Examen langage V H D L

Durée 1H30

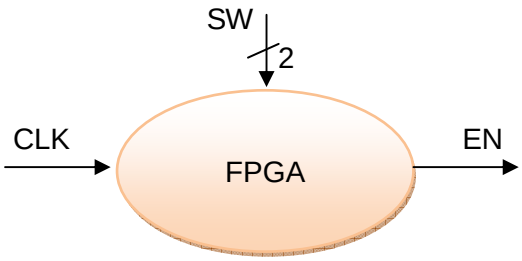
1- Dans un FPGA, il existe des broches particulières pour appliquer une horloge extérieure. En quoi ces broches sont différentes des broches « normales » ?

2- A quoi servent les fichiers .ucf ?

3- Construire un signal enable (EN) à partir d'un signal d'horloge tel que (l'entité plus architecture) :

a. La période T de ce signal est une fonction de SW :
   $T = 2T_0$ si SW= « 00 »,
   $T = 5T_0$ si SW= « 01 » ;
   $T = 9T_0$ si SW= « 10 » ;
   $T = 18T_0$ si SW= « 11 » ;

b. Il est à l'état haut uniquement durant un seul cycle d'horloge.



4- A quoi sert le programme ci-contre :

Tracer un signal d'horloge (CLK) et le signal ENTREE et montrer l'état des signaux tempo et EN.

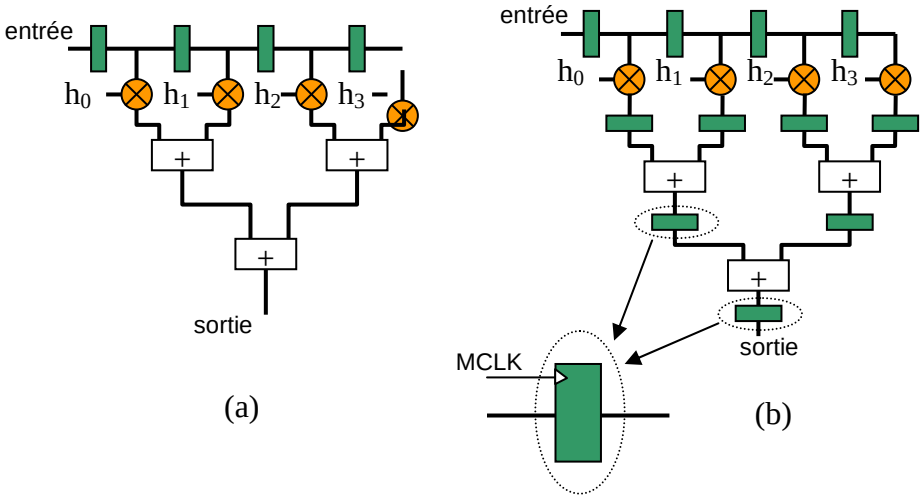
```
process(CLK)
begin
        If CLK'event and CLK='1' then
                EN <= '0';
                tempo <= ENTREE ;
                If tempo = '0' and ENTREE = '1' then
                        EN <= '1'
                end if;
        end if;
end process;
```

5- I) A est un entier sur 6 bits et B est un entier sur 4 bit. Faire un programme en VHDL pour calculer la somme des deux sur 7 bits (pour qu'il n'y ait pas de dépassement). (toutes les valeurs sont signées)
L'entité est donnée ci-dessous :

```
entity somme is
port  (   A : in std_logic_vector(5 downto 0) ;
          B : in std_logic_vector(3 downto 0) ;
          S: out std_logic_vector(6 downto 0));
end somme;
```

II) Si le format est de 1,5 pour A et de 1,3 pour B, quel est le format de S et quelle sera votre solution ?

6- Nous souhaitons réaliser un filtre FIR à 4 coefficients. Deux architectures ont été proposées. La seconde ressemble à la première sauf que nous avons ajouté des registres dans le circuit (architecture à pipeline).
Expliquer le fonctionnement du circuit (a).
Expliquer le fonctionnement du circuit (b).
A votre avis quel est l'avantage du circuit (b) ?



Commandes séquentielles

```
if <condition> then
      <statement>
elsif <condition> then
      <statement>
else
      <statement>
end if;
```

```
   case (<2-bit select>) is
      when "00" =>
         <statement>;
      when "01" =>
         <statement>;
      when "10" =>
         <statement>;
      when "11" =>
         <statement>;
      when others =>
         <statement>;
   end case
```

Déclarations

```
constant <name>: <type> := <value>;
type <type_name> is (<string1>, <string2>, ...);
constant ROM : array 0 to <value> of <type> ;
```

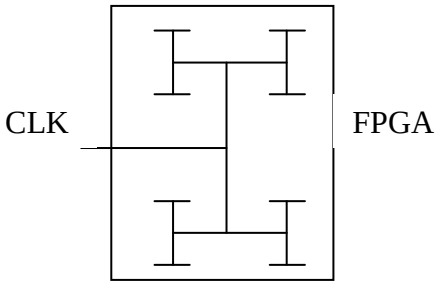
Commandes concurrentes

```
with <choice_expression> select
   <name> <= <expression> when <choices>,
             <expression> when <choices>,
             <expression> when others;

<name> <= <expression> when <condition> else
          <expression> when <condition> else
          <expression>;
```

Corrigé

1- Le FPGA assure que le signal appliqué aux broches dédiées à l'horloge arrive en même temps à toutes les bascules dans tout le FPGA. Alors, si on construit un circuit synchrone, on est sûr que les bascules change d'état toutes en même temps. Ceci est nécessaire pour un bon fonctionnement des machines à états. Alors que cette garantie n'existe pas si d'autres broches de FPGA sont utilisées pour horloge.



2- Toutes les contraints d'optimisation du circuit sont sauvegardées dans le fichier UCF. En TP nous avons utilisé ce fichier pour contrainte le routeur à assigner les broches spécifiques du FPGA à des signaux spécifiques définis dans l'entité. Ce fichier peut aussi prendre les contraintes de timing.

3-

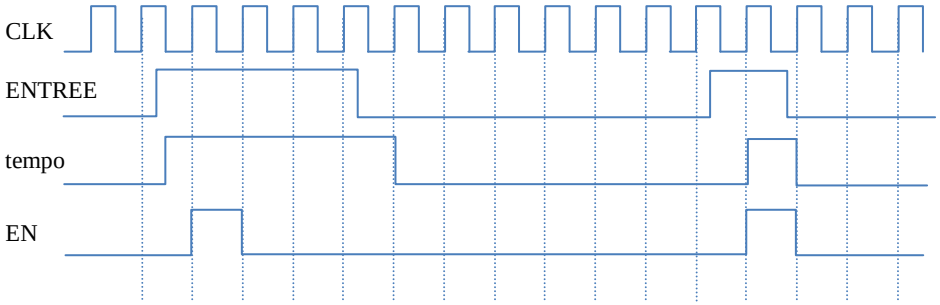
```
entity test is
port(   CLK : in std_logic;
        SW : in std_logic_vector(1 downto 0);
        EN: out std_logic);
end test;

architecture SPARTAN of test is
        signal cmp, cmp_max : integer range 0 to 31;
begin
        process(CLK)
        begin
                if CLK'event and CLK='1' then
                        EN <= '0';
                        cmp <= cmp + 1;
                        if cmp = cmp_max then
                                cmp <= 0;
                                EN <= '1';
                        end if;
                end if;
end process;

with SW select
cmp_max <= 1 when "00",
                4 when "01",
                8 when "10",
                17 when others;

end SPARTAN;
```

4- Ce programme permet de nettoyer et synchroniser un signal qui vient de l'avec l'horloge du FPGA.



5- I)

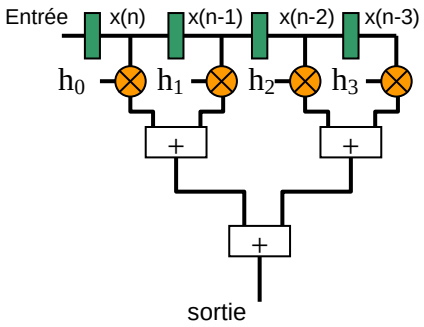
```
entity somme is
port  (   A : in std_logic_vector(5 downto 0) ;
          B : in std_logic_vector(3 downto 0) ;
          S : out std_logic_vector(6 downto 0));
end somme;

architecture FPGA de somme is
begin
          S <= (A(5) & A) + (B(3) & B(3) & B(3) & B);
end architecture;
```

II) le format de sortie sera 2,5.

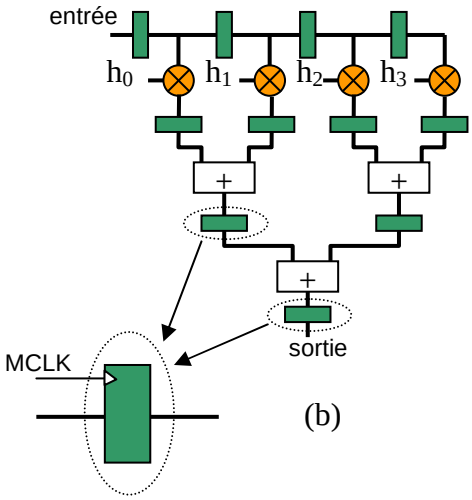
```
architecture FPGA de somme is
begin
          S <= (A(5) & A) + (B(3) & B & "00");
end architecture;
```

6- Le circuit calcule $h_0x(n)+ h_1x(n-1)+ h_2x(n-2)+ h_3x(n-3)$



1- Dans le circuit (a), les signaux x(n), x(n-1), x(n-2) et x(n-3) sont appliqués en même temps aux multiplieurs. T1 secondes plus tard les résultats sont prêts aux entrées des additionneurs. Le résultat final est prêt après T_M+2T_a secondes. C'est un circuit combinatoire. La période d'horloge doit être au moins T_M+2T_a.

Le second circuit, après le premier coup d'horloge, les x(n), x(n-1), x(n-2) et x(n-3) sont présents aux entrée des multiplieurs. Pendant une période d'horloge les multiplieurs ont le temps de calculer leur sortie. Avec le deuxième coup d'horloge, les nouvelles entrées sont appliquées aux multiplieurs alors que les sorties des multiplieurs sont passées aux additionneurs. Maintenant pendant que les multiplieurs traitent le jeu d'entrée suivant, les additionneurs traitent les données précédentes. Et ainsi de suite. C'est à dire que pour que la sortie soit prête, il faut que la période d'horloge soit supérieure à maximum de T_M et T_a. Ainsi la fréquence d'horloge peut augmenter. Par contre la sortie est calculée après plusieurs coups de retard.



(b)