浙江大学

本科实验报告

课程名称：计算机逻辑设计基础

姓　　名：刘晨

学　　院：计算机科学与技术学院

专　　业：图灵班

学　　号：3190104666

指导教师：董亚波

2020 年 12 月 17 日

浙江大学实验报告

课程名称： 计算机逻辑设计基础 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： 刘晨 专业： 图灵 1901 学号： 3190104666

同组学生姓名：白文静 指导老师： 董亚波

实验地点： 东 4-509 实验日期：2020 年 12 月 17 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

内容：

任务 1：采用寄存器传输原理设计计数器

任务 2：基于多路选择器总线的寄存器传输

任务 3：基于 ALU 的数据传输应用设计

原理：

1.寄存器

1.1 寄存器原理

寄存器是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。

寄存器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。

1.2.采取门控时钟的寄存器

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。如下图：

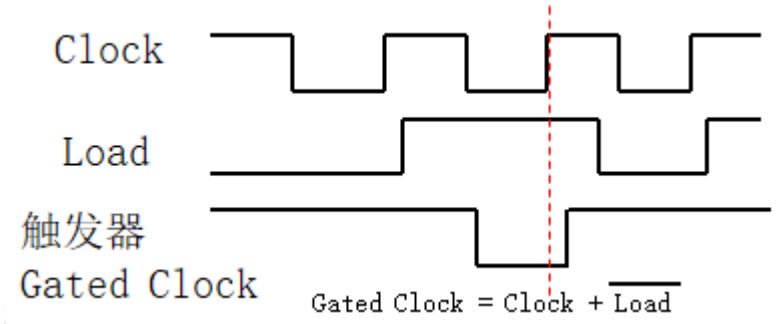

图 1 采取门控时钟的寄存器

1.3. 采用 Load 控制反馈的寄存器

进行有选择地加载寄存器的更可靠方法是：

保证时钟的连续性，且选择性地使用加载控制来改变寄存器的内容。

2.寄存器传输

寄存器传输：寄存器中数据的传输和处理

包含三个基本单元：寄存器组、操作、操作控制

进行的基本操作:加载、计数、移位、加法、按位操作等

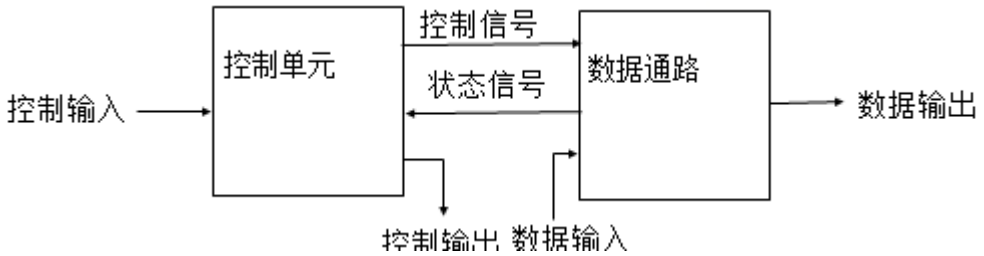

图 2 寄存器传输原理

3.基于多路选择器总线的寄存器传输

优势：由一个多路选择器驱动的总线可以降低硬件开销

劣势：这个结构不能实现多个寄存器相互之间的并行传输操作

4.寄存器传输应用设计

功能：在上一个任务基础上，可以用 A 与 B 的 ALU 计算结果初始化 C

SW[15]=0：初始化 A、B 寄存器，ALU 运算输出控制

SW[15]=1：A、B、C 寄存器之间相互传输

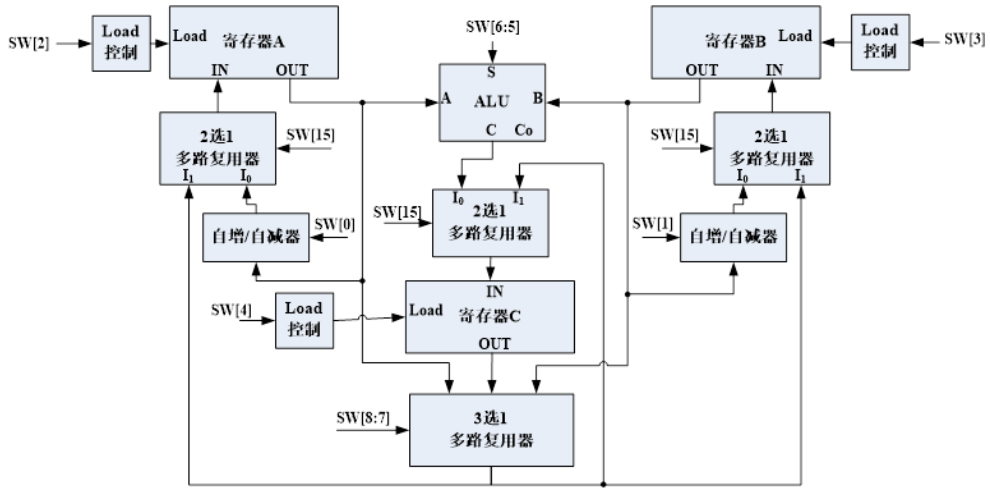

图 3 寄存器传输应用设计示意图

三、实验过程和数据记录

任务 1：采用寄存器传输原理设计计数器

寄存器核心模块设计代码：

```
21  module MyRegister4b(
22      input wire clk,
23      input wire [3:0] IN,
24      input wire Load,
25      output reg [3:0] OUT
26       );
27      always @ (posedge clk) begin
28          if (Load) OUT <= IN;
29      end
30  endmodule
```

Top 模块核心代码：

```
21  module Top(
22     input wire clk,
23     input wire [15:0] SW,
24     output wire [3:0] AN,
25     output wire [7:0] SEGMENT
26      );
27     wire Load_A;
28     wire [3:0] A, A_IN, A1;
29     wire [31:0] clk_div;
30     MyRegister4b RegA(.clk(clk),.IN(A_IN),.Load(Load_A),.OUT(A));
31     Load_Gen m0(.clk(clk),.clk_1ms(clk_div[17]),.btn_in(SW[2]),.Load_out(Load_A));    //寄存器A的Load信号
32     clkdiv m3(clk,1'b0,clk_div);
33     AddSub4b m4(.A(A),.B(4'b0001),.Ctrl(SW[0]),.S(A1));//自增/自减逻辑
34     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
35     DispNum m8(.clk(clk),.HEXS({A, A1, A_IN, 4'b0000}),.LES(4'b0),.points(4'b0),.RST(1'b0),.AN(AN),.Segment(SEGMENT));
36  endmodule
```

任务 2：基于多路选择器总线的寄存器传输

Load_Gen 模块代码：

```
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))  //bt
            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
endmodule
```

Top 模块代码：

```
module Top(
    input wire clk,
    input wire [15:0] SW,
    output wire [3:0] AN,
    output wire [7:0] SEGMENT
    );
    wire [31:0] clk_div;
    wire [3:0] A, A_IN, A1;
    wire [3:0] B, B_IN, B1;
    wire [3:0] C, C_IN, C1;
    wire Load_A, Load_B, Load_C;
    Mux4to14b m1(.I0(A),.I1(B),.I2(C),.I3(C),.s(SW[8:7]),.o(C1));
    clkdiv m3(clk,1'b0,clk_div);
    MyRegister4b RegA(.clk(clk),.IN(A_IN),.Load(Load_A),.OUT(A));
    MyRegister4b RegB(.clk(clk),.IN(B_IN),.Load(Load_B),.OUT(B));
    MyRegister4b RegC(.clk(clk),.IN(C_IN),.Load(Load_C),.OUT(C));
    Load_Gen LGA(.clk(clk),.clk_lms(clk_div[17]),.btn_in(SW[2]),.Load_out(Load_A));  //寄存器A的Load信号
    Load_Gen LGB(.clk(clk),.clk_lms(clk_div[17]),.btn_in(SW[3]),.Load_out(Load_B));  //寄存器B的Load信号
    Load_Gen LGC(.clk(clk),.clk_lms(clk_div[17]),.btn_in(SW[4]),.Load_out(Load_C));  //寄存器B的Load信号
    assign A_IN = (SW[15] == 1'b0)? A1: C1;   //2选1多路复用器，复位寄存器初值
    assign B_IN = (SW[15] == 1'b0)? B1: C1;   //2选1多路复用器，复位寄存器初值
    assign C_IN = (SW[15] == 1'b0)? 4'b0000: C1; //2选1多路复用器，复位寄存器初值
    AddSub4b AS4b0(.A(A), .B(4'b0001), .Ctrl(SW[0]), .S(A1));//自增/自减逻辑
    AddSub4b AS4b1(.A(B), .B(4'b0001), .Ctrl(SW[1]), .S(B1));//自增/自减逻辑
    DispNum m8(.clk(clk),.HEXS({A, B, C, C1}),.LES(4'b0),.points(4'b0),.RST(1'b0),.AN(AN),.Segment(SEGMENT));
endmodule
```

任务 3：基于 ALU 的数据传输应用设计

Top 模块核心代码：

```
module Top(
   input wire clk,
   input wire [15:0] SW,
   output wire [3:0] AN,
   output wire [7:0] SEGMENT
   );
   wire [31:0] clk_div;
   wire [3:0] A, A_IN, A1;
   wire [3:0] B, B_IN, B1;
   wire [3:0] C, C_IN, C1;
   wire Load_A, Load_B, Load_C;
   wire [3:0] ALU_C;
   Mux4to14b m1( .I0(A), .I1(B), .I2(C), .I3(C), .s(SW[8:7]),.o(C1));
   clkdiv m3(clk,1'b0,clk_div);
   MyRegister4b RegA(.clk(clk),.IN(A_IN),.Load(Load_A),.OUT(A));
   MyRegister4b RegB(.clk(clk),.IN(B_IN),.Load(Load_B),.OUT(B));
   MyRegister4b RegC(.clk(clk),.IN(C_IN),.Load(Load_C),.OUT(C));
   Load_Gen LGA(.clk(clk),.clk_lms(clk_div[17]),.btn_in(SW[2]),.Load_out(Load_A));  //寄存器A的Load信号
   Load_Gen LGB(.clk(clk),.clk_lms(clk_div[17]),.btn_in(SW[3]),.Load_out(Load_B));  //寄存器B的Load信号
   Load_Gen LGC(.clk(clk),.clk_lms(clk_div[17]),.btn_in(SW[4]),.Load_out(Load_C));  //寄存器B的Load信号
   assign A_IN = (SW[15] == 1'b0)? A1: C1;   //2选1多路复用器，复位寄存器初值
   assign B_IN = (SW[15] == 1'b0)? B1: C1;   //2选1多路复用器，复位寄存器初值
   assign C_IN = (SW[15] == 1'b0)? ALU_C: C1;   //2选1多路复用器，复位寄存器初值
   AddSub4b AS4b0(.A(A), .B(4'b0001), .Ctrl(SW[0]), .S(A1));//自增/自减逻辑
   AddSub4b AS4b1(.A(B), .B(4'b0001), .Ctrl(SW[1]), .S(B1));//自增/自减逻辑
   myALU ALU0(.A(A),.B(B),.C(ALU_C),.S(SW[6:5]));
   DispNum m8(.clk(clk),.HEXS({A, B, C,C1}),.LES(4'b0), .points(4'b0),.RST(1'b0),.AN(AN),.Segment(SEGMENT));
endmodule
```

四、实验结果分析

总体和预期实验结果内容一致。

硬件描述代码和原理图所要实现的功能一致。

仿真结果符合预期。

Verliog 代码内容实现了预期功能。

在开发板上的实验结果验证了应有的功能。

五、讨论与心得

在这次实验中，我们学会了寄存器和寄存器传输设计，了解了寄存器和寄存器传输设计的方法。我在实验中深刻体会到了实验当中出现的问题和解决的办法，以及协作的重要性。在这次实验当中，我在实验当中遇到了编写 Verilog 代码和 top 模块这几个问题，所幸通过老师和同学的帮助，这些问题顺利解决了，在开发板上面的实验结果也验证了这个结果。