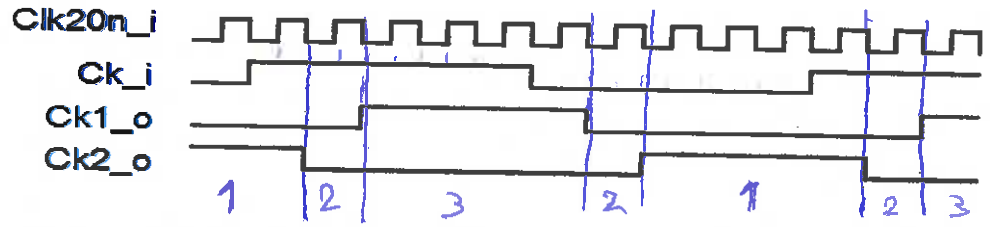
Travail écrit 1: Système Numérique I 2017/2018

Filière : Télécommunication
Classe : T-2a, T-2d
Date : 10 novembre 2017, 13:00 à 14h35
Professeur : Fabio Cunha
Nom et prénom : Zambon Yannick

Points : 22,5 /~~25~~ 23 **Note :** 5.9

Problème 1: Machine d'état

Soit le diagramme temporel (chronogramme) d'un générateur d'horloges ci-dessous :



a) Dessinez la machine d'état pour réaliser ce générateur d'horloge, description de chaque état ainsi que les conditions de transition entre les états.

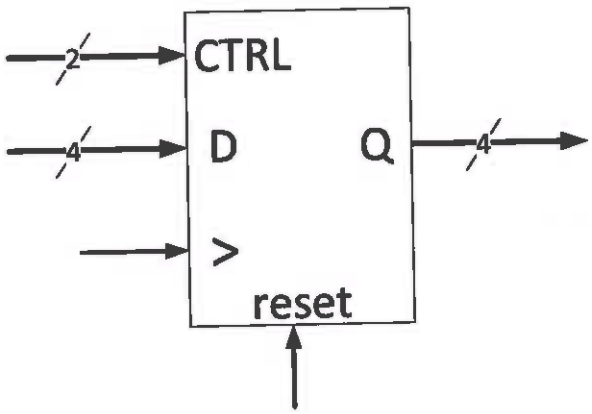
5/ **5 pts**

b) Ecrivez le code VHDL de cette machine. La description VHDL de ce système séquentiel doit être décomposée en trois parties, selon le modèle vu en cours.

5/ **5 pts**

Problème 2: Universal Shift Register

Soit, l'Universal Shift Register suivant :



Avec la table de vérité suivante :

RESET	CTRL	D	CLK	$Q3^+$ $Q2^+$ $Q1^+$ $Q0^+$	Remarque
1	X	X	X	0000	Zéro
0	00	X	↗	Q3 Q2 Q1 Q0	Maintien
0	01	X	↗	Q2 Q1 Q0 Q3	Décalage à droite
0	10	X	↗	Q0 Q3 Q2 Q1	Décalage à gauche
0	11	D3 D2 D1 D0	↗	D3 D2 D1 D0	Chargé valeur

Vous devez :

a) Dessiner à l'aide de flip-flop, portes logiques (AND, OR...) et de fonctions combinatoires standards (Mux, Demux, Shift, Concaténation...) un schéma électronique de ce composant

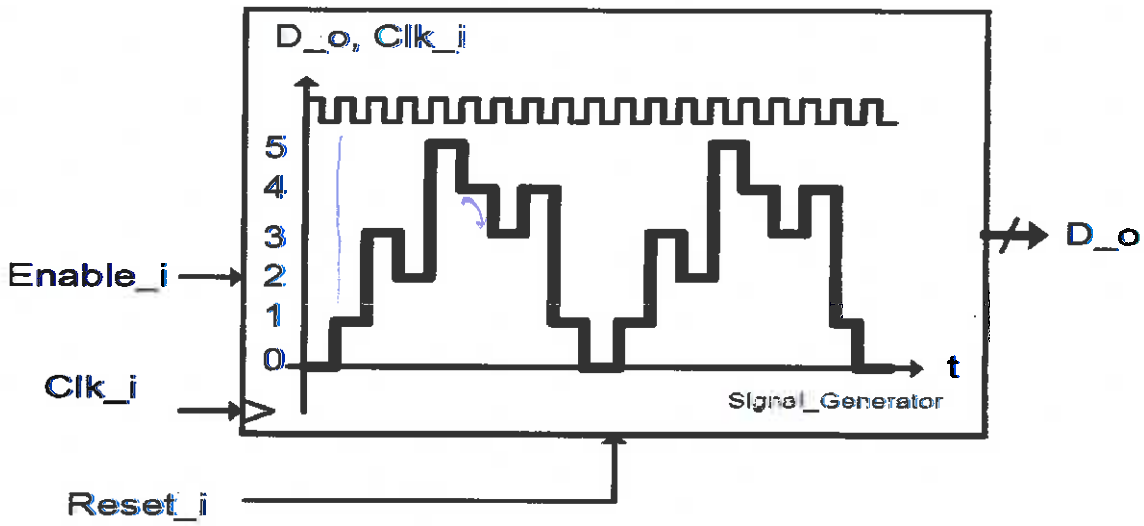
5/ **5 pts**

b) Écrire le code VHDL du composant Universal Shift Register

3.5/ **5 pts**

Problème 3: Signal périodique

Soit le signal périodique suivant à générer :



Si Enable_i = '0', alors la sortie D_o se bloque sur sa dernière valeur.

a) Écrivez le code VHDL du composant Signal_Generator

4/ **5 pts**

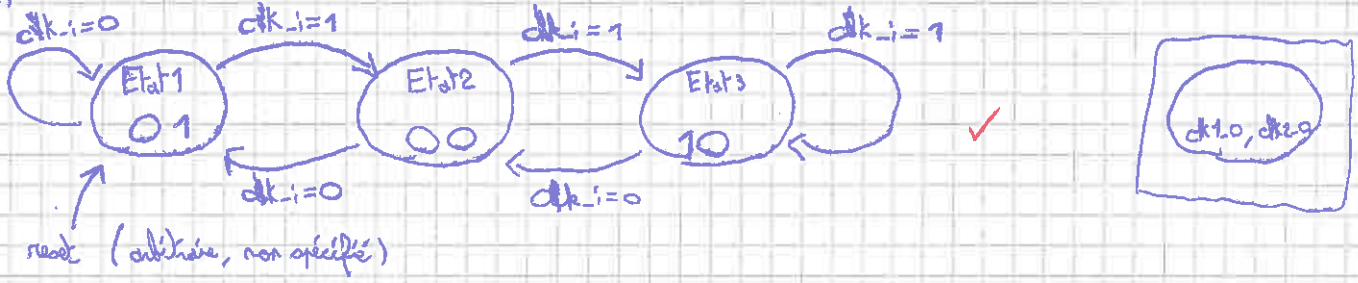
TE 1 : Système numérique

Zanbon Yanick
T2a

Problème 1

Les événements (changement d'état) ont lieu sur le flanc descendant de l'horloge (clk20n_i)

a)



✓

Etat1 : Séquence où ck1_o vaut '0' et ck2_o vaut '1'.
↳ Séquence de 10 cycles dans notre système. ✓

Etat2 : Etat de transition entre l'Etat1 et l'Etat2, dure un cycle : ck1_o = ck2_o = '0' ✓

Etat3 : Séquence de 10 cycles inversée par rapport à l'Etat1 : ck1_o = '1' et ck2_o = '0' ✓

b)

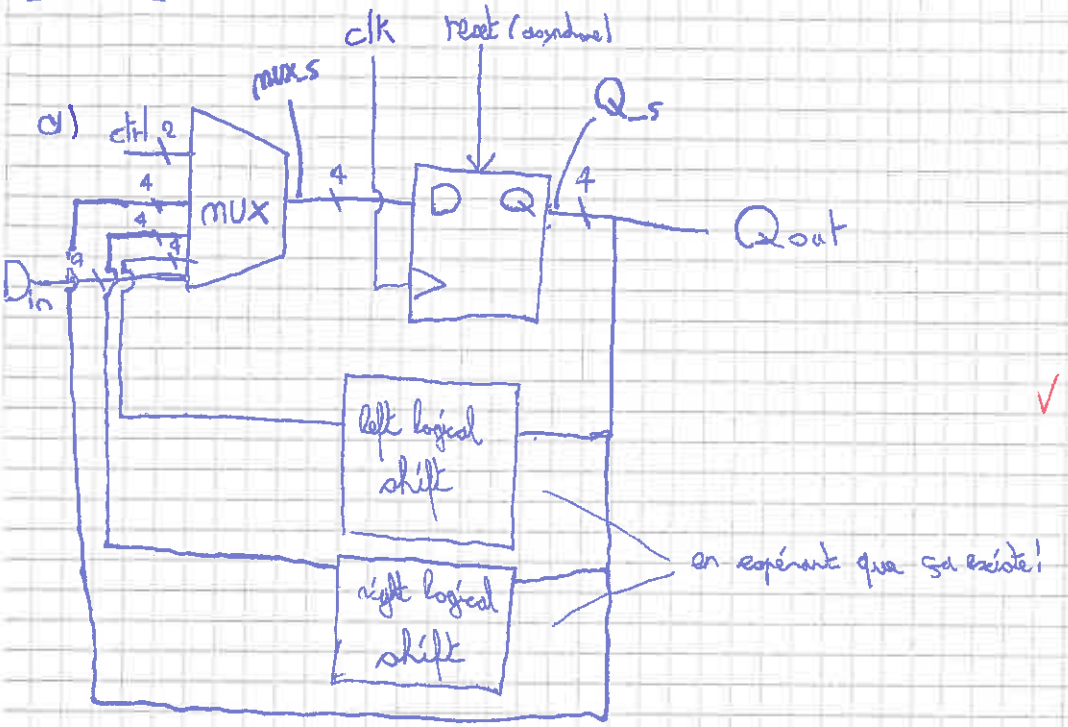
```
library IEEE;
use IEEE.std_logic_1164.all;
entity clock_generator is
        port ( clk20n_1, clk_i, rst : in  std_logic;   -- reset non spécifié (mais plus joli !!
               ck1_o, ck2_o : out std_logic);          ✓
end clock_generator;
architecture Behavior of clock_generator is
    Type state is (Etat1, Etat2, Etat3);
    Signal etat_present, etat_futur : state;          ✓
begin
   registre : process (clk20n_1, rst) (is)
   begin
     if rst='1' then
          etat_present <= Etat1;
     elsif falling_edge(clk20n_1) then   -- presque un le nom exact ☺ pour le flanc descendant!
          etat_present <= etat_futur;    ✓ Très bien
     end if;
   end process;
```

```
machine_etat : process (ck_i , etat_present) is
  begin
    case etat_present is
      when Etat1 =>
          if ck_i='1' then
              etat_futur <= Etat2;
          else
              etat_futur <= Etat1;
          end if;

      when Etat2 =>
          if ck_i='1' then
              etat_futur <= Etat 3;
          else
              etat_futur <= Etat1;
          end if;
      when others =>      -- Etat3
          if ck_i='1' then
              etat_futur <= Etat 3;
          else
              etat_futur <= Etat2;
          end if;
      end case;
  end process;

output : process (etat_present) is
  begin
    if etat_present = Etat1 then
        ck1_o <='0';
        ck2_o <='1';
    elsif etat_present = Etat2 then
        ck1_o <= '0'
        ck2_o <= '0'
    else   -- Etat3
        ck1_o <= '1';
        ck2_o <='0';
    end if;
end process;                                  end architecture;
```

Problème 2

Zemban Yannick
T2-d

a)



✓

b)

```
library IEEE;
use IEEE.std_logic_1164.all;
entity universal_shift_register is
        port ( ctrl : in std_logic_vector (1 downto 0);
               D : in std_logic_vector (3 downto 0);
               clk, reset : in std_logic;
               Q : out std_logic_vector (3 downto 0) );
end entity;
architecture Behavioral of universal_shift_register is
   SIGNAL Q_o, mux_o : std_logic_vector (3 downto 0)
begin
   mux : process ( D, Q_o, CTRL)
     begin
     case CTRL is
          when "00" => mux_s <= Q_o;
          when "01" => mux_s <= Q_o rls 1 ; -- right logical shift?
          when "10" => mux_s <= Q_o ll 1 ; -- left logical shift?
          when "11" => mux_s <= D;
          when others => mux_s <= "XXXX";
     end case;
   end process;
  Q <= Q_o;
```

correction: ror et rol, pas de shift

-1

-1/2 "srl" ne fait pas exactement la fonction demandée

```
flipflop : process (clk, reset) is
  begin
    if reset = '1' then
       Q_s <= "0000";
    elsif rising_edge (clk) then
       Q_s <= mux_o;
    end if;
end process;

end architecture;
```

Problème 3

```
library IEEE;
use IEEE.std_logic_1164.all
entity P3 is
      port ( clk_i, reset_i, enable_i : in std_logic;
             D_o : out std_logic_vector (2 downto 0));
end P3;

architecture Behavioral of P3 is
SIGNAL   D_o : std_logic_vector (2 downto 0);
SIGNAL S1, S3, S4 : std_logic ;
begin
  horloge : process (clk_i, reset_i) is
     begin
        if reset_i = '1' then
           D_o <= "000";
          S1 <='0';   S3 <='0' ; S4 <='0';
        elsif rising_edge (clk_i) then
          if enable_1 = '1' then
             if D_o = "000" then    D_o <= "001";        -- 0
            elsif D_o = "001" then                       -- 1
                        if S1='0' then
                           [illegible];
                           D_o <= "011";
                        else
                           D_o <= "000";
                        end if;
                      S1 <= not S1;
```

```
elsif D_o = "010" then  D_o <= "101";
elsif D_o = "011" then
        if S3 = '0' then
            D_o <= "010";
        else
            D_o <= "100";
        end if;
        S3 <= not S3;
elsif D_o = "100" then
        if S4 = '0' then
            D_o <= "011";
        else
            D_o <= "001";
        end if;
        S4 <= not S4;
elsif D_o = "110" then  D_o <= "100";
else D_o = "XXX";
end if;
end if;
end if;
end process;

D_o <= D_o;
end architecture;
```

un peu compliqué par la compréhension mais ok!

n'existent pas (-1)

"101" je ne sais pas compter jusqu'à 5