# Universidad Nacional de Córdoba.

# Facultad de Ciencias Exactas, Físicas y Naturales.


Arquitectura de Computadoras

---

# Trabajo Práctico 1:

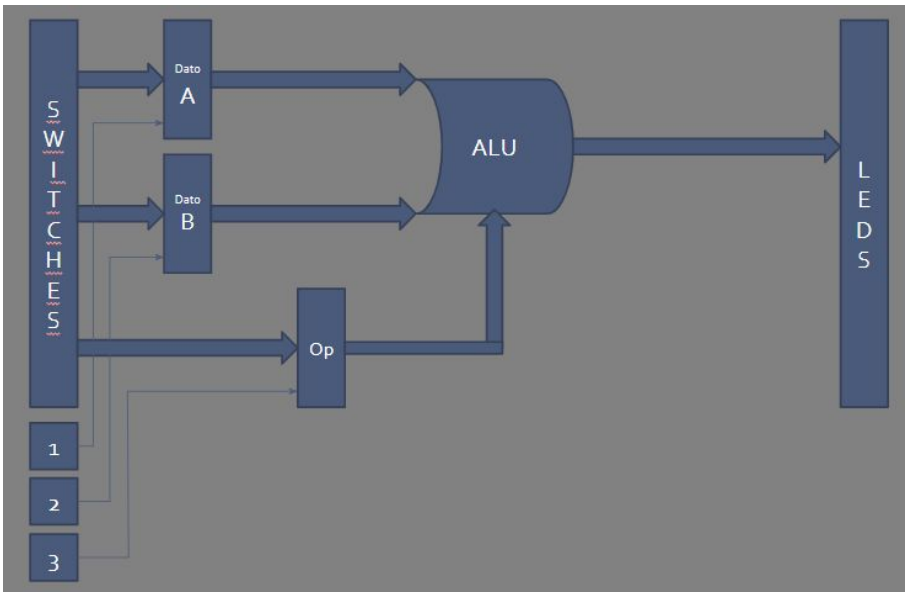
# *ALU*


## Integrantes:

- Drudi, Leandro.
- Goldman, Nicolás.

## CONSIGNA:

Para el primer trabajo de la materia se pide desarrollar una ALU parametrizable en una FPGA (se utilizó la Basys III en nuestro caso). El circuito de dicha ALU es el siguiente:




## DESARROLLO:

Para la implementación del trabajo, se realizaron dos módulos correspondientes a la parte del ***top level*** (switches y botones) y a la parte de la ALU propiamente dicha.

### ALU - TOP LEVEL MODULE:

```
1  `timescale 1ns / 1ps
2
3  module TOPLVL_TP1_ALU
4      #(
5      parameter   N_BUS   = 8,
6      parameter   N_OP    = 6,
7      parameter   N_BTN   = 3
8      )
9      (
10     //ENTRADAS
11     input   signed  [N_BUS-1:0]     i_switch,
12     input           [N_BTN-1:0]     i_btn,
13     input                           i_clk,
14
15     //SALIDAS
16     output  signed [N_BUS-1:0]  o_led
17     );
18
19     reg     signed      [N_BUS-1:0] RegA, RegB;
20     reg                 [N_OP-1:0]  RegOP;
21     wire    signed      [N_BUS-1:0] LEDs;
22
23     assign o_led = LEDs;
24
```

```
25     always @(posedge i_clk)
26         begin
27             casez (i_btn)
28                 3'b??1  :   RegA    <=  i_switch;
29                 3'b?1?  :   RegB    <=  i_switch;
30                 3'b1??  :   RegOP   <=  i_switch; // Ver si toma los MSB o LSB
31             endcase
32         end
33
34     TP1_ALU
35     #(
36         .N_BUS  (N_BUS),
37         .N_OP   (N_OP)
38     )
39
40     alu1
41     (
42         .i_A    (RegA),
43         .i_B    (RegB),
44         .i_OP   (RegOP),
45         .o_RES  (LEDs)
46     );
47
48 endmodule
```

Como podemos ver en las imágenes de arriba, tenemos el código de la sección del *top level*.

Tenemos 3 entradas correspondiente al switch (de 8 elementos), los botones (de 3 elementos) y el clock que será utilizado posteriormente para poder comprobar los estados de cada uno de los switches de la placa.
A su vez podemos ver una salida (*o_led)* que luego de realizar los cálculos correspondientes en la ALU, serán los que nos permitan observar el resultado de la operación seleccionada.

Una vez declaradas las variables que tendrá nuestro sistema, pasamos a la parte de la funcionalidad de nuestro **top_level ALU** en el que podemos ver los registros que se crean para guardar los valores que tengamos en nuestros switches dependiendo del botón que se haya presionado. Este guardado, se realiza en la sección del ***always*** la cual comprueba con cada flanco positivo dicho valor de botón y asigna el valor al registro correspondiente.

Teniendo la sección del guardado de valores en registros, resta realizar los cálculos correspondientes con estos. Para ello, se implementó el módulo de ALU.

## ALU MODULE:

Como se requería en la consigna, el trabajo debía desarrollarse de tal manera que se pueda reutilizar el módulo de ALU para próximas tareas, por lo que se realizó este módulo parametrizable.
Dicho módulo (instanciado desde el **top_level**), recibe como parámetros los registros que guardan los valores de los datos A, B y Operación.

```
`timescale 1ns / 1ps

module TP1_ALU
  #(
    //parametros
    parameter   N_BUS   = 8,
    parameter   N_OP    = 6
   )
   (
    //  ENTRADAS:
    input   signed  [N_BUS-1:0] i_A,
    input   signed  [N_BUS-1:0] i_B,
    input           [N_OP-1:0]  i_OP,

    //  SALIDAS:
    output  signed      [N_BUS-1:0] o_RES
   );
    //Parametros de operacion
    localparam  OP_ADD  = 6'b100000;    //SUMA
    localparam  OP_SUB  = 6'b100010;    //RESTA
    localparam  OP_AND  = 6'b100100;    //AND
    localparam  OP_OR   = 6'b100101;    //OR
    localparam  OP_XOR  = 6'b100110;    //XOR
    localparam  OP_SRA  = 6'b000011;    //SHIFT R (DER) ARITMETICO >>>
    localparam  OP_SRL  = 6'b000010;    //SHIFT R (DER) LOGICO >>
    localparam  OP_NOR  = 6'b100111;    //NOR

reg   signed  [N_BUS-1:0]     RegR;


    always @(*)
        begin
            case (i_OP)
                OP_ADD  :       RegR            <=              i_A + i_B;
                OP_SUB  :       RegR            <=              i_A - i_B;
                OP_AND  :       RegR            <=              i_A & i_B;
                OP_OR   :       RegR            <=              i_A | i_B;
                OP_XOR  :       RegR            <=              i_A ^ i_B;
                OP_SRA  :       RegR            <=              i_A >>> i_B;
                OP_SRL  :       RegR            <=              i_A >> i_B;
                OP_NOR  :       RegR            <=              ~(i_A | i_B);
                default :       RegR            <=              6'b000000;
            endcase
        end

assign o_RES = RegR;

endmodule
```

Cuando se instancia el módulo de ALU, este se encarga de analizar los cambios en las variables y en base al valor de la operación, realizar el cálculo correspondiente.

Por último tenemos la asignación del resultado a la variable de **o_RES** correspondiente a uno de los parámetros pasados por el top_level, para de esta forma, lograr ver los resultados correctos en la FPGA.