浙江大学

本科实验报告

课程名称：	数字逻辑设计
姓　　名：	
学　　院：	竺可桢学院
专　　业：	混合班
指导教师：	董亚波
报告日期：	2025 年 5 月 3 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： 学号： 同组学生姓名：无

实验地点：紫金港东四 509 室 实验日期：2025 年 4 月 30 日

一、 实验目的

- 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、操作方法与实验步骤

1. 实验原理

寄存器

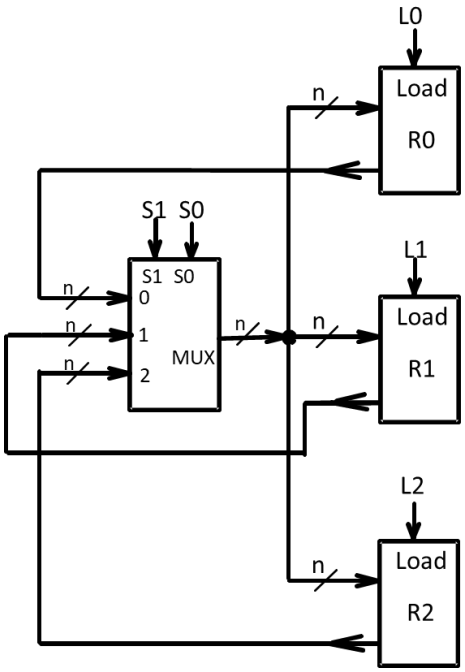
- 一组二进制存储单元
- 一个寄存器可以用于存储一列二进制值，通常用于进行简单数据存储、移动和处理等操作
- 能存储信息并保存多个时钟周期，能用信号来控制“保存”或“加载”信息

寄存器传输

- 寄存器传输：寄存器中数据的传输和处理
- 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

基于多路选择器总线的寄存器传输

- 由一个多路选择器驱动的总线可以降低硬件开销
- 这个结构不能实现多个寄存器相互之间的并行传输操作
- 原理图如下：



2. 采用寄存器传输原理设计计数器

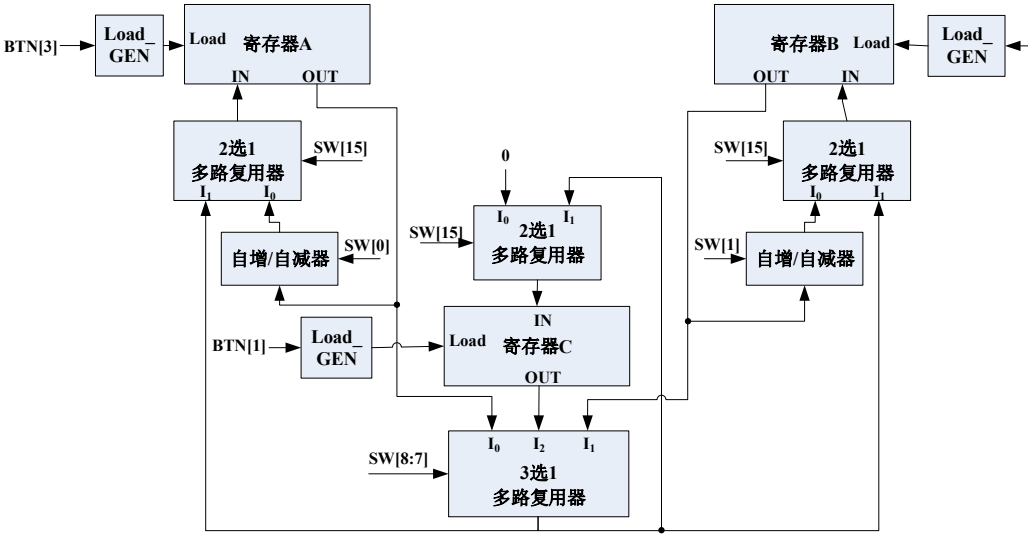
实验步骤如下：

- 在 vivado 中新建工程
- 新建源文件，设计 Load_Gen.v 及 MyRegister4b.v 文件，分别用于生成 Load 信号及寄存器。
- 新建 top.v 源文件，用行为描述进行设计
- 设计 top 文件的仿真波形代码，进行行为仿真
- 在开发板上进一步验证

3. 基于多路选择器总线的寄存器传输

实验步骤如下：

- 在 vivado 中新建工程
- 根据以下原理图新建 top. v 文件，实现传输功能：

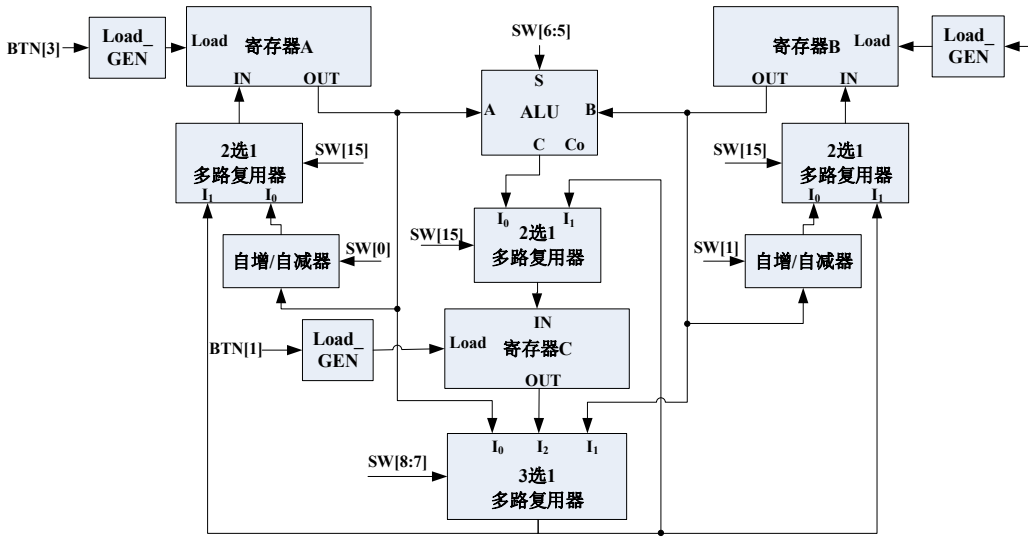


- 设计 top 文件的仿真波形代码，进行行为仿真

4. 基于 ALU 的数据传输应用设计

实验步骤如下：

- 在 vivado 中新建工程
- 根据以下原理图新建 top.v 文件，实现带运算的传输功能：

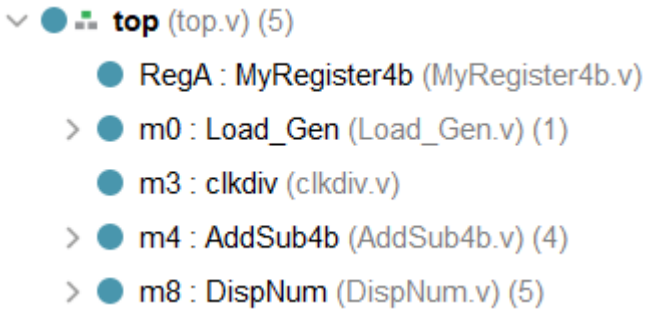


- 在开发板上对其功能进行验证

三、实验结果与分析

1. 采用寄存器传输原理设计计数器

按要求创建工程，需要添加之前的文件 clkdiv.v, DispNum.v, AddSub4b.v（不再重复展示），并新建 Load_Gen.v, MyRegister4b.v 及 top.v。
文件结构如图所示：



Load_Gen.v 代码如下：

```
`timescale 1ns / 1ps
module Load_Gen(
    input wire clk,
    input wire clk_1ms,
    input wire btn_in,
```

```
    output reg Load_out
    );
     initial Load_out = 0;
     wire btn_out;
     reg old_btn;
     pbdebounce p0(.clk_1ms(clk_1ms), .button(btn_in), .pbreg(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
     always@(posedge clk) begin     //保存上一个周期btn 的状态
        old_btn <= btn_out;
     end
//   assign btn_out = btn_in;
endmodule
```

MyRegister4b.v 代码如下：

```
`timescale 1ns / 1ps
module MyRegister4b(
    input wire clk,
    input wire [3:0] IN,
    input wire Load,
    output reg [3:0] OUT
);
    initial OUT = 4'b0000;
    always @ (posedge clk) begin
        if (Load) OUT <= IN;
    end
endmodule
```

top.v 代码如下：

```
`timescale 1ns / 1ps
module top(
    input wire clk,
    input wire [15:0] SW,
    input wire [3:0] BTN,
    output wire [3:0] AN,
    output wire [7:0] SEGMENT,
    output wire BTNX4
);
    wire [3:0] Load_A, Co;
    wire [3:0] A,  A_IN, A1;
    wire [31:0] clk_div;
    wire [15:0] num;
```

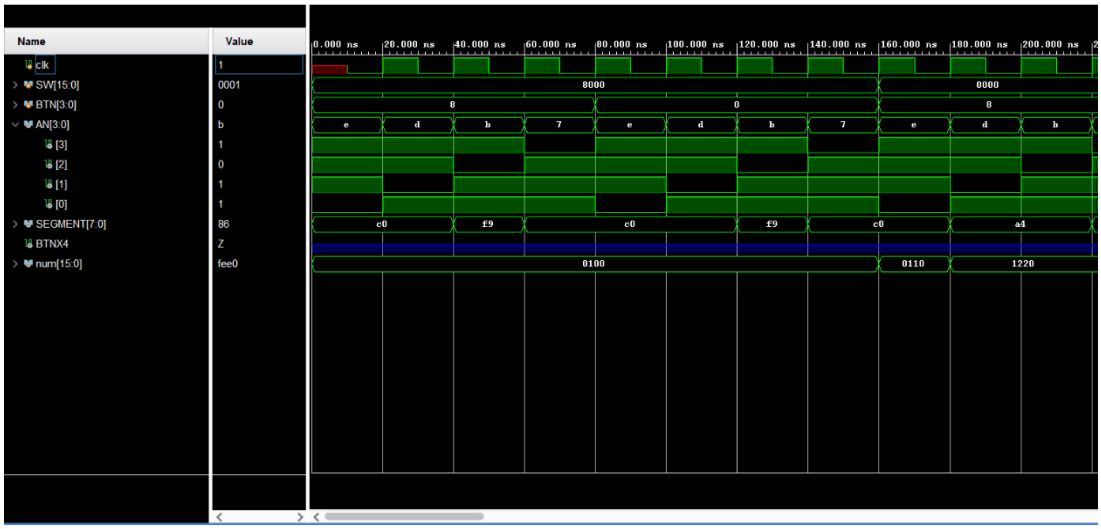
```
    assign num = {A, A1, A_IN, 4'b0000};
    MyRegister4b RegA(.clk(clk), .IN(A_IN), .Load(Load_A), .OUT(A));
    Load_Gen m0(.clk(clk), .clk_1ms(clk_div[17]), .btn_in(BTN[3]),
        .Load_out(Load_A));    //寄存器A的Load信号
    clkdiv m3(.clk(clk), .rst(1'b0), .clk_div(clk_div));
    AddSub4b m4(.A(A), .B(4'b0001), .Ctrl(SW[0]), .S(A1));//自增/自减逻辑
    assign A_IN = (SW[15] == 1'b0)? A1: 4'b0000;   //2选1多路复用器，复位寄存器初值
    DispNum m8(.scan(clk_div[18:17]), .HEXS(num), .LES(4'b0), .point(4'b0), .AN(AN), .SEGMENT(SEGMENT));
    assign BTNX4 = 1'b0;
endmodule
```

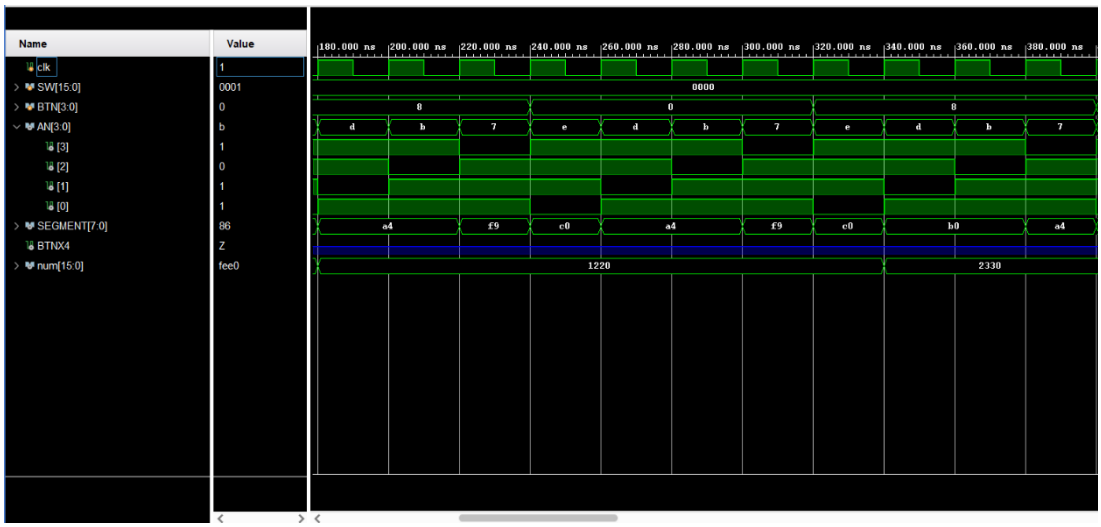
对 top.v 进行仿真，仿真代码如下：

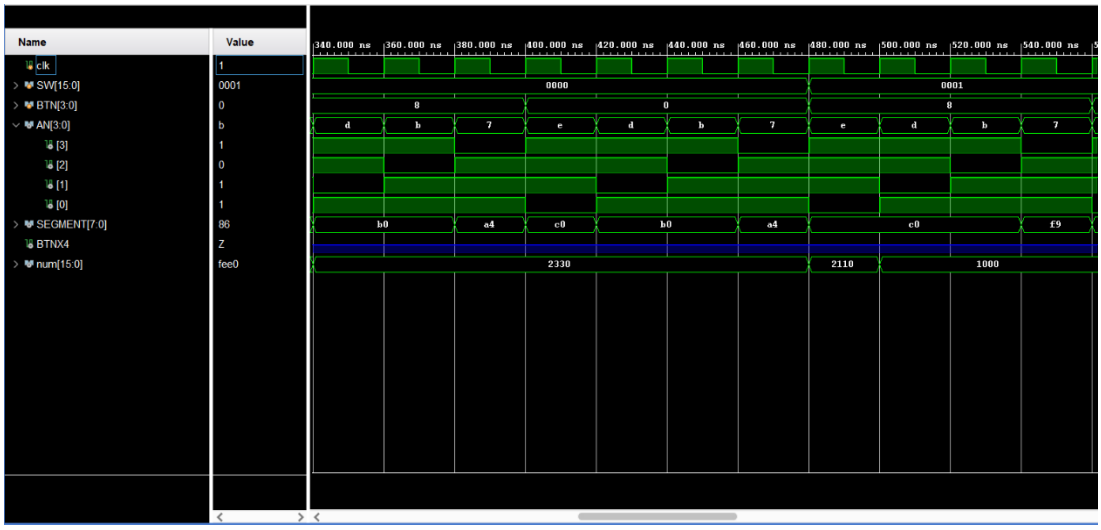
```
`timescale 1ns / 1ps
module top_testbench();
    reg clk;
    reg [15:0] SW;
    reg [3:0] BTN;
    wire [3:0] AN;
    wire [7:0] SEGMENT;
    wire BTNX4;
    top uut(.clk(clk),.SW(SW),.BTN(BTN),.AN(AN),.SEGMENT(SEGMENT),.BTNX4(BTNX4));
    initial begin
       SW=0;
       BTN=0;
       SW[15]=1;
       BTN[3]=1;
       #80;
       BTN[3]=0;
       #80;
       SW[15]=0;
       SW[0]=0;
       BTN[3]=1;
       #80;
       BTN[3]=0;
       #80;
       BTN[3]=1;
       #80;
       BTN[3]=0;
       #80;
       SW[0]=1;
```

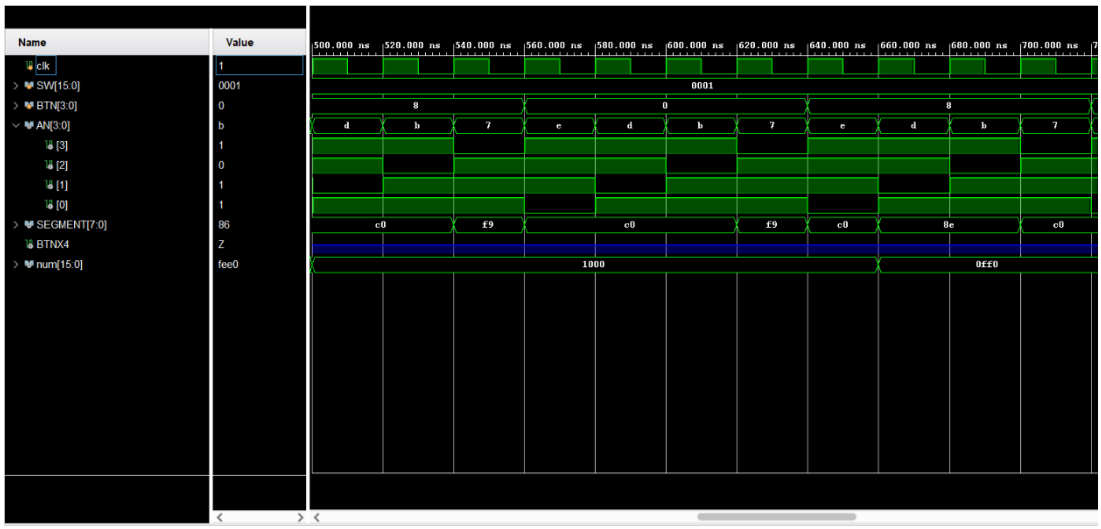
```
        BTN[3]=1;
        #80;
        BTN[3]=0;
        #80;
        BTN[3]=1;
        #80;
        BTN[3]=0;
        #80;
        BTN[3]=1;
        #80;
        BTN[3]=0;
    end
    always begin
        #10;clk = 0;
        #10;clk = 1;
    end
endmodule
```

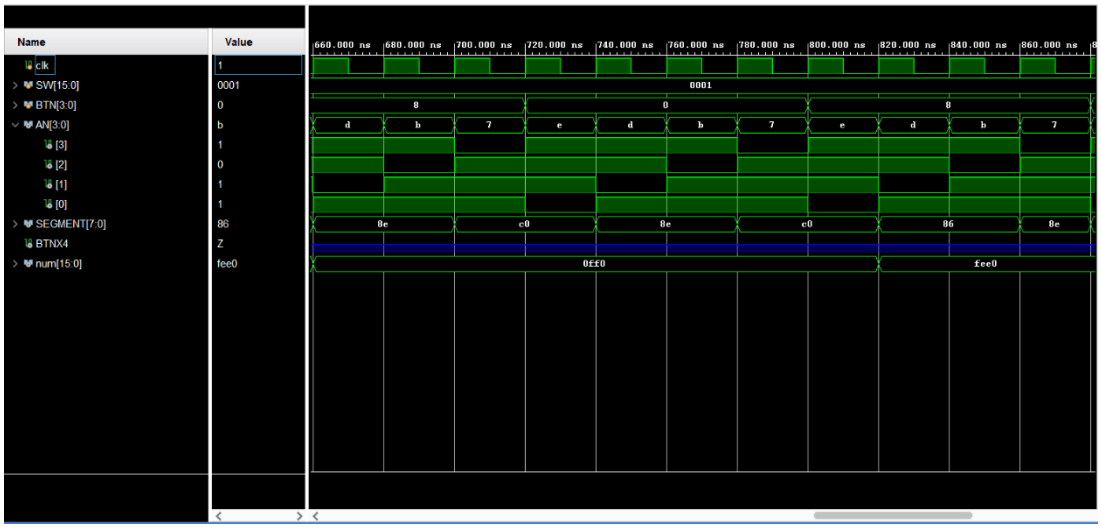
仿真波形如下（注：仿真时并未设置 BTNX4 的值，故为 Z）：











仿真波形中，num 的第一个数为 A 的值；第二个数为通过加减法器运算，A 的下一个值；第三个数为 A 下一个值（不一定通过加减法器运算）。可以从仿真波形看出这一工程成功实现了要求的功能。

再下载到板上进行验证，约束文件如下（第三个任务的约束文件完全一致，不再重复展示）：

```
create_clock -name clk100MHZ -period 10.0 [get_ports clk]
set_property PACKAGE_PIN AC18 [get_ports clk]
set_property IOSTANDARD LVCMOS18 [get_ports clk]

# Button
set_property PACKAGE_PIN W14 [get_ports BTN[0]]
set_property PACKAGE_PIN V14 [get_ports BTN[1]]
set_property PACKAGE_PIN V19 [get_ports BTN[2]]
set_property PACKAGE_PIN V18 [get_ports BTN[3]]
set_property PACKAGE_PIN W16 [get_ports BTNX4]
set_property IOSTANDARD LVCMOS18 [get_ports BTN[0]]
set_property IOSTANDARD LVCMOS18 [get_ports BTN[1]]
set_property IOSTANDARD LVCMOS18 [get_ports BTN[2]]
set_property IOSTANDARD LVCMOS18 [get_ports BTN[3]]
set_property IOSTANDARD LVCMOS18 [get_ports BTNX4]
# Switch
set_property PACKAGE_PIN AA10 [get_ports {SW[0]}]
set_property PACKAGE_PIN AB10 [get_ports {SW[1]}]
set_property PACKAGE_PIN AA13 [get_ports {SW[2]}]
set_property PACKAGE_PIN AA12 [get_ports {SW[3]}]
set_property PACKAGE_PIN Y13 [get_ports {SW[4]}]
set_property PACKAGE_PIN Y12 [get_ports {SW[5]}]
set_property PACKAGE_PIN AD11 [get_ports {SW[6]}]
set_property PACKAGE_PIN AD10 [get_ports {SW[7]}]
set_property PACKAGE_PIN AE10 [get_ports {SW[8]}]
set_property PACKAGE_PIN AE12 [get_ports {SW[9]}]
set_property PACKAGE_PIN AF12 [get_ports {SW[10]}]
set_property PACKAGE_PIN AE8 [get_ports {SW[11]}]
set_property PACKAGE_PIN AF8 [get_ports {SW[12]}]
set_property PACKAGE_PIN AE13 [get_ports {SW[13]}]
set_property PACKAGE_PIN AF13 [get_ports {SW[14]}]
set_property PACKAGE_PIN AF10 [get_ports {SW[15]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[0]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[1]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[2]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[3]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[4]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[5]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[6]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[7]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[8]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[9]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[10]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[11]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[12]}]
```

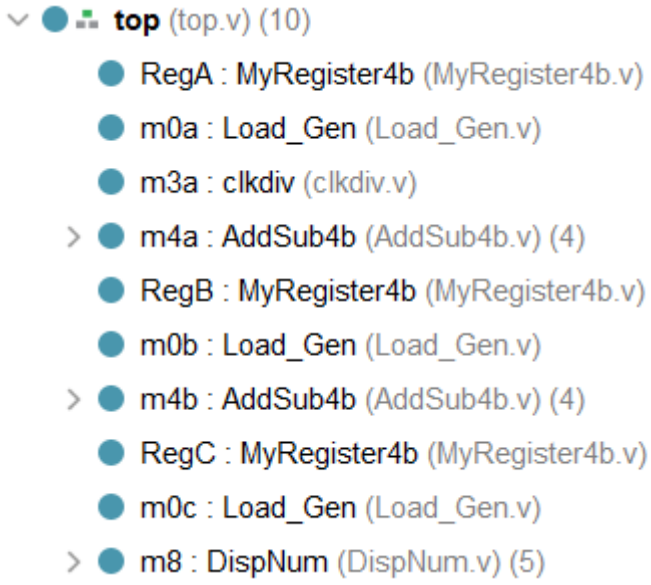
```
set_property IOSTANDARD LVCMOS15 [get_ports {SW[13]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[14]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[15]}]

# 7-Segment LED
set_property PACKAGE_PIN AB22 [get_ports {SEGMENT[0]}]
set_property PACKAGE_PIN AD24 [get_ports {SEGMENT[1]}]
set_property PACKAGE_PIN AD23 [get_ports {SEGMENT[2]}]
set_property PACKAGE_PIN Y21 [get_ports {SEGMENT[3]}]
set_property PACKAGE_PIN W20 [get_ports {SEGMENT[4]}]
set_property PACKAGE_PIN AC24 [get_ports {SEGMENT[5]}]
set_property PACKAGE_PIN AC23 [get_ports {SEGMENT[6]}]
set_property PACKAGE_PIN AA22 [get_ports {SEGMENT[7]}]
set_property PACKAGE_PIN AD21 [get_ports {AN[0]}]
set_property PACKAGE_PIN AC21 [get_ports {AN[1]}]
set_property PACKAGE_PIN AB21 [get_ports {AN[2]}]
set_property PACKAGE_PIN AC22 [get_ports {AN[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {AN[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {AN[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {AN[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {AN[3]}]
```

上板结果如下：


这里只在视频中截取了几个关键状态，完成的任务与仿真时一致。可以看到，上板验证时当最左边的开关闭合时，按下按钮，A 会清零，而最左边的开关断开时，按下按钮，A 会根据最右边的按钮进行增减。

2. 基于多路选择器总线的寄存器传输

按要求创建工程，需要添加之前的文件 clkdiv.v, DispNum.v, AddSub4b.v, Load_Gen.v, MyRegister4b.v（不再重复展示），并新建 top.v。
文件结构如图所示（本工程中无需 pbdebounce）：



top.v 代码如下（多路选择器直接通过?:语句实现）：

```
`timescale 1ns / 1ps
module top(
    input wire clk,
    input wire [15:0] SW,
    input wire [3:0] BTN,
    output wire [3:0] AN,
    output wire [7:0] SEGMENT,
    output wire BTNX4
);
    wire [3:0] Load_A, Load_B, Load_C, Co;
    wire [3:0] A,  A_IN, A1;
    wire [3:0] B,  B_IN, B1;
```

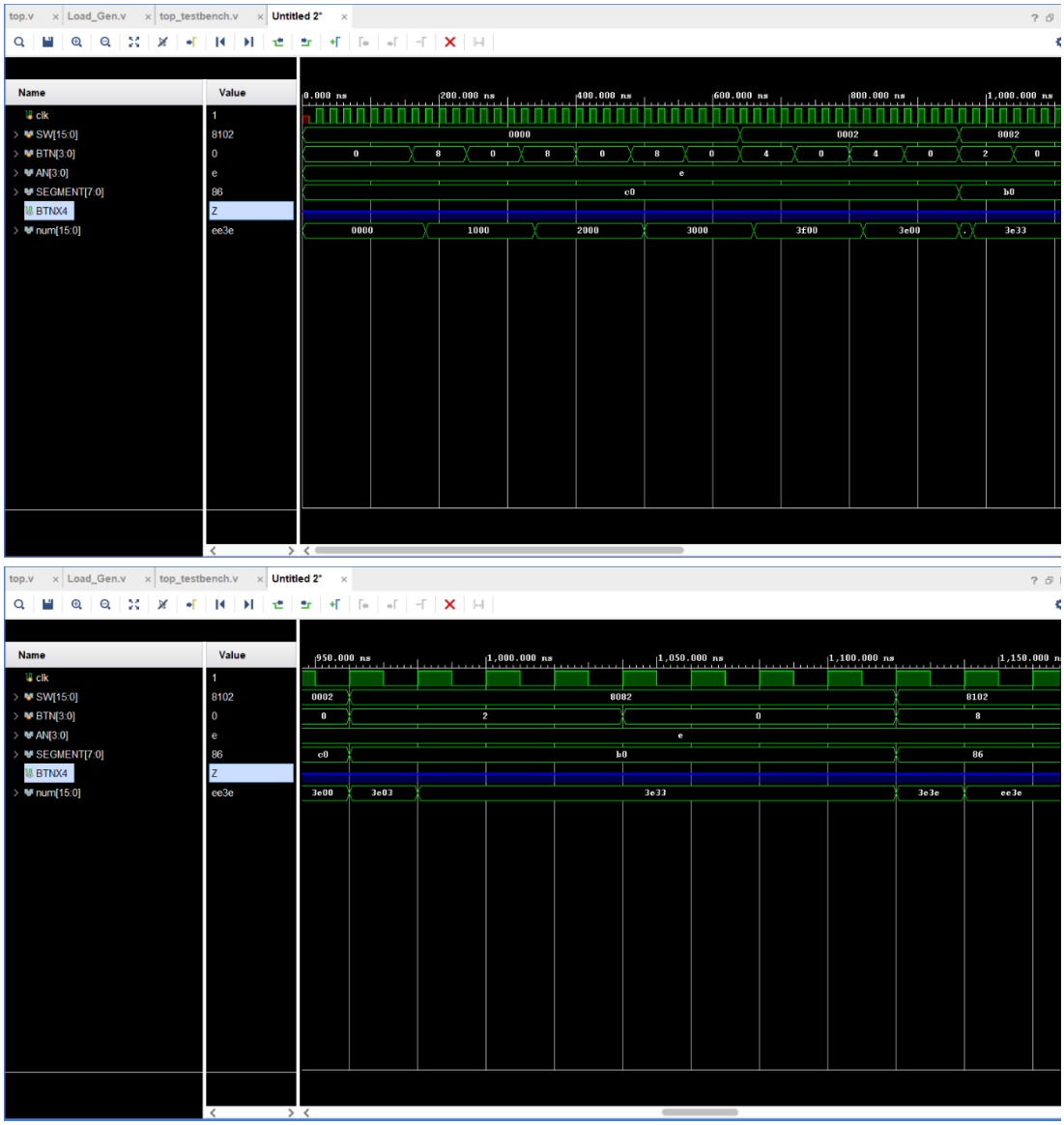
```
    wire [3:0] C,  C_IN, C1;
    wire [3:0] Bus;
    wire [31:0] clk_div;
    wire [15:0] num;
    assign num={A,B,C,Bus};
    assign Bus=(SW[8:7]==0)?C:(SW[8:7]==1)?A:(SW[8:7]==2)?B:0;
    MyRegister4b RegA(.clk(clk), .IN(A_IN), .Load(Load_A), .OUT(A));
    Load_Gen m0a(.clk(clk), .clk_1ms(clk_div[17]), .btn_in(BTN[3]),
        .Load_out(Load_A));
    clkdiv m3a(.clk(clk), .rst(1'b0), .clk_div(clk_div));
    AddSub4b m4a(.A(A), .B(4'b0001), .Ctrl(SW[0]), .S(A1));
    assign A_IN = (SW[15] == 1'b0)? A1: Bus;

    MyRegister4b RegB(.clk(clk), .IN(B_IN), .Load(Load_B), .OUT(B));
    Load_Gen m0b(.clk(clk), .clk_1ms(clk_div[17]), .btn_in(BTN[2]),
        .Load_out(Load_B));
    AddSub4b m4b(.A(B), .B(4'b0001), .Ctrl(SW[1]), .S(B1));
    assign B_IN = (SW[15] == 1'b0)? B1: Bus;

    MyRegister4b RegC(.clk(clk), .IN(C_IN), .Load(Load_C), .OUT(C));
    Load_Gen m0c(.clk(clk), .clk_1ms(clk_div[17]), .btn_in(BTN[1]),
        .Load_out(Load_C));
    assign C1 = C;
    assign C_IN = (SW[15] == 1'b0)? 4'b0000: Bus;

    DispNum
m8(.scan(clk_div[18:17]), .HEXS(num), .LES(4'b0), .point(4'b0), .AN(AN)
, .SEGMENT(SEGMENT));
endmodule
```

对 top.v 进行仿真，仿真代码如下：

```
`timescale 1ns / 1ps
module top_testbench();
    reg clk;
    reg [15:0] SW;
    reg [3:0] BTN;
    wire [3:0] AN;
    wire [7:0] SEGMENT;
    wire BTNX4;
    top
uut(.clk(clk),.SW(SW),.BTN(BTN),.AN(AN),.SEGMENT(SEGMENT),.BTNX4(BTNX4)
);
    initial begin
       SW=0;
       BTN=0;
```

```
        SW[15]=0;
        #160;
        BTN[3]=1;
        #80;
        BTN[3]=0;
        #80;
        BTN[3]=1;
        #80;
        BTN[3]=0;
        #80;
        BTN[3]=1;
        #80;
        BTN[3]=0;
        #80;
        SW[1]=1;
        BTN[2]=1;
        #80;
        BTN[2]=0;
        #80;
        BTN[2]=1;
        #80;
        BTN[2]=0;
        #80;
        SW[15]=1;
        SW[7]=1;
        BTN[1]=1;
        #80
        BTN[1]=0;
        #80;
        SW[8]=1;
        SW[7]=0;
        BTN[3]=1;
        #80;
        BTN[3]=0;
    end
    always begin
        #10;clk = 0;
        #10;clk = 1;
    end
endmodule
```

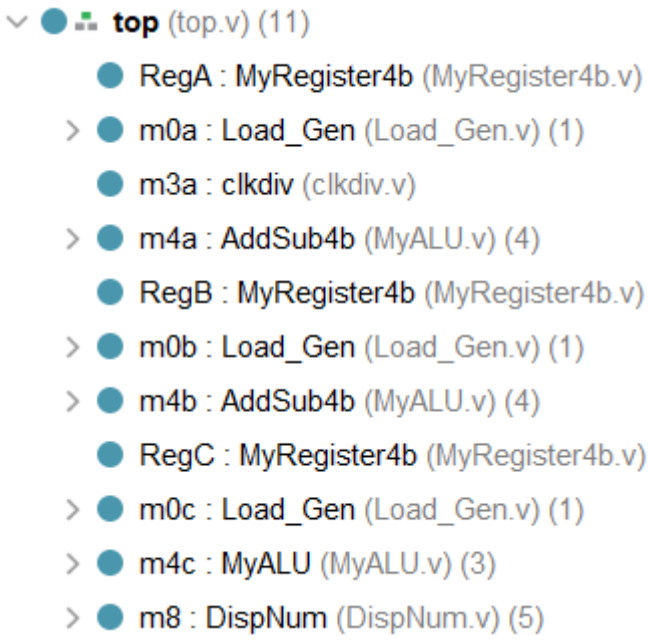
仿真波形如图所示（注：仿真时并未设置 BTNX4 的值，故为 Z）：



可以看到，当 SW[15]为 1 时，通过调节 SW[8:7]，Bus 总线将选择与之相对应的值，再按下与数字相对应的按钮，即可实现值的传输。

3. 基于 ALU 的数据传输应用设计

按要求创建工程，需要添加之前的文件 clkdiv.v, DispNum.v, AddSub4b.v, Load_Gen.v, MyRegister4b.v（不再重复展示），并新建 top.v。
文件结构如图所示：

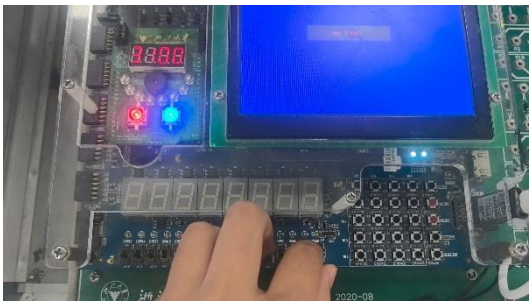
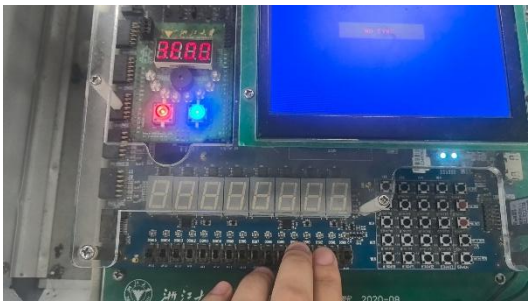
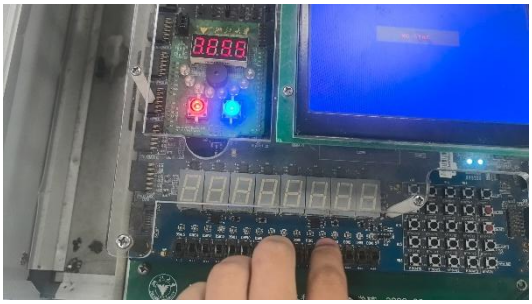
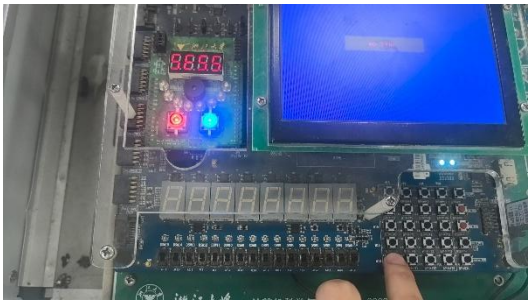
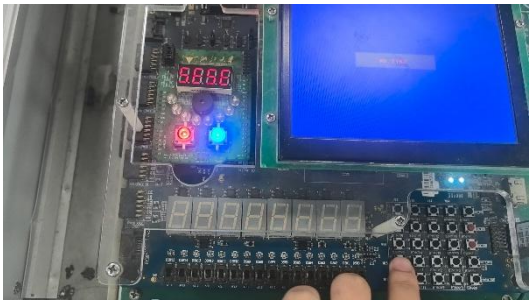
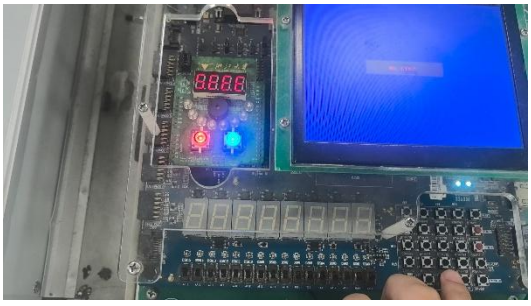


top.v 代码如下（多路选择器直接通过?:语句实现）：

```
`timescale 1ns / 1ps
module top(
    input wire clk,
    input wire [15:0] SW,
    input wire [3:0] BTN,
    output wire [3:0] AN,
    output wire [7:0] SEGMENT,
    output wire BTNX4
);
    wire [3:0] Load_A, Load_B, Load_C, Co;
    wire [3:0] A,  A_IN, A1;
    wire [3:0] B,  B_IN, B1;
    wire [3:0] C,  C_IN, C1;
    wire [3:0] Bus;
    wire [31:0] clk_div;
    wire [15:0] num;
    assign num={A,B,C,Bus};
    assign Bus=(SW[8:7]==0)?C:(SW[8:7]==1)?A:(SW[8:7]==2)?B:0;
    MyRegister4b RegA(.clk(clk), .IN(A_IN), .Load(Load_A), .OUT(A));
    Load_Gen m0a(.clk(clk), .clk_1ms(clk_div[17]), .btn_in(BTN[3]),
         .Load_out(Load_A));
    clkdiv m3a(.clk(clk), .rst(1'b0), .clk_div(clk_div));
    AddSub4b m4a(.A(A), .B(4'b0001), .Ctrl(SW[0]), .S(A1));
    assign A_IN = (SW[15] == 1'b0)? A1: Bus;

    MyRegister4b RegB(.clk(clk), .IN(B_IN), .Load(Load_B), .OUT(B));
    Load_Gen m0b(.clk(clk), .clk_1ms(clk_div[17]), .btn_in(BTN[2]),
```

```
        .Load_out(Load_B));
    AddSub4b m4b(.A(B), .B(4'b0001), .Ctrl(SW[1]), .S(B1));
    assign B_IN = (SW[15] == 1'b0)? B1: Bus;

    MyRegister4b RegC(.clk(clk), .IN(C_IN), .Load(Load_C), .OUT(C));
    Load_Gen m0c(.clk(clk), .clk_1ms(clk_div[17]), .btn_in(BTN[1]),
        .Load_out(Load_C));
    MyALU m4c(.A(A),.B(B),.S(SW[6:5]),.C(C1));
    assign C_IN = (SW[15] == 1'b0)? C1: Bus;

    DispNum
m8(.scan(clk_div[18:17]), .HEXS(num), .LES(4'b0), .point(4'b0), .AN(AN)
, .SEGMENT(SEGMENT));
    assign BTNX4 = 1'b0;
endmodule
```

将其下载到板上进行验证，结果如下：

这里只在视频中截取了几个关键状态，分别为将 A 置为 3，将 B 置为 E，令 C=A+B，将 C 的值传给 A，将 B 的值传给 C。可以看到，这一份工程可以成功完成该功能，说明整个工程无误。

四、讨论、心得

这一次实验中，我学会了寄存器的设计及应用，了解了寄存器的作用，这为后续的复杂的逻辑电路的设计做了铺垫。