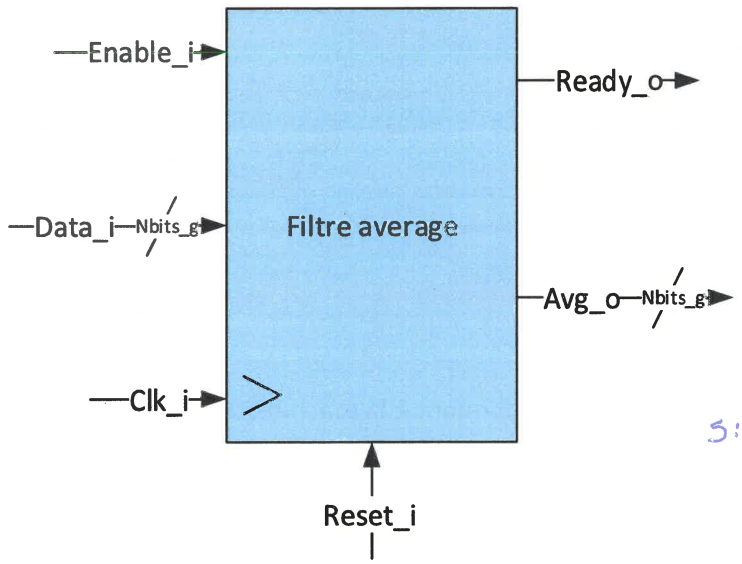
# Travail écrit 1: Système Numérique I 2017/2018

Filière : Télécommunication
Classe : T-2a, T-2d
Date : 22 décembre 2017, 13:00 à 14h35
Professeur : Fabio Cunha
**Nom et prénom :** Zambon Yannick

**Points :** 14.5 /15 **Note :** 5.8

## *Problème 1: Filtre Average*

Soit, un filtre qui consiste à faire la moyenne sur 4 échantillons non-signés:




Vous devez concevoir ce circuit en utilisant un generic « Nbits_g » déterminant la taille des bus d'entrée et sortie ainsi que des bus internes. Notez que :

- Enable_i active le bloc lorsqu'il est à '1' et met les sorties à '0' lorsqu'il n'est pas activé.
- La sortie Ready_o indique lorsque la sortie Avg_o contient une première valeur moyennée (après 4 échantillons) et reste à '1' tant que le filtre continue actif

Vous devez:

a) Écrire le code VHDL correspond à la description ci-dessus

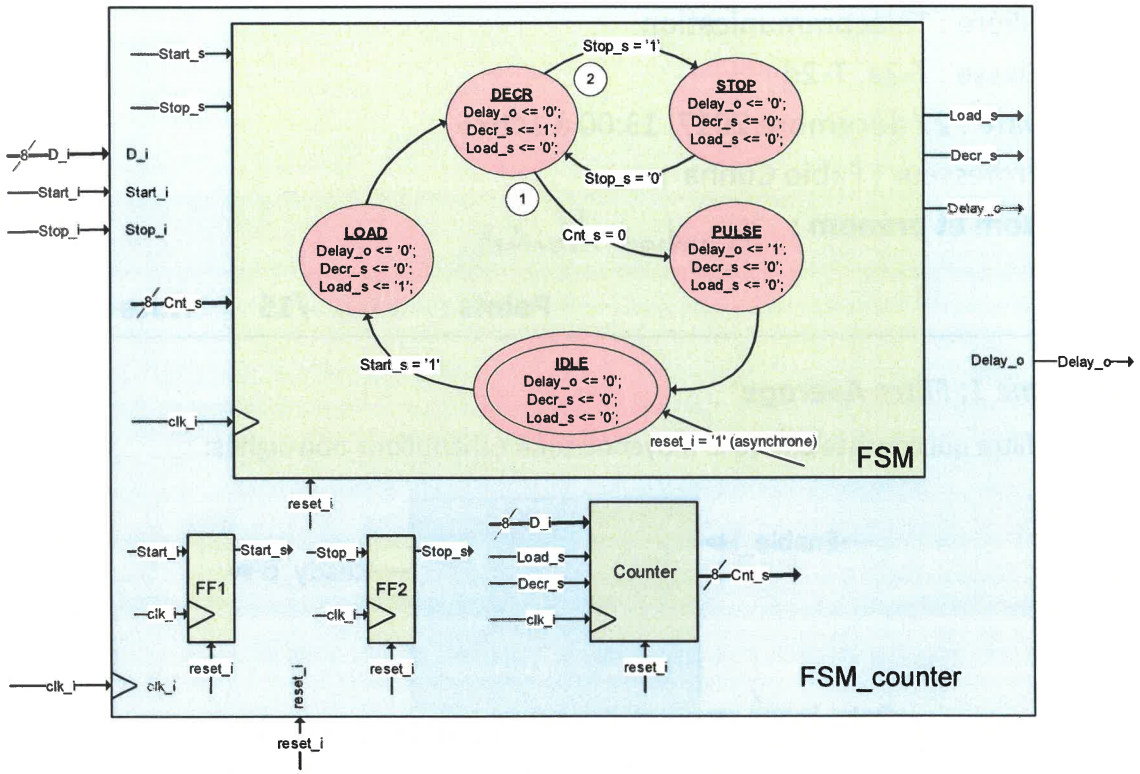
4/ **4 pts**

b) Donner un exemple de chronogramme afin de démontrer le fonctionnement du bloc. Décrivez les signaux internes si nécessaire.

2/ **2 pts**

## *Problème 2: FSM*

Soit, le composant suivant, contenant une machine d'état, des flips-flops et un counter :




Vous devez :

a) Écrire le code VHDL correspondant à la machine d'état ci-dessus

4,5 **5 pts**

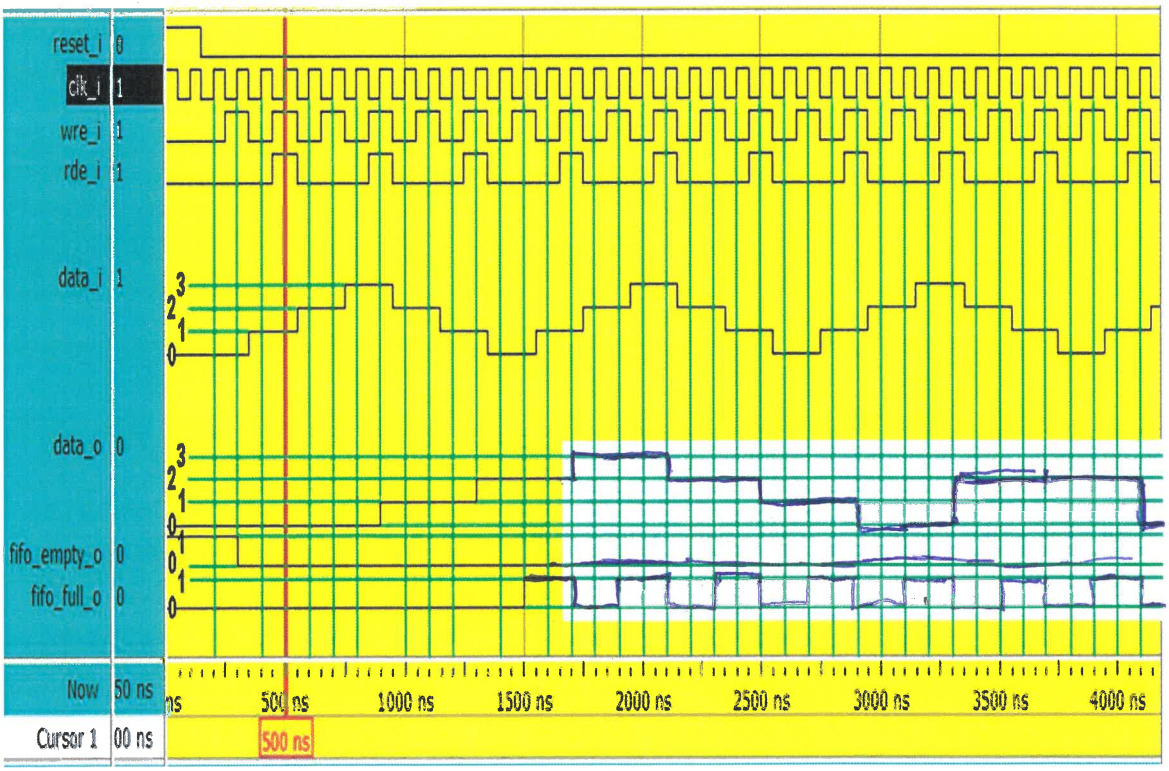
b) Décrire le comportement et la fonctionnalité de ce circuit. Soyez clair dans votre explication.

2/**2 pts**

## *Problème 3 : FIFO*

Soit un FIFO à 4 places mémoires. Le mécanisme d'écriture et de lecture est exactement le même que celui développé au cours. Chaque place mémoire peut contenir un mot de 8 bits. L'horloge système clk_i appliquée à une fréquence de 10MHz. Ce FIFO travaille au flanc montant de clk_i.

Lorsqu'il est plein, il faut faire en premier une lecture avant de pouvoir, le cycle d'après, faire une écriture (ou une écriture et lecture simultanée). Lorsque le FIFO est plein, si l'utilisateur fait quand-même une écriture et une lecture simultanée alors seulement la lecture est effectuée. Lorsqu'il est vide, il faut faire en premier une écriture avant de pouvoir, le cycle d'après, faire une lecture (ou une écriture et lecture simultanée). Lorsque le FIFO est vide, si l'utilisateur fait quand-même une écriture et une lecture simultanée alors seulement l'écriture est effectuée. Dans tous les autres cas une lecture et une écriture peuvent être faites simultanément. Un chronogramme incomplet d'écriture et de lecture est donné ci-dessous:




Les données à écrire data_i vont de 0 à 3, 2 à 1 puis de nouveau de 0 à 3 etc.

a) Complétez, sur cette donnée, le chronogramme

2/ **2 pts**

fifo_empty : jamais (on alterne lecture → écriture dans la zone d'intérêt)

fifo_full : alternée (on lit quand c'est plein (écriture ignorée) puis on écrit et remplit à nouveau)

Zambon Yannick



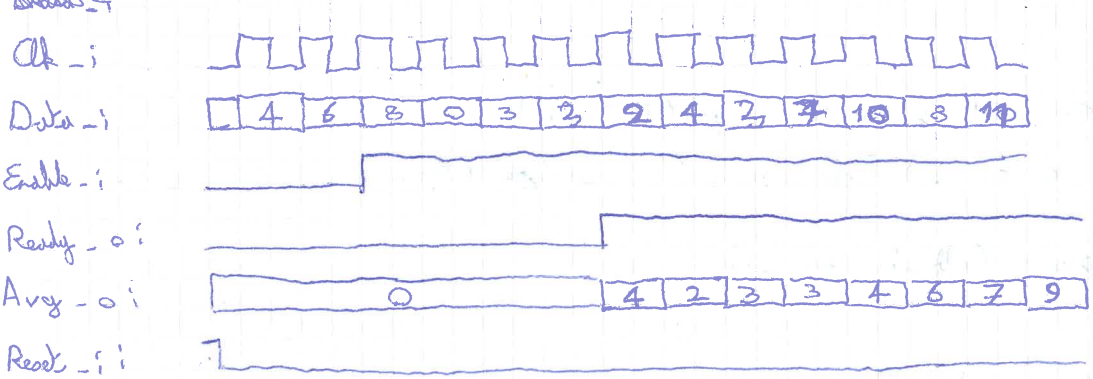
# TE02 Sys. Num.

## Problème 1

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

entity filtre is
    generic ( Nbit_g : integer := 4); -- arbitraire
    port ( Enable_i, clk_i, reset_i : in std_logic;
           Data_i : in std_logic_vector(Nbit_g-1 downto 0);
           Ready_o : out std_logic;
           Avg_o : out std_logic_vector ( Nbit_g-1 downto 0));
end filtre;

architecture behavioral of filtre is

    SIGNAL Datain_s : integer range 0 to 2**(Nbit_g-1);   -- (Dataout_s)
    Type ht_Array is array (0 to 3) of integer range 0 to 2**(Nbit_g-1);
    SIGNAL memory : ht_Array;
    SIGNAL n_value_s : integer range 0 to 4;

    Begin
    Datain_s <= to_integer(unsigned(Data_i));
    Avg_o <= std_logic_vector(to_unsigned(Dataout_s, Nbit_g));

    memory : process (clk_i, reset_i)
        Begin
        if Reset_i = '1' then
            memory <= (others=>(others=>'0'))
            n_value_s <= 0
        elsif rising_edge(clk_i) then
            if enable_i = '1' then
            memory(3) <= memory(2);
            memory(2) <= memory(1);
            memory(1) <= memory(0);
            memory(0) <= Datain_s;
```

```
        if n_value_o <= 4 then
            n_value_o <= n_value_o + 1;
        end if;
    else  -- enable = '0'
        n_value_o <= '0';
        memory <= (others => (others => 0));
    end if;
end if;
end process;

moyenne : process( n_value_o, memory )
    begin
        if n_value_o < 4  then      -- 3 premiers kick ou enable_i = '0'
            Ready_o <= '0';
            Dataout_o <= 0;
        else
            Ready_o <= '1';
            Avg
            Dataout_o <= ( memory(0) + memory(1) + memory(2) + memory(3) ) / 4;
        end if;
end process;
end behavioral;
```

b)



Zamboa Yanicka

T2a
③

Problème 2

```
library IEEE;
use IEEE.std_logic_1164.all;
entity FSM is
    port ( start_i, stop_i, clk_i, reset_i : in std_logic;
           Cnt_i : in std_logic_vector (7 downto 0);
           Load_o, Decr_o, Delay_o : out std_logic );

end FSM;

architecture behavioral of FSM is
 Type state is ( IDLE, LOAD, DECR, STOP, PULSE);
 SIGNAL etat_futur, etat_present : State;
 begin
   sortie : process ( etat_present)
        begin  Delay_o<='0'; Decr_o<='0'; Load_o<='0';
        if (etat_present = IDLE) then
            ~~Delay_o<='0'; Decr_o<='0'; Load_o<='0';~~   -- Rien
        elsif (etat_present = Load) then
            Load_o <='1';
        elsif (etat_present = Decr) then
            Decr_o <='1';
        elsif (etat_present = Pulse) then
            Delay_o <='1';
        else      -- Stop
            -- Rien
        end if;
      end process;
   Registre : process (clk_i, reset_i)
      Begin
          if reset_i = '1' then
              etat_present <= IDLE;
          elsif rising_edge (clk_i) then
              etat_present <= etat_futur;
          end if;
      end process;
```

```
state_machine : process ( start_i , stop_i , cnt_i , etat_present )
  begin
    if ( etat_present = IDLE ) then
          if  start_i ='1' then
              etat_futur <= LOAD;
          else
              etat_futur <= IDLE;
          end if;
    elsif ( etat_present = LOAD ) then
              etat_futur <= Decr;
    elsif ( etat_present = Decr ) then
              if cnt_i ="0000 0000" then
                  etat_futur <= Pulse;
              elsif  Stop_i = '1' then
                  etat_futur <= Stop;
              else
                  etat_futur <= Decr;
              end if
    elsif ( etat_present = Stop) then
              if Stop_i ='0' then
                  etat_futur <= Decr;
              else
                  etat_futur <= Stop;
              end if
    else
              etat_futur <= Idle;

    end if;
  end process;
end behavioral;
```

-1/2

b) Le composant permet de générer une impulsion de 1 cycle d'horloge dans "D_i"+2 cycles. Une valeur d'entrée est placée, on la charge en lançant le compteur (1 cycle), on la décrémente jusqu'à 0 (D_i cycle) puis on génère un pulse de 1 (1 cycle).

Le "chrono" peut également être stoppé et repris là où il s'est arrêté, avec l'entrée Stop_i.

PS : si le compteur recommence, il y aura alors une impulsion tous les ~255+2 cycles.