中国科学技术大学计算机学院
《计算机组成原理实验》报告

实验题目：运算器及其应用

学生姓名：阿卜杜赛米江·萨吾提

学生学号：PB19111752

完成日期：2021-4-7

计算机实验教学中心制

2020 年 09 月

【实验题目】

运算器及其应用

【实验步骤】

- **掌握算术逻辑单元（ALU）的功能**

  掌握数据通路和控制器的设计方法

- **掌握组合电路和时序电路，以及参数化和结构化的 Verilog 描述方法**

- **了解查看电路性能和资源使用情况**

【实验环境】

Vivado 软件

【实验各模块】

//根据选择信号输送使能

1.译码器：

```
module decoder
(input en,clk,
input [1:0] selct,
output reg ef,reg ea,reg eb);
always@(posedge clk)
if(en)
begin
    case(selct)
        2'b11:ef<=1;
        2'b0 module decoder
(input en,clk,
input [1:0] selct,
output reg ef,reg ea,reg eb);
always@(posedge clk)
if(en)
begin
    case(selct)
        2'b11:ef<=1;
        2'b01:ea<=1;
```

```
        2'b10:eb<=1;
        endcase
end
endmodule 1:ea<=1;
        2'b10:eb<=1;
        endcase
end
endmodule
```
//将数据传送到指定的数据块端口

2.复用器：

```
module reuser
 #(parameter width=6)
 (input [width-1:0] data,
 input [1:0] selct,
 input clk,
 output reg[width-1:0]a,b,
 output reg [2:0] f);

always@(posedge clk)
begin
case(selct)
    2'b11: f<=data[2:0];
    2'b01: a<=data;
    2'b10: b<=data;
endcase
end
```
//操作类型块

3.f 模块：

```
module f
 #(parameter width=3,ef_val=0)
 (input [width-1:0] f,
 input ef,clk,
 output reg [width-1:0] f_o);
 always@(posedge clk)
 begin
    if(ef==ef_val)
    f_o<=0;
    else f_o<=f;
 end
 endmodule
```
//操作数块

4.a 模块：

```
module a
 #(parameter width=6,ea_val=0)
```

```
 (input [width-1:0] a,
 input ea,clk,
 output reg [width-1:0] a_o);
 always@(posedge clk)
 begin
    if(ea==ea_val)
    a_o<=0;
    else a_o<=a;
 end
 endmodule
```

操作数快

5.b 模块：

```
 module b
 #(parameter width=6,eb_val=0)
 (input [width-1:0] b,
 input eb,clk,
 output reg [width-1:0] b_o);
 always@(posedge clk)
 begin
    if(eb==eb_val)
    b_o<=0;
    else b_o<=b;
 end
 endmodule
```

//根据用户的输入数据和选择的操作类型，对数据进行操作并输出

6.alu 模块：

```
module alu
#(parameter width=6)
(input [width-1:0]a,b,
input [2:0]f,
input clk,
output reg z,
output reg [width-1:0] y);
always@(posedge clk)
begin
    case(f)
        3'b000:begin
            y<=(a+b);
            end
        3'b001:begin
            y<=(a-b);
            end
        3'b010:begin
            y<=(a&b);
```

```
            end
        3'b011:begin
            y<=(a|b);
            end
        3'b100:begin
            y<=(a^b);
            end
        default:begin
                y<=0;z<=0;
            end
         endcase
    if(y==0)z<=1;
end
endmodul
```

//将很多个信号周期采样，以适应人的使用习惯

7.信号采样模块：

```
module signal_edge(
    input clk,
    input button,
    output button_edge,
    output reg button_r1,button_r2);
    always@(posedge clk) button_r1 <= button;
    always@(posedge clk) button_r2 <= button_r1;
    assign button_edge = button_r1 & (~button_r2);
endmodule
```

//组织协调各模块，并负责向外提供统一接口

8.组织各模块形成 alu:

```
module using
#(parameter width=6)
(input   en, clk,
input [1:0] selct,
input [width-1:0] data,
output z,
output  [width-1:0] y);

wire e_f;
wire e_a;
wire e_b;
wire [1:0] t_selct;
wire[width-1:0] t_d;
wire t_en;
wire [2:0] f;
wire [width-1:0] a;
wire [width-1:0] b;
```

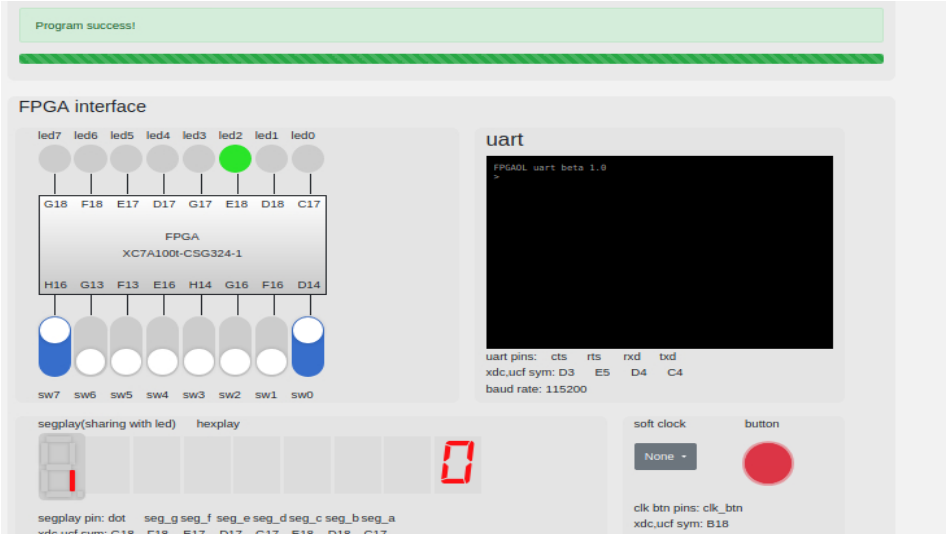
```
/*always@( clk)
begin
    t_en<=en;
    t_selct<=selct;
    t_d<=data;
end*/
assign t_en =en;
assign t_selct=selct;
assign t_d=data;

wire [2:0] f_c;
wire [width-1:0] a_c;
wire [width-1:0] b_c;
decoder de1(.en(t_en),.clk(clk),.selct(t_selct),.ef(e_f),.ea(e_a),.eb(e_b));

reuser  #(.width(width))
re1(.selct(t_selct),.clk(clk),.data(t_d),.f(f),.a(a),.b(b));
f #(.width(3)) f1(.f(f),.ef(e_f),.clk(clk),.f_o(f_c));
a #(.width(width)) a1(.a(a),.ea(e_a),.clk(clk),.a_o(a_c));
b #(.width(width)) b1(.b(b),.eb(e_b),.clk(clk),.b_o(b_c));
alu #(.width(width)) al_o(.f(f_c),.clk(clk),.a(a_c),.b(b_c),.z(z),.y(y));
endmodule
```

//alu 模块的实现

8.调用 alu:

```
module fpga
#(parameter width=6)
(input btn,clk,
input [width-1:0] sw,
input [1:0] selct,
output [width-1:0] led,
output z);
using  #(.width(width))
t_using(.en(btn),.data(sw),.clk(clk),.selct(selct),.y(led),.z(z));
endmodule
```

效果图：



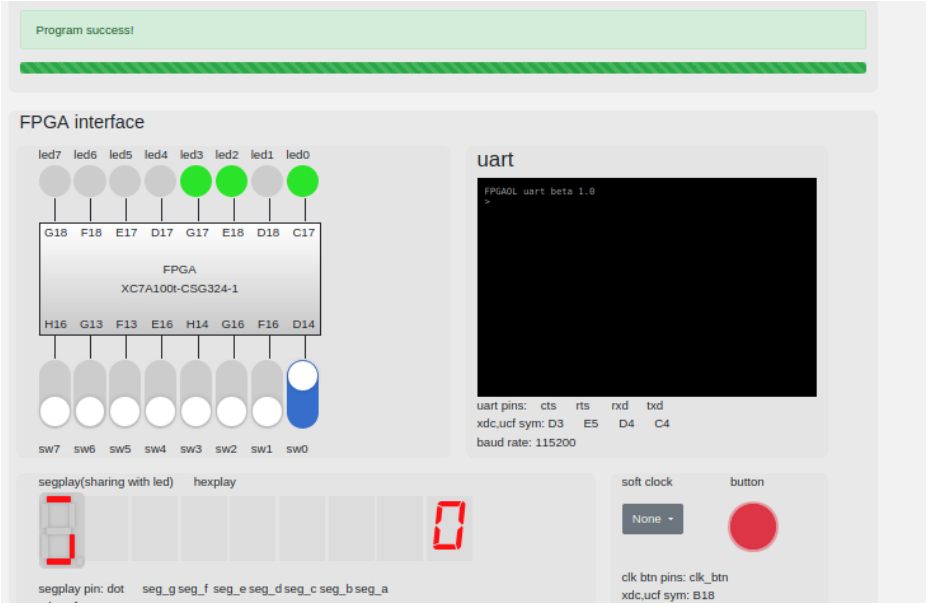
！！至此为：alu 模块

二：fls 模块

```
//模块的定义
module fls
#(parameter width=7)
(input [width-1:0] sw,
input btn,clk,rst,
output reg [width-1:0] led);
reg [width-1:0] temp1;
//reg [width-1:0] temp2;
reg [1:0] count;
wire clk1;
//clk_wiz_0  clk_u(.reset(rst),.clk_in1(clk),.clk_out1(clk1));
signal_edge sig_edg(.clk(clk),.button(btn),.button_edge(clk1));
always@(posedge clk1)
begin

    if(rst)begin
        count<=0;
        led<=0;
        end
    else
    begin
       if(btn==1&&count<=2) begin
       count<=count+1;
       temp1<=led;
       led<=sw;
       end
       else if(btn)
```

```
        begin
            temp1<=led;
            led<=temp1+led;
        end
        end

end

endmodule
```

效果图：



【总结与思考】

1. 首先，alu 设计时模块分的比较细，为后面的分析带来了方便，便于自己 debug.
2. 设计 fls 时，btn 按下时，计数太快，因此导入了去年数电实验室的 signal_edge 模块，解决了此问题。
3. 发现 always 块内的语句是，此次循环结束时才对值进行赋值，利用了这一点设计了 fls.
4. Verilog 编程还得进一步练习和加强。