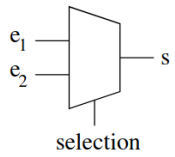
Architecture des ordinateurs - TD 1

On désire implanter et tester un multiplexeur 2 bits vers 1 en VHDL et ceci en utilisant les différentes
vues abordées en cours (vue structurelle, vue flot de données, vue comportementale).

L'interface du multiplexeur est la suivante



`s` vaut `e1` si `selection = '0'`
`s` vaut `e2` si `selection = '1'`

Nous disposons de 3 composants porte et, porte ou et inverseur dont voici les interfaces :

```
-- temps de traversée : 2 ns
entity porte_et is
  port ( i1, i2 : in std_logic;
           s  : out std_logic );
  end porte_et;

-- temps de traversée : 2 ns
entity porte_ou is
  port ( i1, i2 : in std_logic;
           s  : out std_logic );
  end porte_ou;

-- temps de traversée : 1 ns
entity inverseur is
  port ( i : in std_logic;
         s : out std_logic );
  end inv;
```

1. donner l'interface vhdl du multiplexeur

```
entity multiplexeur is
    port ( e1, e2, selection : in std_logic;
                           s : out std_logic;
    );
end multiplexeur
```

2. décrire les architectures de ce composant selon les trois vues

```
architecture dataflow1 of multiplexeur is
    begin
        s <= e1 after 5 ns when ( selection = '0') else
             e2 after 5 ns when ( selection = '1' ) else
             'X' after 5 ns;
    end dataflow1
```

Ce qui est factorisable avec un switch case vhdl

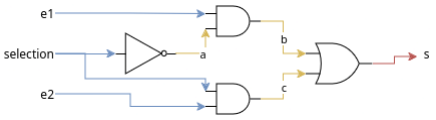
```
architecture dataflow1 of multiplexeur is
    begin
        with selection select
        s <= e1 after 5 ns when '0',
             e2 after 5 ns when '1',
             'X' after 5 ns when others;
    end dataflow1

```

Le fait d'avoir défini la structure architectural de notre composant permet de factoriser du code, dans les faits si on voulait on aurait pu écrire tout le comportement du composant avec ces signaux interne (faut être un enculé pour)

```
architecture dataflow1 of multiplexeur is
    signal a, b, c : std_logic
    begin
        with selection select
        s <= b or c after 2 ns
        a <= not e1 after 1 ns
        ...

    end dataflow1

```

On a comme description comportemental du composant :

```
architecture behavoural of myultiplexeur is
    begin
        process(e1, e2, selection)
            begin
                s <= (e1 and not selection) or (e2 and selection) after 5ns;
        end process;
end;

```




```
architecture structural of multiplexeur is

    -- Définition des composants
    component porte_et
        port(i1, i2 : in std_logic;
                  s : out std_logic);
    end component;
    component inverseur
        port(i1 : in std_logic;
              s : out std_logic);
    end component
    component porte_ou
        port(i1, i2 : in std_logic;
                  s : out std_logic);
    end component;

    --- Définition signal
    signal a, b, c: std_logic;

begin

    -- Architecture du composant et signaux intermédiaire.
    u1 : inverseur port map (selection, a); -- équivalent à port map (i1 => se
    u2 : porte_et port map (i1 => e1, i2 => a, s => b);
    u3 : porte_et port map (i1 => selection, i2 => e2, s => c);
    u4 : porte_ou port map (i1 => b, i2 => c, s);

end structural;
```

Equation logique:
s = e1 . !selection + e2.selection

Ainsi on a le dataflow suivant : (Attention a pas oublier qu'on est dans le domaine concurant pour ce genre de description!)

```
architecture dataflow1 of multiplexeur is
    begin
        s <= (e1 and (not selection)) or (e2 and selection) after 5 ns
    end dataflow1
```

On peut aussi faire avec des if :