

CIRCUITS SÉQUENTIELS

EXERCICES

- ❶ Décrire une bascule D avec une remise à zéro (*reset*) asynchrone.
- ❷ Rajouter à la bascule D précédente une remise à 1 (*preset*) asynchrone
- ❸ Rajouter à la bascule D précédente un signal d'activation (*enable*) synchrone. Décrire un fichier testbench permettant de tester la bascule D.
- ❹ Décrire en VHDL un registre 8 bits avec un *reset* asynchrone. Décrire un fichier testbench permettant de tester le registre.
- ❺ Décrire en VHDL un compteur générique (8 bits par défaut) avec chargement parallèle et *reset* synchrone. Décrire un fichier testbench permettant de tester le compteur 8 bits.
- ❻ Décrire en VHDL un compteur/décompteur générique (8 bits par défaut) avec chargement parallèle, *reset* asynchrone, signal **enable** et un signal de sens (**count/decount**). Décrire également un fichier testbench permettant de tester le circuit.

EXERCICE 1

SOLUTION

```
library ieee;
use ieee.std_logic_1164.all;
entity dffr is
    port(
        clk: in std_logic;
        reset: in std_logic;
        d: in std_logic;
        q: out std_logic
    );
end dffr;

architecture arch of dffr is
begin
```

EXERCICE 1

SOLUTION

```
process(clk,reset)
begin
    if (reset='1') then
        q <='0';
    elsif (clk'event and clk='1') then
        q <= d;
    end if;
end process;
end arch;
```

EXERCICE 2

SOLUTION

```
library ieee;
use ieee.std_logic_1164.all;
entity dffrp is
    port(
        clk: in std_logic;
        reset, preset: in std_logic;
        d: in std_logic;
        q: out std_logic
    );
end dffrp;

architecture arch of dffrp is
begin
```

EXERCICE 2

SOLUTION

```
process(clk,reset,preset)
begin
    if (reset='1') then
        q <='0';
    elsif (preset='1') then
        q <= '1';
    elsif (clk'event and clk='1') then
        q <= d;
    end if;
end process;
end arch;
```

EXERCICE 3

SOLUTION

```
library ieee;
use ieee.std_logic_1164.all;
entity dff_en is
    port(
        clk: in std_logic;
        reset: in std_logic;
        en: in std_logic;
        d: in std_logic;
        q: out std_logic
    );
end dff_en;

architecture two_seg_arch of dff_en is
```

EXERCICE 3

SOLUTION

```
signal q_reg: std_logic;  
signal q_next: std_logic;  
begin  
  -- D FF  
  process(clk,reset)  
  begin  
    if (reset='1') then  
      q_reg <= '0';  
    elsif (clk'event and clk='1') then  
      q_reg <= q_next;  
    end if;  
  end process;  
  -- next-state logic  
  q_next <= d when en ='1' else
```

EXERCICE 3

SOLUTION

```
                q_reg;
-- output logic
q <= q_reg;
end two_seg_arch;
architecture one_seg_arch of dff_en is
begin
    process(clk,reset)
    begin
        if (reset='1') then
            q <='0';
        elsif (clk'event and clk='1') then
            if (en='1') then
                q <= d;
            end if;
        end if;
    end process;
end one_seg_arch;
```


EXERCICE 3

SOLUTION

```
        end if;  
    end process;  
end one_seg_arch;
```

EXERCICE 4

SOLUTION

```
library ieee;
use ieee.std_logic_1164.all;
entity reg8 is
    port(
        clk: in std_logic;
        reset: in std_logic;
        d: in std_logic_vector(7 downto 0);
        q: out std_logic_vector(7 downto 0)
    );
end reg8;

architecture arch of reg8 is
begin
```

EXERCICE 4

SOLUTION

```
process(clk,reset)
begin
    if (reset='1') then
        q <=(others=>'0');
    elsif (clk'event and clk='1') then
        q <= d;
    end if;
end process;
end arch;
```

EXERCICE 5

SOLUTION

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity COMPTEUR is
  generic (TAILLE : integer := 8 );
  port(
    DIN    : in  std_logic_vector(TAILLE-1 downto 0);
    CLK    : in  std_logic;
    LOAD   : in  std_logic;
    DOUT   : out std_logic_vector(TAILLE-1 downto 0));
end COMPTEUR;
architecture behavior of COMPTEUR is begin
  clk_proc:process(CLK)
```

EXERCICE 5

SOLUTION

```
variable COUNT:unsigned(TAILLE-1 downto 0) := (others
=> '0');
begin
  if rising_edge(CLK) then
    if LOAD = '1' then
      COUNT := unsigned(DIN);
    else COUNT := COUNT + 1;
    end if;
  end if;
  DOUT <= std_logic_vector(COUNT);
end process clk_proc;
end behavior;
```

EXERCICE 6

SOLUTION

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity compteur_decompteur is
  generic (TAILLE : integer := 16 );
  port(
    din    : in  std_logic_vector(TAILLE-1 downto 0);
    clk    : in  std_logic;
    load   : in  std_logic;
    reset  : in  std_logic;
    updown : in  std_logic;
    dout   : out std_logic_vector(TAILLE-1 downto 0));
end;
architecture behavior of compteur_decompteur is begin
```

EXERCICE 6

SOLUTION

```
clk_proc:process(clk)
  variable count:unsigned(TAILLE-1 downto 0) := (others
    => '0');
begin
  if rising_edge(clk) then
    if reset = '1' then
      count := (others => '0');
    elsif load = '1' then
      count := unsigned(din);
    elsif updown = '0' then
      if count = to_unsigned(2**TAILLE-1,TAILLE) then
        count:=(others=> '0');
      else
        count := count + 1;
      end if;
    else
```

EXERCICE 6

SOLUTION

```
        if count = 0 then
            count:= to_unsigned(2**TAILLE-1,TAILLE);
        else
            count := count - 1;
        end if;
    end if;
end if;
dout <= std_logic_vector(count);
end process clk_proc;
end behavior;
```


SOMMAIRE

- Types d'objets
- Process
- Circuits séquentiels
- Utilisation de la carte FPGA

EXEMPLES SIMPLES

CIRCUIT COMBINATOIRE

- ❶ Décrire en VHDL un circuit qui réalise différentes opérations arithmétiques et logiques sur des opérandes 4 bits en fonction d'un sélecteur sur 2 bits.
- ❷ Tester le fonctionnement sur la carte

On utilisera les interrupteurs SW[9] et SW[8] pour la sélection des fonctions (00=>ET ;01=>OU ;10=>OU-Excl ;11=>+) et les opérandes seront codés par SW[7..4] et SW[3..0] respectivement. Les résultats seront visualisés sur les LEDS LEDR[3..0] Pour les contraintes de placement, importer le fichier qsf disponible sur Arche ou en créer un nouveau.

EXEMPLES SIMPLES

CIRCUIT COMBINATOIRE

```
set_location_assignment PIN_AE26 -to HEX0[0]
set_location_assignment PIN_AE27 -to HEX0[1]
set_instance_assignment -name IO_STANDARD LVTTTL -to
    HEX0[0]
set_instance_assignment -name IO_STANDARD LVTTTL -to
    HEX0[1]
```

Après compilation vérifier qu'il n'y a pas d'avertissements critiques (problèmes de placement). Pour la programmation, choisir le circuit de type 5CSEMA puis lui associer le fichier sof du projet.