

---

# Pràctica 3: ALU (Unitat Aritmètica-Lògica)

Curs 2024/25

---

Darius Natan SANTA  
Paul-Cristian CRISTEA

## Índex

<b>Plantejament de la solució.....</b>	<b>2</b>
<b>Codi VHDL.....</b>	<b>3</b>
1. Declaració de llibreries i entitat.....	3
2. Declaració de l'arquitectura i senyals.....	4
3. Inici del procés principal.....	4
4. Estructura case per seleccionar l'operació.....	5
5. Operacions.....	6
5. Assignació final del resultat.....	6
<b>Demostració del funcionament de la testbench.....</b>	<b>7</b>
<b>Simulació amb GTKWAVE.....</b>	<b>7</b>

## Plantejament de la solució

Plantejament per a la Solució de les Operacions

L'arquitectura implementada utilitza VHDL per descriure el comportament de l'ALU.

Les operacions que es poden realitzar inclouen:

### 1. Comparació (sel = "0001")

Es comparen els valors d'entrada A i B. En funció del resultat:

- Si  $A < B$ , s'envien valors predefinits als displays per indicar que A és menor.
- Si  $A = B$ , s'envien valors per indicar que són iguals.
- Si  $A > B$ , s'envien valors per indicar que A és més gran.
- El resultat de l'ALU (resultat) es fixa en zero, ja que la comparació no genera un valor numèric.

### 2. Suma (sel = "0010")

Es calcula la suma de A i B utilitzant representació no signada (unsigned). El resultat es retorna en resultat.

### 3. Resta (sel = "0100")

Es realitza la resta entre A i B. Si el resultat és negatiu, un senyal especial (codificat) es mostra al tercer display per indicar-ho.

#### 4. Desplaçament (sel = "1000")

Es desplaça a l'esquerra el valor d'inp\_a segons el valor numèric d'inp\_b. Aquesta operació és útil en càlculs que impliquen multiplicacions per potències de 2.

#### 5. Operació per defecte (sel = others)

Si no es reconeix l'operació indicada, l'ALU fixa el resultat en zero i mostra un missatge predefinit als displays.

## Codi VHDL

A continuació, es presenta el codi de l'ALU per fragments, amb una explicació detallada del que fa cada part.

### 1. Declaració de llibreries i entitat

```
vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
entity alu is
    Port (
        inp_a : in STD_LOGIC_VECTOR(4 downto 0);
        inp_b : in STD_LOGIC_VECTOR(4 downto 0);
        sel : in STD_LOGIC_VECTOR(3 downto 0);
        display1 : out STD_LOGIC_VECTOR(7 downto 0);
        display2 : out STD_LOGIC_VECTOR(7 downto 0);
        display3 : out STD_LOGIC_VECTOR(7 downto 0);
        out_alu : out STD_LOGIC_VECTOR(4 downto 0)
    );
end alu;
```

Practica 3

La entitat alu defineix els ports:

Entrades:

- *inp\_a* i *inp\_b*: Operands de 5 bits.
- *sel*: Selector de l'operació de 4 bits.

Sortides:

- *display1*, *display2*, *display3*: Mostren informació en displays de 7 segments.
- *out\_alu*: Resultat de l'operació en 5 bits.

## 2. Declaració de l'arquitectura i senyals

```
vhdl
architecture behaviour of alu is
    signal A, B, resultat : STD_LOGIC_VECTOR(4 downto 0);
    signal temp : unsigned(4 downto 0);
begin
```

Practica 3

Es declaren els senyals:

- *A i B*: Representen les còpies dels valors d'*inp\_a* i *inp\_b*.
- *resultat*: Emmagatzema el resultat de l'operació seleccionada.
- *temp*: Variable temporal, utilitzada per fer càlculs amb números unsigned.

## 3. Inici del procés principal

```
vhdl
process(inp_a, inp_b, sel)
begin
    A <= inp_a;
    B <= inp_b;
```

Practica 3

Explicació:

- Es declara un procés que s'executa quan es produeixen canvis a *inp\_a*, *inp\_b* o *sel*.
- Les senyals A i B s'assignen directament des de les entrades *inp\_a* i *inp\_b*.

#### 4. Estructura case per seleccionar l'operació

```
vhdl
case sel is
  when "0001" => -- Comparació
    resultat <= "00000";
    if (A < B) then
      display1 <= "00000000";
      display2 <= STD_LOGIC_VECTOR(to_unsigned(80, 8));
      display3 <= STD_LOGIC_VECTOR(to_unsigned(111, 8));
    elsif (A = B) then
      display1 <= "00000000";
      display2 <= STD_LOGIC_VECTOR(to_unsigned(121, 8));
      display3 <= STD_LOGIC_VECTOR(to_unsigned(103, 8));
    else
      display1 <= "00000000";
      display2 <= STD_LOGIC_VECTOR(to_unsigned(56, 8));
      display3 <= STD_LOGIC_VECTOR(to_unsigned(111, 8));
    end if;
```

Practica 3

Explicació:

- Quan *sel* = "0001", es compara A amb B.
- En funció del resultat ( $A < B$ ,  $A = B$ ,  $A > B$ ), s'envien valors codificats als *display2* i *display3* per indicar l'estat de la comparació.
- El resultat final (*resultat*) es fixa en zero perquè aquesta operació no genera un valor numèric.

## 5. Operacions

```
vhdl
when "0010" => -- Suma
    resultat <= STD_LOGIC_VECTOR(unsigned(A) + unsigned(B));
when "0100" => -- Resta
    temp <= unsigned(A) - unsigned(B);
    display1 <= "00000000";
    display2 <= "00000000";
    display3 <= "00000000";
    if (temp < 0) then
        display3 <= STD_LOGIC_VECTOR(to_unsigned(64,8));
    end if;
when "1000" => -- Shift
    resultat <= STD_LOGIC_VECTOR(shift_left(unsigned(inp_a),
    to_integer(unsigned(inp_b))));
when others =>
    display1 <= STD_LOGIC_VECTOR(to_unsigned(121, 8));
    display2 <= STD_LOGIC_VECTOR(to_unsigned(80, 8));
    display3 <= STD_LOGIC_VECTOR(to_unsigned(80, 8));
    resultat <= "00000";
```

Practica 3

## 5. Assignació final del resultat

```
vhdl
process(inp_a, inp_b, sel)
begin
    A <= inp_a;
    B <= inp_b;
```

Practica 3

Explicació:

- El valor final calculat en *resultat* es transmet a la sortida *out\_alu*.
- Es tanca el procés i l'arquitectura.

## Demostració del funcionament de la testbench

Mitjançant *assert*, es comprova si el resultat obtingut en cada operació és correcte, i si no ho és, s'activa un missatge d'error amb *report* que especifica quina part de la prova ha fallat. En cas que totes les comprovacions siguin correctes, la simulació es completa sense errors, confirmant que l'ALU funciona segons l'especificació.

```
bash
alu_tb.vhdl:28:5:@0ms:(assertion note): Display 1 correcta
alu_tb.vhdl:29:5:@0ms:(assertion note): Display 2 correcta
alu_tb.vhdl:30:5:@0ms:(assertion note): Display 3 correcta
alu_tb.vhdl:35:5:@10ns:(assertion note): Suma correcta
alu_tb.vhdl:40:5:@20ns:(assertion note): Resta correcta
alu_tb.vhdl:45:5:@30ns:(assertion note): Shift correcte
```

Practica 3

## Simulació amb GTKWAVE

