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：仇国智

学　　院：竺可桢学院

专　　业：计算机科学与技术

学　　号：3220102181

指导教师：董亚波

2023 年 11 月 10 日

# 一、 实验目的和要求

1　实验目的

1.1　掌握寄存器传输电路的工作原理

1.2　掌握寄存器传输电路的设计方法

1.3　掌握 ALU 和寄存器传输电路的综合应用

2　实验任务

2.1　任务：基于 ALU 的数据传输应用设计

# 二、 实验内容和原理

1　实验内容

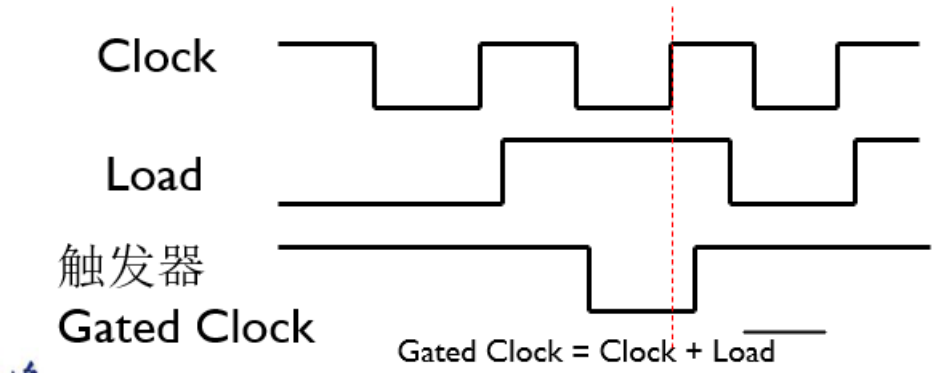
1.1　任务 1：采用寄存器传输原理设计计数器

1.2　任务 2：基于多路选择器总线的寄存器传输

1.3　任务 3：基于 ALU 的数据传输应用设计

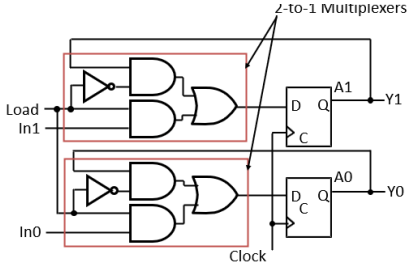
2　实验原理

2.1　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。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。例如： 对于上升沿触发的边沿触发器或负向脉冲触发的主从触发器：

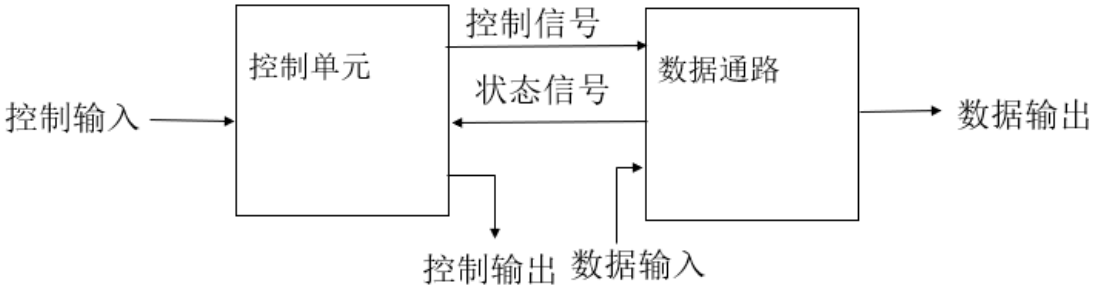



2.2　进行有选择地加载寄存器的更可靠方法是：保证时钟的连续性，

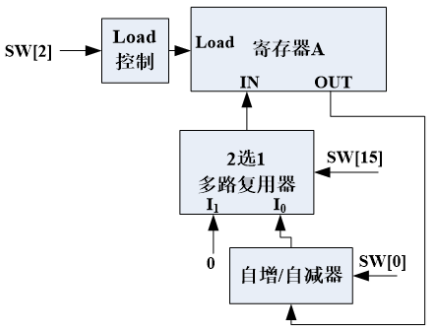
且选择性地使用加载控制来改变寄存器的内容




2.3 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

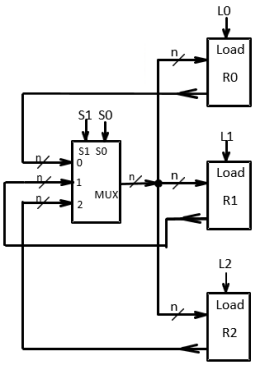



2.4 采用寄存器传输原理的计数器。功能：SW[2]拨动一次，计一次数。Load 控制模块——在 SW[2]的上升沿产生 1 个时钟周期宽度的 Load 信号。自增/自减器可以用 4 位加减法器实现。功能：SW[2]：寄存器加载，SW[0]：向上/下计数，SW[15]：寄存器清零。




2.5 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。




2.6 分频器设计：100MHz 信号通过 50,000,000 次分频后，得到 1Hz 的秒脉冲方波，作为计数器的脉冲输入。

## 三、 实验过程和数据记录

1 采用寄存器传输原理设计计数器

1.1 针对原理框图，设计任务 1 的主要代码。

读写控制 Load_Gen 模块

```
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, 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
     always@(posedge clk) begin     //保存上一个周期 btn 的状态
        old_btn <= btn_out;
     end
endmodule
```

寄存器 MyRegister4b 模块

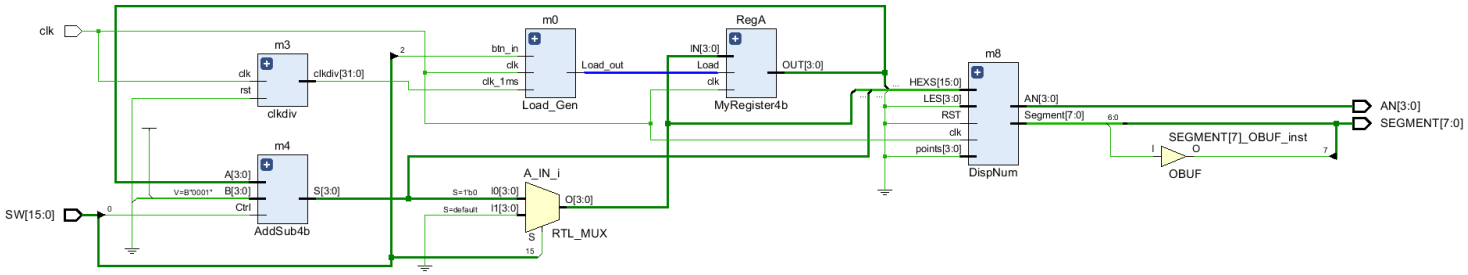
```
module MyRegister4b (
```

```
    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) begin
            OUT <= IN;
        end
    end
endmodule
```

Top 模块

```
module top1 (input wire clk, input wire [15:0] SW, output wire [3:0]
AN, output wire [7:0] SEGMENT);
  wire Load_A;
  wire 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)
  );  //寄存器 A 的 Load 信号
  clkdiv m3 (
      clk,
      1'b0,
      clk_div
  );
  AddSub4b m4 (
      .A(A),
      .B(4'b0001),
      .Ctrl(SW[0]),
      .S(A1),
      .Co(Co)
  );  //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 (
      .clk(clk),
      .HEXS({A, A1, A_IN, 4'b0000}),
      .LES(4'b0),
      .points(4'b0),
      .RST(1'b0),
      .AN(AN),
      .Segment(SEGMENT)
  );
endmodule
```




1.2 仿真测试模块设计

修改设计文件以进行仿真：

Load_Gen 模块修改

```
module Load_Gen_Test(
    input wire clk,
    input wire btn_in,
    output reg Load_out
    );
     initial Load_out = 0;
     wire btn_out=btn_in;
     reg old_btn;
     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
endmodule
```

Top 模块修改

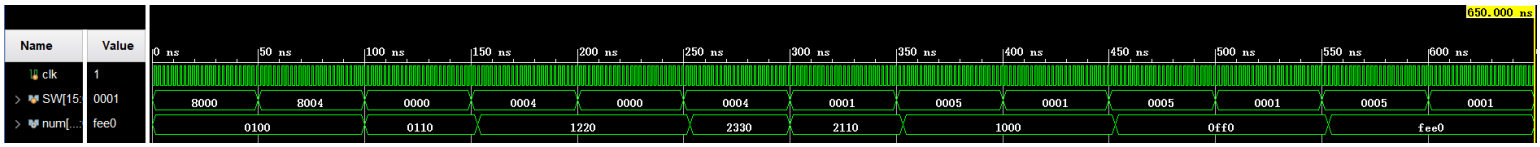
```
module top1_test_source (
    input wire clk,
    input wire [15:0] SW,
    output wire [15:0] num
);
  wire Load_A;
  wire Co;
  wire [3:0] A, A_IN, A1;
  wire [31:0] clk_div;
  MyRegister4b RegA (
      .clk (clk),
      .IN  (A_IN),
      .Load(Load_A),
      .OUT (A)
  );
  Load_Gen_Test m0 (
      .clk(clk),
      .btn_in(SW[2]),
      .Load_out(Load_A)
  );  //寄存器 A 的 Load 信号
  clkdiv m3 (
      clk,
      1'b0,
      clk_div
  );
  AddSub4b m4 (
      .A(A),
      .B(4'b0001),
      .Ctrl(SW[0]),
      .S(A1),
      .Co(Co)
  );  //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
  assign num  = {A, A1, A_IN, 4'b0000};
endmodule
```

仿真激励代码：

```
module top1_test1();
reg clk;
reg [15:0] SW;
wire [15:0]num;
top1_test_source test1 (.clk(clk), .SW(SW), .num(num));
initial begin
```

```
    clk=0;
    while (1) begin
        #1;
        clk=~clk;
    end
end
initial begin
    SW=16'b1000000000000000;
    //SW[15]=1;
    //SW[2]=0;
    #50;
    SW[2]=1;
    #50;
    SW[2]=0;
    SW[15]=0;
    SW[0]=0;
    #50;
    SW[2]=1;
    #50;
    SW[2]=0;
    #50;
    SW[2]=1;
    #50;
    SW[0]=1;
    SW[2]=0;
    #50;
    SW[2]=1;
    #50;
    SW[2]=0;
    #50;
    SW[2]=1;
    #50;
    SW[2]=0;
    #50;
    SW[2]=1;
    #50;
    SW[2]=0;
    #50;
    $finish;
end
endmodule
```

仿真波形：




1.3 编写引脚约束文件。

```
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 IOSTANDARD LVCMOS33 [get_ports {SEGMENT[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[0]}]

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

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 {AN[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {AN[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {AN[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {AN[0]}]

set_property IOSTANDARD LVCMOS15 [get_ports {SW[15]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[14]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[13]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[12]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[11]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[10]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[9]}]
```

```
set_property IOSTANDARD LVCMOS15 [get_ports {SW[8]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[7]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[6]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[5]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[4]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[3]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[2]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[1]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[0]}]
set_property PACKAGE_PIN AA10 [get_ports {SW[15]}]
set_property PACKAGE_PIN AB10 [get_ports {SW[14]}]
set_property PACKAGE_PIN AA13 [get_ports {SW[13]}]
set_property PACKAGE_PIN AA12 [get_ports {SW[12]}]
set_property PACKAGE_PIN Y13 [get_ports {SW[11]}]
set_property PACKAGE_PIN Y12 [get_ports {SW[10]}]
set_property PACKAGE_PIN AD11 [get_ports {SW[9]}]
set_property PACKAGE_PIN AD10 [get_ports {SW[8]}]
set_property PACKAGE_PIN AE10 [get_ports {SW[7]}]
set_property PACKAGE_PIN AE12 [get_ports {SW[6]}]
set_property PACKAGE_PIN AF12 [get_ports {SW[5]}]
set_property PACKAGE_PIN AE8 [get_ports {SW[4]}]
set_property PACKAGE_PIN AF8 [get_ports {SW[3]}]
set_property PACKAGE_PIN AE13 [get_ports {SW[2]}]
set_property PACKAGE_PIN AF13 [get_ports {SW[1]}]
set_property PACKAGE_PIN AF10 [get_ports {SW[0]}]
```

1.4 将 bit 下载至实验版实际验证。

## 2　基于多路选择器总线的寄存器传输

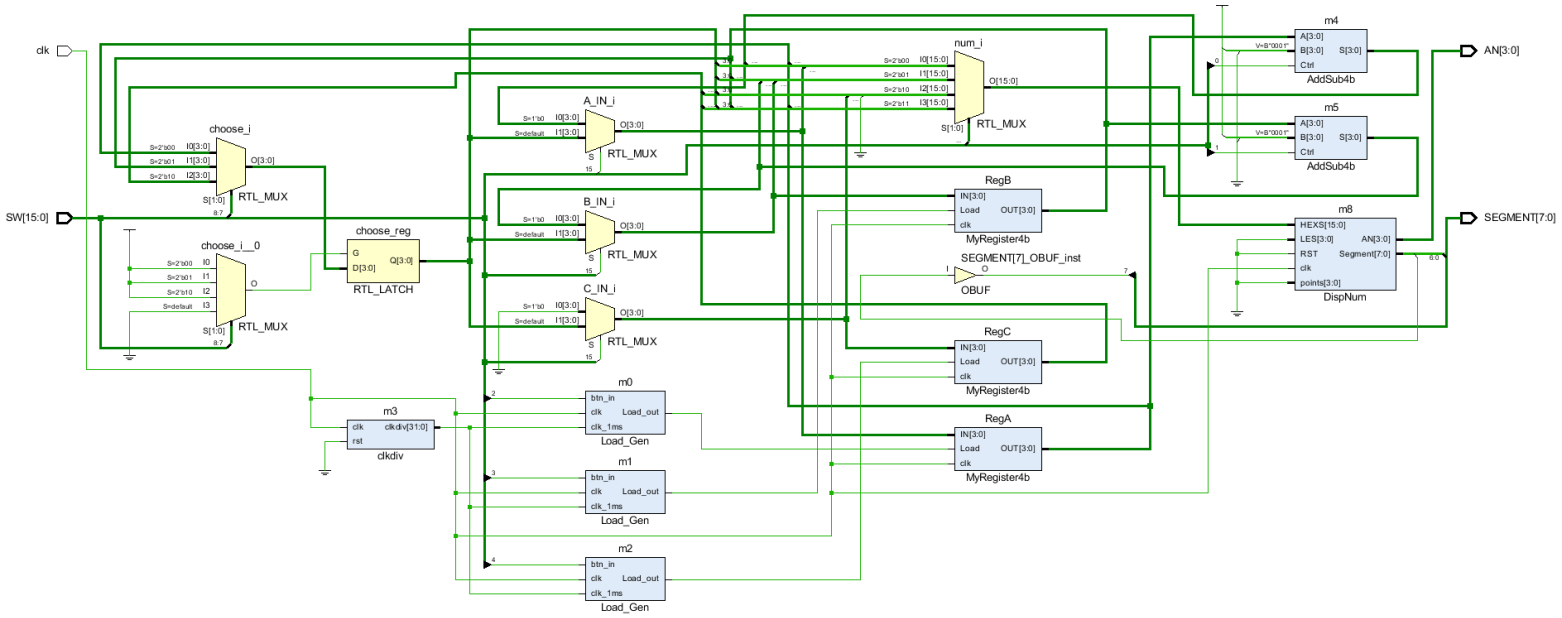
2.1 针对原理框图，设计任务 2 的主要代码。

Top 模块，其余模块同上

```
module top2 (
    input wire clk,
    input wire [15:0] SW,
    output wire [3:0] AN,
    output wire [7:0] SEGMENT
);
  wire Load_A, Load_B, Load_C;
  wire [3:0] Co;
  wire [3:0] A, A_IN, A1;
  wire [3:0] B, B_IN, B1;
  wire [3:0] C, C_IN;
  reg  [ 3:0] choose;
```

```
reg  [15:0] num;
wire [31:0] clk_div;
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 m0 (
    .clk(clk),
    .clk_1ms(clk_div[17]),
    .btn_in(SW[2]),
    .Load_out(Load_A)
);
Load_Gen m1 (
    .clk(clk),
    .clk_1ms(clk_div[17]),
    .btn_in(SW[3]),
    .Load_out(Load_B)
);
Load_Gen m2 (
    .clk(clk),
    .clk_1ms(clk_div[17]),
    .btn_in(SW[4]),
    .Load_out(Load_C)
);
AddSub4b m4 (
```

```
      .A(A),
      .B(4'b0001),
      .Ctrl(SW[0]),
      .S(A1),
      .Co(Co[0])
  );
  AddSub4b m5 (
      .A(B),
      .B(4'b0001),
      .Ctrl(SW[1]),
      .S(B1),
      .Co(Co[1])
  );
  assign A_IN = (SW[15] == 1'b0) ? A1 : choose;
  assign B_IN = (SW[15] == 1'b0) ? B1 : choose;
  assign C_IN = (SW[15] == 1'b0) ? 4'b0000 : choose;
  always @(A, B, C, SW[7:8]) begin
    case (SW[8:7])
      2'b00: choose <= A;
      2'b01: choose <= B;
      2'b10: choose <= C;
    endcase
  end
  always @(SW[9:10], A, A_IN, A1, B, B_IN, B1, C, C_IN, choose) begin
    case (SW[10:9])
      2'b00: num <= {A, A_IN, A1, choose};
      2'b01: num <= {B, B_IN, B1, choose};
      2'b10: num <= {C, C_IN, 4'b0000, choose};
      2'b11: num <= {A, B, C, choose};
    endcase
  end
  DispNum m8 (
      .clk(clk),
      .HEXS(num),
      .LES(4'b0),
      .points(4'b0),
      .RST(1'b0),
      .AN(AN),
      .Segment(SEGMENT)
  );
endmodule
```




2.2 按照同样的引脚约束文件，下载至实验板实际验证

3　基于 ALU 的数据传输应用设计

3.1 针对原理框图，设计任务 3 的主要代码。

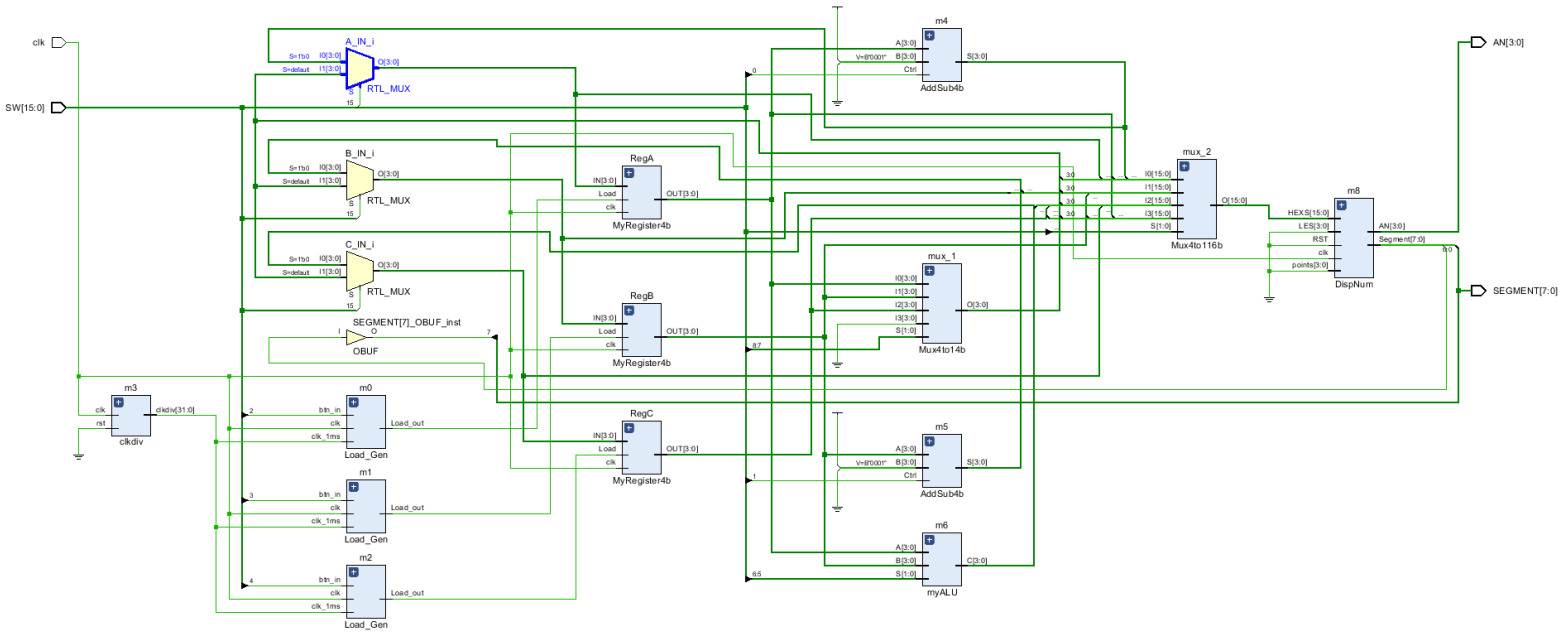
Top 模块，其余模块同上

```
module top3 (
    input wire clk,
    input wire [15:0] SW,
    output wire [3:0] AN,
    output wire [7:0] SEGMENT
);
  wire Load_A, Load_B, Load_C;
  wire [3:0] Co;
  wire [3:0] A, A_IN, A1;
  wire [3:0] B, B_IN, B1;
  wire [3:0] C, C_IN;
  wire [ 3:0] result;
  wire [ 3:0] choose;
  wire [15:0] num;
  wire [31:0] clk_div;
  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 m0 (
    .clk(clk),
    .clk_1ms(clk_div[17]),
    .btn_in(SW[2]),
    .Load_out(Load_A)
);
Load_Gen m1 (
    .clk(clk),
    .clk_1ms(clk_div[17]),
    .btn_in(SW[3]),
    .Load_out(Load_B)
);
Load_Gen m2 (
    .clk(clk),
    .clk_1ms(clk_div[17]),
    .btn_in(SW[4]),
    .Load_out(Load_C)
);
AddSub4b m4 (
    .A(A),
    .B(4'b0001),
    .Ctrl(SW[0]),
    .S(A1),
    .Co(Co[0])
);
```

```
AddSub4b m5 (
    .A(B),
    .B(4'b0001),
    .Ctrl(SW[1]),
    .S(B1),
    .Co(Co[1])
);
myALU m6 (
    .A (A),
    .B (B),
    .S ({SW[6:5]}),
    .C (result),
    .Co(Co[2])
);
assign A_IN = (SW[15] == 1'b0) ? A1 : choose;
assign B_IN = (SW[15] == 1'b0) ? B1 : choose;
assign C_IN = (SW[15] == 1'b0) ? result : choose;
//assign choose = (SW[8:7]==2'b00)? A:
//    (SW[8:7]==2'b01)?B:
//    (SW[8:7]==2'b10)?C:4'b0000;
Mux4to14b mux_1 (
    SW[8:7],
    A,
    B,
    C,
    4'b0000,
    choose
);
Mux4to116b mux_2 (
    SW[10:9],
    {A, A_IN, A1, choose},
    {B, B_IN, B1, choose},
    {C, C_IN, result, choose},
    {A, B, C, choose},
    num
);
//always @(SW[10:9], A, A_IN, A1, B, B_IN, B1, C, C_IN, choose) begin
//case (SW[10:9])
//2'b00: num <= {A, A_IN, A1, choose};
//2'b01: num <= {B, B_IN, B1, choose};
//2'b10: num <= {C, C_IN, result, choose};
//endcase
//end
DispNum m8 (
```

```
      .clk(clk),
      .HEXS(num),
      .LES(4'b0),
      .points(4'b0),
      .RST(1'b0),
      .AN(AN),
      .Segment(SEGMENT)
  );
endmodule
```




3.2 修改设计文件进行仿真

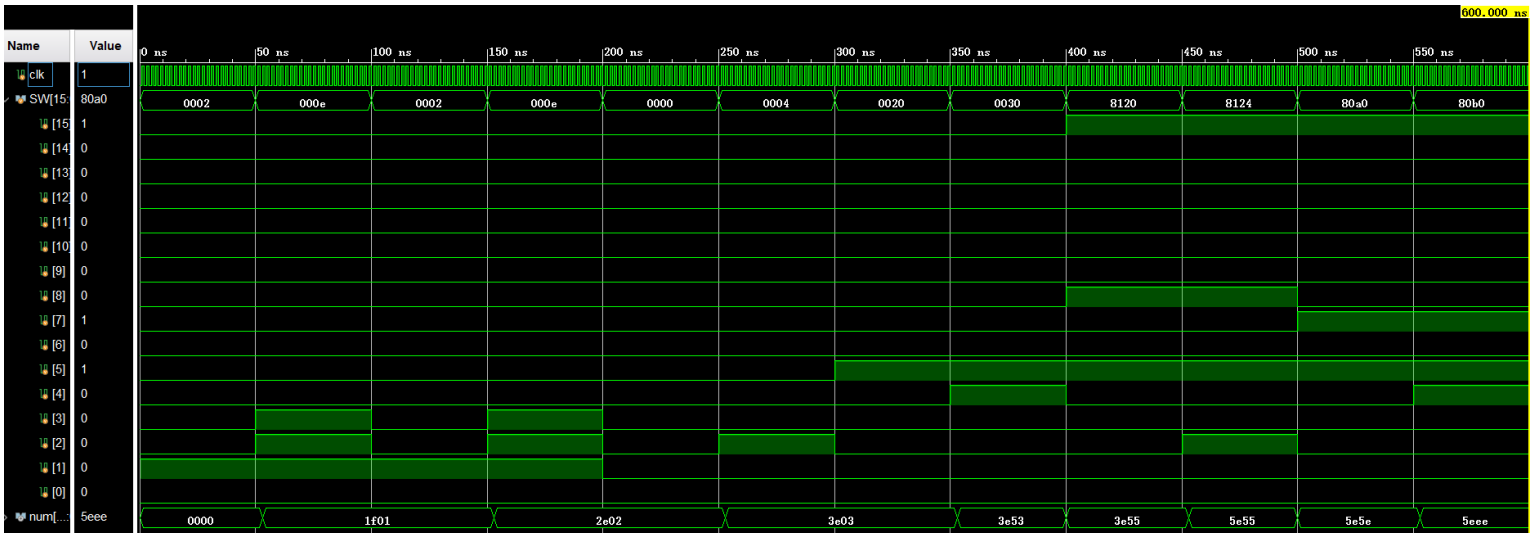
修改设计文件：

Top 模块修改，其余修改同上

```
module top3_test_source (
    input wire clk,
    input wire [15:0] SW,
    output wire [15:0] num
);
  wire Load_A, Load_B, Load_C;
  wire [3:0] Co;
  wire [3:0] A, A_IN, A1;
  wire [3:0] B, B_IN, B1;
  wire [3:0] C, C_IN;
  wire [ 3:0] result;
  wire [ 3:0] choose;
  wire [31:0] clk_div;
```

```
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_Test m0 (
    .clk(clk),
    .btn_in(SW[2]),
    .Load_out(Load_A)
);
Load_Gen_Test m1 (
    .clk(clk),
    .btn_in(SW[3]),
    .Load_out(Load_B)
);
Load_Gen_Test m2 (
    .clk(clk),
    .btn_in(SW[4]),
    .Load_out(Load_C)
);
AddSub4b m4 (
    .A(A),
    .B(4'b0001),
    .Ctrl(SW[0]),
    .S(A1),
    .Co(Co[0])
```

```
  );
  AddSub4b m5 (
      .A(B),
      .B(4'b0001),
      .Ctrl(SW[1]),
      .S(B1),
      .Co(Co[1])
  );
  myALU m6 (
      .A (A),
      .B (B),
      .S ({SW[6:5]}),
      .C (result),
      .Co(Co[2])
  );
  assign A_IN = (SW[15] == 1'b0) ? A1 : choose;
  assign B_IN = (SW[15] == 1'b0) ? B1 : choose;
  assign C_IN = (SW[15] == 1'b0) ? result : choose;
  //assign choose = (SW[8:7]==2'b00)? A:
  //    (SW[8:7]==2'b01)?B:
  //    (SW[8:7]==2'b10)?C:4'b0000;
  Mux4to14b
      mux_1 (
          SW[8:7],
          A,
          B,
          C,
          4'b0000,
          choose
      );
  //always @(SW[10:9], A, A_IN, A1, B, B_IN, B1, C, C_IN, choose) begin
  //case (SW[10:9])
  //2'b00: num <= {A, A_IN, A1, choose};
  //2'b01: num <= {B, B_IN, B1, choose};
  //2'b10: num <= {C, C_IN, result, choose};
  //endcase
  //end
  assign num={A,B,C,choose};
endmodule
```




3.3 按照上面的引脚文件，生成 bit 文件，下载至实验板验证。

四、 实验结果分析

仿真图像和实际操作结果均符合预期

五、 实验心得体会

直接采用原理图设计对于大型工程来说过于复杂，应当转用 verilog 代码。