ARQUITECTURA DE COMPUTADORAS

2019

Trabajo Práctico Nº 1

ALU

MEGEVAND, Nicolás

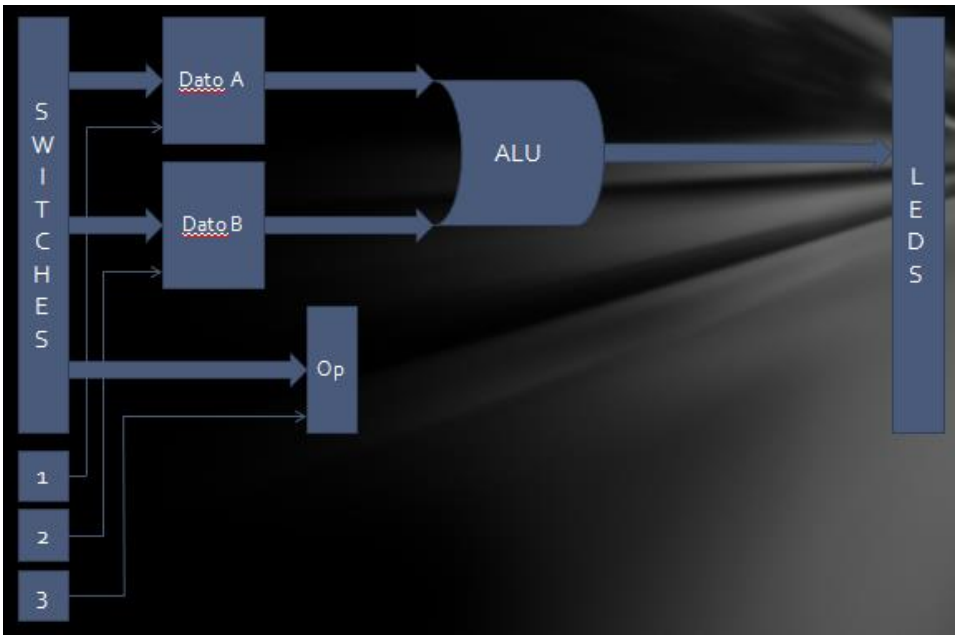
MORALES, Franco

OBJETIVOS

- Implementar en FPGA una ALU
- Utilizar las placas de desarrollo Basys II o Nexys III
- La ALU debe ser parametrizable (bus de datos) para poder ser utilizada posteriormente en el trabajo final
- Validar el desarrollo por medio de Test Bench

CONSIDERACIONES

La ALU a desarrollar debe poseer una arquitectura como la siguiente



Y ser capaz de realizar determinadas operaciones, de acuerdo al opcode ingresado

OPERACION	CODIGO
ADD	100000
SUB	100010
AND	100100
OR	100101
XOR	100110
SRA	000011
SRL	000010
NOR	100111

DESARROLLO

La placa que se utilizará para grabar el proyecto posee las siguientes características:

Family	Spartan3E
Device	XC3S100E
Package	CP132

Se definen las entradas de la ALU, y los parámetros correspondientes a las operaciones que deberá realizar:

```
module alu #(
    //Parameters
    parameter BUS_SIZE = 32)
    (
    //Inputs
    input wire [BUS_SIZE-1:0] i_data_1,
    input wire [BUS_SIZE-1:0] i_data_2,
    input wire [5:0] i_ctrl,
    //Outputs
    output reg [BUS_SIZE-1:0] o_out
    );

    localparam SRL = 6'b000010;
    localparam SRA = 6'b000011;
    localparam ADD = 6'b100000;
    localparam SUB = 6'b100010;
    localparam AND = 6'b100100;
    localparam OR = 6'b100101;
    localparam XOR = 6'b100110;
    localparam NOR = 6'b100111;
```

La lógica se implementa a través de un case sencillo que, según la operación que se ingrese a través de **i_ctrl**, produce la salida entre los operandos **i_data_1** e **i_data_2** y la coloca a la salida **0_out**.

Para facilitar la identificación visual en la placa, se muestra en el default del case (en caso de que la operación ingresada no se corresponda con las predefinidas) una salida que encienda los 8 leds disponibles:

```
begin: alu
   case (i_ctrl)
      ADD: begin
             o_out = i_data_1 >> i_data_2;
           end
      SRA: begin
             o_out = i_data_1 >>> i_data_2;
           end
      SRL: begin
             o_out = i_data_1 + i_data_2;
           end
      SUB: begin
             o_out = i_data_1 - i_data_2;
           end
      AND: begin
             o_out = i_data_1 & i_data_2;
           end
      OR: begin
             o_out = i_data_1 | i_data_2;
           end
      XOR: begin
             o_out = i_data_1 ^ i_data_2;
           end
      NOR: begin
             o_out = ~(i_data_1 | i_data_2);
           end
      default: begin
            o_out = 8'bllllllll;
           end
   endcase
end
```

Se utilizarán además 3 latches para los datos de ingreso, definidos a partir del siguiente módulo:

```
module latch_data
   #(parameter N=32)
   (
    input [N-1:0] data_in,
    input A,
    output reg [N-1:0] data_out
    );

    always @(A)
      if (A == 1)begin
         data_out = data_in;
      end
endmodule
```

Instanciados todos los componentes de la siguiente manera:

```
module top(
   input clock,
    input [31:0] bus_in,
    input At,
    input Bt,
    input Ct,
    output [31:0] bus_out
    );

    wire [31:0] latch_a_alu;
    wire [31:0] latch_b_alu;
    wire [5:0] latch_c_alu;

    //Instanciar Latch A
    latch_data #(.N(32)) latch_a
      (.data_in(bus_in),.A(At),.data_out(latch_a_alu));

    //Instanciar Latch B
    latch_data #(.N(32)) latch_b
      (.data_in(bus_in),.A(Bt),.data_out(latch_b_alu));

    //Instanciar Latch C
    latch_data #(.N(6)) latch_c
      (.data_in(bus_in),.A(Ct),.data_out(latch_c_alu));

   //Instanciar ALU
   alu alu_top
      (.i_data_1(latch_a_alu),.i_data_2(latch_b_alu),.i_ctrl(latch_c_alu),.o_out(bus_out));

endmodule
```

Por último en el testbench utilizamos valores sencillos de visualizar tanto desde el software como desde la placa:

```
// Initialize Inputs
i_data_1 = 0;
i_data_2 = 0;
i_ctrl = 0;

// Wait 100 ns for global reset to finish
#100;

// Add stimulus here
//ADD
i_data_1 = 32'b11;
i_data_2 = 32'b10;
#10 i_ctrl = 6'b100000;
#100
//SUB
i_data_1 = 32'b11;
i_data_2 = 32'b10;
#10 i_ctrl = 6'b100010;
//AND
i_data_1 = 32'b11;
i_data_2 = 32'b10;
#10 i_ctrl = 6'b100100;
#100
//OR
i_data_1 = 32'b11;
i_data_2 = 32'b10;
#10 i_ctrl = 6'b100101;
#100
//XOR
i_data_1 = 32'b11;
i_data_2 = 32'b10;
#10 i_ctrl = 6'b100110;
#100
//NOR
```