多功能 ALU 设计及与桶形移位器连接实验报告

2020 年 4 月 19 日　　　　　　　　　　　　　成绩：

姓名	周俊佐	学号	19071337	班级	19185312
姓名	颜伟鹏	学号	19071334	班级	19185312
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	2	实验名称	多功能 ALU 设计及与桶形移位器连接实验		
实验时间	2020.4.19	实验地点	1 教 225	实验设备号	14

一、实验目的和实验要求

实验目的：

（1）学习多功能 ALU 运算器的工作原理，掌握其设计方法;

（2）实现多功能 ALU 模块与桶形移位器模块的连接与调试;

（3）学习板级实验中解决外设资源不够的三种方法。

学习板级实验中解决外设资源不够的三种方法。

实验要求：

（1）学习多功能 ALU 运算器的工作原理，掌握多功能 ALU 运算器的设计方法;

（2）使用 Verilog HDL 的行为级描述方式，在 xilinx Vivado Design Suite 平台上编程实现一个最多支持 16 种运算的多功能 ALU 运算器;

(3)编写一个顶层调用模块，测试多功能 ALU 运算器的功能的正确性;

(4)合并前一个实验的顶层模块，对多功能 ALU 运算器与桶形移位器模块的连接的正确性进行调试;

(5)学习板级实验中，解决外设资源不够的三种方法;

(6)学习封装 IP 核的方法，将带桶形移位器的多功能 ALU 运算器封装成 IP 核供后续实验调用;

(7)撰写实验报告・

实验要求细化说明：

（1）打开实验- -的工程使用 Verilog HDL 的行为级描述方式，编程实现- -个新的多功能 ALU 运算器模块:

（2）编写一个顶层调用模块，测试 ALU 运算器模块的正确性。调试 ALU 模块的目的是，检查每- -种运算设计和生成标志位 NZCV 的逻辑是否正确?调试时应考虑:

1)对每- -种运算，选择具有典型性的操作数或边界值进行运算:

2)对每- -种运算，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:操作数 A.操作数 B、ALU_ OP.移位输出进位标志

位 Shift carry. _out. 来自 CPSR 寄存器的进位/借位标志 CF 和溢出标志 VF、运算结果了 F、标志位 NZCV;

（3）

合并并修改实验--和实验二的两个顶层调用模块成为-个新的项层调用模块，测试连接桶形移位器模块和 ALU 运算器模块后的正确性。调试的目的是，检查桶形移位器和 ALU 运算器之间数据传送是否正确，为后续整机实验预备功能正确的模块。所以，调试时应该考虑:

1)桶形移位器的输出数据 Shift _out,是否与 ALU 的操作数 B 连接在一起:

2)当 ALU 进行逻辑运算时，ALU 输出的进位标志 CF 是否与移位器输出的 Shift carry. _out 相等? ALU 输出的溢出标志 VF 是否与从 CPSR 读入的 VF 一样?

（4）板级验证:

1)配置管脚:采用 12.2.2 小节所述的解决外设不够的三种方法之一来设计本实验的模块输入输出端口结构。如果采用方法二来进行设计，那么请参考表 12-5,为顶层模块的输入输出引脚添加管脚约束。请注意:表 12-5 给出的是桶形移位器模块和多功能 ALU 模块联调的引脚配置方案。如果采用方法三来设计本实验的模块输入输出端口结构，那么请另行自定义顶层模块的输入输出引脚配置方案。如果采用其他型号的板卡进行板级验证，那么请根据该板卡的外设资源，自行设计管脚约束方案:

2) 生成*.bit 文件，下载到 HDU-XL-01 教学开发板上的 FPGA 中，完成板级调试验证。

（5）x 将调试好的桶形移位器和多功能 ALU 运算器模块连接后，封装成 IP 核供后续实验调用:

（6）撰写实验报告:

撰写报告时要求叙述以下内容，以及你对本实验的思考与探索。

1)请自行选择不少于 8 组典型数据，对 ALU 运算器进行功能测试，将实验结果记录到表中，分析你的实验结果是否符合你的预期:如果不符合，请分析原因;

2)建议:

■用一组典型的有代表性的数据或者边界值数据，测试某 1 种或几种功能，不一定用 1 组数据测试完桶形移位器或 ALU 运算器的全部功能;另一方面，对于 ALU 运算器的每一种功能，**至少安排 2 组具有典型性的数据进行测试**:

■根据需要复制表格 12-6 的数量，**设计几组测试数据就复制几张表格**，每组测试数据对应表格内的运算功能，可根据需要删减。

3)请自行选择不少于 5 组典型数据，对桶形移位器和 ALU 运算器模块连接后的功能进行测试，将实验结果记录到表中，分析你的实验结果是否符合你的预期:如果不符合，请分析原因。

二、实验程序源代码

ALU_Board.v:顶层模块

```verilog
`timescale 1ns / 1ps

module ALU_Board(sw, swb, led, clk, which, seg, enable);
    input [1:32] sw;//????
    input [1:6] swb;//��ť

    reg [32:1] A,B,data;
    reg [4:1] ALU_OP;
    reg [3:1] SHIFT_OP;
    reg Shift_Carry_Out = 0,CF,VF;
    wire [32:1] F;
    wire [4:1] NZCV;
    reg [2:1]cnt = 0;
    output [1:32] led;

    input clk;
    output [2:0] which;
    output [7:0] seg;
    output reg enable = 1;
    always @(posedge swb[1]) A = sw;
    always @(posedge swb[2]) B = sw;
    always @(posedge swb[3]) {ALU_OP,CF,VF,Shift_Carry_Out} = sw[1:7];
    always @(posedge swb[4])
    begin
        begin
            case(cnt)
            0:begin data <= A; end //first time hit swb[1]: input A
            1:begin data <= B; end //second time hit swb[1]: input B
            2:begin data <= F; end //third time hit swb[1]: input [32:29]ALU_OP; [28, 27, 26] CF, VF,Shift_Carry_Out
            endcase
            cnt <= (cnt+1) % 3;
        end
    end
    ALU ALU_Instance(ALU_OP,A,B,Shift_Carry_Out,CF,VF,NZCV,F);

    assign led[1:4] = NZCV;
    assign led[5]=Shift_Carry_Out;
    assign led[31:32] = cnt;

    Display Display_Instance(.clk(clk), .data(data),
    .which(which), .seg(seg));
```

```
endmodule
```

ALU.v:运算器

```
`timescale 1ns / 1ps

module ALU(ALU_OP, A, B, Shift_Carry_Out, CF, VF, NZCV, F);
```

```
    // INPUT
    input [31:0] A;
    input [3:0] ALU_OP;
    input CF; // from CSPR
    input VF; // from CSPR
    // FROM BARRELSHIFTER
    input  [31:0] B;
    input  Shift_Carry_Out;
    // OUTPUT
    output reg [31:0] F; // result
    output reg [3:0] NZCV; // push into CSPR
    // C32 FLAG
    reg C32;

    // ALU_CF:
    // ADD: C32 = 1 -> C = 1
    // SUB: C32 = 1 -> C = 0
    // ALU_VF: C32 ^ C31

    always@(*) begin
    case(ALU_OP)
         4'b0000:begin
            F <= A & B;
            // C = NZCV[1]
            NZCV[1] <= Shift_Carry_Out;
            // V = NZCV[0]
            NZCV[0] <= VF;end
         4'b0001:begin
            F = A ^ B;
            // C = NZCV[1]
            NZCV[1] <= Shift_Carry_Out;
            // V = NZCV[0]
            NZCV[0] <= VF;end
         4'b0010:begin
            {C32, F} <= A - B;
            // C = NZCV[1]
            NZCV[1] <= ~C32;
            // V = NZCV[0]
            NZCV[0] <= C32 ^ F[31];end
         4'b0011:begin
            {C32, F} <= B - A;
            // C = NZCV[1]
            NZCV[1] <= ~C32;
            // V = NZCV[0]
            NZCV[0] <= C32 ^ F[31];end
         4'b0100:begin
            {C32, F} <= A + B;
            // C = NZCV[1]
            NZCV[1] <= C32;
            // V = NZCV[0]
            NZCV[0] <= C32 ^ F[31];end
         4'b0101:begin
            {C32, F} <= A + B + CF;
            // C = NZCV[1]
            NZCV[1] <= C32;
            // V = NZCV[0]
            NZCV[0] <= C32 ^ F[31];end
         4'b0110:begin
            {C32, F} <= A - B + CF - 1;
            // C = NZCV[1]
            NZCV[1] <= ~C32;
            // V = NZCV[0]
            NZCV[0] <= C32 ^ F[31];end
         4'b0111:begin
            F <= B - A + CF - 1;
            // C = NZCV[1]
            NZCV[1] <= ~C32;
            // V = NZCV[0]
            NZCV[0] <= C32 ^ F[31];end
         4'b1000:begin
            F <= A;
            // C = NZCV[1]
            NZCV[1] <= Shift_Carry_Out;
            // V = NZCV[0]
            NZCV[0] <= VF;end
         4'b1010:begin
            F <= A - B + 4;
            // C = NZCV[1]
            NZCV[1] <= ~C32;
            // V = NZCV[0]
            NZCV[0] <= C32 ^ F[31];end
```

```
        4'b1100:begin
            F <= A|B;
            // C = NZCV[1]
            NZCV[1] <= Shift_Carry_Out;
            // V = NZCV[0]
            NZCV[0] <= VF;end
        4'b1101:begin
            F <= B;
            // C = NZCV[1]
            NZCV[1] <= Shift_Carry_Out;
            // V = NZCV[0]
            NZCV[0] <= VF;end
        4'b1110:begin
            F <= A & (~B);
            // C = NZCV[1]
            NZCV[1] <= Shift_Carry_Out;
            // V = NZCV[0]
            NZCV[0] <= VF;end
        4'b1111:begin
            F <= ~B;
            // C = NZCV[1]
            NZCV[1] <= Shift_Carry_Out;
            // V = NZCV[0]
            NZCV[0] <= VF;end
    endcase
    // N = NZCV[3]
    NZCV[3] <= F[31];
    // Z = NZCV[2]
    NZCV[2] <= (F == 0) ? 1 : 0;
    end

endmodule
```

Desplay.v

```
`timescale 1ns / 1ps

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
```

ALU_barrelShifter.v 顶层模块

```
`timescale 1ns / 1ps
module ALU_barrelShifter(
    // input from shifter
    input [31:0] Shift_Data,
    input [31:0] Shift_Num,
    input [3:0] SHFT_OP,
    // input from alu
    input [31:0] A,
    input [3:0] ALU_OP,
    input CF,
    input VF,
    // output
    output [31:0] F,
    output [3:0] NZCV
);
barrelShifter shifter(
    // input
    .Shift_Data     (Shift_Data),
    .Shift_Num      (Shift_Num),
    .SHFT_OP        (SHFT_OP),
    .Carry_flag     (CF),
    // output
    .Shift_Out      (Shift_Out),
    .Shift_Carry_Out(Shift_Carry_Out)
);
```

```
ALU alu(
    // input
    .A              (A),
    .B              (Shift_Out),
    .Shift_Carry_Out(Shift_Carry_Out),
    .ALU_OP         (ALU_OP),
    .CF             (CF),
    .VF             (VF),
    // output
    .F              (F),
    .NZCV           (NZCV)
);
endmodule
```

BarrelShIfte.v 桶形移位器

```
`timescale 1ns / 1ps
module barrelShifter(Shift_Data, Shift_Num, SHFT_OP, Carry_flag, Shift_Out, Shift_Carry_Out);

    // INPUT
    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
output logic [31:0] Shift_Out;
output logic Shift_Carry_Out;

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

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

    2'b10: begin
        if (Shift_Num == 0 && SHFT_OP[0] == 1) begin
```

```
                Shift_Out <= Shift_Data;
                Shift_Carry_Out <= 1'bx;
            end
            if (Shift_Num == 0 && SHFT_OP[0] == 0) begin
                Shift_Out <={32{Shift_Data[31]}};
                Shift_Carry_Out <= Shift_Data[31];
            end
            else if (Shift_Num >= 1 && Shift_Num <= 31) begin
                Shift_Out <= ({{32{Shift_Data[31]}},Shift_Data}>>Shift_Num);
                Shift_Carry_Out <= Shift_Data[Shift_Num-1];
            end
            else begin
                Shift_Out <= {32{Shift_Data[31]}};
                Shift_Carry_Out <= Shift_Data[31];
            end
        end

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

测试代码 tb_ALU.v

```
module tb_ALU();
    // INPUT
    reg [31:0] A;
    reg [3:0] ALU_OP;
    reg CF; // from CSPR
    reg VF; // from CSPR

    // FROM BARRELSHIFTER
    reg  [31:0] B;
    reg  Shift_Carry_Out;
```

```
    // OUTPUT
    wire [31:0] F; // result
    wire [3:0] NZCV; // push into CSPR
```

```
    initial begin
        // 1
        A = 32'hac963a55;
        B = 32'h365aacf9;
        CF = 1'b0;
        VF = 1'b0;
        ALU_OP = 4'b0000;
        Shift_Carry_Out = 1'b1;
        #50;

        //2
        A = 32'hac963a55;
        B = 32'h365aacf9;
        CF = 1'b0;
        VF = 1'b0;
        ALU_OP = 4'b0001;
        Shift_Carry_Out = 1'b1;
        #50;
```

```
        //3
        A = 32'hac963a55;
        B = 32'h365aacf9;
        CF = 1'b0;
        VF = 1'b0;
        ALU_OP = 4'b0010;
        Shift_Carry_Out = 1'b1;
        #50;
```

```
        //4
        A = 32'hac963a55;
        B = 32'h365aacf9;
        CF = 1'b0;
        VF = 1'b1;
        ALU_OP = 4'b0011;
        Shift_Carry_Out = 1'b1;
        #50;
```

```
        //5
        A = 32'hac963a55;
        B = 32'h365aacf9;
        CF = 1'b0;
        VF = 1'b0;
        ALU_OP = 4'b0100;
        Shift_Carry_Out = 1'b1;
        #50;
```

```
        //6
        A = 32'hac963a55;
        B = 32'h365aacf9;
        CF = 1'b1;
        VF = 1'b0;
        ALU_OP = 4'b0101;
        Shift_Carry_Out = 1'b1;
        #50;
```

```
        //7
        A = 32'hac963a55;
        B = 32'h365aacf9;
        CF = 1'b0;
        VF = 1'b0;
        ALU_OP = 4'b0110;
        Shift_Carry_Out = 1'b1;
        #50;
```

```
        //8
        A = 32'hac963a55;
        B = 32'h365aacf9;
        CF = 1'b1;
        VF = 1'b0;
        ALU_OP = 4'b0111;
        Shift_Carry_Out = 1'b1;
        #50;
```

```
        //9
        A = 32'hac963a55;
        B = 32'h365aacf9;
        CF = 1'b1;
        VF = 1'b1;
        ALU_OP = 4'b1000;
        Shift_Carry_Out = 1'b1;
        #50;
```

```
        //10
        A = 32'hac963a55;
        B = 32'h365aacf9;
        CF = 1'b0;
        VF = 1'b0;
        ALU_OP = 4'b1010;
        Shift_Carry_Out = 1'b1;
        #50;
```

```
        //11
        A = 32'hac963a55;
        B = 32'h365aacf9;
        CF = 1'b0;
        VF = 1'b0;
        ALU_OP = 4'b1100;
        Shift_Carry_Out = 1'b1;
        #50;
```

```
        //12
        A = 32'hac963a55;
        B = 32'h365aacf9;
        CF = 1'b0;
        VF = 1'b1;
        ALU_OP = 4'b1101;
        Shift_Carry_Out = 1'b1;
        #50;
```

```
        //13
        A = 32'hac963a55;
        B = 32'h365aacf9;
        CF = 1'b0;
        VF = 1'b0;
        ALU_OP = 4'b1110;
        Shift_Carry_Out = 1'b1;
        #50;
```

```
        //14
        A = 32'hac963a55;
        B = 32'h365aacf9;
        CF = 1'b0;
        VF = 1'b0;
        ALU_OP = 4'b1111;
        Shift_Carry_Out = 1'b1;
        #50;
    end
```

```
    ALU alu(
        .A(A),
        .B(B),
        .ALU_OP(ALU_OP),
        .CF(CF),
        .VF(VF),
        .Shift_Carry_Out(Shift_Carry_Out),
        .F(F),
        .NZCV(NZCV));
```

```
endmodule
```

测试代码 tb_ALU_barrelShifter.v

```
module tb_ALU_barrelShifter();
    reg [32:1] A;
    reg [32:1] Shift_Data;
    reg [8:1] Shift_Num;
    reg [3:1] SHFT_OP;
    reg [4:1] ALU_OP;
    reg CF,VF;

    wire [32:1] Shift_Out;
    wire Shift_Carry_Out=0;
    wire [32:1] F;
    wire [4:1] NZCV;

    initial begin
        SHFT_OP=3'b001; Shift_Num=8'd3;
        Shift_Data = 32'h3ac50001;
        ALU_OP=4'b0101;
        A = 32'h9a4d882b;
        CF=1'b1; VF=1'b0;
        #100;

        SHFT_OP=3'b000; Shift_Num=8'd35;
        Shift_Data = 32'h2342689a;
        ALU_OP=4'b1100;
        A = 32'hee34fa12;
        CF=1'b0; VF=1'b1;
        #100;

        SHFT_OP=3'b010; Shift_Num=8'd0;
        Shift_Data = 32'hf0000000;
        ALU_OP=4'b0001;
        A = 32'h8b49da1f;
        CF=1'b0; VF=1'b0;
        #100;

        SHFT_OP=3'b010; Shift_Num=8'd12;
        Shift_Data = 32'h5f5555f5;
        ALU_OP=4'b0111;
        A = 32'h98463f4f;
        CF=1'b0; VF=1'b0;
        #100;

        SHFT_OP=3'b100; Shift_Num=8'd6;
        Shift_Data = 32'h8a9d029d;
        ALU_OP=4'b1010;
        A = 32'h4b5c6d7e;
        CF=1'b1; VF=1'b1;
        #100;

        SHFT_OP=3'b101; Shift_Num=8'd40;
        Shift_Data = 32'h8a9d029d;
        ALU_OP=4'b1101;
        A = 32'hbcdef123;
        CF=1'b1; VF=1'b1;
        #100;
```

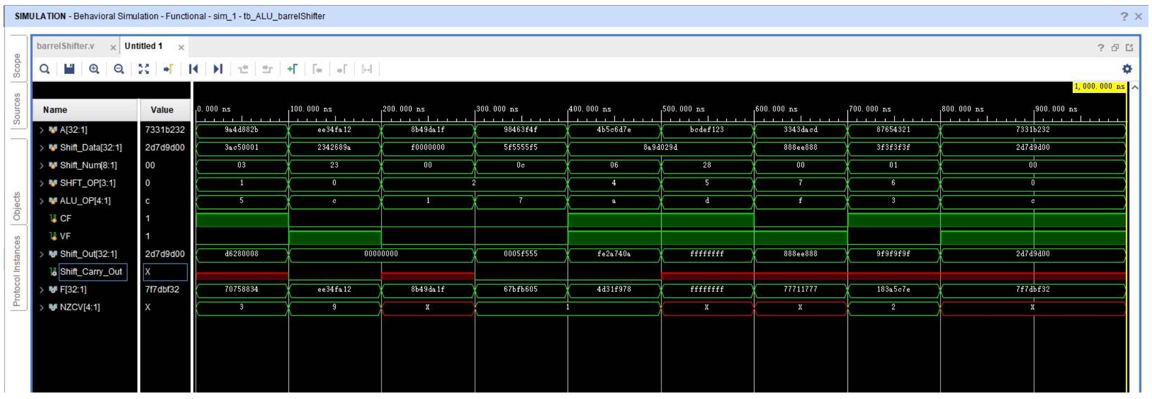
```
        SHFT_OP=3'b111; Shift_Num=8'd0;
        Shift_Data = 32'h888ee888;
        ALU_OP=4'b1111;
        A = 32'h3343dacd;
        CF=1'b0; VF=1'b1;
        #100;
```

```
        SHFT_OP=3'b110; Shift_Num=8'd1;
        Shift_Data = 32'h3f3f3f3f;
        ALU_OP=4'b0011;
        A = 32'h87654321;
        CF=1'b1; VF=1'b0;
        #100;
```

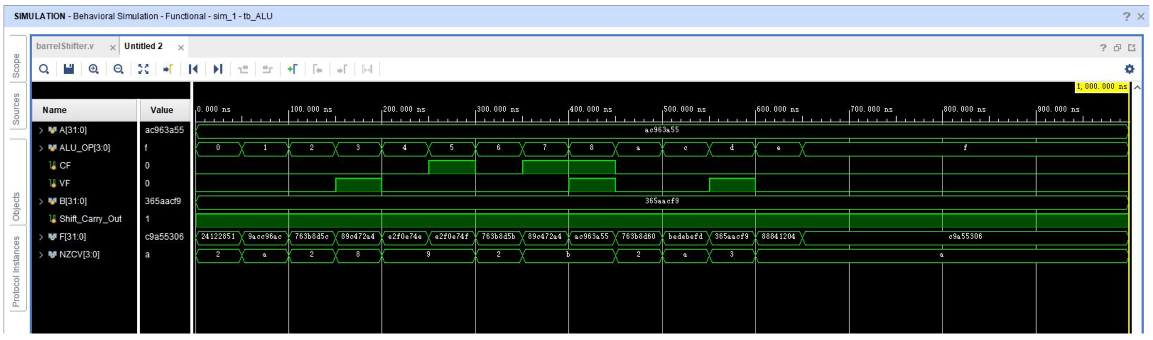
```
        SHFT_OP=3'b000; Shift_Num=8'd0;
        Shift_Data = 32'h2d7d9d00;
        ALU_OP=4'b1100;
        A = 32'h7331b232;
        CF=1'b1; VF=1'b1;
        //#100;

    end

    barrelShifter Shift_Instance(.SHFT_OP(SHFT_OP),.Shift_Data(Shift_Data),.Shift_Num(Shift_Num),
.Carry_flag(CF),.Shift_Out(Shift_Out),.Shift_Carry_Out(Shift_Carry_Out));
    ALU ALU_Instance(ALU_OP,A,Shift_Out,Shift_Carry_Out,CF,VF,NZCV,F);
endmodule
```

三、仿真文件和波形图（有就填，没有就不填）

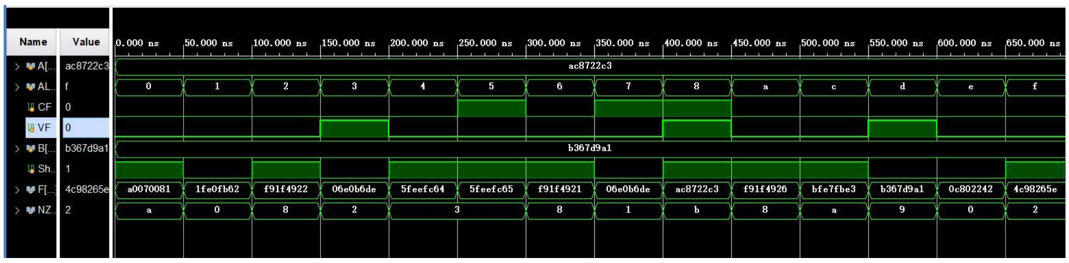
ALU_barrelShifter 仿真波形



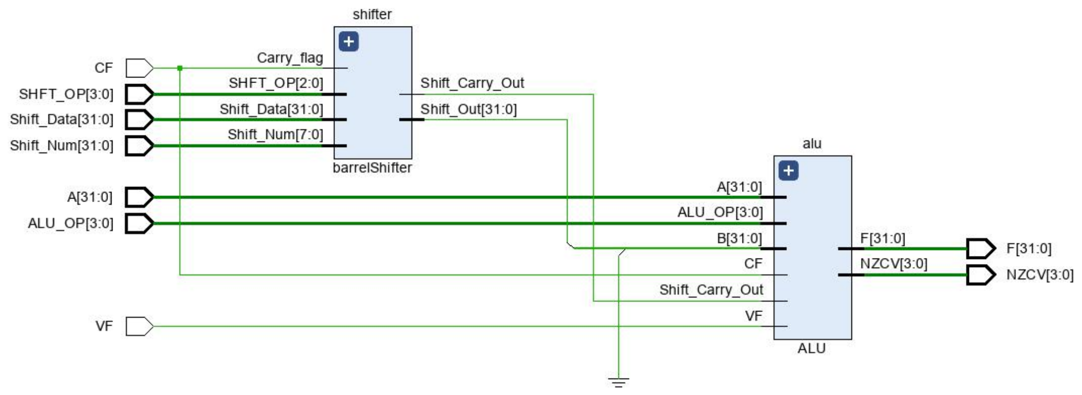
ALU 仿真波形 1



ALU 仿真波形 2


Name
Value
0.000 ns
50.000 ns
100.000 ns
150.000 ns
200.000 ns
250.000 ns
300.000 ns
350.000 ns
400.000 ns
450.000 ns
500.000 ns
550.000 ns
600.000 ns
650.000 ns
A[...] ac8722c3 — ac8722c3
AL... f — 0 1 2 3 4 5 6 7 8 a c d e f
CF 0
VF 0
B[...] b367d9a1 — b367d9a1
Sh... 1
F[...] 4c98265e — a0070081 1fe0fb62 f91f4922 06e0b6de 5feefc64 5feefc65 f91f4921 06e0b6de ac8722c3 f91f4926 bfe7fbe3 b367d9a1 0c802242 4c98265e
NZ... 2 — a 0 8 2 3 8 1 b 8 a 9 0 2

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
周俊佐：实验报告，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）

课后总结：

完成编程和仿真后，需要使用 HDU-XL-01 板卡来验证多功能 ALU 运算器的设计是否正确，那么，单独调试 ALU 模块就需要 70 个逻辑开关，需要 36 个 LED 显示灯，对应 32 位输出运算结果 F 和 4 位标志位 NZCV。若与桶形移位器模块联调，减少 1 个对应桶形移位器 Shift_carry_out 的开关，另外还需要增加 44 个逻辑开关，对应 32 位移位数据 Shift_Data、8 位移位次数 Shift_Num、3 位的 Shift_OP 和 1 位来自 CPSR 的 CF 标志位。HDU-XL-01 板卡上只有 32 个逻辑开关和 32 个 LED 灯，因此，**板卡提供的外设资源不够支持一次性操作，如何解决这个问题呢**?

（1）解决办法之一:
为本实验设置一个顶层模块，结合分时复用逻辑开关和 LED 显示灯，用按键来协助选择当前
输入/输出数据的方法，来完成输入输出的功能。这种方法的优点在于不必固定操作数 A、B 的
值，可以随意设置，因此在选择不同运算的边界值进行测试时具有优越性。

（2）解决方法之二:
当按键数量紧张时，还可以用按键轮询的方法来解决逻辑开关不够的问题。一般采用 2 个
按键，1 个用于轮询，另一个用于确认输入。对用于轮询的按键，应在顶层模块中编程对该按
键的次数以输入数据的总量为模取余数后加 1，以便将按键次数控制在一定范围内。注意:多
个短数据拼接为 I 个 32 位以内的数据，且一次性输入的，计为 1 次。例如:按键 swb[1]〕用于
轮询，按键 swb[2]用于确认输入，两个按键都是按下为 1，松开为 0。

（3）解决办法之三:
为本实验设置一个顶层模块,在顶层模块中通过多路选器，在若干组预设常数组合中选一组，
送入 ALU 运算器模块;也通过多路选择器选择输出结果或标志位，来完成板级验证。该方法的
优点是，可以预先设置好测试数据，节省实验时输入数据的时间。

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：整个 FPGA 实验流程，先进行仿真测试，其次再进行板级测试，我们这次实验的板级调试之前出现过错误，后来在课堂上与同学讨论并解决，出错在 data 的输出固定在 alu 的 F 上，使得其他信号量无法显示。

2：在 ALU 运算器的设计，与桶形移位器的连接过程之中，应当注意的是模块设计的思想，模块与模块之间的层次与对应关系，若模块之间连接错误，对之后实验的更大规模模块拼接会造成很大的错误。

3：alu 和 barrelshifter 均是组合逻辑结构，故不需要引入 clock 时钟

5：在之后的实验中，在更复杂的工程之中，我们能够完成所有既定的逻辑功能之后，便需要追求更少的能耗，更快的运行时间

我们小组编制的 ALU 单个模块测试结果记录表详情如下：

ALU 运算器模块测试结果记录表

A= ac8722c3 **B=b367d9a1**

ALU_OP	**运算方式**	Shift_carry_out，来自 CPSR 的 CF、VF	F	**NZCV**
0000	F=A＆ B	1 0 0	a0070081	a
0001	F=A ^ B	0 0 0	1fe0fb62	0
0010	F=A－ B	1 0 0	fb1f4922	8
0011	F=B - A	0 0 1	06e0b6d1	2
0100	F=A＋ B	1 0 0	5feefc64	3
0101	F=A＋ B +CF	1 1 0	5feefc65	3
0110	F=A－ B + CF - 1	1 0 0	f91f4921	8
0111	F=B - A + CF - 1	0 1 0	06e0b6de	1
1000	F=A	1 1 1	ac8722c3	b
1010	F=A－B + 4	1 0 0	f91f4926	8
1100	F=A｜ B	1 0 0	bfe7fbe3	a
1101	F=B	0 0 1	b367d9a1	9
1110	F=A & (~B)	0 0 0	0c802242	0
1111	F=~B	1 0 0	4c982651	2

A= ac963a55 **B=365aacf9**

ALU_OP	**运算方式**	Shift_carry_out，来自 CPSR 的 CF、VF	F	**NZCV**
0000	F=A＆ B	0 0 0	24122851	0
0001	F=A ^ B	1 0 0	9acc96ac	a
0010	F=A－ B	0 0 0	763b8d5c	3
0011	F=B - A	1 0 1	89c472a4	9
0100	F=A＋ B	0 0 0	e2f0e74e	8
0101	F=A＋ B +CF	1 1 0	e2f0e74f	8
0110	F=A－ B + CF - 1	0 0 0	763b8d5b	3

0111	F=B - A + CF - 1	1 1 0	89c472a4	9
1000	F=A	1 1 0	ac963a55	b
1010	F=A－B + 4	1 0 0	763b8d60	2
1100	F=A ｜ B	1 0 0	bedebefd	a
1101	F=B	1 0 1	365aacf9	3
1110	F=A & (~B)	0 0 0	88841204	8
1111	F=~B	1 0 0	c9a55306	A

桶形移位器和 ALU 运算器模块联调实验记录表

SHIFT_OP	移位次数 Shift_Num	被移位数据 Shift_Data	ALU_OP	**A**	来自 CPSR 的 CF、VF	F	**NZCV**
111	0	888ee888	1111	3343 dacd	0 1	77711777	x
110	1	3f3f3f3f	0011	8765 4321	1 0	183a5c7e	2
000	0	2d7d9d00	1100	7331 b232	1 1	7f7dbf32	x
001	3	3ac50001	0101	9a4d 882b	1 0	70758834	0011
000	35	2342689a	1100	ee34 fa12	0 1	ee34fa12	1001
010	0	f0000000	0001	8b49 da1f	0 0	8b49da1f	x
010	12	5f5555f5	0111	9846 3f4f	0 0	67bfb605	0001
100	6	8a9d029d	1010	4b5c 6d7e	1 1	4d31f978	0001
101	40	8a9d029d	1101	bcde f123	1 1	fffffff	x