

浙江大学

本科实验报告

课程名称: 数字逻辑设计

姓 名: 颜晗

学 院: 计算机科学与技术学院

系:

专 业: 计算机科学与技术

学 号: 3200105515

指导教师: 蔡铭

2021 年 12 月 31 日

浙江大学实验报告

课程名称： 数字逻辑设计 实验类型： 综合

实验项目名称： 寄存器和寄存器传输设计

学生姓名： 颜晗 专业： 计算机科学与技术 学号： 3200105515

同组学生姓名： _____ 指导老师： 蔡铭

实验地点： 东四 509 实验日期： 2021 年 12 月 20 日

一、实验目的和要求

掌握寄存器传输电路的工作原理

掌握寄存器传输电路的设计方法

掌握 ALU 和寄存器传输电路的综合应用

二、实验内容和原理

实验内容：基于 ALU 的数据传输应用设计

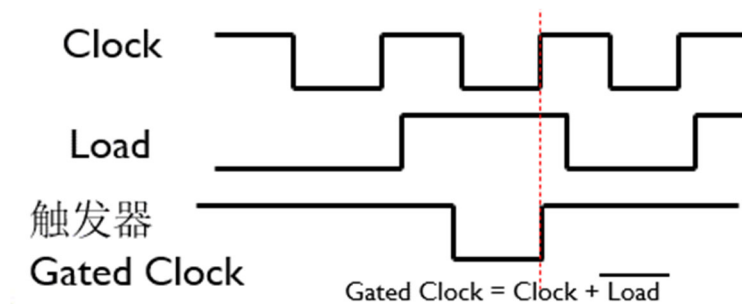
实验原理：

1. 寄存器

寄存器是计算机中的一组二进制存储单元，可用于暂时存储一系列二进制值，例如参与运算的数据，运算结果，指令等，能保存信息多个周期，可通过信号来控制“保存”还是“加载”信息。

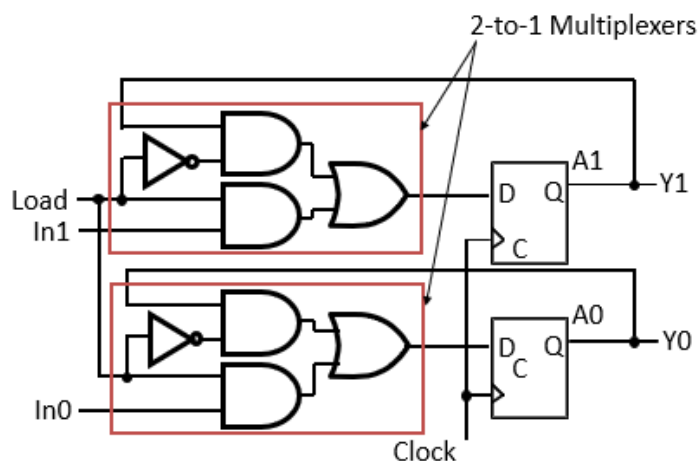
1.1 采用门控时钟的寄存器

如果 Load 信号为 1，允许时钟信号通过，如果为 0 则阻止时钟信号通过。如此控制寄存器是否需要执行“加载”操作。



1.2 采用 Load 控制反馈的寄存器

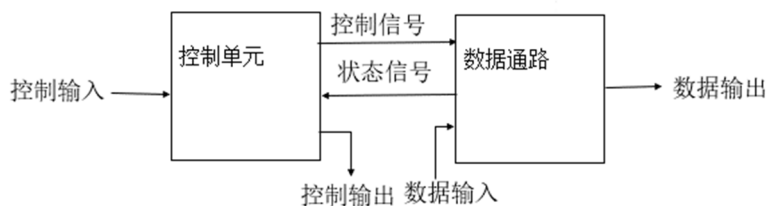
要想有选择的加载寄存器，更加可靠的方式是保证时钟的连续性，选择性地使用加载控制信号来改变寄存器地内容。如下原理图：



根据原理图，寄存器使用 D 触发器控制输出，输入相当于借助二选一多路选择器，Load 信号即为选择信号，Load 为 0 时相当于选择上一周期输出信号再作为输入，即“保存”操作；Load 为 1 时即选择外界输入作为触发器输入，即“加载”操作。

2. 寄存器传输

即寄存器中数据地传输和处理。使寄存器能从外界加载数据，能在不同寄存器之间转移数据，能将寄存器中地数据进行计算后再加载进寄存器中。



三个基本单元：寄存器组、操作、操作控制；基本操作:加载、计数、移位、加法、按位操作等。

2.1 Load 信号的产生

代码如下：

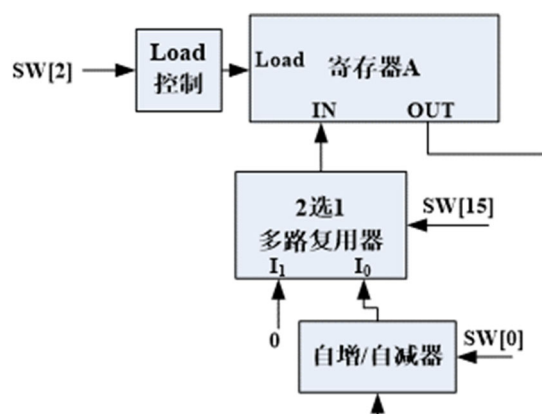
```

module Load_Gen(
    input wire clk,
    input wire clk_lms,
    input wire btn_in,
    output reg Load_out
);
    initial Load_out = 0;
    wire btn_out;
    reg old_btn;
    pbdebounce p0(clk_lms, btn_in, btn_out);
    always@(posedge clk) begin
        if ((old_btn == 1'b0) && (btn_out == 1'b1)) //btn出现上升沿
            Load_out <= 1'b1;
        else
            Load_out <= 1'b0;
        end
    end
    always@(posedge clk) begin //保存上一个周期btn的状态
        old_btn <= btn_out;
    end
end
endmodule

```

通过一个按钮或开关来产生信号，具体原理为不断保存按钮旧状态，仅当按钮旧状态为 0，转变为 1 的那个上升沿才执行一次“加载”操作，其余时间“加载”信号都为 0。

2.2 采用寄存器传输原理的计数器



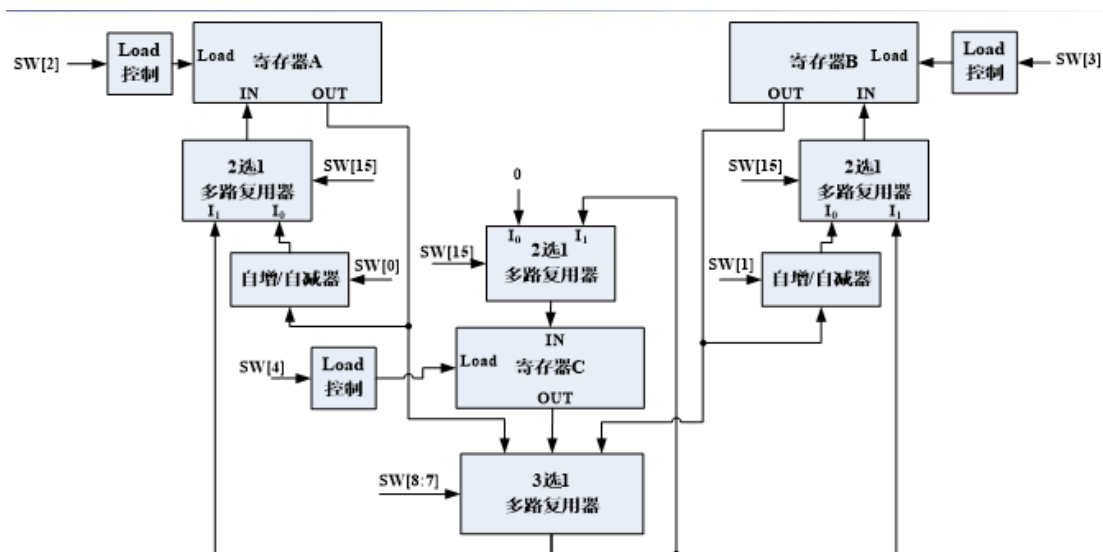
最简单的寄存器应用，寄存器的输出在经过自增自减后与清零信号一起作为 2 选 1 多路复用器的输入，再用另外两组开关控制选择与加载信号，计数器便可以实现逐一增减或清零。

3. 基于多路选择器总线的寄存器传输

多个寄存器共用一条数据总线，通过多路选择器可以选择需要进行操作的寄存器数据，再控制不同寄存器的 Load 信号，便可实现寄存器之间的数据传输，另外同计数器相同，寄存器自身数据的自增自减数也是多路复用器的一个输入。

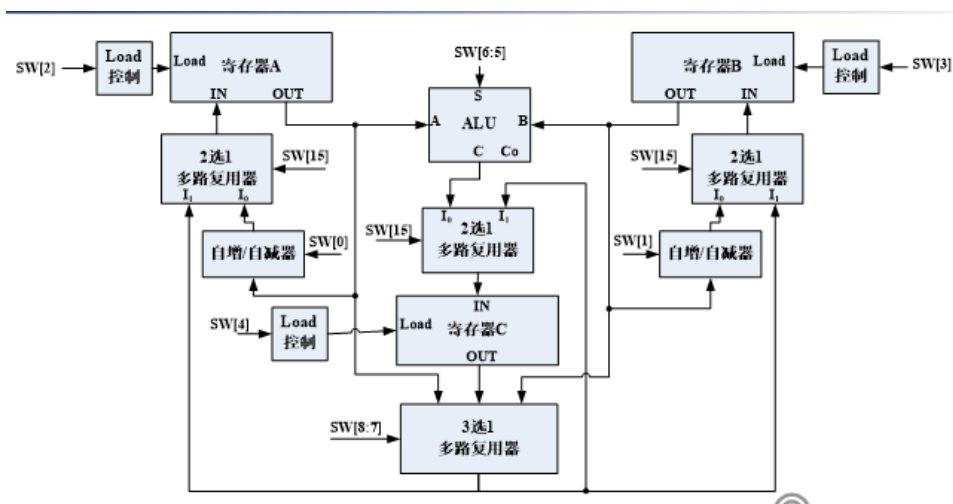
需要注意的是，由于共用数据总线，虽然降低了硬件开销，但是也导致无法实现寄存器之间的并行传输操作。

如图实现了三个寄存器之间的数据传输。



4. 寄存器传输应用设计

相比上一级任务增加了 ALU，可以将寄存器 A、B 的数据进行运算后的数据作为寄存器 C 的一个输入选项，仅需将 A、B 的输入线路再分一条至 ALU 模块即可。



三、实验过程和数据记录

共同步骤：

1. 创建源文件，设计寄存器模块。通过条件语句选择输出即可。

```

module register4b(
    input wire clk,
    input wire [3:0] IN,
    input wire Load,
    output reg [3:0] OUT
);
    always @(posedge clk)begin
        if(Load) OUT = IN;
    end
endmodule

```

2. 将任务所需的模块添加进工程。

任务 1: 采用寄存器传输原理设计计数器

1. 依据任务要求设计顶层模块

要求实现寄存器的设置初值功能，寄存器自增、自减功能。

```

module Top(
    input wire clk,
    input wire [15:0] SW,
    output wire [3:0] AN,
    output wire [7:0] SEGMENT
);

    wire Load_A;
    wire [3:0] A, A_IN, A1;
    wire [31:0] clk_div;

    register4b RegA(.clk(clk), .IN(A_IN), .Load(Load_A), .OUT(A));
    Load_Gen m0(.clk(clk),
        .clk_lms(clk_div[17]),
        .btn_in(SW[2]),
        .Load_out(Load_A));
    clkdiv m3(clk, 1'b0, clk_div);
    addsub4b m4(.Ctrl(SW[0]), .A(A), .B(4'b0001), .S(A1));
    assign A_IN = (SW[15] == 1'b0) ? A1 : 4'b0000;
    Disp_num m5(clk, 1'b0, {A, A1, A_IN, 4'b0000}, SEGMENT, AN);
endmodule

```

任务 2: 基于多路选择器总线的寄存器传输

1. 依据任务要求设计顶层模块

验证 A、B、C 寄存器的设置初值功能，验证 A、B 寄存器的自增、自减功能，验证 A、B、C 寄存器之间的传输功能。

```

module Top(
    input clk,
    input [15:0] SW,
    output [3:0] AN,
    output [7:0] SEGMENT
);
    wire [31:0] clk_div;
    wire [3:0] A_IN, B_IN, C_IN;
    wire Load_A, Load_B, Load_C;
    wire [3:0] A_OUT, B_OUT, C_OUT;
    wire [3:0] I1, IO_A, IO_B, IO_C;
    wire [3:0] C;

    Load_Gen m0(.clk(clk), .clk_lms(clk_div[17]), .btn_in(SW[2]),.Load_out(Load_A));
    Load_Gen m1(.clk(clk), .clk_lms(clk_div[17]), .btn_in(SW[3]),.Load_out(Load_B));
    Load_Gen m2(.clk(clk), .clk_lms(clk_div[17]), .btn_in(SW[4]),.Load_out(Load_C));
    clkdiv m3(clk, 1'b0, clk_div);
    addsub4b m4(.Ctrl(SW[0]),.A(A_OUT), .B(4'b0001), .S(IO_A));
    addsub4b m5(.Ctrl(SW[1]),.A(B_OUT), .B(4'b0001), .S(IO_B));
    assign A_IN = (SW[15] == 1'b0)? IO_A: I1;
    assign B_IN = (SW[15] == 1'b0)? IO_B: I1;
    assign C_IN = (SW[15] == 1'b0)? 4'b0000: I1;
    Mux4to1b4 m9(SW[8:7],A_OUT, B_OUT, C_OUT, 4'b0000, I1);
    register4b RegA( clk, A_IN, Load_A, A_OUT );
    register4b RegB( clk, B_IN, Load_B, B_OUT );
    register4b RegC( clk, C_IN, Load_C, C_OUT );
    Disp_num m11(clk,1'b0, {A_OUT, B_OUT, C_OUT, I1}, SEGMENT, AN);
endmodule

```

任务 3：基于 ALU 的数据传输应用设计

1. 依据任务要求设计顶层模块

验证 A、B、C 寄存器的设置初值功能；验证 A、B 寄存器的自增、自减功能；
验证 ALU 运算功能；验证寄存器传输功能。

```

module Top(
    input clk,
    input [15:0] SW,
    output [3:0] AN,
    output [7:0] SEGMENT
);
    wire [31:0] clk_div;
    wire [3:0] A_IN, B_IN, C_IN;
    wire Load_A, Load_B, Load_C;
    wire [3:0] A_OUT, B_OUT, C_OUT;
    wire [3:0] I1, IO_A, IO_B, IO_C;
    wire Co;

    Load_Gen m0(.clk(clk), .clk_lms(clk_div[17]), .btn_in(SW[2]),.Load_out(Load_A));
    Load_Gen m1(.clk(clk), .clk_lms(clk_div[17]), .btn_in(SW[3]),.Load_out(Load_B));
    Load_Gen m2(.clk(clk), .clk_lms(clk_div[17]), .btn_in(SW[4]),.Load_out(Load_C));
    clkdiv m3(clk, 1'b0, clk_div);
    addsub4b m4(.A(A_OUT), .B(4'b0001), .Ctrl(SW[0]), .S(IO_A));
    addsub4b m5(.A(B_OUT), .B(4'b0001), .Ctrl(SW[1]), .S(IO_B));
    assign A_IN = (SW[15] == 1'b0)? IO_A: I1;
    assign B_IN = (SW[15] == 1'b0)? IO_B: I1;
    assign C_IN = (SW[15] == 1'b0)? IO_C: I1;
    Mux4to1b4 m9(SW[8:7],A_OUT, B_OUT, C_OUT, 4'b0000, I1);
    register4b RegA( clk, A_IN, Load_A, A_OUT );
    register4b RegB( clk, B_IN, Load_B, B_OUT );
    register4b RegC( clk, C_IN, Load_C, C_OUT );
    ALU4 m10( A_OUT, B_OUT, SW[6:5], IO_C, Co);
    Disp_num m11(clk,1'b0, {A_OUT, B_OUT, C_OUT, 3'b000, Co}, SEGMENT, AN);
endmodule

```

四、实验结果分析

任务 1：采用寄存器传输原理设计计数器

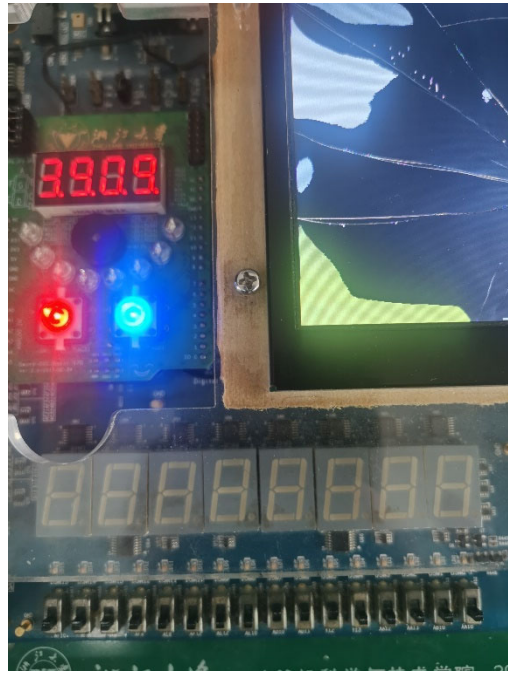
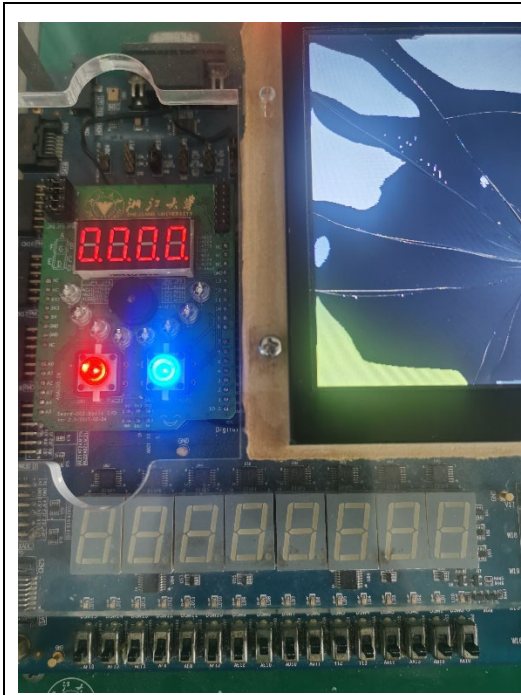
左起数码管分别表示 1.寄存器的输出，2.寄存器自增/减的值，3.寄存器等待的输入，4.无意义常值 0.

自增、数据选择	清零信号选择	自减
清零复位		

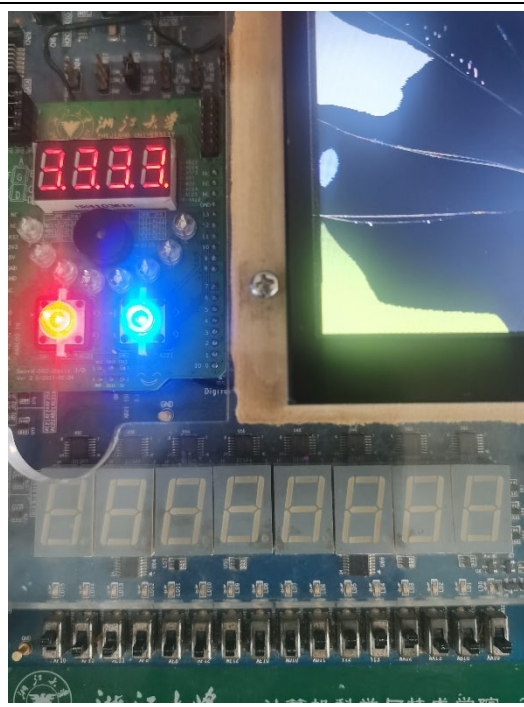
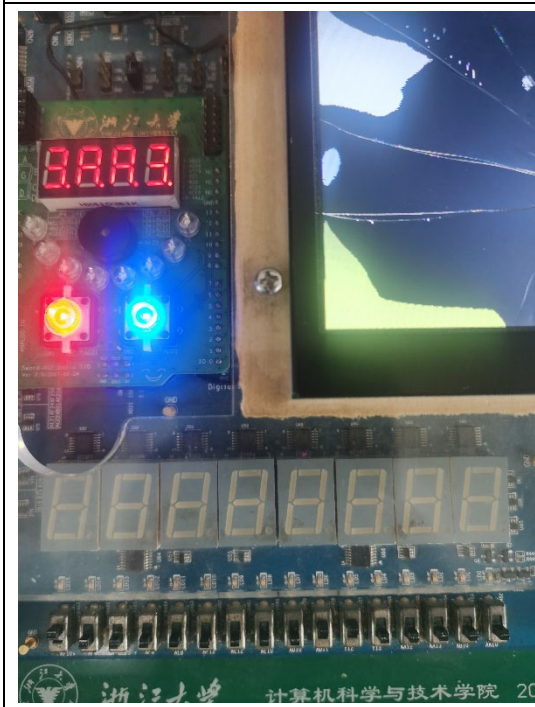
任务 2：基于多路选择器总线的寄存器传输

左起数字分别表示 1.A 寄存器的输出，2. B 寄存器的输出，3. C 寄存器的输出，4.多路复用器选择的数据。

初始数据	A、B 自增，选择 B
------	-------------

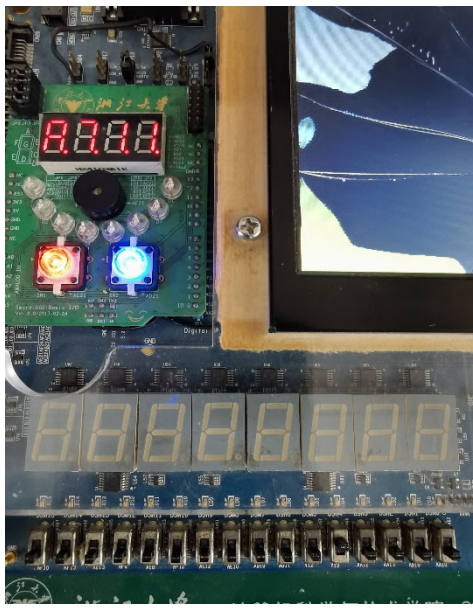
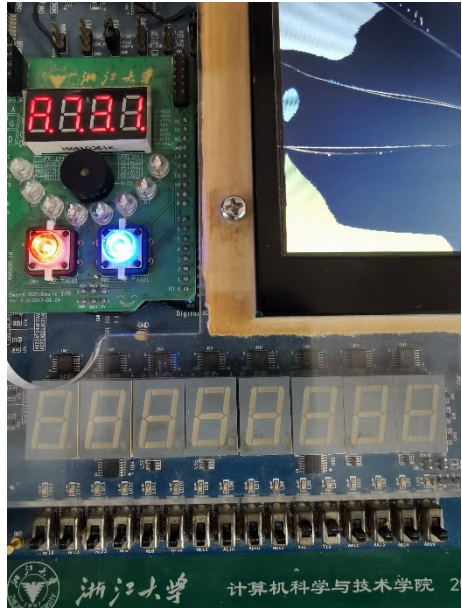
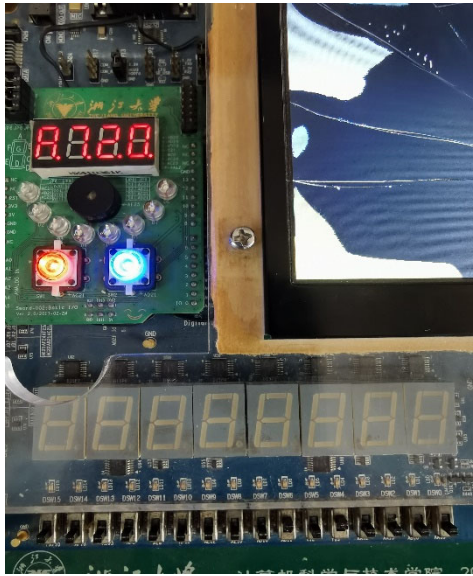
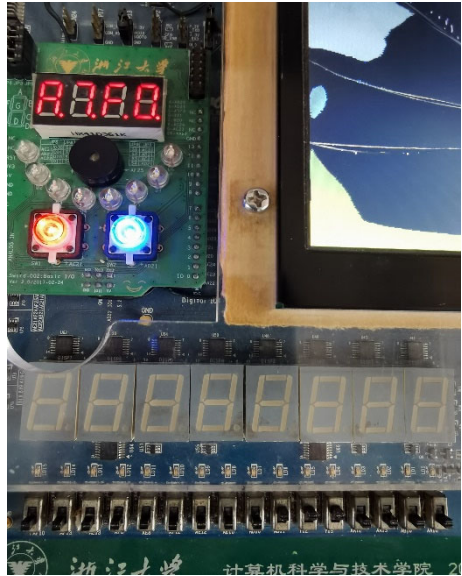


选择 A 数据，并传输给 B、C 寄存器



任务 3：基于 ALU 的数据传输应用设计

左起数字分别表示 1.A 寄存器的输出，2. B 寄存器的输出，3. C 寄存器的输出，4.运算的进位。（结果可由 C 的数据表示）；不进行与任务一、二相同的操作，仅展示运算后赋值的功能。

加法 $A+7=11(16)$ 	减法 $A-7=3(\text{补码表示有进位 } 1)$ 
与 $1010 \& 0111 = 0010$ 	或 $1010 \mid 0111 = 1111$ 

五、实验总结与反思

本次实验我们接触了硬件中非常重要的结构——寄存器。通过亲自设计寄存器模块并实现其应用，加深了原理的理解，也对硬件中的数据传输和处理步骤有了一个较为清晰的认识，为接下来的实验打好基础。