here
        // Finish simulation
        $finish;
    end
endmodule
```

（4）pcreg

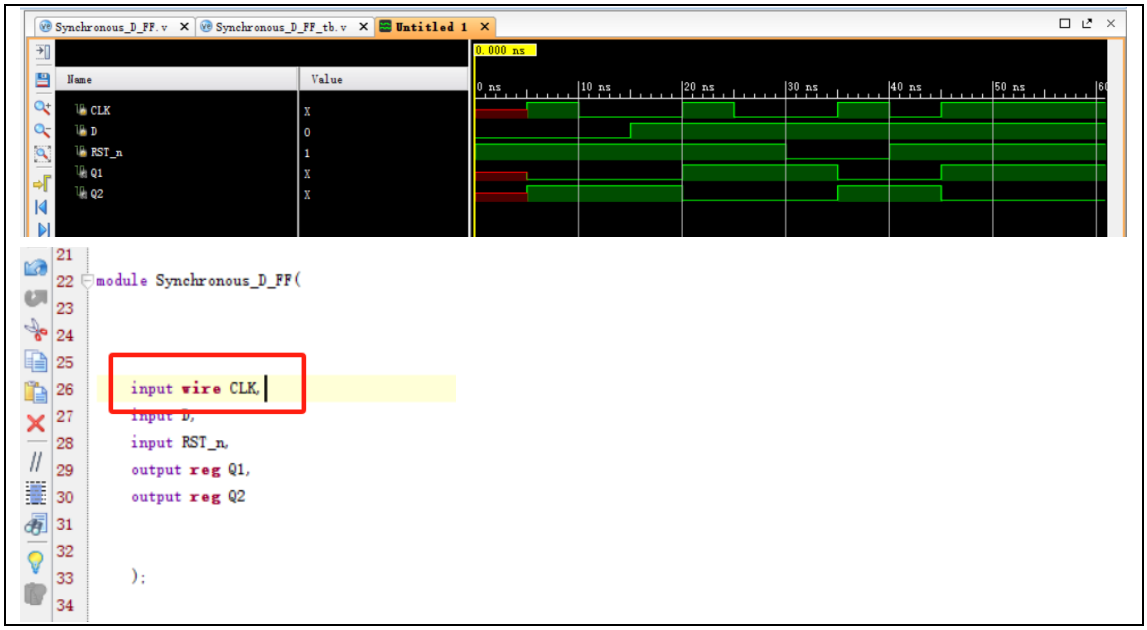
```
module pcreg_tb();
    reg clk;
    reg rst;
    reg ena;
    reg [31:0] data_in;
    wire [31:0] data_out;

    initial
    begin
        data_in = 32'b11111111_00000000_11110000_00001111;
        rst = 0;
        ena = 1;
        clk = 1;
        #5;
        clk = 0;
        #5;
        data_in = 32'b11111111_11111111_11111111_11111111;
        #5;
        clk = 1;
        #5;
        clk = 0;
        #5;
        rst = 1;
        #5;
        rst = 0;
        ena = 0;
        #5;
        data_in = 32'b11111111_00000000_11110000_00001111;
        #5;
        clk = 1;
        #5;
        clk = 0;
        ena = 1;
        #5;
        clk = 1;
    end
    pcreg pcreg_inst(clk, rst, ena, data_in, data_out);
endmodule
```

五、实验结果

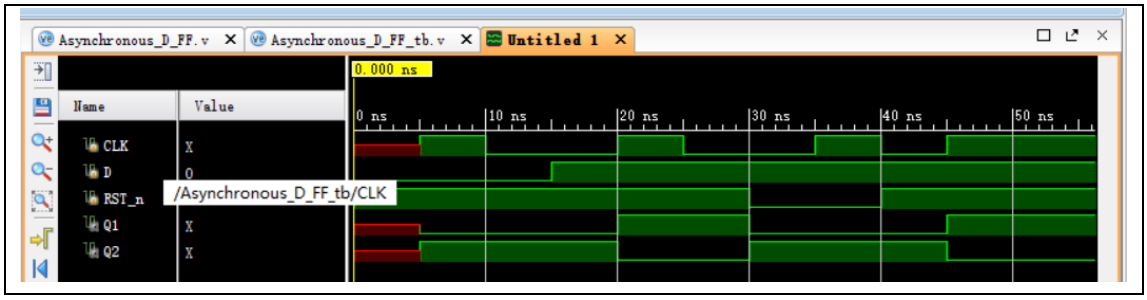
（该部分可截图说明，要求 logisim 逻辑验证图、modelsim 仿真波形图、以及下板后的实验结果贴图（实验步骤中没有下板要求的实验，不需要下板贴图））

1. modelsim 仿真波形图

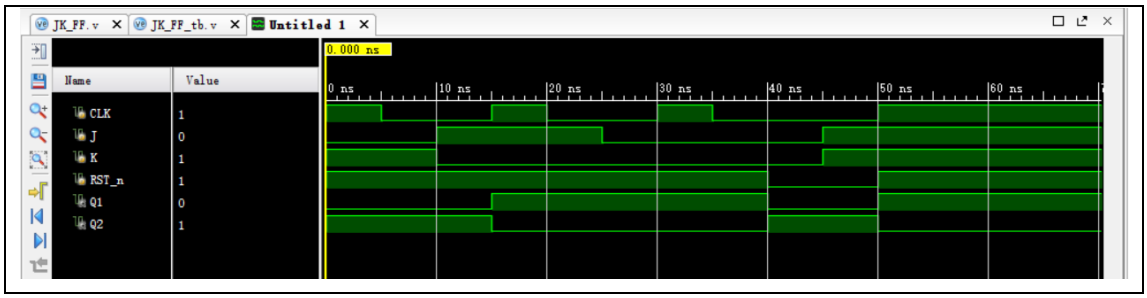
（1）Synchronous_D_FF



（2）Asynchronous_D_FF

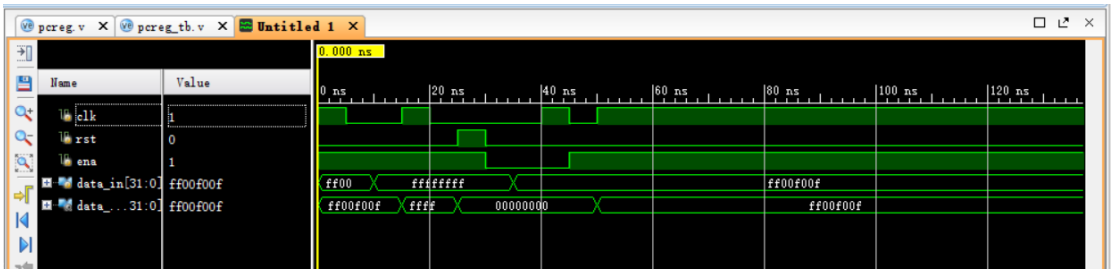


（3）JK_FF

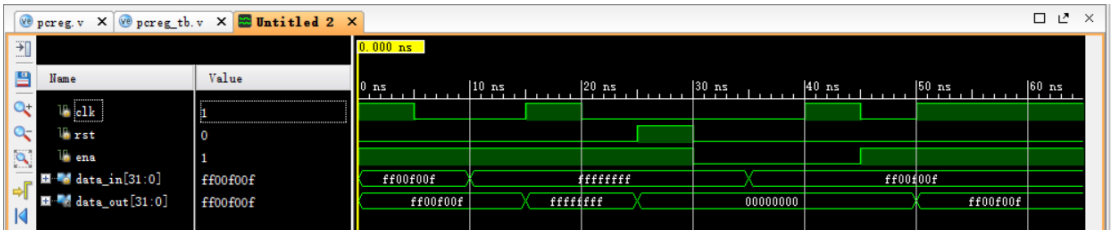


（4）pcreg

.v 文件没有使用实例化：

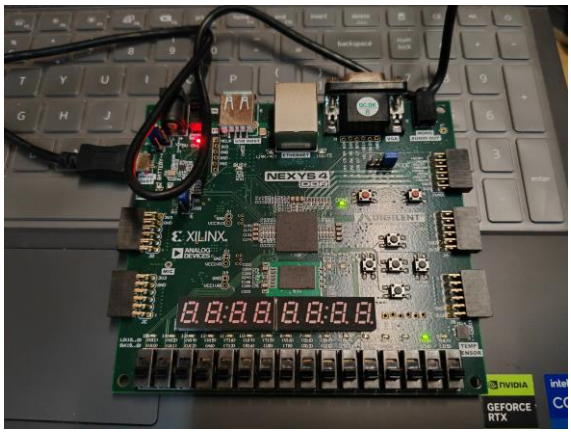


实例化后：

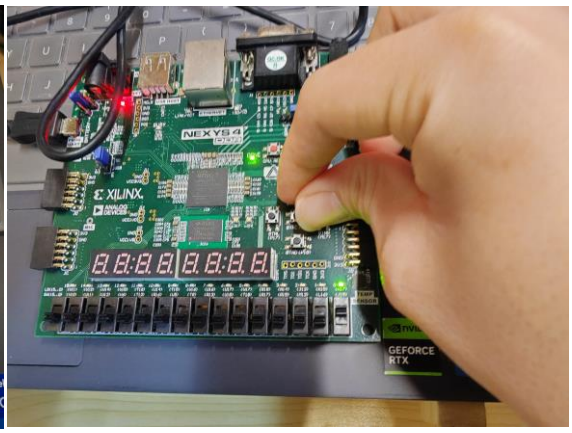


2. 下板结果

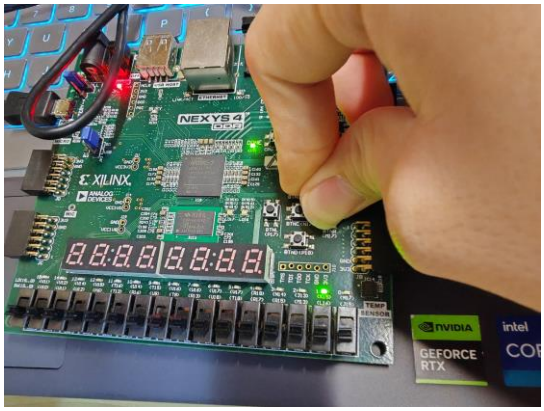
（1）Synchronous_D_FF

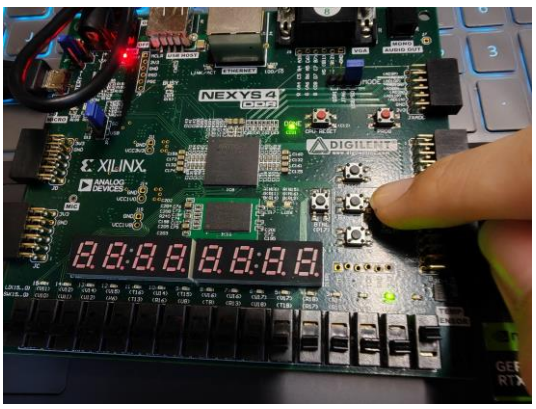


初始：没有任何按键：K15 亮

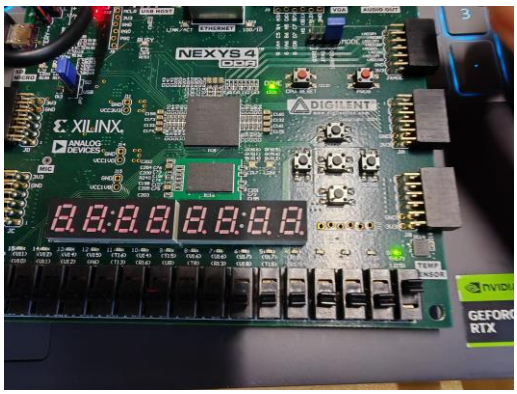


打开 J15，按下 M18（保持）后按下 M17：H17 亮





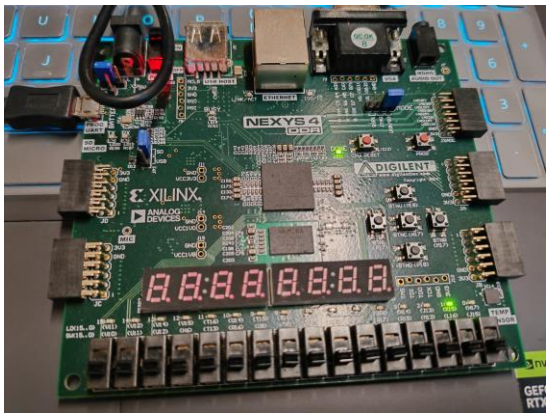
如果不打开任何开关，直接按下 M18（保持）后按下 M17：
K15 亮不变



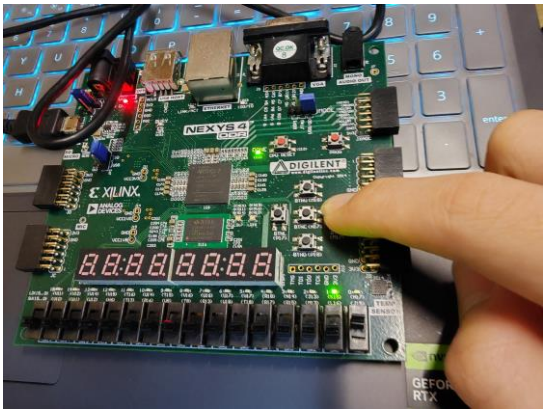
接上方情况，此时单独按下 M17，K15 和 H17 交换

如果不单独按下 M17，松手 M18 和 M17，H17 保持亮不灭。

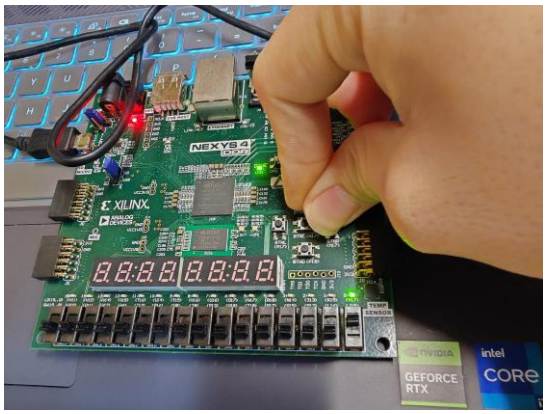
（2）Asynchronous_D_FF



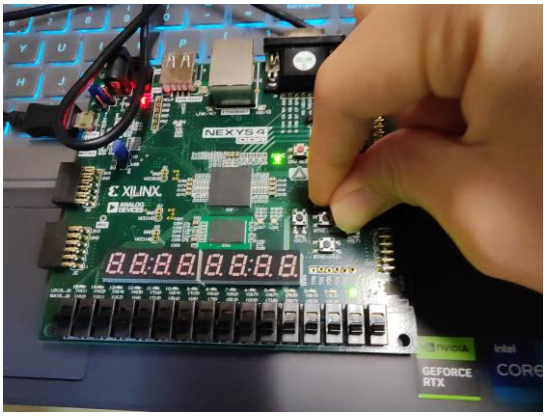
初始：K15 亮



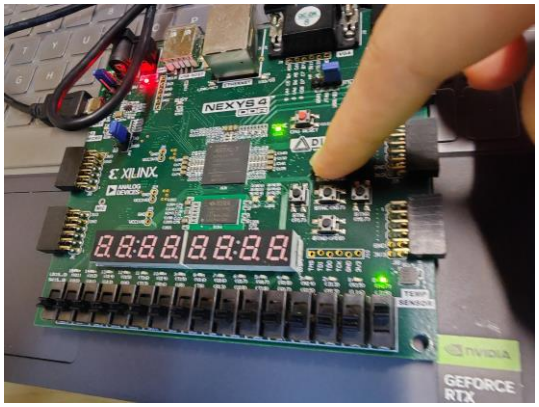
此时按 M17，H17 和 K15 值交换



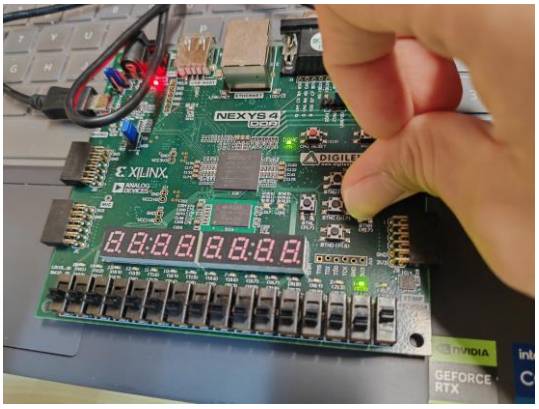
打开 J15 开关，
按下 M18（保持）后按下 M17：
由 K15 亮变为 H17 亮



按下 M18（保持）后按下 M17：
K15 亮保持不变



此时按 M18，H17 亮不变

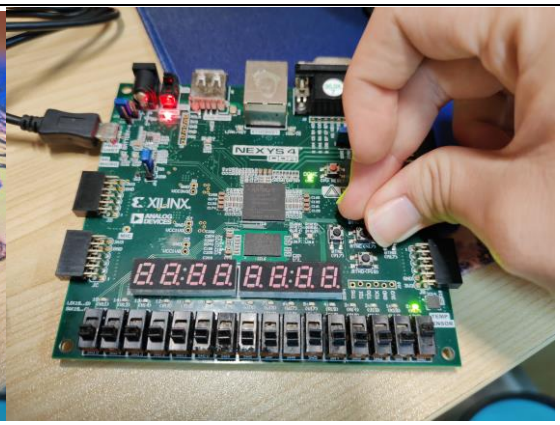


此时再按 M17，值不变

（3）JK_FF

（本次作业忘记携带板子，借用同学板子）

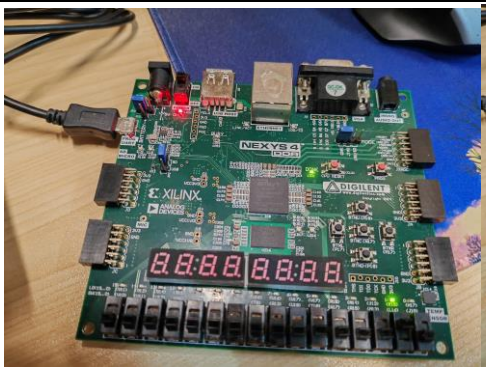
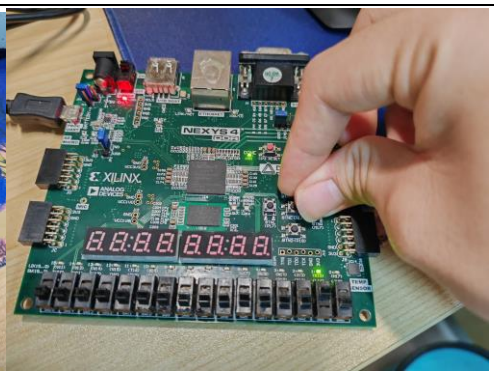
没有任何按键时，K15 亮



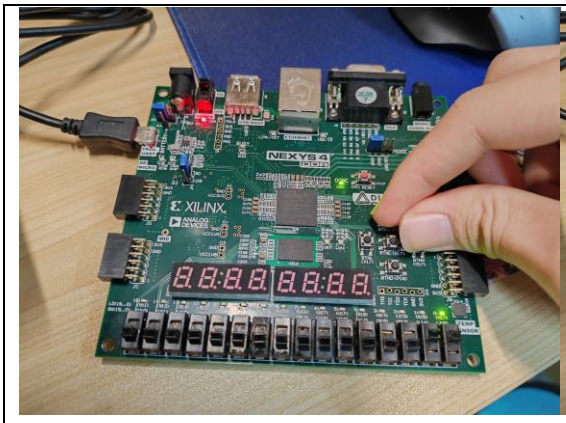
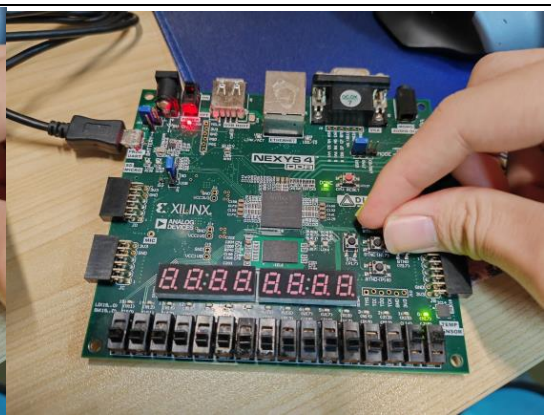
打开 J15，K15 亮

按下 M18（保持）后按下 M17：

H17 亮

打开 L16，K15 亮
按下 M18（保持）后按下 M17：
K15 亮不变

打开 L16 和 J15，H17 亮
按下 M18（保持）后按下 M17：
H17 亮不变