实验报告

2021 年 4 月 13 日　　　　　　　　　　　　　　成绩：

姓名	周俊佐	学号	19071337	班级	19184115
姓名	颜伟鹏	学号	19071334	班级	19184115
姓名	高炜哲	学号	18081811	班级	18181411
专业	计算机科学与技术		课程名称	计算机组成原理课程设计	
任课老师	章复嘉	指导老师	章复嘉	机位号	14
实验序号	1	实验名称	桶形移位器设计实验		
实验时间	2021.4.12	实验地点	1 教 215	实验设备号	14

一、实验目的和实验要求

1、实验目的：

1.学习桶形移位器的工作原理，掌握桶形移位器的设计方法;

2.学习利用顶层模块来协助实现板级验证的方法;

3.学习采用开关复用的方法来解决输入开关不够用的问题。

2、实验要求：

(1)新建一个工程，使用 Verilog HDL 的行为级描述方式编程实现桶形移位器模块;

(2)编写一个顶层调用模块，测试桶形移位器模块的正确性。调试桶形移位器模块的目的是，检查每一种移位运算和移位控制信号 SHIIFT_OP 的编码对应关系设计是否正确?调试时应该考虑:

1.对每一种移位运算，选择典型的操作数进行运算;

2.对每一种移位运算，至少采用 2 组及以上不同的操作数进行运算。

3.仿真验证，仿真时请观察信号：移位数据 Shift_Data、移位次数 Shift_Num、SHIFT_OP、输入进位标志 Carry_flag、移位结果 Shift_out、移位输出进位标志位 Shift_carry_out;

(3）引脚配置

使用每一个 FPGA 引脚均进行低电平 1.8v 约束；

对于 HDL 模块中的时间边沿端口信号，可以使用 CLOCK_DEDICATED_ROUTE FALSE 取消时钟引脚约束；

开关对应的 FPGA 引脚，设置 PULLDOWN true 选项，使得上电初始化为无效；

在管脚约束文件的第一条语句里开启比特压缩，优化 .bit 文件的大小，语句为：

set_property BITSTREAM.GENERAL.COMPRESS TRUE [current_design]

(4）板级调试

按下按键 1 时：逻辑开关输入 32 为数赋值给被移除数据 Shift_Data；

按下按键 2 时：逻辑开关输入 32 为数赋值给 8 位移位次数 Shift_Num 和 3 位 Shift_OP；

输出信号 Shift_Out 用 8 片数码管显示；

输出标志 Shift_Carry_out 约束 1 个 LED 灯上显示输出

二、实验程序源代码

```
`timescale 1ns / 1ps
// 桶形移位器
module barrelShifter(Shift_Data, Shift_Num, SHFT_OP, Carry_flag, Shift_out,
Shift_carry_out);
	input [31:0]Shift_Data; // Shift Data
	input [7:0]Shift_Num;	// Shift bits
	//SHFT_OP[2:1] shift function
	//SHFT_OP[0] decide shift bits with Shift_Num
	input [2:0]SHFT_OP; //Shift OP
	input Carry_flag;
	output reg [31:0]Shift_out;
	output reg Shift_carry_out;


//	reg [31:0]Shift_out_tmp;
```

```
//      reg Shift_carry_out_tmp;

    always@(*) begin
    case(SHFT_OP[2:1])
        2'b00: begin
            if(Shift_Num == 0)
                Shift_out <= Shift_Data;
            if(Shift_Num >= 0 && Shift_Num <= 32)
                Shift_out <= Shift_Data << Shift_Num;
                Shift_carry_out <= Shift_Data[32-Shift_Num];
            if(Shift_Num > 32)
                Shift_out <= 0;
                Shift_carry_out <= 0;
            end

        2'b01: begin
            if(SHFT_OP[0]==0 && Shift_Num==0)
                Shift_out <= 0;
                Shift_carry_out <= Shift_Data[31];
            if(SHFT_OP[0]==1 && Shift_Num==0)
                Shift_out <= Shift_Data;
            if(Shift_Num >= 1 && Shift_Num <= 32)
                Shift_out <= Shift_Data >> Shift_Num;
                Shift_carry_out <= Shift_Data[Shift_Num-1];
            if(Shift_Num > 32)
                Shift_out <= 0;
                Shift_carry_out <= 0;
            end

        2'b10: begin
            if(SHFT_OP[0]==0 && Shift_Num==0)
                Shift_out <= {32{Shift_Data[31]}};
                Shift_carry_out <= Shift_Data[31];
            if(SHFT_OP[0]==1 && Shift_Num==0)
```

```
                Shift_out <= Shift_Data;
            if(Shift_Num >= 1 && Shift_Num <= 31)
                Shift_out <= {{32{Shift_Data[31]}},Shift_Data}>>Shift_Num;
                Shift_carry_out <= Shift_Data[Shift_Num-1];
            if(Shift_Num >= 32)
                Shift_out <= {32{Shift_Data[31]}};
                Shift_carry_out <= Shift_Data[31];
            end

        2'b11: begin
            if(SHFT_OP[0]==0 && Shift_Num==0)
                Shift_out <= {Carry_flag, Shift_Data[31:1]};
                Shift_carry_out <= Shift_Data[31];
            if(SHFT_OP[0]==1 && Shift_Num==0)
                Shift_out <= Shift_Data;
            if(Shift_Num >= 1 && Shift_Num <= 32)
                Shift_out <= {Shift_Data, Shift_Data} >> Shift_Num;
                Shift_carry_out <= Shift_Data[Shift_Num-1];
            if(Shift_Num > 32)
                Shift_out <= {{32{Shift_Data}},Shift_Data}>>Shift_Num[4:0];
                Shift_carry_out <= Shift_Data[Shift_Data[4:0]-1];
            end
        endcase
    end
endmodule

// 显示模块
module Display(clk, data, which, seg, count, digit);
    input clk;
    input [32:1] data;
    output reg [2:0] which = 0;
    output reg [7:0] seg;

    output reg [10:0] count = 0;
```

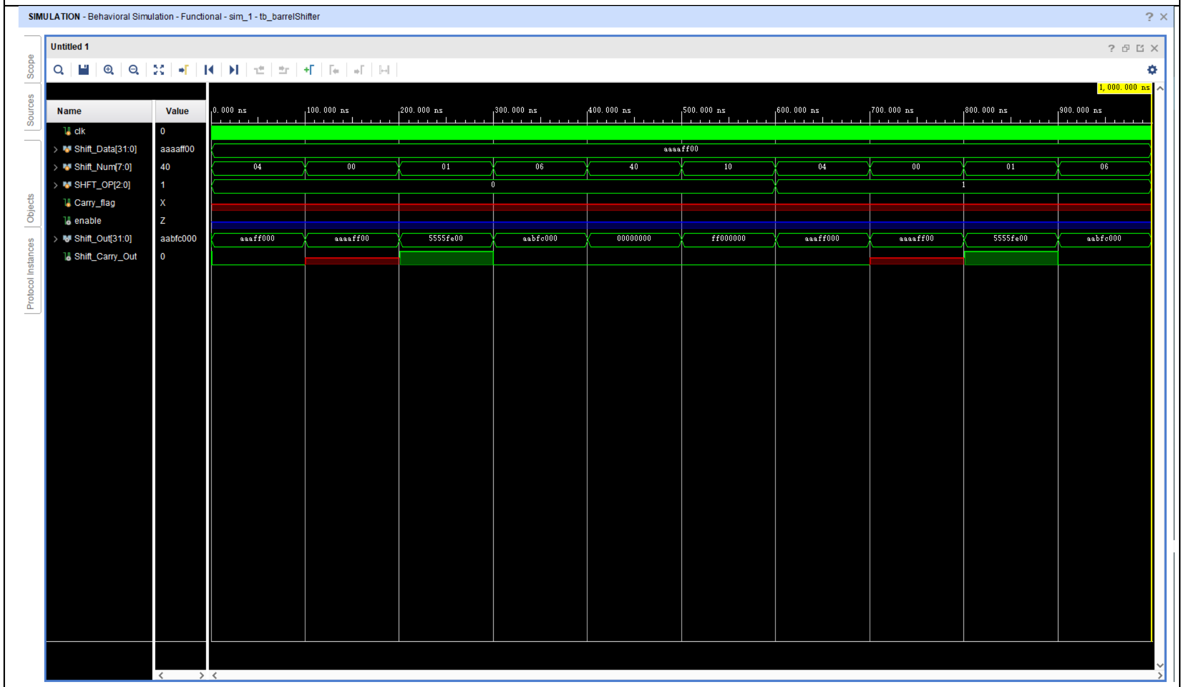
```
always @(posedge clk) count <= count + 1'b1;
always @(negedge clk) if (&count) which <= which + 1'b1;

output reg [3:0] digit;
always @* case (which)
      0: digit <= data[32:29];
      1: digit <= data[28:25];
      2: digit <= data[24:21];
      3: digit <= data[20:17];
      4: digit <= data[16:13];
      5: digit <= data[12:09];
      6: digit <= data[08:05];
      7: digit <= data[04:01];
endcase

always @* case (digit)
      4'h0: seg <= 8'b0000_0011;
      4'h1: seg <= 8'b1001_1111;
      4'h2: seg <= 8'b0010_0101;
      4'h3: seg <= 8'b0000_1101;
      4'h4: seg <= 8'b1001_1001;
      4'h5: seg <= 8'b0100_1001;
      4'h6: seg <= 8'b0100_0001;
      4'h7: seg <= 8'b0001_1111;
      4'h8: seg <= 8'b0000_0001;
      4'h9: seg <= 8'b0000_1001;
      4'hA: seg <= 8'b0001_0001;
      4'hB: seg <= 8'b1100_0001;
      4'hC: seg <= 8'b0110_0011;
      4'hD: seg <= 8'b1000_0101;
      4'hE: seg <= 8'b0110_0001;
      4'hF: seg <= 8'b0111_0001;
endcase
```

```
endmodule // Display

// 顶层模块
module Board(sw, swb, led, clk, which, seg, enable);
    input [1:32] sw; // switch
    input [1:6] swb; // button

    reg [32:1] Shift_Data;
    reg [8:1] Shift_Num;
    reg [3:1] SHFT_OP;
    wire [32:1] Shift_Out;
    wire Shift_Carry_Out;

    output [1:32] led;

    input clk;
    output [2:0] which;
    output [7:0] seg;
    output reg enable = 1;

    always @(posedge swb[1]) Shift_Data    <= sw;
    always @(posedge swb[2]) {Shift_Num, SHFT_OP} <= sw[1:11];

    barrelShifter barrelShifter(
    .clk(clk),
    .SHFT_OP(SHFT_OP),
    .Shift_Data(Shift_Data),
    .Shift_Num(Shift_Num),
    .Carry_flag(swb[6]),
    .Shift_Out(Shift_Out),
    .Shift_Carry_Out(Shift_Carry_Out)
    );

    assign led = {Shift_Carry_Out,31'h00000000};
    Display Display(
    .clk(clk),
    .data(Shift_Out),
    .which(which),
    .seg(seg));

Endmodule

// testbench 测试代码
module tb_barrelShifter();

    reg clk = 0;
    reg [31:0] Shift_Data;
    reg [7:0] Shift_Num;
    reg [2:0] SHFT_OP;
    reg Carry_flag;
    wire enable;
    wire [31:0] Shift_Out;
    wire Shift_Carry_Out;

    always #0.01 clk = ~clk;
    initial begin
        Shift_Num = 8'h04;
        SHFT_OP = 3'b000;
        Shift_Data = 32'haaaaff00;
        #100 Shift_Num = 8'h00;

        #100 Shift_Num = 8'h01;
        #100 Shift_Num = 8'h06;
        #100 Shift_Num = 8'h40;
        #100 Shift_Num = 8'h10;

        #100 Shift_Num = 8'h04;
        SHFT_OP = 3'b001;
        Shift_Data = 32'haaaaff00;
        #100 Shift_Num = 8'h00;
        #100 Shift_Num = 8'h01;
        #100 Shift_Num = 8'h06;
        #100 Shift_Num = 8'h40;
        #100 Shift_Num = 8'h10;

        #100 Shift_Num = 8'h04;
        SHFT_OP = 3'b010;
        Shift_Data = 32'haaaaff00;
        #100 Shift_Num = 8'h00;
        #100 Shift_Num = 8'h01;
        #100 Shift_Num = 8'h06;
        #100 Shift_Num = 8'h40;
        #100 Shift_Num = 8'h10;

        #100 Shift_Num = 8'h04;
        SHFT_OP = 3'b011;
        Shift_Data = 32'haaaaff00;
        #100 Shift_Num = 8'h00;
        #100 Shift_Num = 8'h01;
        #100 Shift_Num = 8'h06;
        #100 Shift_Num = 8'h40;
        #100 Shift_Num = 8'h10;

        #100 Shift_Num = 8'h04;
        SHFT_OP = 3'b100;
        Shift_Data = 32'haaaaff00;
        #100 Shift_Num = 8'h00;
        #100 Shift_Num = 8'h01;
        #100 Shift_Num = 8'h06;
        #100 Shift_Num = 8'h40;
        #100 Shift_Num = 8'h10;
```

```
#100 Shift_Num = 8'h04;
SHFT_OP = 3'b101;
Shift_Data = 32'haaaaff00;
#100 Shift_Num = 8'h00;
#100 Shift_Num = 8'h01;
#100 Shift_Num = 8'h06;
#100 Shift_Num = 8'h40;
#100 Shift_Num = 8'h10;

#100 Shift_Num = 8'h04;
SHFT_OP = 3'b110;
Carry_flag = 0;
Shift_Data = 32'haaaaff00;
#100 Shift_Num = 8'h00;
#100 Shift_Num = 8'h01;
#100 Shift_Num = 8'h06;
#100 Shift_Num = 8'h40;
#100 Shift_Num = 8'h10;

#100 Shift_Num = 8'h04;
SHFT_OP = 3'b110;
Carry_flag = 1;
Shift_Data = 32'haaaaff00;
#100 Shift_Num = 8'h00;
#100 Shift_Num = 8'h01;
#100 Shift_Num = 8'h06;
#100 Shift_Num = 8'h40;
#100 Shift_Num = 8'h10;

#100 Shift_Num = 8'h04;
SHFT_OP = 3'b111;
Shift_Data = 32'haaaaff00;
#100 Shift_Num = 8'h00;
```

```
        #100 Shift_Num = 8'h01;
        #100 Shift_Num = 8'h06;
        #100 Shift_Num = 8'h40;
        #100 Shift_Num = 8'h10;
    end

barrelShifter barrelshifter(
    .clk(clk),
    .SHFT_OP(SHFT_OP),
    .Shift_Data(Shift_Data),
    .Shift_Num(Shift_Num),
    .Carry_flag(Carry_flag),
    .Shift_Out(Shift_Out),
    .Shift_Carry_Out(Shift_Carry_Out));
endmodule
```

三、仿真文件和波形图（有就填，没有就不填）

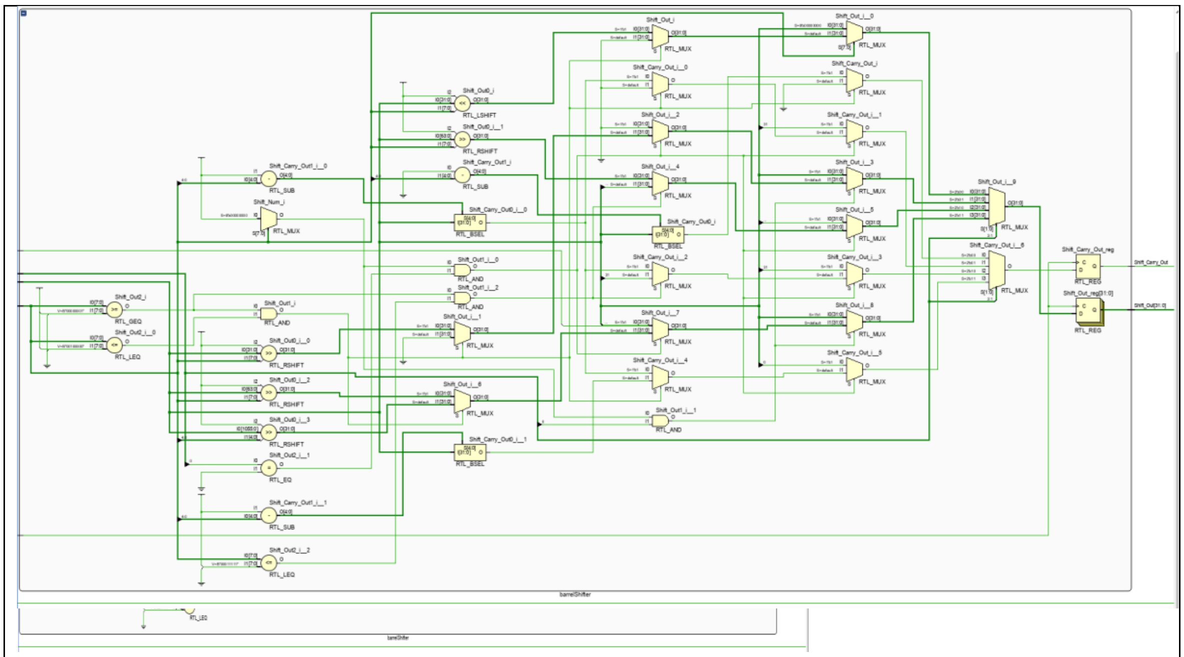


SIMULATION - Behavioral Simulation - Functional - sim_1 - tb_barrelShifter

Untitled 1

Name	Value
clk	0
Shift_Data[31:0]	aaaaff00
Shift_Num[7:0]	40
SHFT_OP[2:0]	1
Carry_flag	X
enable	Z
Shift_Out[31:0]	aabfc000
Shift_Carry_Out	0

四、电路图（有就填，没有就不填）



五、引脚配置（约束文件，有就填，没有就不填））

```
set_property BITSTREAM.GENERAL.COMPRESS TRUE [current_design]


set_property PULLDOWN true [get_ports sw]
set_property IOSTANDARD LVCMOS18 [get_ports sw]
set_property PACKAGE_PIN T3 [get_ports {sw[1]}]
set_property PACKAGE_PIN U3 [get_ports {sw[2]}]
set_property PACKAGE_PIN T4 [get_ports {sw[3]}]
set_property PACKAGE_PIN V3 [get_ports {sw[4]}]
set_property PACKAGE_PIN V4 [get_ports {sw[5]}]
set_property PACKAGE_PIN W4 [get_ports {sw[6]}]
set_property PACKAGE_PIN Y4 [get_ports {sw[7]}]
set_property PACKAGE_PIN Y6 [get_ports {sw[8]}]
set_property PACKAGE_PIN W7 [get_ports {sw[9]}]
set_property PACKAGE_PIN Y8 [get_ports {sw[10]}]
set_property PACKAGE_PIN Y7 [get_ports {sw[11]}]
set_property PACKAGE_PIN T1 [get_ports {sw[12]}]
set_property PACKAGE_PIN U1 [get_ports {sw[13]}]
set_property PACKAGE_PIN U2 [get_ports {sw[14]}]
set_property PACKAGE_PIN W1 [get_ports {sw[15]}]
set_property PACKAGE_PIN W2 [get_ports {sw[16]}]
set_property PACKAGE_PIN Y1 [get_ports {sw[17]}]
set_property PACKAGE_PIN AA1 [get_ports {sw[18]}]
set_property PACKAGE_PIN V2 [get_ports {sw[19]}]
set_property PACKAGE_PIN Y2 [get_ports {sw[20]}]
set_property PACKAGE_PIN AB1 [get_ports {sw[21]}]
set_property PACKAGE_PIN AB2 [get_ports {sw[22]}]
set_property PACKAGE_PIN AB3 [get_ports {sw[23]}]
set_property PACKAGE_PIN AB5 [get_ports {sw[24]}]
set_property PACKAGE_PIN AA6 [get_ports {sw[25]}]
set_property PACKAGE_PIN R2 [get_ports {sw[26]}]
set_property PACKAGE_PIN R3 [get_ports {sw[27]}]
set_property PACKAGE_PIN T6 [get_ports {sw[28]}]
set_property PACKAGE_PIN R6 [get_ports {sw[29]}]
set_property PACKAGE_PIN U7 [get_ports {sw[30]}]
set_property PACKAGE_PIN AB7 [get_ports {sw[31]}]
set_property PACKAGE_PIN AB8 [get_ports {sw[32]}]

# Switch Button
set_property IOSTANDARD LVCMOS18 [get_ports swb]
set_property PACKAGE_PIN R4 [get_ports {swb[1]}]
set_property PACKAGE_PIN AA4 [get_ports {swb[2]}]
set_property PACKAGE_PIN AB6 [get_ports {swb[3]}]
set_property PACKAGE_PIN T5 [get_ports {swb[4]}]
set_property PACKAGE_PIN V8 [get_ports {swb[5]}]
set_property PACKAGE_PIN AA8 [get_ports {swb[6]}]

# LED
set_property IOSTANDARD LVCMOS18 [get_ports led]
set_property PACKAGE_PIN R1 [get_ports {led[1]}]
set_property PACKAGE_PIN P2 [get_ports {led[2]}]
set_property PACKAGE_PIN P1 [get_ports {led[3]}]
set_property PACKAGE_PIN N2 [get_ports {led[4]}]
set_property PACKAGE_PIN M1 [get_ports {led[5]}]
set_property PACKAGE_PIN M2 [get_ports {led[6]}]
set_property PACKAGE_PIN L1 [get_ports {led[7]}]
set_property PACKAGE_PIN J2 [get_ports {led[8]}]
set_property PACKAGE_PIN G1 [get_ports {led[9]}]
set_property PACKAGE_PIN E1 [get_ports {led[10]}]
set_property PACKAGE_PIN D2 [get_ports {led[11]}]
set_property PACKAGE_PIN A1 [get_ports {led[12]}]
set_property PACKAGE_PIN L3 [get_ports {led[13]}]
set_property PACKAGE_PIN G3 [get_ports {led[14]}]
set_property PACKAGE_PIN K4 [get_ports {led[15]}]
set_property PACKAGE_PIN G4 [get_ports {led[16]}]
set_property PACKAGE_PIN K1 [get_ports {led[17]}]
set_property PACKAGE_PIN J1 [get_ports {led[18]}]
set_property PACKAGE_PIN H2 [get_ports {led[19]}]
set_property PACKAGE_PIN G2 [get_ports {led[20]}]
set_property PACKAGE_PIN F1 [get_ports {led[21]}]
set_property PACKAGE_PIN E2 [get_ports {led[22]}]
set_property PACKAGE_PIN D1 [get_ports {led[23]}]
set_property PACKAGE_PIN B1 [get_ports {led[24]}]
set_property PACKAGE_PIN B2 [get_ports {led[25]}]
set_property PACKAGE_PIN N3 [get_ports {led[26]}]
set_property PACKAGE_PIN M3 [get_ports {led[27]}]
set_property PACKAGE_PIN K3 [get_ports {led[28]}]
set_property PACKAGE_PIN H3 [get_ports {led[29]}]
set_property PACKAGE_PIN N4 [get_ports {led[30]}]
set_property PACKAGE_PIN L4 [get_ports {led[31]}]
set_property PACKAGE_PIN J4 [get_ports {led[32]}]


set_property IOSTANDARD LVCMOS18 [get_ports seg]
set_property PACKAGE_PIN H19 [get_ports {seg[7]}]
set_property PACKAGE_PIN G20 [get_ports {seg[6]}]
set_property PACKAGE_PIN J22 [get_ports {seg[5]}]
set_property PACKAGE_PIN K22 [get_ports {seg[4]}]
set_property PACKAGE_PIN K21 [get_ports {seg[3]}]
set_property PACKAGE_PIN H20 [get_ports {seg[2]}]
set_property PACKAGE_PIN H22 [get_ports {seg[1]}]
set_property PACKAGE_PIN J21 [get_ports {seg[0]}]
set_property IOSTANDARD LVCMOS18 [get_ports which]
set_property PACKAGE_PIN N22 [get_ports {which[0]}]
set_property PACKAGE_PIN M21 [get_ports {which[1]}]
set_property PACKAGE_PIN M22 [get_ports {which[2]}]
set_property -dict {IOSTANDARD LVCMOS18 PACKAGE_PIN L21} [get_ports enable]
set_property -dict {IOSTANDARD LVCMOS18 PACKAGE_PIN H4} [get_ports clk]

# [Place 30-574] Poor placement for routing between an IO pin and BUFG.If this
# sub optimal condition is acceptable for this design, you may use the
# CLOCK_DEDICATED_ROUTE constraint in the .xdc file to demote this message to a
# WARNING. However, the use of this override is highly discouraged.
set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets clk_IBUF]
set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets swb_IBUF[1]]
set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets swb_IBUF[2]]
set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets swb_IBUF[3]]
set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets swb_IBUF[4]]
set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets swb_IBUF[5]]
set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets swb_IBUF[6]]
```

六、实验分工（需写清楚每个人负责的工作，以及个人贡献在小组总工作量中的占比）

总计：高炜哲负责 40%的实验，周俊佐和颜伟鹏各负责 30%的实验。

具体分工如下：

高炜哲：代码编写，代码调试，代码仿真测试，报告修改

周俊佐：代码修改，资料收集，测试数据编写，资料分析

颜伟鹏：错误分析，板卡调试，报告初稿

七、实验收获（心得体会等，请以个人为单位分别写）

思考与探索

1：假如桶形移位器的被移位数据 Shift_Date 有 3 种输入来源，如何修改被移位数据输入端呢，是否需要添加附件和控制信号？如何添加？

通过开关复用的方法来完成。

2：某同学对桶形移位器仿真调试时采用了 2 组测试数据分别为：

第一组是：shift__Date=05H,lsl,Shift__Num=2,Carry_flag=0;

第二组是：shift__Date=03H,lsl,Shift__Num=3,Carry_flag=0;

请结合本实验介绍的方法，对这一仿真测试方案做出评价。

这一仿真测试方案不够合理，因为

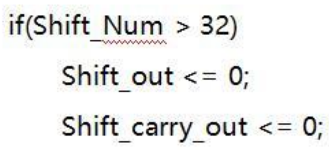
1. 只有 lsl 测试，无法覆盖所有移位运算，无法排查所有错误
2. 0000 0101 逻辑左移 2 位得到 0001 0100，算术左移得到 0001 0100

0000 0011 逻辑左移 3 位得到 0001 1000，算术左移得到 0001 0100
由此可见，这两个测试案例无法区分算数左移和逻辑左移之间的区别，或许存在设计错误的情况。

本次实验进行的过程中，包括在实验分享经验的过程中，遇到了一些自己小组与别人小组的困难，罗列如下：

高炜哲

1：非阻塞赋值，也就是在 always 里，用小于等于号进行赋值，这样可以充分得发挥非阻塞式的赋值所带来的高效率：
也就是形如：



```
if(Shift_Num > 32)
      Shift_out <= 0;
      Shift_carry_out <= 0;
```

2：由于对板卡并不熟悉，在写代码+波形仿真阶段一切顺利，到约束文件以及顶层模块板卡实验阶段时有些生疏。而在波形仿真阶段，开始有波形错误，后经过检查发现是信号定义中位数设置错误。

3：在某些板级测试之中，会出现灯光半明半暗的现象，这一类问题通常会是约束文件有问题。

4：关于上升沿与下降沿，取上升沿意味着，在 0-1 这个过程发挥功能，相反，取下降沿意味着，在 1-0 这个过程发挥功能。
而对于远程实验的电子板而言，点击一下按键，则意味着从 0-1 或者从 1-0 的改变。
但是，对于实验室中的电子板而言，按键还会有自动反弹的效果，例如原来是 0，那么按键意味着由 0 变成 1，再由 1 回到 0，即进行完毕了上升与下降的两个过程。

5：关于实验室中电子板测试，在将已有的数据输入板中之后，进行了一次输入指令的移位测试，这时，如果不继续按数据输入键进行数据的重新输入，或者更新的话。而是继续修改新的指令做移位的操作，那么，这一新指令操作的对象仍然是最开始输入板中的数据，而不是此前电板显示区显示的已经经过第一次移位操作的数据。

我们小组编制的桶形移位器 Barrel shifter 测试结果记录表详情如下：

移位类型	**Carry_flag**	**SHIFT_OP**	**移位次数** Shift_Num	**被移位数据** Shift_Data	**输出数据** **Shift_Out**	**移位进位** **Shift_Carry_Out**
逻辑左移 lsl	0	000	0	1101 1000 0000 0000 0000 1000 0000 0000	D800 0800	无
	0	000	4	1101 1000 0000 0000 0000 1000 0000 0000	8000 0800	1
	0	001	40	1101 1000 0000 0000 0000 1000 0000 0000	0	0
逻辑右移 lsr	0	010	0	1101 1000 0000 0000 0000 1000 0001 1101	0	1
	0	011	0	1101 1000 0000 0000 0000 1000 0001 1101	D800 081D	无
	0	010	4	1101 1000 0000 0000 0000 1000 0001 1101	0D80 0081	1
	0	010	40	1101 1000 0000 0000 0000 1000 0001 1101	0	0
	0	011	4	1101 1000 0000 0000 0000 1000 0001 1101	0D80 0081	0
	0	011	40	1101 1000 0000 0000 0000 1000 0001 1101	0	0
算术右移 asr	0	100	0	1101 1000 0000 0000 0000 1000 0001 1101	FFFF FFFF	1
	0	101	0	1101 1000 0000 0000 0000 1000 0001 1101	D800 081D	无
	0	101	4	1101 1000 0000 0000 0000 1000 0001 1101	FD80 0081	1
	0	100	40	1101 1000 0000	FFFF FFFF	1

				0000 0000 1000 0001 1101		
（带扩展）循环右移 rrx/ror	0	110	0	1101 1000 0000 0000 0000 1000 0001 1101	6C00 040E	1
	0	111	0	1101 1000 0000 0000 0000 1000 0001 1101	D800 081D	无
	0	110	4	1101 1000 0000 0000 0000 1000 0001 1101	0D80 0081	1
	0	110	40	1101 1000 0000 0000 0000 1000 0001 1001	9D800081	1