同济大学计算机系

数字逻辑课程实验报告

学　　号 2253156

姓　　名 闫浩扬

专　　业 计算机科学与技术（精英班）

授课老师 郭玉臣

一、 实验内容

实验介绍:

在本次实验中，我们将使用 Verilog HDL 语言实现 32 位桶形移位器的设计和仿真。

实验目标:

- 深入了解桶形移位器的原理。
- 使用 logicsim 软件搭建一个 8 位的桶形移位器。
  学习使用 Verilog HDL 语言设计实现一个 32 位桶形移位器。

二、 硬件逻辑图

实验原理: 桶式移位器是一种组合逻辑电路，通常作为微处理器 CPU 的一部分。它具有 n 个数据输入和 n 个数据输出，以及指定如何移动数据的控制输入，指定移位方向、移位类型（循环、算术还是逻辑移位）及移动的位数等等。

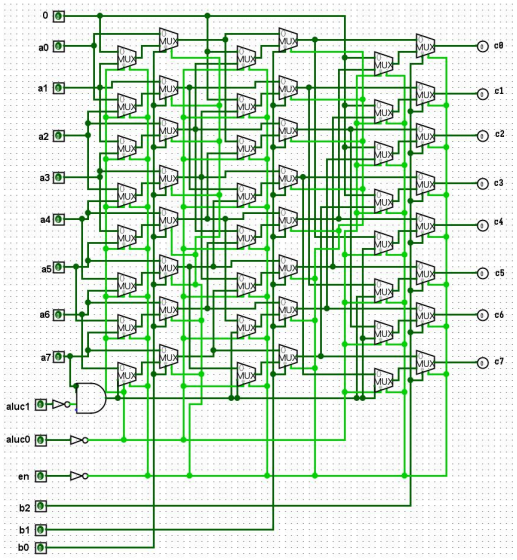


图 1. 8 位桶形移位器原理图

图一给出了一个简单的 4 位桶形移位器原理图示例。其中的主要部件为二选一数据选择器 MUX，该数据选择器的 S1 和 S2 为两路数据输入端， D 为数据输出端， C 为选择控制端， ENB 为使能控制端。该桶形移位器的所有输入输出信号规定如下：

输入信号 a0~a7 左移以及右移一位的数据被输入第一列数据选择器；

- 输入信号 aluc0 选择左移还是右移，将数据输入第二列数据选择器；
- 输入信号 aluc1 控制右移时进行算数右移还是逻辑右移；
- 输入信号 a0~a7 的原始数据也被输入第二列数据选择器；
- 输入信号 b0 和 b1 和 b2 的值决定移多少位；
- 输出信号 c0~c7；
- 输入信号 en 为桶形移位器的使能端。

表 6.4.1 aluc1 和 aluc0 的值所对应的逻辑运算

MIPS 指令	aluc1	aluc0	说明
算术右移(sra)	0	0	a 向右移动 b 位，最高位补 b 位符号位
逻辑右移(srl)	1	0	a 向右移动 b 位，最高位补 b 位 0
算术左移(sll)	0	1	a 向左移动 b 位，最低位补 b 位 0
逻辑左移(sll)	1	1	a 向左移动 b 位，最低位补 b 位 0

根据上面所描述的 8 位桶形移位器的原理，设计者可以开发其他位数的桶形移位器器。 当移位器的位数较多时，采用原理图设计的方式将非常繁琐。而采用 Verilog HDL 行为 描述方式则能够很容易的对其建模。

三、 模块建模

- 接口定义
  这里我们给出本实验所要求建模的 32 位桶形移位器的 接口定义。

```
module barrelshifter32(
    input [31:0] a, //32 位原始输入数据
    input [4:0] b, //5 位输入信号，控制移位的位数
    input [1:0] aluc, //2 位输入信号，控制移位的方式
    output reg [31:0] c //32 位移位后的输出数据
```

Verilog 代码描述

```
module barrelshifter32(
    input [31:0] a, // 32 位原始输入数据
    input [4:0] b, // 5 位输入信号，控制移位的位数
    input [1:0] aluc, // 2 位输入信号，控制移位的方式
    output reg [31:0] c // 32 位移位后的输出数据
);
always @(*)
begin
    case ({aluc[1], aluc[0]})
        2'b00:  // 算术右移 (sra)
            begin
                c = $signed(a) >>> b[4:0];
            end
        2'b10:  // 逻辑右移 (srl)
            begin
                c = a >> b[4:0];
            end
        2'b01:  // 算术左移 (sll)
            begin
                c = a << b[4:0];
            end
        2'b11:  // 逻辑左移 (sll)
            begin
                c = a << b[4:0];
            end
    endcase
end
```

这个模块采用了行为型描述，通过 aluc[1]和 aluc[0]的不同组合采取不同的移位方式。同时，采用了组合逻辑的描述方式，输出信号只依赖于当前输入信号的值，而没有时序元素或状态储存器，不受时钟信号的影响。

这个模块通过移位操作符实现了移位操作。对于算术右移，使用$signed(a)>>>b[4:0],$signed(a)将 a 视作带符号整数并执行算数右移，右移时保留符号位。对于逻辑移位操作，不需要考虑符号位，需要在相应位置填充 0.

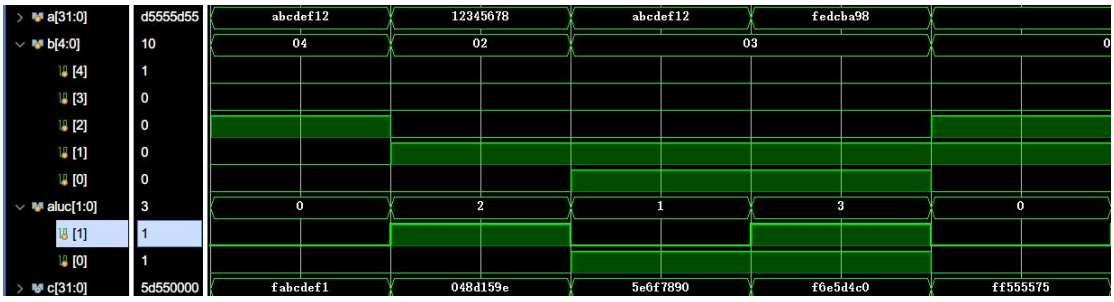
这个模块可以用于各种应用场景，如处理器指令执行、数据加密解密、图像处理、数字信号处理以及数据压缩编解码等，提供了对 32 位数据进行灵活位移操作

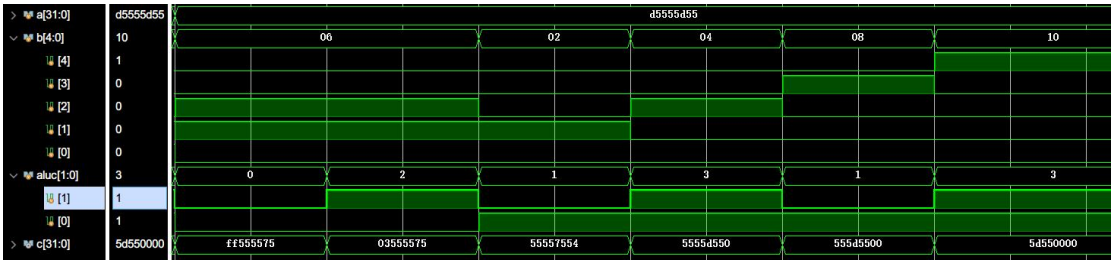
四、 测试模块建模

```
`timescale 1ns/1ps
module tb_barrelshifter32;
    reg [31:0] a;
    reg [4:0] b;
    reg [1:0] aluc;
    wire [31:0] c;
    barrelshifter32 uut (
        .a(a),
        .b(b),
        .aluc(aluc),
        .c(c)
    );
    initial begin
        a = 32'hABCDEF12;
        b = 5'b00100;
        aluc = 2'b00;
     #40
        a = 32'h12345678;
        b = 5'b00010;
        aluc = 2'b10;
     #40
        a = 32'hABCDEF12;
        b = 5'b00011;
        aluc = 2'b01;
     #40
        a = 32'hFEDCBA98;
        b = 5'b00011;
        aluc = 2'b11;
     #40
        a = 32'b11010101_01010101_01011101_01010101;
        b = 5'b0011_0;
        aluc = 2'b00;
     #40
        a = 32'b11010101_01010101_01011101_01010101;
        b = 5'b0011_0;
        aluc = 2'b10;
     #40
        a = 32'b11010101_01010101_01011101_01010101;
        b = 5'b0001_0;
        aluc = 2'b01;
     #40
        a = 32'b11010101_01010101_01011101_01010101;
        b = 5'b0010_0;
        aluc = 2'b11;
     #40
        a = 32'b11010101_01010101_01011101_01010101;
        b = 5'b0100_0;
        aluc = 2'b01;
     #40
        a = 32'b11010101_01010101_01011101_01010101;
        b = 5'b1000_0;
        aluc = 2'b11;
    end
endmodule
```

五、 实验结果

- 波形图仿真





• logisim 逻辑验证