Question 1. (5 points)

a- Qu'est-ce que le temps de propagation *td*? À quoi correspond-il dans une bascule D?

temps nécessaire pour que la sortie de la composan-te se stabilise suite à un changement à l''une de ses entrées. Pour les loquets et les bascules, on mesure en général le délai de propagation à partir de la transition de l''horloge

b- Qu'est-ce que le temps de préparation *tsu*?

temps minimal pendant lequel le signal d''entrée de la bascule ou du loquet ne doit pas changer avant la transition active de l''horloge

c- Qu'est-ce que le temps de maintien *th*?

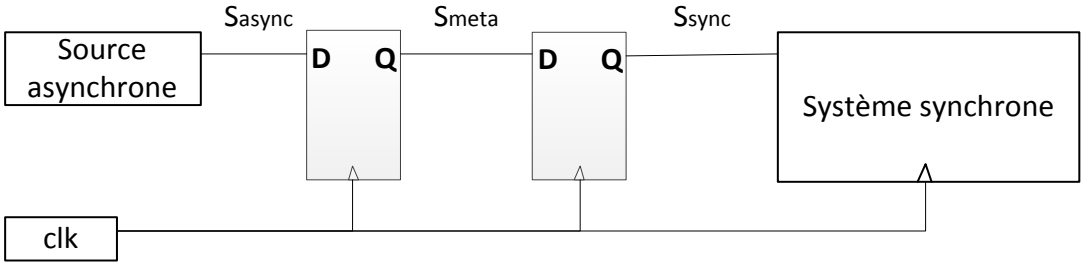
temps minimal pendant lequel le signal d''entrée de la bascule ou du loquet ne doit pas changer après la transition active de l''horloge.

d- Qu'est-ce que la métastabilité? Quand peut-elle se produire?

Si le temps de préparation ou le temps de maintien ne sont pas respectés, alors la bascule ou le loquet risquent d''entrer dans un état métastable, c'est-à-dire que leur sortie aura un niveau imprévisible entre 0 et 1.

e- Proposez une méthode permettant de sortir de la métastabilité (avec schéma à l'appui).

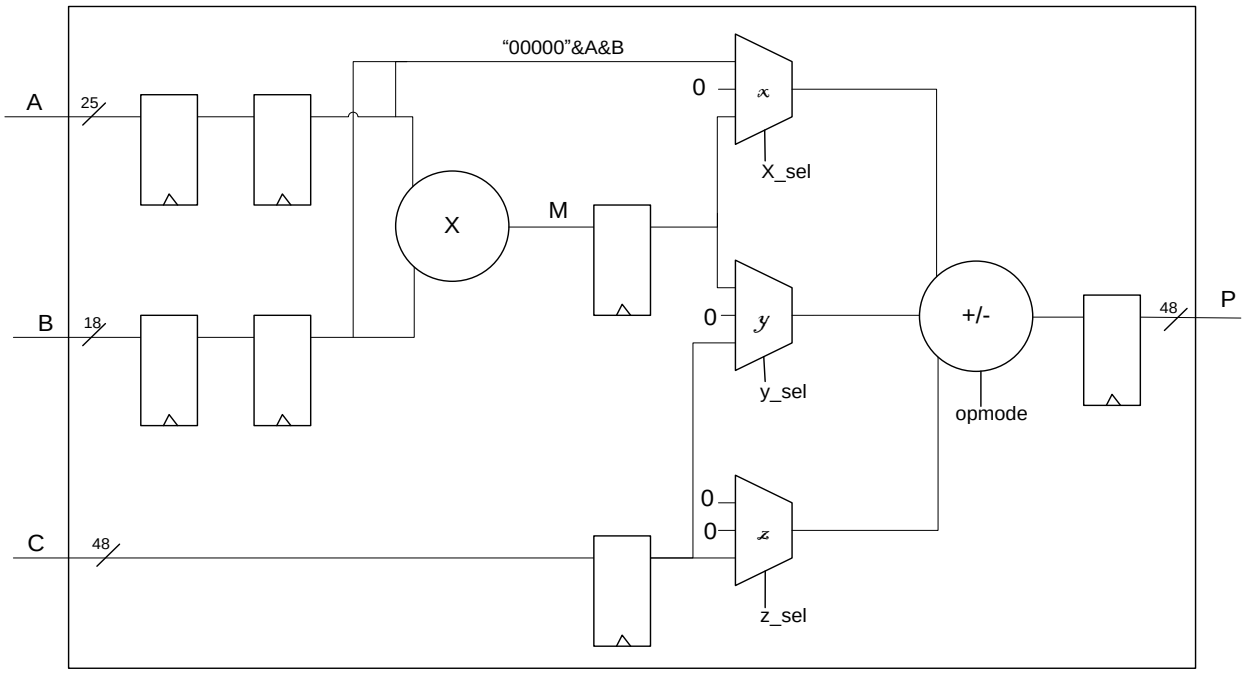
Une solution habituelle consiste à utiliser deux bascules connectées en cascade. Dans ce circuit, la première bascule est reliée à la source asynchrone. Son signal de sortie, S_{meta}, peut être métastable.. Si la première bascule entre dans un état métastable, elle finira en pratique par se stabiliser sur un 0 ou un 1. La deuxième bascule permet d''accorder le plus long temps possible à la première pour se stabiliser, soit une période d''horloge.



Question 2. (12 points)

Dans ses FPGAs, Xilinx met à la disposition des concepteurs des blocs permettant d'effecteur de puissants calculs sans avoir besoin d'utiliser les look-up tables (LUTs) du FPGA. Ces blocs sont des DSP48. Les DSP48 sont des blocs qui supportent plusieurs fonctions : multiplication, multiplication-accumulation, addition…

Une version simplifiée d'un DSP48 est représentée dans la figure ci-dessous :



Les paramètres de configuration sont

A_reg= 0,1 ou 2 : nombre de registres en série sur l'entrée A (deux sont montrés sur le diagramme)

B_reg= 0,1 ou 2 : nombre de registres en série sur l'entrée B (deux sont montrés sur le diagramme)

C_reg= 0,1 : nombre de registres en série sur l'entrée C (un est montré sur le diagramme)

M_reg= 0,1 : nombre de registres en série sur la sortie du multiplicateur (un est montré sur le diagramme)

P_reg= 0,1 : nombre de registres en série sur la sortie P (un est montré sur le diagramme)

x_sel	*x*
0	*A&B*
1	*0*
2	*M*

y_sel	*y*
0	*M*
1	*0*
2	*C*

z_sel	*z*
0	*0*
1	*0*
2	*C*

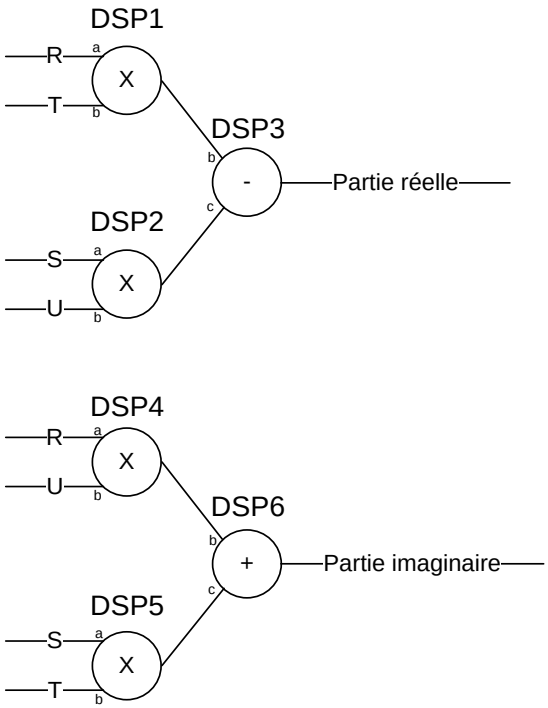
Opmode	*operation*
0	*addition*
1	*soustraction*

On aimerait effectuer la multiplication de deux nombres complexes : (R +*i*S)* (T +*i*U*)*. Sous forme de schéma bloc, proposez trois architectures basées sur les DSP48 :

- Une qui minimise la latence
- Une qui maximise la fréquence maximale
- Une qui minimise le nombre de ressources utilisées

Les trois architectures doivent maximiser le débit. Pour chaque DSP48 utilisé, précisez les paramètres de configuration choisis. Ne faites pas attention à la largeur des données.

Minimisation de la latence



DSP1
A_reg=0
B_reg=0
C_reg=0
M_reg=0
P_reg=0
X_sel=1
Y_sel=0
Z_sel=0
Opmode=0

DSP4
A_reg=0
B_reg=0
C_reg=0
M_reg=0
P_reg =0
X_sel=1
Y_sel=0
Z_sel=0
Opmode=0

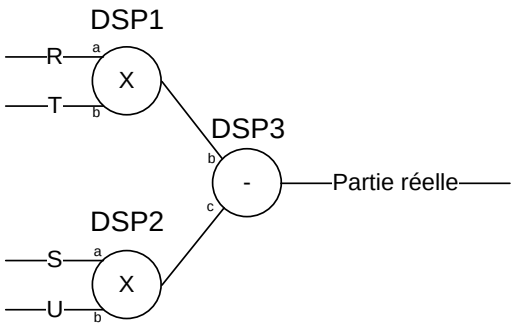
DSP2
A_reg=0
B_reg=0
C_reg=0
M_reg=0
P_reg=0
X_sel=1
Y_sel=0
Z_sel=0
Opmode=0

DSP5
A_reg=0
B_reg=0
C_reg=0
M_reg=0
P_reg =0
X_sel=1
Y_sel=0
Z_sel=0
Opmode=0

DSP3
A_reg=0
B_reg=0
C_reg=0
M_reg=0
P_reg =0
X_sel=0
Y_sel=1
Z_sel=2
Opmode=1

DSP6
A_reg=0
B_reg=0
C_reg=0
M_reg=0
P_reg=0
X_sel=0
Y_sel=1
Z_sel=2
Opmode=0

Maximisation de la fréquence maximale

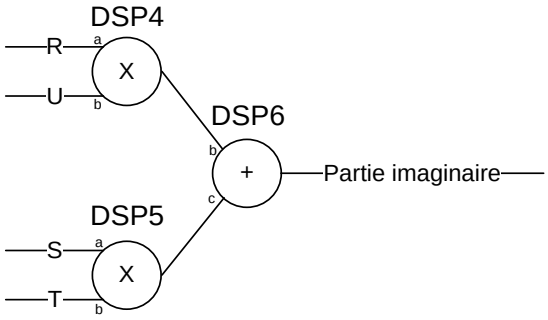


DSP1
A_reg=2
B_reg=2
C_reg=0
M_reg=1
P_reg=1
X_sel=1
Y_sel=0
Z_sel=0
Opmode=0

DSP2
A_reg=2
B_reg=2
C_reg=0
M_reg=1
P_reg=1
X_sel=1
Y_sel=0
Z_sel=0
Opmode=0

DSP4
A_reg=2
B_reg=2
C_reg=0
M_reg=1
P_reg =1
X_sel=1
Y_sel=0
Z_sel=0
Opmode=0

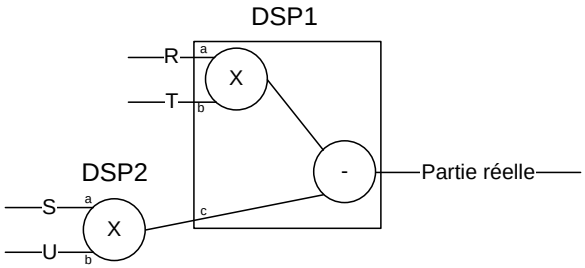
DSP5
A_reg=2
B_reg=2
C_reg=0
M_reg=1
P_reg =1
X_sel=1
Y_sel=0
Z_sel=0
Opmode=0



DSP3
A_reg=0
B_reg=1
C_reg=1
M_reg=0
P_reg =1
X_sel=0
Y_sel=1
Z_sel=2
Opmode=1

DSP6
A_reg=0
B_reg=1
C_reg=1
M_reg=0
P_reg=1
X_sel=0
Y_sel=1
Z_sel=2
Opmode=0

Minimisation de l'utilisation des ressources

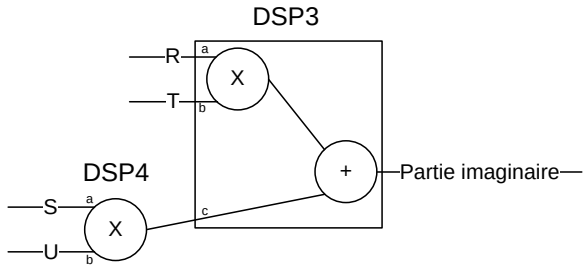


DSP1
A_reg=0
B_reg=0
C_reg=0
M_reg=0
P_reg=0
X_sel=1
Y_sel=0
Z_sel=2
Opmode=1

DSP2
A_reg=0
B_reg=0
C_reg=0
M_reg=0
P_reg=0
X_sel=1
Y_sel=0
Z_sel=0
Opmode=0

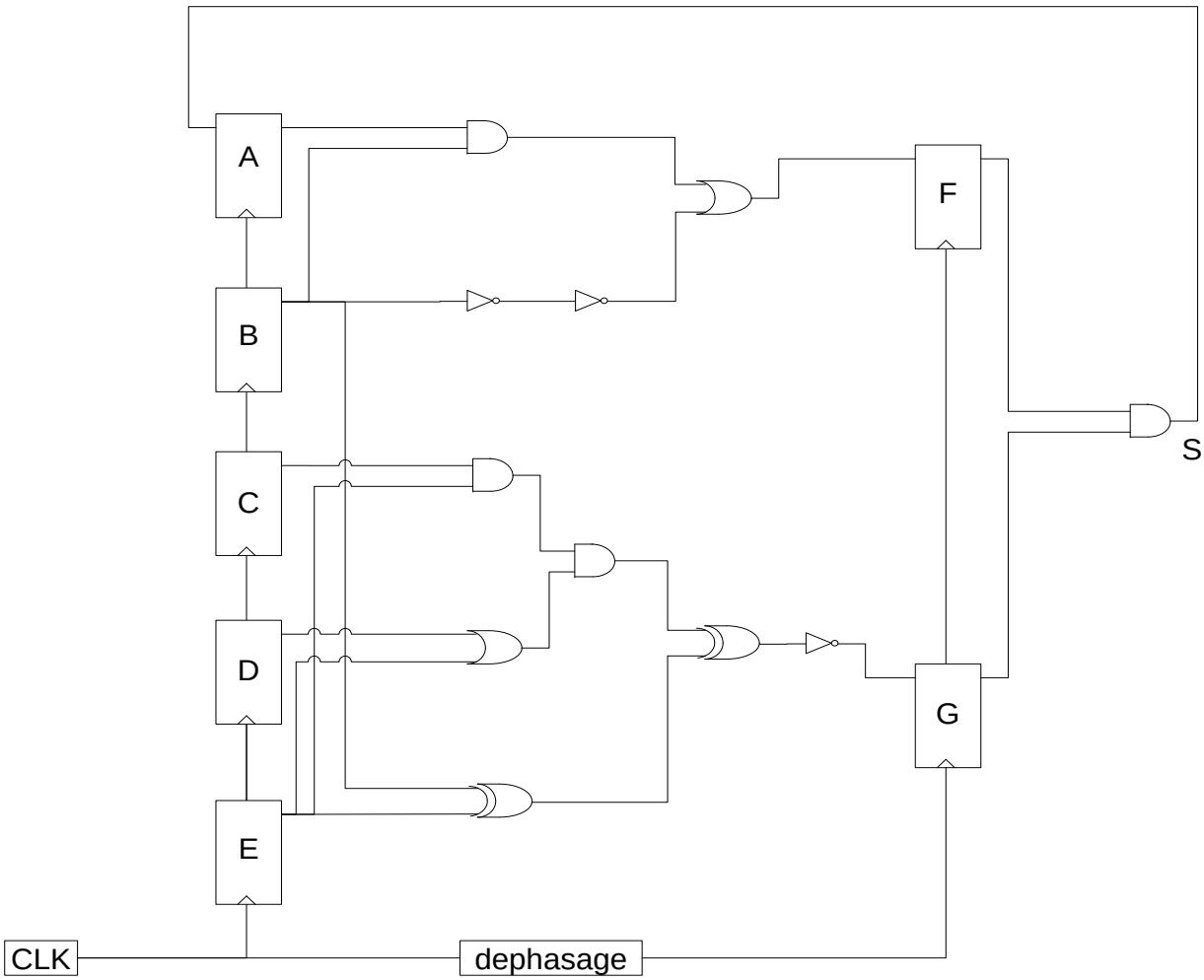
DSP3
A_reg=0
B_reg=0
C_reg=0
M_reg=0
P_reg =0
X_sel=1
Y_sel=0
Z_sel=2
Opmode=0

DSP4
A_reg=0
B_reg=0
C_reg=0
M_reg=0
P_reg =0
X_sel=1
Y_sel=0
Z_sel=0
Opmode=0



Question 3. (8 points)

Soit le circuit suivant :



a- Déterminez sa fréquence maximale en considérant un temps de préparation de 1 ns, un temps de maintien de 0.5 ns, un temps de propagation dans les bascules de 2 ns et un temps de propagation dans les portes ET, OU, XOR et INV de 2 ns, 2 ns, 3 ns, 1 ns respectivement. Le temps de propagation dans les interconnexions est considéré nul.

Pour déterminer la période minimale de chaque chemin: $T = tcomb+tsu+td$

A vers F: T=2+2+2+1 = 7ns
B vers F: T=1+1+2+2+1= 7ns
B vers G: T=3+3+1+2+1= 10 ns
C vers G: T=2+2+3+1+2+1= 11 ns
D vers G: T=2+2+3+1+2+1= 11 ns
E vers G: T=2+2+3+1+2+1=11ns
F vers A: T=2+2+1=5ns
G vers A: T=2+2+1=5ns

Le chemin le plus long fait 11 ns. La fréquence maximale du circuit est donc 1/11ns= 90.9Mhz

b- Déterminez la marge libre de préparation de chaque chemin.

On applique la formule *Tmin-T*, T etant la période minimale de chaque chemin, et *Tmin* la période minimale du circuit

A vers F: 11-7 = 4ns
B vers F: 11-7= 4ns
B vers G: 11-10= 1 ns
C vers G: 11-11= 0 ns

D vers G: 11-11= 0 ns
E vers G: 11-11=0ns
F vers A: 11-5=6ns
G vers A: 11-5=6ns

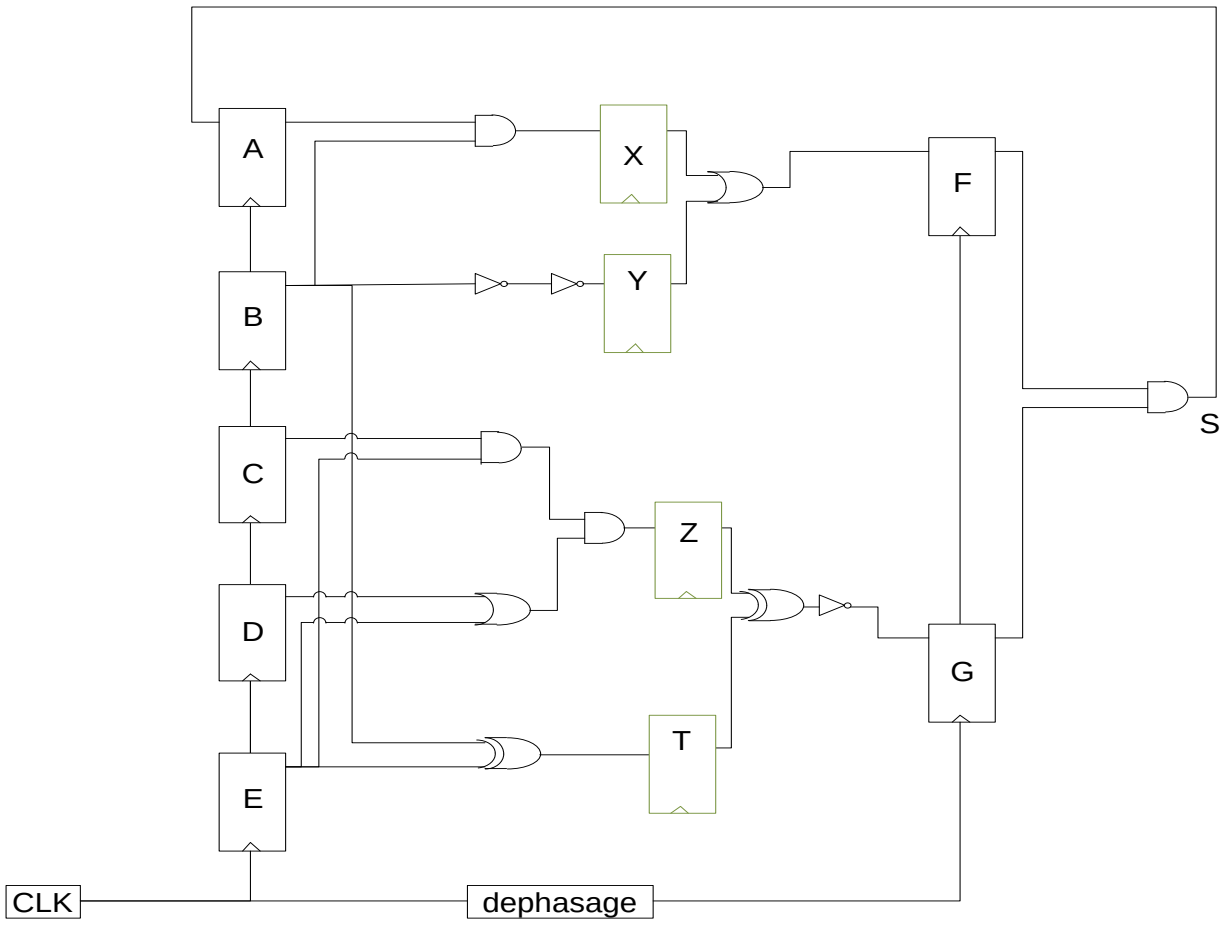
c- Quelles sont les bornes minimales et maximales du déphasage d'horloge acceptables tout en garantissant cette fréquence maximale?

Pour chaque chemin, on va determiner les bornes du déphasage d'horloge
T>= tcomb+tsu+td-tcs d'où la borne tcs >= tcomb+tsu+td-T
tcomb +td-tcs>th d'où la borne tcs < tcomb+td-th

A vers F: -4ns<=tcs<5.5ns
B vers F: -4ns<=tcs<5.5ns
B vers G: -1 ns <= tcs <8.5ns
C vers G: 0ns<=tcs<9.5ns
D vers G: 0ns<=tcs<9.5ns
E vers G: 0ns<=tcs<9.5ns
F vers A: -6ns<=tcs<3.5ns
G vers A: -6ns<=tcs<3.5ns

Pour que le circuit continue de bien fonctionner, les bornes de déphasage acceptables sont:
0ns<=tcs<3.5ns

d- En considérant un déphasage d'horloge nul, ajoutez un étage de pipeline permettant d'augmenter la fréquence maximale. Déterminez la nouvelle fréquence maximale.



Z vers G T= 3+1+2+1=7ns. Fréquence max= 142.8 Mhz

Question 4. (6 points)

Le code VHDL suivant est une portion d'un contrôleur de lecture d'une mémoire. Lorsque le contrôleur reçoit la commande de début de lecture *read_en*, il envoie les adresses de lecture au bloc mémoire.
Deux modes de fonctionnement existent: le mode calibration et le mode hors-calibration.

- Lorsque le contrôleur est en mode hors-calibration (signal *calibration_mode* à 0), 8192 échantillons sont lus de la mémoire. Chaque échantillon est lu à chaque front montant du signal *pulse_read.*
- Lorsque le controleur est en mode calibration (signal *calibration_mode* à 1), 1024 échantillons sont lus de la mémoire. Chaque échantillon est lu au huitième front montant du signal *pulse_read*.

```
library ieee;
  use ieee.std_logic_1164.all;
  use ieee.std_logic_unsigned.all;



entity ctl_fifo_example is
  port (
    clk_sys          : in  std_logic;  --horloge du systeme
    rst_n            : in  std_logic;  -- reset du systeme, actif bas
    read_en          : in  std_logic;  -- indique au controleur qu'il peut demarrer la
                                       -- lecture
    pulse_read       : in  std_logic;
    calibration_mode : in std_logic;   -- mode: calibration hors_calibration?
    rd_addr          : buffer std_logic_vector(12 downto 0)
  );
end ctl_fifo_example ;

architecture ctl_fifo_example_a of ctl_fifo_example is

-------------------------------------------------------
-- types definition
-------------------------------------------------------

type t_state is (IDLE, SEND_DATA);

-------------------------------------------------------
-- signals definition
-------------------------------------------------------
signal state          : t_state;
signal max_rd_addr    : std_logic_vector(12 downto 0);
signal read_en_1q     : std_logic;
signal read_en_2q     : std_logic;
signal read_en_r      : std_logic;
signal read_en_f      : std_logic;
signal calib_mode     : std_logic;
signal count_pulse    : std_logic_vector(3 downto 0);
signal pulse_read_1q  : std_logic;
signal pulse_read_r   : std_logic;



begin

delay_p:process(clk_sys, rst_n)
begin
       if rst_n = '0' then
         read_en_1q <= '0';
         read_en_2q <= '0';
         pulse_read_1q <= '0';
       elsif rising_edge(clk_sys) then
         read_en_1q    <= read_en;
```

```
        read_en_2q    <= read_en_1q;
        pulse_read_1q <= pulse_read;
    end if;
end process;

read_en_r    <= read_en_1q and not read_en_2q; --détection du front montant de read_en
read_en_f    <= read_en_2q and not read_en_1q; --détection du front descendant de
                                               --read_en
pulse_read_r <= pulse_read and not pulse_read_1q; --détection du front montant de
                                                  --pulse_read


state_machine_p:process(clk_sys, rst_n)
begin
    if rst_n = '0' then
        state   <= IDLE;
        rd_addr <= (others => '0');
        max_rd_addr <= (others => '0');
        count_pulse <= (others => '0');
        calib_mode <= '0';
    elsif rising_edge(clk_sys) then
        case state is
            when IDLE =>
               if read_en_r = '1' then
                   if calibration_mode = '1' then
                       rd_addr <= (others => '0');
                       max_rd_addr <= '0'& X"3FF"; --1024 échantillons à lire en mode
                                                   --calibration
                        calib_mode <= '1';
                   else
                       rd_addr <= (others => '0');
                       max_rd_addr <= (others => '1'); --8192 échantillons à lire en
                                                       --mode hors-calibration
                       calib_mode <= '0';
                   end if;
                      state <= SEND_DATA;
                      count_pulse <= "0000";
               end if;
            when SEND_DATA =>
                if pulse_read_r = '1' then
                    count_pulse <= count_pulse + 1;
                    if count_pulse = "0111" then
                        count_pulse <= (others => '0');
                    end if;

                    if calib_mode = '0' then
                        rd_addr <= rd_addr + 1;
                    elsif count_pulse = "0111" then
                        rd_addr <= rd_addr + 1;
                    end if;
                end if;
               if (rd_addr = max_rd_addr and count_pulse = "0111")
                                                 or (read_en_f = '1')   then
                   state <= IDLE;
               end if;
            when others => NULL;
        end case;
    end if;
end process;

end ctl_fifo_example_a;
```

Donnez un banc d'essai pour ce contrôleur. Le banc d'essai devra garantir une couverture de branchement de 100% tout en minimisant le nombre de vecteurs de tests appliqués.

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;


ENTITY ctl_fifo_tb IS
END ctl_fifo_tb;

ARCHITECTURE behavior OF ctl_fifo_tb IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT ctl_fifo_example
    PORT(
         clk_sys : IN  std_logic;
         rst_n : IN  std_logic;
         read_en : IN  std_logic;
         pulse_read : IN  std_logic;
         calibration_mode : IN  std_logic;
         rd_addr : buffer  std_logic_vector(12 downto 0)
        );
    END COMPONENT;


   --Inputs
   signal clk_sys : std_logic := '0';
   signal rst_n : std_logic := '0';
   signal read_en : std_logic := '0';
   signal pulse_read : std_logic := '0';
   signal calibration_mode : std_logic := '0';

      --Outputs
   signal rd_addr : std_logic_vector(12 downto 0);

   -- Clock period definitions
   constant clk_sys_period : time := 10 ns;

BEGIN

      -- Instantiate the Unit Under Test (UUT)
   uut: ctl_fifo_example PORT MAP (
          clk_sys => clk_sys,
          rst_n => rst_n,
          read_en => read_en,
          pulse_read => pulse_read,
          calibration_mode => calibration_mode,
          rd_addr => rd_addr
        );

   -- Clock process definitions
   clk_sys_process :process
   begin
             clk_sys <= '0';
             wait for clk_sys_period/2;
             clk_sys <= '1';
             wait for clk_sys_period/2;
   end process;

   process(clk_sys,rst_n)
   begin
      if rst_n = '0' then
         pulse_read <= '0';
      elsif rising_edge(clk_sys) then
         pulse_read <= not pulse_read;
      end if;
   end process;
```

```
    -- Stimulus process
    stim_proc: process
    begin
       -- hold reset state for 100 ns.
       wait for 100 ns;
             rst_n <= '1';
       wait for clk_sys_period*10;
          read_en <= not read_en; -- premier front montant de read enable
       wait for clk_sys_period*100;
          read_en <= not read_en;  -- premier front descendant de read enable
       wait for clk_sys_period*10;
          calibration_mode <= '1'; -- second front montant de read enable
       wait for clk_sys_period*10;
          read_en <= not read_en; -- second front montant de read enable
       wait for clk_sys_period*100;
          read_en <= not read_en; -- second front descendant de read enable


       wait;
    end process;

END;....
```

….

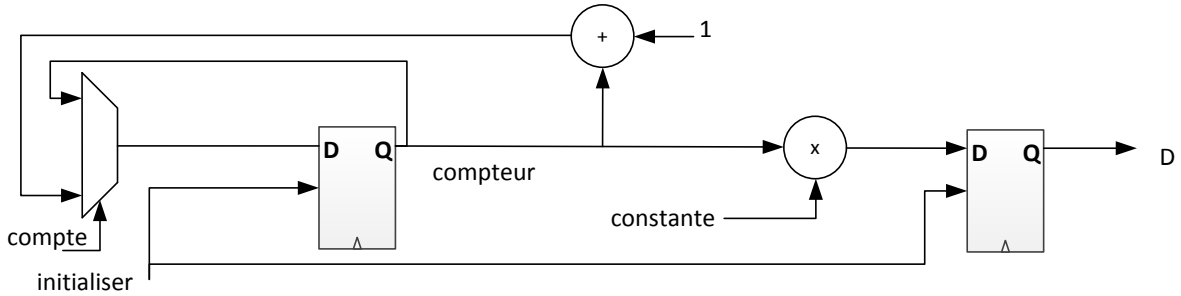
Question 5. (9 points)

Faites la conception d'un processeur pour un télémètre laser. Le télémètre a un bouton pour déclencher la prise de mesure. Quand le bouton est pressé, une impulsion lumineuse est générée et un chronomètre est activé. L'impulsion lumineuse se propage dans l'air, frappe la cible et revient vers un détecteur. Quand l'écho de l'impulsion lumineuse est perçu par le détecteur, le chronomètre est arrêté et la distance peut être calculée en tenant compte de la vitesse de propagation de la lumière dans l'air. Soit T le temps mesuré par le chronomètre, en secondes. Alors la distance D est donnée par $D = T \times c / 2$, où $c = 3 \times 10^8$ m/s.

Le processeur a deux entrées : le bouton et un signal provenant du détecteur indiquant qu'une impulsion lumineuse a été reçue. Il a deux sorties : un signal vers le laser pour déclencher une impulsion lumineuse et un autre signal indiquant la distance mesurée.

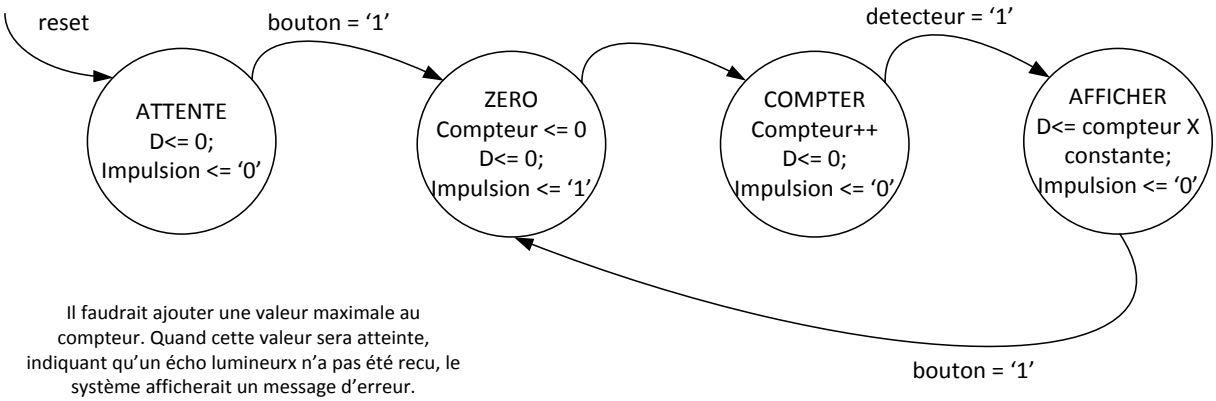
a- Donnez un diagramme représentant le chemin des données du processeur.

Solution



b- Donnez le diagramme d'états de l'unité de contrôle du processeur.

Solution



Il faudrait ajouter une valeur maximale au compteur. Quand cette valeur sera atteinte, indiquant qu'un écho lumineurx n'a pas été recu, le système afficherait un message d'erreur.