Université M’hamed BOUDIAF de M’SILA

Faculté de Technologie

Département Electronique

TP n°4 Electronique Numérique Avancée

Développement d’un deuxième exemple de circuit : multiplexeur.

1. Objectifs

Ecrire un programme VHDL d'un multiplexeur 2 vers 1 en utilisant les deux implémentations parallèle et séquentielle. Ensuite on va construire un multiplexeur 4 vers 1 en utilisant le composant (multiplexeur 2 vers 1) déjà réalisé.

2. Manipulations

1) Multiplexeur MUX 2-1

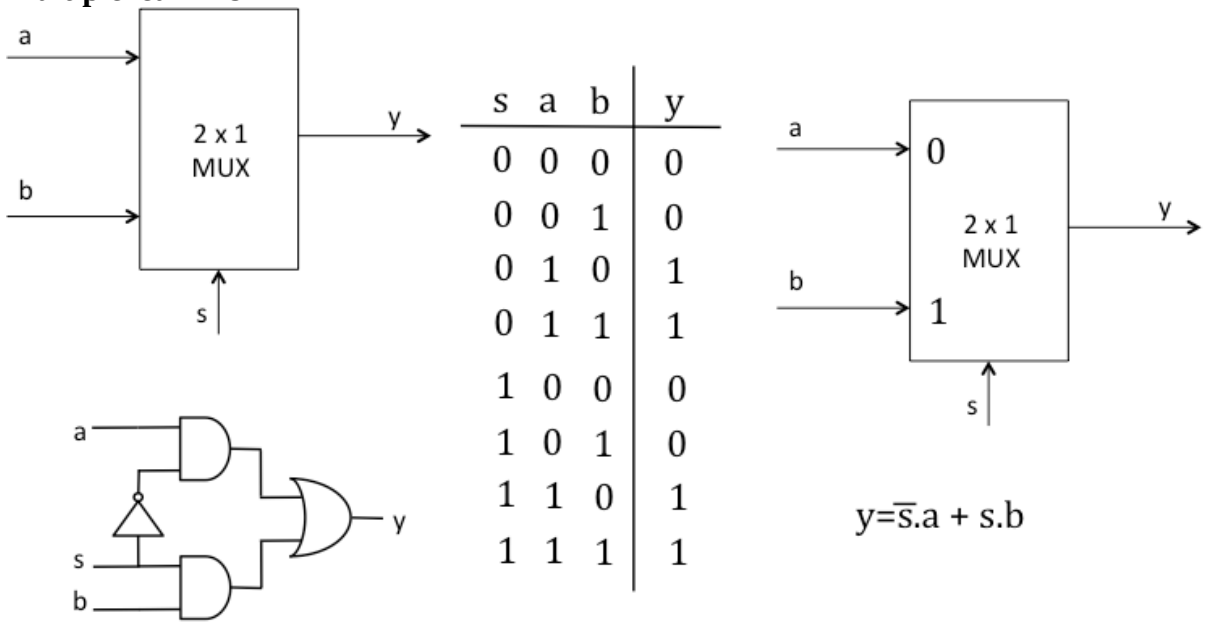

Figure : Schéma bloc d'un MUX 2x1, table de vérité et circuit logique

Ce commutateur à 2 entrées est contrôlé par une seule ligne de contrôle s. Ce bit sélectionne une des 2 entrées à connecter à la sortie. Ce qui signifie que la valeur logique de sortie y sera la même que la valeur logique de l'entrée sélectionnée.

Implémentation parallèle :

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
  entity mux21a is
   port(
          a: in std_logic;
          b: in std_logic;
          s: in std_logic;
          y: out std_logic
         );
end mux21a;
   architecture Behavioral of mux21a is
      begin
          y <= (not s and a) or (s and b) ;
end Behavioral;
```

- écrire le testbench (remarque : on peut utiliser 3 clocks sur a, b, s) et faire la simulation

Implémentation séquentielle:
On utilise un process (remarque, que se passe-t-il si on enlève un des paramètres de la liste de sensibilité ?)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
     entity mux21b is
         port(
                a: in std_logic;
                b: in std_logic;
                s: in std_logic;
                y: out std_logic
              );
     end mux21b;
     architecture Behavioral of mux21b is
       begin
         p1 : process(a,b,s)
         begin
            if s = '0' then y <= a ;
              else y <= b ;
            end if ;
        end process ;
     end Behavioral;
```

- écrire le testbench (remarque : on peut utiliser 3 clocks sur a, b, s) et faire la simulation

Implémentation parallèle différente:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
     entity mux21c is
       port(
             a: in std_logic;
            b: in std_logic;
            s: in std_logic;
            y: out std_logic
             );
    end mux21c;
        architecture Behavioral of mux21c is
           begin
             y <= a when s='0', else b;
        end Behavioral;
```

- écrire le testbench (remarque : on peut utiliser 3 clocks sur a, b, s) et faire la simulation

Synthèse et implémentation sur SPARTAN 3E :

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
    entity mux21_top is
       port(
              sw: in std_logic_vector(1 downto 0);
             btn: in std_logic_vector(0 downto 0);
             led: out std_logic_vector(0 downto 0)
             );
       end mux21_top;
```

```
architecture Behavioral of mux21_top is
     component mux21b
         port(
          a: in std_logic;
          b: in std_logic;
          s: in std_logic;
          y: out std_logic
          );
     end component ;
 begin
      c1 : mux21b port map( a=> sw(0), b => sw(1), s => btn(0), y=> ld(0)) ;
 end Behavioral;
```

- Faire la synthèse et l'implémentation de ce composant.

2) Multiplexeur MUX 4-1

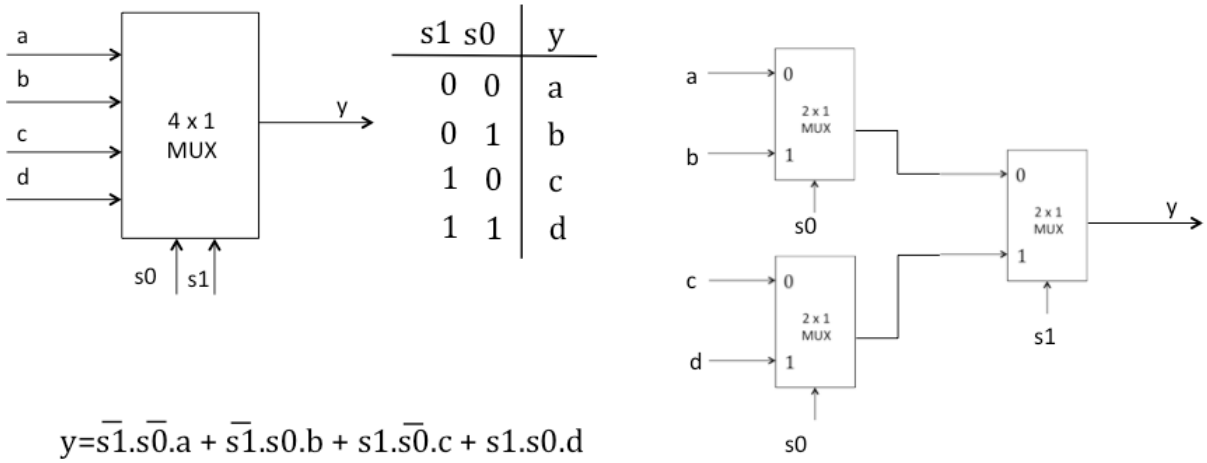

Figure : Schéma bloc d'un MUX 4x1, table de vérité et circuit logique

Ce MUX nécessite 2 lignes de contrôle, s0 et s1. Ces 2 bits sélectionnent une des 4 entrées à connecter à la sortie. Il est possible de faire un MUX 4 vers 1 à partir d'un arbre de 3 MUX 2x1. De la même manière, on peut construire des MUX 8x1, 16x1, etc.

- Ecrire le code VHDL et le testbench de ce composant

3) Multiplexeur MUX 2-1 4 bits (MUX 2-1 Quad)

Figure : Schéma bloc d'un MUX 2x1 quad, table de vérité et circuit à base de MUX 2x1
Ce type de MUX est appelé Quad car il nécessite 4 MUX 2-1 pour sa réalisation. On peut de la même manière réaliser des MUX n-bits.

- Ecrire le code VHDL et le testbench de ce composant