

# 浙江大学

## 本科实验报告

课程名称:	数字逻辑设计
姓 名:	王浩雄
学 院:	竺可桢学院
系:	混合班
专 业:	计算机科学与技术
学 号:	3230106032
指导教师:	马德

2025 年 5 月 7 日

# 浙江大学实验报告

课程名称： 数字逻辑设计 实验类型：                     

实验项目名称： 寄存器和寄存器传输设计

学生姓名： 王浩雄 专业： 混合班 学号： 3230106032

同组学生姓名： 无 指导老师： 马德

实验地点： 紫金港东 4-509 实验日期： 2025 年 5 月 7 日

## 一、 实验目的和要求

- ① 掌握寄存器传输电路的工作原理；
- ② 掌握寄存器传输电路的设计方法；
- ③ 掌握 ALU 和寄存器传输电路的综合应用。

## 二、 实验内容

- ① 采用寄存器传输原理设计计数器
- ② 基于多路选择器总线的寄存器传输
- ③ 基于 ALU 的数据传输应用设计

## 三、 主要仪器设备

- ① 装有 Vivado 2024.2 Enterprise 的计算机 1 台
- ② SWORD 开发板 1 套

## 四、 操作方法与实验步骤

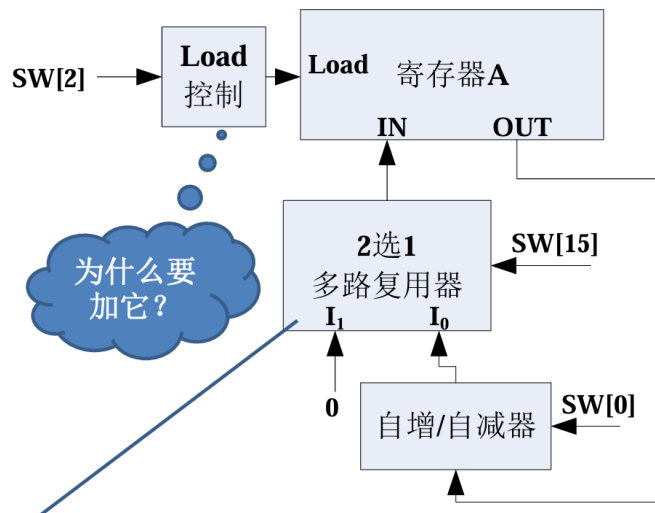
### 1、采用寄存器传输原理设计计数器

① 根据设计要求，新建 Verilog 文件，以行为描述方式完成四位寄存器模块的设计。该寄存器采用 Load 控制反馈，即仅在 Load 为真的时钟上升沿处更新寄存器的存储值。

```
module MyRegister4b(  
    input clk,  
    input Load,  
    input [3:0] IN,  
    output reg [3:0] OUT  
);  
  
always @(posedge clk) begin  
    if (Load) OUT <= IN;  
end  
  
endmodule
```

② 新建 Verilog 文件，完成顶层模块 top1.v 的设计。该顶层模块基于下方的示意图进行设计，并实现如下功能：

- (1) SW[2]：控制寄存器加载
- (2) SW[0]：控制寄存器自增/自减
- (3) SW[15]：控制寄存器清零



```

module top1(
    input wire clk,
    input wire [15:0] SW,
    output wire [7:0] SEGMENT,
    output wire [3:0] AN
);

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

    MyRegister4b RegA(.clk(clk), .IN(A_IN), .Load(Load_A), .OUT(A));
    Load_Gen
m0(.clk(clk), .clk_1ms(clk_div[17]), .btn_in(SW[2]), .Load_out(Load_A));
    clkdiv m3(clk, 1'b0, clk_div);
    addsub_4b m4(.A(A), .B(4'b0001), .Ctrl(SW[0]), .S(A1));
    assign A_IN = (SW[15] == 1'b0)? A1: 4'b0000;
    DispNum m8(.clk(clk), .HEXS({A, A1, A_IN,
4'b0000}), .LES(4'b0), .points(4'b0), .RST(1'b0), .AN(AN), .Segment(SEG
MENT));

endmodule

```

其中，clkdiv、addsub\_4b、DispNum 等模块由过往实验完成编写，此处不再展示代码。

③ 根据顶层模块的设计，四位七段数码管显示的数字含义如下（从左至右）：

第 1 位：寄存器 A 的输出值

第 2 位：自增/自减模块的输出值

第 3 位：寄存器 A 的输入值

第 4 位：恒为 0

④ 根据顶层模块的输入输出，完成如下的引脚约束文件。

```

set_property PACKAGE_PIN AC18 [get_ports clk]
set_property IOSTANDARD LVCMOS18 [get_ports clk]

set_property PACKAGE_PIN AB22 [get_ports {SEGMENT[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[0]}]

```

(省略若干行...)

```
set_property PACKAGE_PIN AA22      [get_ports {SEGMENT[7]}]
set_property IOSTANDARD LVCMOS33   [get_ports {SEGMENT[7]}]

set_property PACKAGE_PIN AD21      [get_ports {AN[0]}]
set_property IOSTANDARD LVCMOS33   [get_ports {AN[0]}]

set_property PACKAGE_PIN AC21      [get_ports {AN[1]}]
set_property IOSTANDARD LVCMOS33   [get_ports {AN[1]}]

set_property PACKAGE_PIN AB21      [get_ports {AN[2]}]
set_property IOSTANDARD LVCMOS33   [get_ports {AN[2]}]

set_property PACKAGE_PIN AC22      [get_ports {AN[3]}]
set_property IOSTANDARD LVCMOS33   [get_ports {AN[3]}]

set_property PACKAGE_PIN AA10      [get_ports {SW[0]}]
set_property IOSTANDARD LVCMOS15   [get_ports {SW[0]}]
```

(省略若干行...)

```
set_property PACKAGE_PIN AF10      [get_ports {SW[15]}]
set_property IOSTANDARD LVCMOS15   [get_ports {SW[15]}]
```

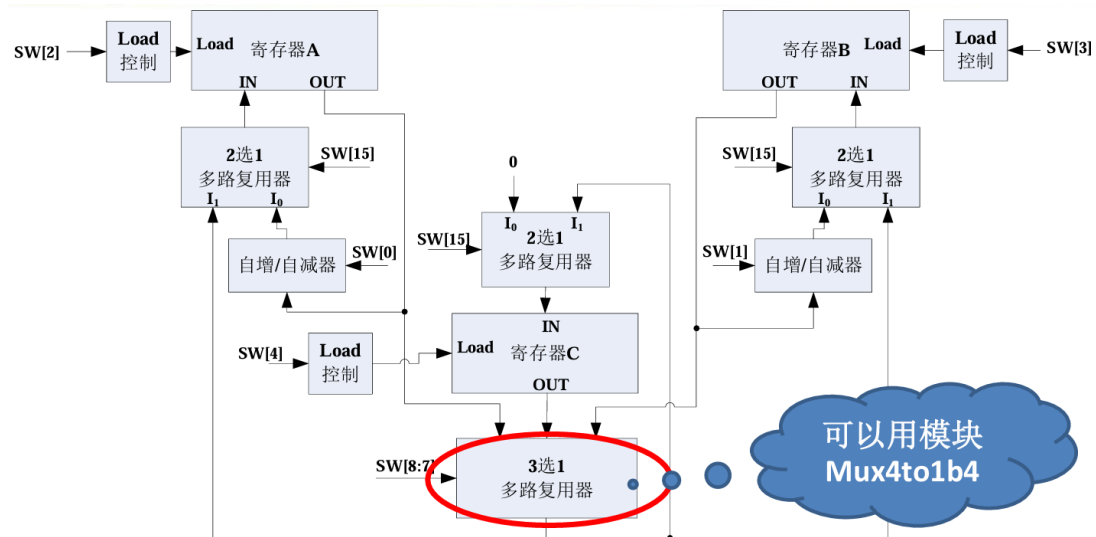
## 2、基于多路选择器总线的寄存器传输

① 新建 Verilog 文件，完成顶层模块 top2.v 的设计。该顶层模块基于下方的示意图进行设计，并实现如下功能：

(1) SW[2]、SW[3]、SW[4]：更新 A、B、C 寄存器，更新的值由 SW[15]和 SW[0]、SW[1]决定

(2) SW[15]=0：通过向上/向下计数修改寄存器 A、B，对 C 清零

(3) SW[15]=1：A、B、C 寄存器之间相互传输，源寄存器由 SW[8:7]确定



```

module top2(
    input wire clk,
    input wire [15:0] SW,
    output wire [7:0] SEGMENT,
    output wire [3:0] AN
);

// 寄存器输出
wire [3:0] A_out, B_out, C_out;
wire [3:0] A_in, B_in, C_in;
wire [3:0] A1, B1; // 自增/自减输出
wire [31:0] clk_div;

// 时钟分频
clkdiv U_div(clk, 1'b0, clk_div);

// Load 控制信号
wire LoadA, LoadB, LoadC;
Load_Gen
uLA(.clk(clk), .clk_1ms(clk_div[17]), .btn_in(SW[2]), .Load_out(LoadA));
Load_Gen
uLB(.clk(clk), .clk_1ms(clk_div[17]), .btn_in(SW[3]), .Load_out(LoadB));
Load_Gen
uLC(.clk(clk), .clk_1ms(clk_div[17]), .btn_in(SW[4]), .Load_out(LoadC));

// 自增/自减模块
addsub_4b addA(.A(A_out), .B(4'b0001), .Ctrl(SW[0]), .S(A1)); // SW[0]
控制加/减
addsub_4b addB(.A(B_out), .B(4'b0001), .Ctrl(SW[1]), .S(B1)); // SW[1]
控制加/减

```

```

// C 输入选择器 (3 选 1)
wire [3:0] muxC_out;
my_mux4to1_4b muxC(
    .s(SW[8:7]),
    .I0(A_out),
    .I1(B_out),
    .I2(C_out),
    .I3(4'b0000),
    .o(muxC_out)
);

// A, B, C 输入选择器
assign A_in = (SW[15] == 1'b0) ? A1 : muxC_out;
assign B_in = (SW[15] == 1'b0) ? B1 : muxC_out;
assign C_in = (SW[15] == 1'b0) ? 4'b0000 : muxC_out;

// 三个寄存器
MyRegister4b RegA(.clk(clk), .IN(A_in), .Load(LoadA), .OUT(A_out));
MyRegister4b RegB(.clk(clk), .IN(B_in), .Load(LoadB), .OUT(B_out));
MyRegister4b RegC(.clk(clk), .IN(C_in), .Load(LoadC), .OUT(C_out));

// 显示模块
DispNum u_disp(
    .clk(clk),
    .HEXS({A_out, B_out, C_out, muxC_out}),
    .LES(4'b0000),
    .points(4'b0000),
    .RST(1'b0),
    .AN(AN),
    .Segment(SEGMENT)
);

endmodule

```

其中，clkdiv、addsub\_4b、DispNum 等模块由过往实验完成编写，此处不再展示代码。

② 根据顶层模块的设计，四位七段数码管显示的数字含义如下（从左至右）：

第 1 位：寄存器 A 的输出值

第 2 位：寄存器 B 的输出值

第 3 位：寄存器 C 的输出值

第 4 位：三选一多路选择器的输出值

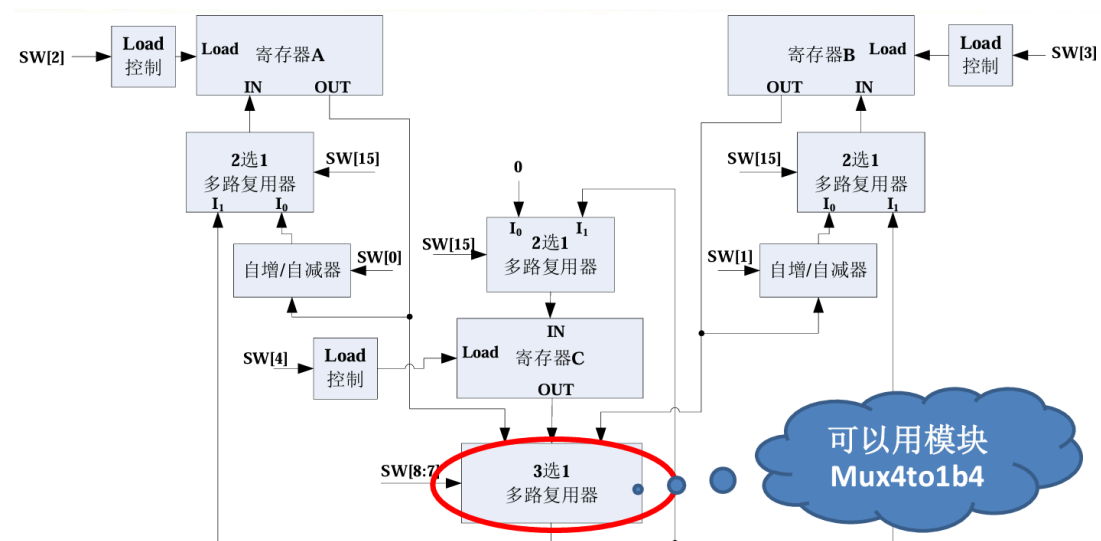
### 3、基于 ALU 的数据传输应用设计

① 新建 Verilog 文件，完成顶层模块 top3.v 的设计。该顶层模块基于下方的示意图进行设计，并实现如下功能：

(1) SW[2]、SW[3]、SW[4]：更新 A、B、C 寄存器，更新的值由 SW[15]和 SW[0]、SW[1]决定

(2) SW[15]=0：通过向上/向下计数修改寄存器 A、B，对 A、B 进行 ALU 运算并使用结果修改寄存器 C

(3) SW[15]=1：A、B、C 寄存器之间相互传输，源寄存器由 SW[8:7]确定



```
module top3(  
    input wire clk,  
    input wire [15:0] SW,  
    output wire [7:0] SEGMENT,  
    output wire [3:0] AN  
);  
  
// 寄存器输出  
wire [3:0] A_out, B_out, C_out;  
wire [3:0] A_in, B_in, C_in;  
wire [3:0] A1, B1; // 自增/自减输出
```



```

wire [31:0] clk_div;

// 时钟分频
clkdiv U_div(clk, 1'b0, clk_div);

// Load 控制信号
wire LoadA, LoadB, LoadC;
Load_Gen
uLA(.clk(clk), .clk_1ms(clk_div[17]), .btn_in(SW[2]), .Load_out(LoadA));
Load_Gen
uLB(.clk(clk), .clk_1ms(clk_div[17]), .btn_in(SW[3]), .Load_out(LoadB));
Load_Gen
uLC(.clk(clk), .clk_1ms(clk_div[17]), .btn_in(SW[4]), .Load_out(LoadC));

// 自增/自减模块
addsub_4b addA(.A(A_out), .B(4'b0001), .Ctrl(SW[0]), .S(A1)); // SW[0]
控制加/减
addsub_4b addB(.A(B_out), .B(4'b0001), .Ctrl(SW[1]), .S(B1)); // SW[1]
控制加/减

// 选择器 (3 选 1)
wire [3:0] muxC_out;
my_mux4to1_4b muxC(
    .s(SW[8:7]),
    .I0(A_out),
    .I1(B_out),
    .I2(C_out),
    .I3(4'b0000),
    .o(muxC_out)
);

// ALU 模块
wire [3:0] ALU_out;
ALU_4b ALU(
    .S(SW[6:5]),
    .A(A_out),
    .B(B_out),
    .C(ALU_out)
);

// A, B, C 输入选择器
assign A_in = (SW[15] == 1'b0) ? A1 : muxC_out;
assign B_in = (SW[15] == 1'b0) ? B1 : muxC_out;
assign C_in = (SW[15] == 1'b0) ? ALU_out : muxC_out;

```

```

// 三个寄存器
MyRegister4b RegA(.clk(clk), .IN(A_in), .Load(LoadA), .OUT(A_out));
MyRegister4b RegB(.clk(clk), .IN(B_in), .Load(LoadB), .OUT(B_out));
MyRegister4b RegC(.clk(clk), .IN(C_in), .Load(LoadC), .OUT(C_out));

// 显示模块
DispNum u_disp(
    .clk(clk),
    .HEXS({A_out, B_out, C_out, ALU_out}),
    .LES(4'b0000),
    .points(4'b0000),
    .RST(1'b0),
    .AN(AN),
    .Segment(SEGMENT)
);

endmodule

```

其中，clkdiv、addsub\_4b、DispNum 等模块由过往实验完成编写，此处不再展示代码。

② 根据顶层模块的设计，四位七段数码管显示的数字含义如下（从左至右）：

第 1 位：寄存器 A 的输出值

第 2 位：寄存器 B 的输出值

第 3 位：寄存器 C 的输出值


第 4 位：ALU 的输出值

四、 实验结果与分析







1、采用寄存器传输原理设计计数器

按该任务的功能设计要求进行下板验证，验证环节如下：

① 验证寄存器的设置初值功能

SW[0]=1，控制寄存器自增		
		
初始状态，寄存器值：5	SW[15]=1	SW[2]=1，寄存器值：0

② 验证寄存器自增、自减功能

SW[0]=0，控制寄存器自增		
		
寄存器值：1	寄存器值：2	寄存器值：3
SW[0]=1，控制寄存器自增		
		
寄存器值：1	寄存器值：6	寄存器值：5

## 2、基于多路选择器总线的寄存器传输

按该任务的功能设计要求进行下板验证，验证环节如下：

- ① 验证 A、B、C 寄存器的设置初值功能
- ② 验证 A、B 寄存器的自增、自减功能

SW[0]=0, SW[15]=0, 拨动 SW[2]控制寄存器 A 自增, B、C 不变		
 A=8	 A=9	 A=10
SW[1]=1, SW[15]=0, 拨动 SW[3]控制寄存器 B 自减, A、C 不变		
 B=5	 B=4	 B=3
SW[15]=0, 拨动 SW[4]控制寄存器 C 清零, A、B 不变		
 初始状态, C=7	 拨动 SW[4]后, C=0	


③ 验证 A、B、C 寄存器之间的传输功能

		
初始状态： A=3，B=5，C=0	SW[15]=1，SW[8:7]=2' b00 SW[4]=1，C 更新为 A 的值	SW[15]=1，SW[8:7]=2' b01 SW[4]=1，C 更新为 B 的值

3、基于 ALU 的数据传输应用设计

按该任务的功能设计要求进行下板验证，验证环节如下：

- ① 验证 A、B、C 寄存器的设置初值功能
- ② 验证 A、B 寄存器的自增、自减功能（与上一实验相同，此处省略现象记录）
- ③ 验证 ALU 运算功能

令 A=3、B=2，数码管最右侧一位表示 A 与 B 的 ALU 运算结果			
			
SW[6:5]=2' b00，加法运算，3+2=5	SW[6:5]=2' b01，减法运算，3-2=1		
			
SW[6:5]=2' b10，与运算，3&2=2	SW[6:5]=2' b11，或运算，3 2=3		

#### ④ 验证寄存器传输功能

 <p>初始状态: A=3, B=5, C=0</p>	 <p>SW[15]=1, SW[8:7]=2' b00 SW[6:5]=2' b00, SW[4]=1 C 更新为 A 的值</p>	 <p>SW[15]=1, SW[8:7]=2' b01 SW[6:5]=2' b00, SW[4]=1 C 更新为 B 的值</p>
 <p>SW[15]=0, SW[8:7]=2' b00 SW[6:5]=2' b00, SW[4]=1 C 更新为 A+B 的值, 即 8</p>	 <p>SW[15]=0, SW[8:7]=2' b00 SW[6:5]=2' b01, SW[4]=1 C 更新为 A-B 的值, 即 0xE</p>	 <p>SW[15]=0, SW[8:7]=2' b00 SW[6:5]=2' b11, SW[4]=1 C 更新为 A B 的值, 即 7</p>

## 五、 讨论、心得

在本次实验中, 我尝试实现了 4 位寄存器模块, 以及一个简单的数据传输应用, 并成功通过下板验证其正确性。本次实验使我对寄存器模块的理解和认识大为加深。此外, 通过自己编写顶层模块, 我对 Verilog 语言的熟悉程度进一步增强, 对分层次、模块化设计的方法更为熟练。