《计算机组成原理》
实验报告

实验题目：运算器和排序

学生姓名：王 志 强

学生学号： PB18051049

完成日期： 2020.04.29

计算机实验教学中心制

2019年9月

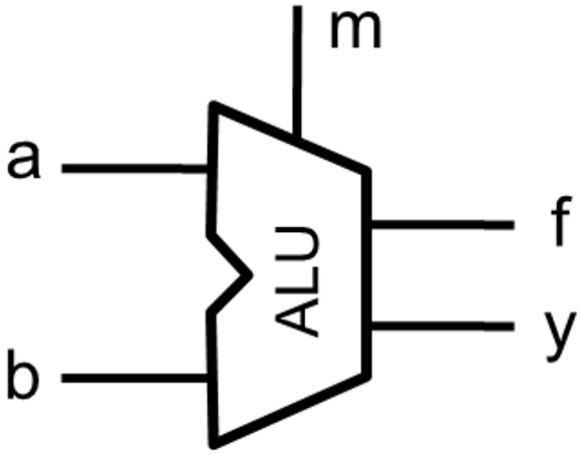
一、实验目标

- 掌握算术逻辑单元（ALU）的功能，加/减运算时溢出、进位/借位、零标志的形成及其应用；
- 掌握数据通路和控制器的设计和描述方法。

二、实验内容

1. ALU的设计

- 待设计的ALU模块的逻辑符号如下图所示。该模块的功能是将两操作数（a，b）按照指定的操作方式（m）进行运算，产生运算结果（y）和相应的标志（f）。



- 操作方式m的编码与ALU的功能对应关系如下图 所示。表中标志f细化为进位/借位标志（cf）、溢出标志（of）和零标志（zf）；“*”表示根据运算结果设置相应值；“x”表示无关项，可取任意值。例如，加法运算后设置进位标志（cf）、（of）和（zf），减法运算后设置借位标志（cf）、（of）和（zf）。

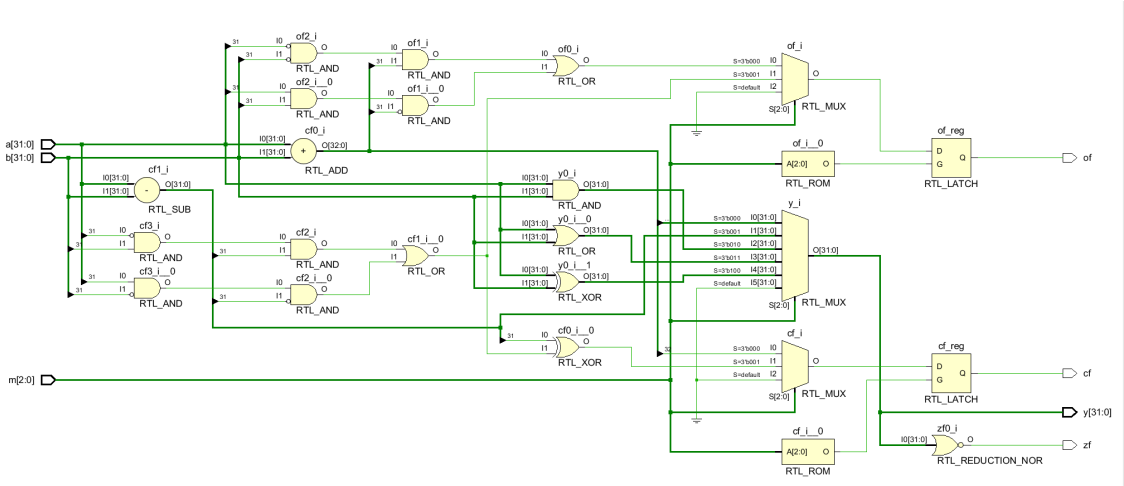
m	y	cf	of	zf
000	a + b	*	*	*
001	a - b	*	*	*
010	a & b	x	x	*
011	a | b	x	x	*
100	a ^ b	x	x	*
其他	x	x	x	x

- 根据端口和功能要求，Verilog代码实现如下（有符号数运算，输入输出为补码）：

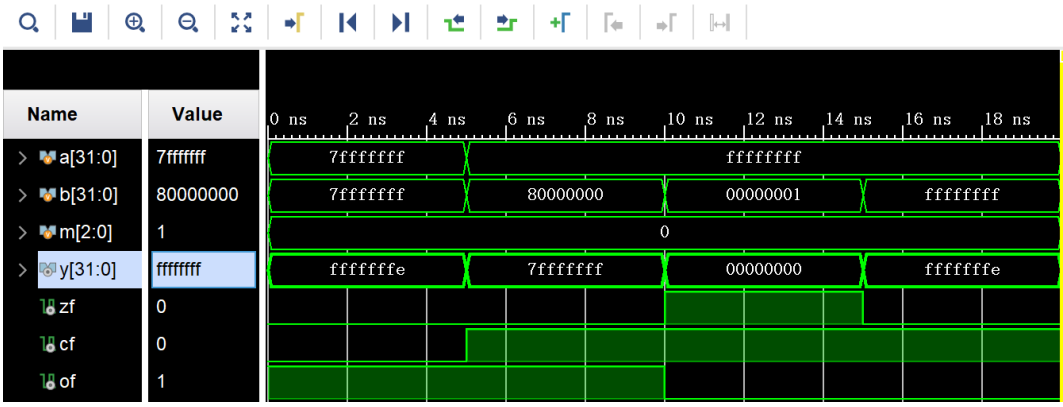
```
module alu  //输入、输出均视为补码
   #(parameter WIDTH=32)
    (output reg [WIDTH-1:0] y,
    output reg zf,
    output reg cf,
    output reg of,
    input [WIDTH-1:0] a,b,
    input [2:0] m
    );
    parameter ADD = 3'b000;
```

```
    parameter SUB = 3'b001;
    parameter AND = 3'b010;
    parameter OR = 3'b011;
    parameter XOR = 3'b100;
    always @(*)
        begin
        case(m)
            ADD:
            begin
                {cf,y} = a + b;
                of = (~a[WIDTH-1]&~b[WIDTH-1]&y[WIDTH-1])
                    |(a[WIDTH-1]&b[WIDTH-1]&~y[WIDTH-1]);
            end
            SUB:
            begin
                y  = a - b; //a<b 产生借位
                of = (~a[WIDTH-1]&b[WIDTH-1]&y[WIDTH-1])
                    |(a[WIDTH-1]&~b[WIDTH-1]&~y[WIDTH-1]);
              //在输入输出均为补码的情况下 ，cf = y[WIDTH-1] ^ of
                cf = y[WIDTH-1] ^ of;
            end
            AND:y = a & b;
            OR:y = a | b;
            XOR:y = a ^ b;
            default: //保证完全赋值
            begin y = 32'd0;zf = 0;cf = 0;of = 0; end
        endcase
        zf = ~|y;
        end
endmodule
```

- **RTL ANALYSIS-Schematic:**



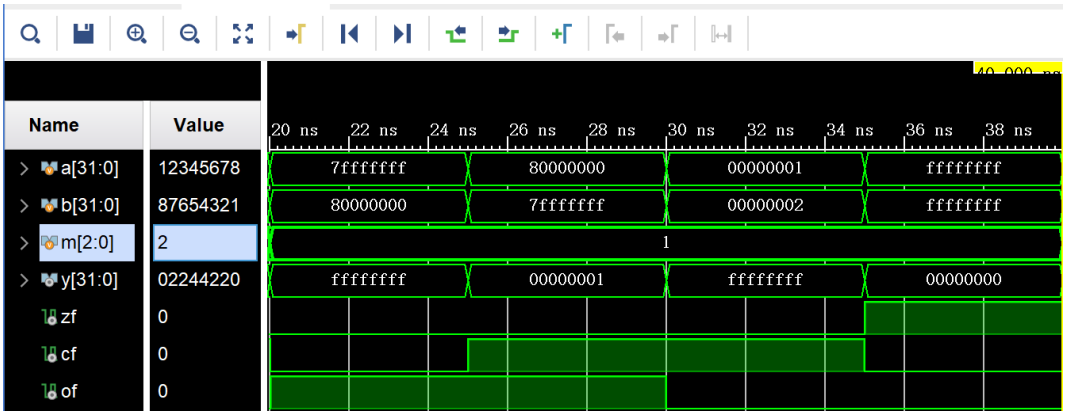
- **Vivado仿真如下:**
  - **ADD测试:**



testcase:

```
    m=3'b000;a=32'h7FFF_FFFF;b=32'h7FFF_FFFF;//正溢
#5  m=3'b000;a=32'hFFFF_FFFF;b=32'h8000_0000;//负溢
#5  m=3'b000;a=32'hFFFF_FFFF;b=32'h0000_0001;//0
#5  m=3'b000;a=32'hFFFF_FFFF;b=32'hFFFF_FFFF;//负数加
```

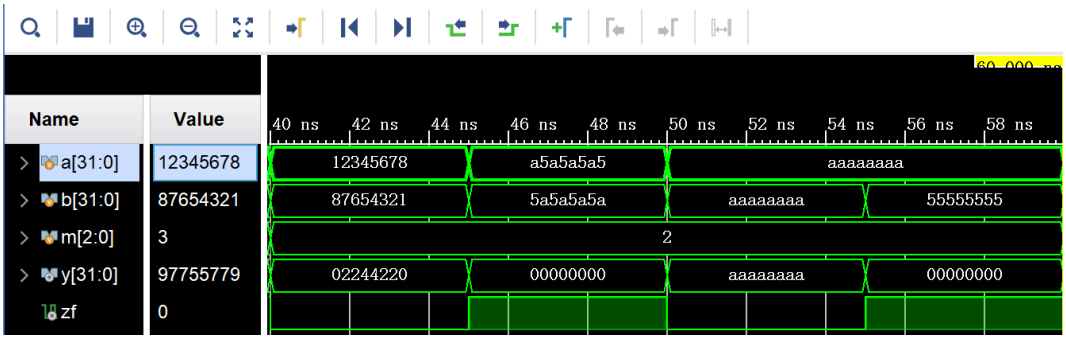
- **SUB测试**:



testcase:

```
#5  m=3'b001;a=32'h7FFF_FFFF;b=32'h8000_0000;//正溢
#5  m=3'b001;a=32'h8000_0000;b=32'h7FFF_FFFF;//负溢
#5  m=3'b001;a=32'h0000_0001;b=32'h0000_0002;//借位
#5  m=3'b001;a=32'hFFFF_FFFF;b=32'hFFFF_FFFF;//0
```

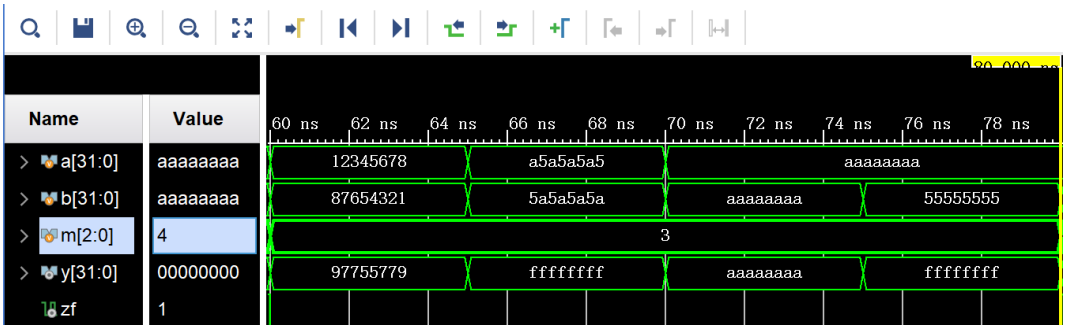
- **AND测试**:



testcase:

```
#5  m=3'b010;a=32'h1234_5678;b=32'h8765_4321;
#5  m=3'b010;a=32'hA5A5_A5A5;b=32'h5A5A_5A5A;
#5  m=3'b010;a=32'hAAAA_AAAA;b=32'hAAAA_AAAA;
#5  m=3'b010;a=32'hAAAA_AAAA;b=32'h5555_5555;//0
```

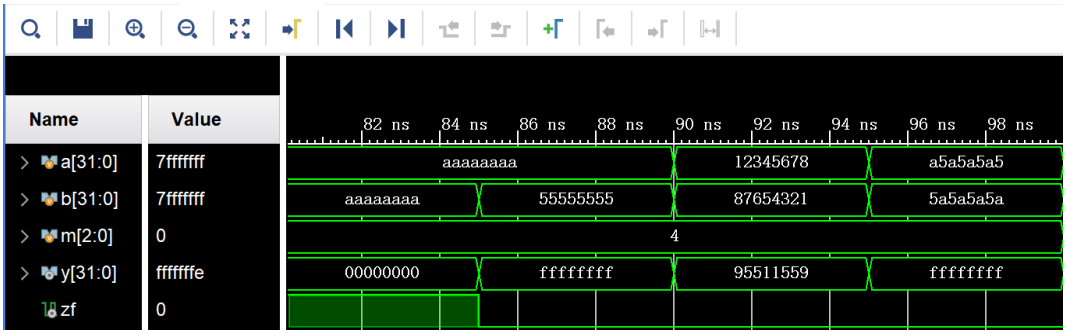
- **OR测试：**



testcase：

```
#5 m=3'b011;a=32'h1234_5678;b=32'h8765_4321;
#5 m=3'b011;a=32'hA5A5_A5A5;b=32'h5A5A_5A5A;//全1
#5 m=3'b011;a=32'hAAAA_AAAA;b=32'hAAAA_AAAA;
#5 m=3'b011;a=32'hAAAA_AAAA;b=32'h5555_5555;//全1
```

- **XOR测试：**



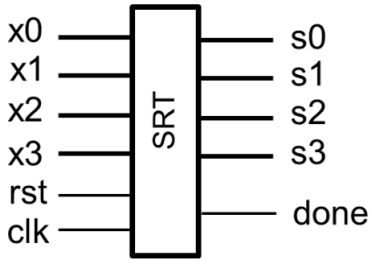
testcase：

```
#5 m=3'b100;a=32'hAAAA_AAAA;b=32'hAAAA_AAAA;
#5 m=3'b100;a=32'hAAAA_AAAA;b=32'h5555_5555;//全1
#5 m=3'b100;a=32'h1234_5678;b=32'h8765_4321;
#5 m=3'b100;a=32'hA5A5_A5A5;b=32'h5A5A_5A5A;//全1
```

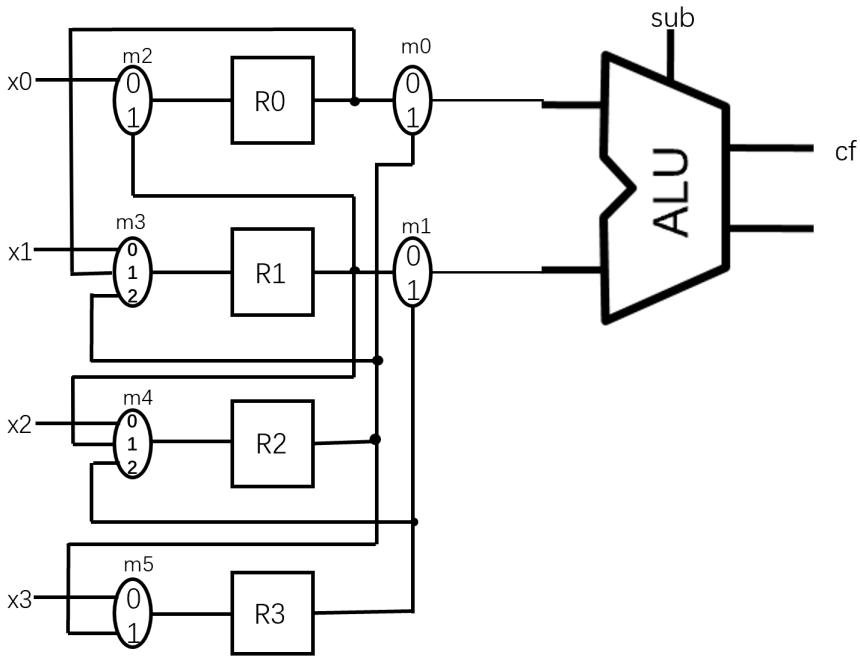
- **FPGA开发板测试如下：**
  - **返校后进行**

2、排序电路的设计：

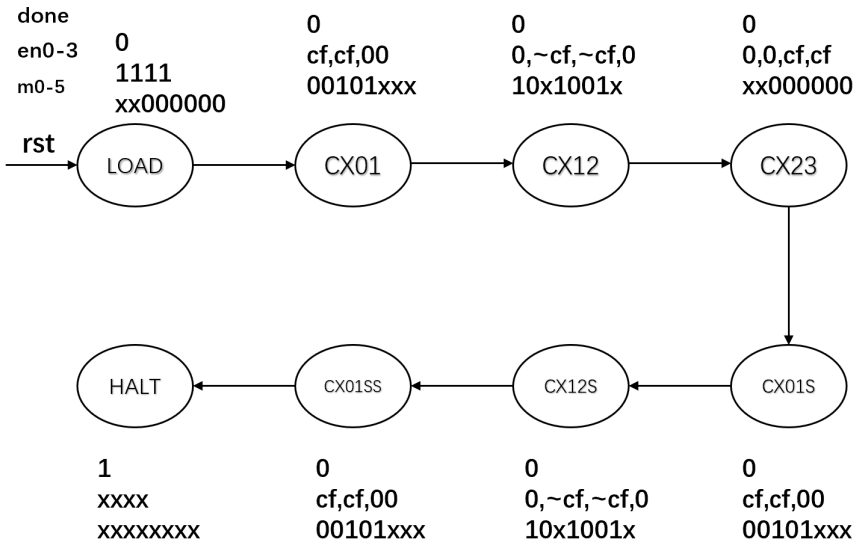
- 利用前面设计的ALU模块，辅之以若干寄存器和数据选择器，以及适当的控制器，设计实现四个4位有符号数的排序电路，其逻辑符号如下图所示所示：



- **四个符号数排序电路的数据通路如下图所示**：



- **排序电路状态控制图如下**:

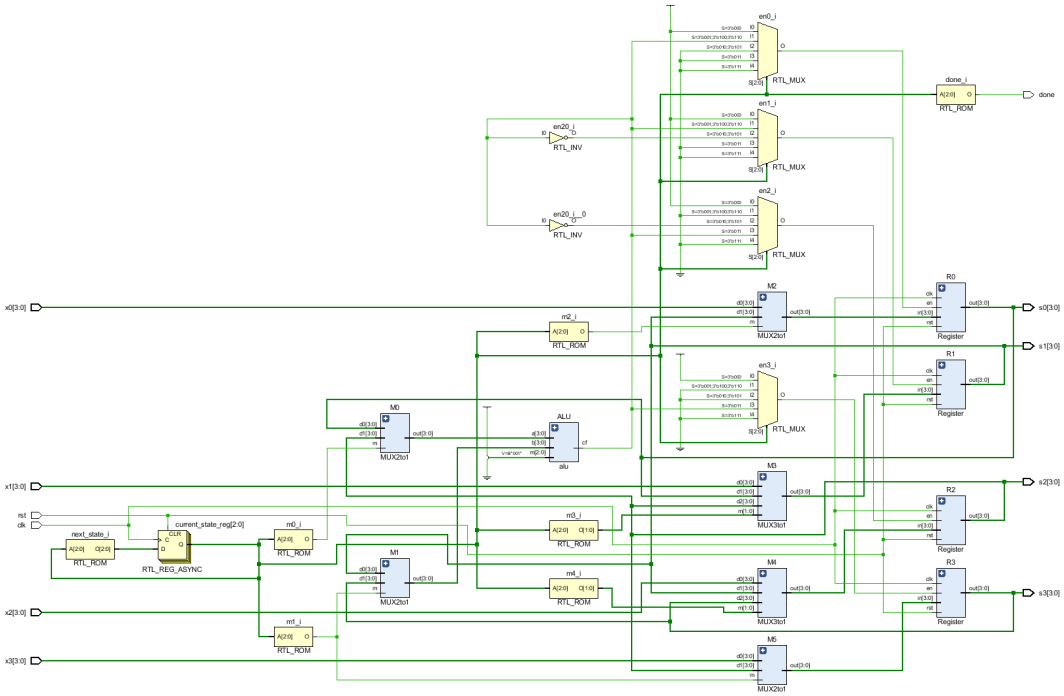


- **排序电路Verilog实现**:

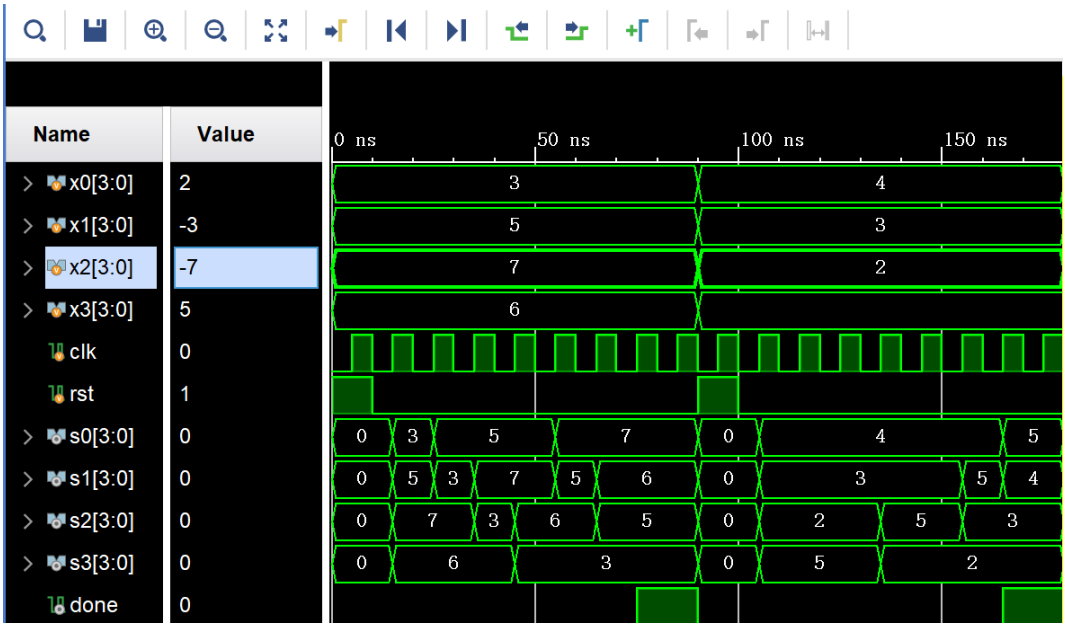
```
module sort #(parameter N = 4)(
    output [N-1:0] s0,s1,s2,s3, //排序后的四个数据
    output reg done,    //排序结束标志
    input [N-1:0] x0,x1,x2,x3,  //待排序的数字
    input clk,rst   //时钟，复位，上升沿有效
    );
    parameter SUB = 3'b001;
    wire [N-1:0] i0,i1,i2,i3;   //寄存器输入端
   // wire [N-1:0] s0,s1,s2,s3;   //寄存器输出端
    wire [N-1:0] a,b;           //ALU输入、输出端
    wire cf; //借位标志
    reg m0,m1,m2,m5;
    reg [1:0] m3,m4;   //多选器选择信号
    reg en0,en1,en2,en3;   //寄存器使能信号
    reg [2:0] current_state,next_state; //状态
    parameter LOAD = 3'b000;
    parameter CX01 = 3'b001;
    parameter CX12 = 3'b010;
    parameter CX23 = 3'b011;
```

```
    parameter CX01S = 3'b100;
    parameter CX12S = 3'b101;
    parameter CX01SS = 3'b110;
    parameter HALT = 3'b111;
    //Data Path
    Register #(4) R0(.in(i0),.en(en0),.rst(rst),.clk(clk),.out(s0));
    Register #(4) R1(.in(i1),.en(en1),.rst(rst),.clk(clk),.out(s1));
    Register #(4) R2(.in(i2),.en(en2),.rst(rst),.clk(clk),.out(s2));
    Register #(4) R3(.in(i3),.en(en3),.rst(rst),.clk(clk),.out(s3));

    MUX2to1 #(4) M0(.m(m0),.d0(s0),.d1(s2),.out(a));
    MUX2to1 #(4) M1(.m(m1),.d0(s1),.d1(s3),.out(b));
    MUX2to1 #(4) M2(.m(m2),.d0(x0),.d1(s1),.out(i0));
    MUX3to1 #(4) M3(.m(m3),.d0(x1),.d1(s0),.d2(s2),.out(i1));
    MUX3to1 #(4) M4(.m(m4),.d0(x2),.d1(s1),.d2(s3),.out(i2));
    MUX2to1 #(4) M5(.m(m5),.d0(x3),.d1(s2),.out(i3));

    alu #(4) ALU(.a(a),.b(b),.m(SUB),.cf(cf),.of(),.zf(),.y());
    //control unit
    always @(posedge clk,posedge rst)
        if(rst)
            current_state <=LOAD;
        else
            current_state <= next_state;
    always @(*)
    begin
        case(current_state)
          LOAD:next_state <= CX01;
          CX01:next_state <= CX12;
          CX12:next_state <= CX23;
          CX23:next_state <= CX01S;
          CX01S:next_state <= CX12S;
          CX12S:next_state <= CX01SS;
          CX01SS:next_state <=HALT;
          HALT:next_state <= HALT;
        endcase
    end
    always @(*)
    begin
    {m0,m1,m2,m3,m4,m5,en0,en1,en2,en3,done} = 13'd0;
    case(current_state)
      LOAD:{en0,en1,en2,en3} = 4'b1111;
      CX01,CX01S,CX01SS:begin {m2,m3} = 3'b101;en0 = cf;en1 = cf;end
      CX12,CX12S:begin {m0,m1,m3,m4} = 6'b101001;en1 = ~cf;en2 =
~cf;end
      CX23:begin {m0,m1,m4,m5} = 5'b11101;en2 = cf;en3 = cf;end
      HALT:done = 1;
    endcase
    end
endmodule
```

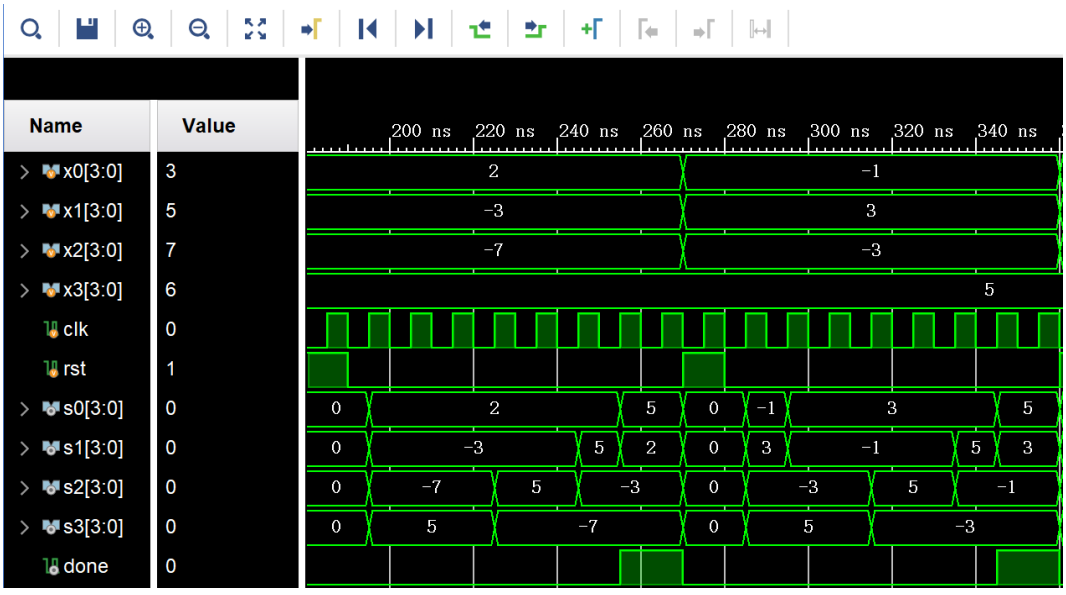
- **RTL ANALYSIS-Schematic:**



- **Vivado仿真如下：**
  - **无符号数排序测试：**



  - **有符号数排序测试（带负数）：**



- **testcase:**

```
module tb_sort;

// sort Parameters
parameter N = 4;
parameter PERIOD  = 10    ;

// sort Inputs
reg   [N-1:0]  x0                         = 0 ;
reg   [N-1:0]  x1                         = 0 ;
reg   [N-1:0]  x2                         = 0 ;
reg   [N-1:0]  x3                         = 0 ;
reg   clk                                 = 0 ;
reg   rst                                 = 0 ;

// sort Outputs
wire  [N-1:0]  s0                         ;
wire  [N-1:0]  s1                         ;
wire  [N-1:0]  s2                         ;
wire  [N-1:0]  s3                         ;
wire  done                                ;

initial
begin
    forever #(PERIOD/2)  clk=~clk;
end

sort u_sort (
    .x0                      ( x0    [N-1:0] ),
    .x1                      ( x1    [N-1:0] ),
    .x2                      ( x2    [N-1:0] ),
    .x3                      ( x3    [N-1:0] ),
    .clk                     ( clk           ),
    .rst                     ( rst           ),

    .s0                      ( s0    [N-1:0] ),
    .s1                      ( s1    [N-1:0] ),
    .s2                      ( s2    [N-1:0] ),
    .s3                      ( s3    [N-1:0] ),
    .done                    ( done          )
);

initial
    begin
    rst = 1;
    #PERIOD rst = 0;

    #(PERIOD*8) rst = 1;
    #PERIOD rst = 0;

    #(PERIOD*8) rst = 1;
    #PERIOD rst = 0;
    #(PERIOD*8) rst = 1;
    #PERIOD rst = 0;
    end
initial
```

```
begin
    x0 = 3;
    x1 = 5;
    x2 = 7;
    x3 = 6;
    #(PERIOD*9)

    x0 = 4;
    x1 = 3;
    x2 = 2;
    x3 = 5;
    #(PERIOD*9)

    x0 = 2;
    x1 = -3;
    x2 = -7;
    x3 = 5;
    #(PERIOD*9)

    x0 = -1;
    x1 = 3;
    x2 = -3;
    x3 = 5;
   #(PERIOD*9)
    $finish;
end

endmodule
```

- **FPGA开发板测试：**
  - **返校后进行**

三、思考题

1. **如果要求排序后的数据是递减顺序，电路如何调整？**

   答：本题设计的即为递减顺序，故回答如何调整至递增顺序，至于要将使能信号中的~cf和cf对换即可。

2. **如果为了提高性能，使用两个ALU，电路如何调整？**

   答：如下图所示，减少了MUX的使用，同时也可以在减少一个时钟周期的情况下完成排序，因为s2、s3比较的同时s0和s1也可以比较。