Fiche 3 : Combinatoires/Séquentiels

✎ Romain Kamdem

⇝ EXERCICE 1 (ModifIN) ✎

L'objectif de cet exercice est de construire le circuit representé par la figure 1. Ce circuit prend une entrée X sur N bits et met sur la sortie soit la valeur de X, soit 0, soit l'inverse de X ou l'inverse de 0 comme résumé dans le tableau 1 suivant les valeurs de zero et de neg.

1. *Écrire en VHDL l'entité de ce circuit.*
2. *Implémenter une architecture structurelle de votre circuit.*
3. *Construire un testbench pour simuler votre circuit*

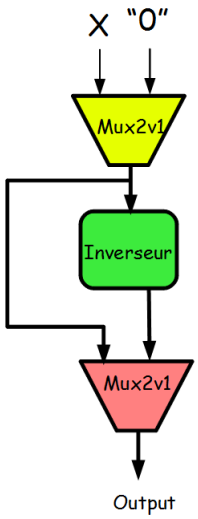

Figure 1: Chemin de données

Table 1: Spécifications

zero	neg	Sortie
0	0	**X**
1	0	**0**
0	1	Inverse de **X**
1	1	Inverse de **0**

⇝ EXERCICE 2 (LoadCount) ✎

L'objectif de cet exercice est d'implémenter le circuit décrit par la figure 2. Le comportement du circuit est le suivant quand le port d'entré st est égal à 0, la sortie du circuit est incrémentée à chaque cycle d'horloge. Quand st est égal à 1 le contenu de l'entrée X est chargé dans le port de sortie.

1. *Ecrire en VHDL une entité de ce circuit.*
2. *Ecrire une architecture structurelle en VHDL de ce circuit*

3. *Grace à un testbench simuler votre implémentation*

4. *Décrire une nouvelle architecture de ce circuit en utilisant un processus clocké asynchrone.*

5. *Ecrire en VHDL une configuration qui permette de simuler cette deuxième architecture.*

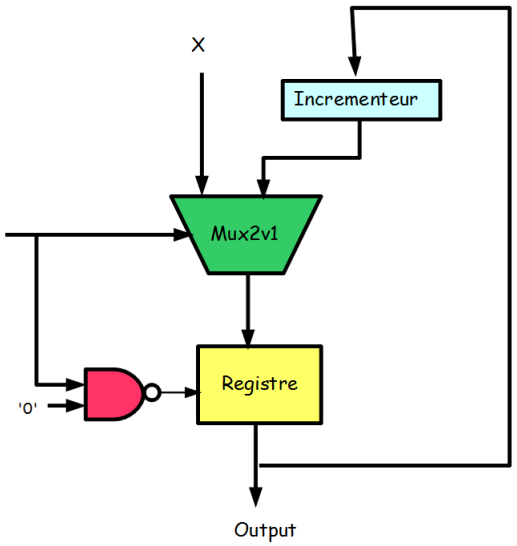

Figure 2: Chemin de données compteur

☞ EXERCICE 3 (ALU) ✎

On se propose ici de construire une unité arithmétique et logique représentée par la figure 3. Les entrées/sorties du circuit seront definies en se basant sur les composant qui sont dans le chemin de données en évitant toute redondance. Les spécifications du circuit sont données par le tableau 2.

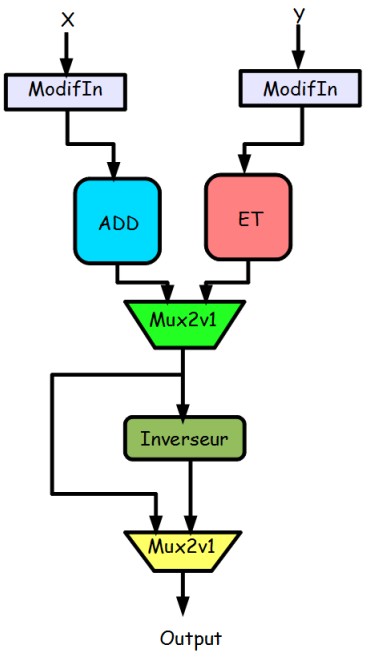

Figure 3: Chemin de données Unité Arithmétique et Logique

Table 2: Spécifications

zx	nx	zy	ny	f	no	Sortie
1	0	1	0	1	0	**0**
1	1	1	1	1	1	**1**
1	1	1	0	1	0	**-1**
0	0	1	1	0	0	**X**
1	1	0	0	0	0	**Y**
0	0	1	1	0	1	**not X**
1	1	0	0	0	1	**not Y**
0	0	1	1	1	1	**-X**
1	1	0	0	1	1	**-Y**
0	1	1	1	1	1	**X + 1**
1	1	0	1	1	1	**Y + 1**
0	0	0	0	1	0	**X + Y**
0	0	1	1	1	0	**X - 1**
1	1	0	0	1	0	**Y - 1**
0	1	0	0	1	1	**X - Y**
0	0	0	1	1	1	**Y - X**
0	0	0	0	0	0	**X and Y**
0	1	0	1	0	1	**X or Y**

1. *Définir et écrire en VHDL l'entité et l'architecture de ce circuit*
2. *Construire un testbench pour simuler votre circuit.*

☞ EXERCICE 4 (Fichiers/RAM) ✎

Le code suivant décrit le comportement d'une mémoire qui lit un fichier et stocke les données. Recopiez ce code, compilez le et construire un test bench qui permet de simuler cette mémoire (créer un fichier, charger le en mémoire puis lire le contenu, écrire/lire en mémoire à une adresse donnée).

```
library ieee;
use ieee.std_logic_textio.all;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;
use std.textio.all;

entity MA_RAM is
  generic (
    load_file_name : string;
    dta  : integer := 8;
    adr  : integer := 4);

  port ( address   : in std_logic_vector(adr - 1 downto 0);
         datain    : in std_logic_vector(dta - 1 downto 0);
         wr_ena    : in std_logic;
         dataout   : out std_logic_vector(dta - 1 downto 0)
          );
  end MA_RAM;

architecture beh of MA_RAM is
begin
  p_init_mem : process

    subtype myword is std_logic_vector(0 to datain'length - 1);
    type storage_array is
      array (natural range 0 to 2**address'length - 1) of myword;
    variable storage  : storage_array;
    variable index    : natural;
    variable mem_data : line;
    variable vdata    : integer;
    file load_file    : text is load_file_name;

     begin
       index := 0;
       while not endfile(load_file) loop
        readline(load_file,mem_data);
        if mem_data'length > 0 then
         read(mem_data, vdata);
         storage(index):=std_logic_vector(conv_unsigned(vdata,datain'length));
         index := index + 1;
        end if;
        if index = (2**address'length) then
          index := 0;
        end if;
       end loop;

       loop
         wait on address, wr_ena, datain;
         if wr_ena = '0' then
           index    := conv_integer(unsigned(address));
```

```
            dataout  <= std_logic_vector(storage(index));
          else
            index           := conv_integer(unsigned(address));
            storage(index) := datain;
          end if;
        end loop;
      end process;
end beh;
```