INF3500 : Conception et réalisation de systèmes numériques

Examen intra

1er novembre 2007

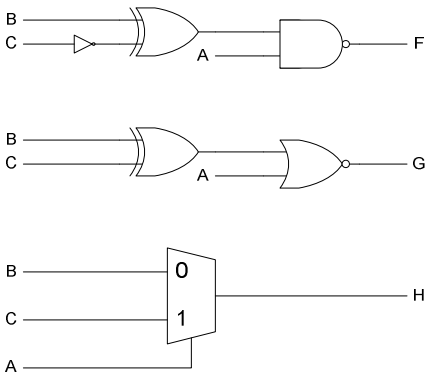
Durée: 2h.
Pondération: 20%.
Documentation: Toute permise.
Calculatrice: Programmable permise.
Directives particulières: Ordinateurs interdits, répondre à toutes les questions, la valeur de chaque question est indiquée.
Ne posez pas de questions. En cas de doute sur le sens d'une question, énoncez clairement toute supposition que vous faites.

Question 1. (2 points)

Considérez l'extrait de code VHDL suivant. Donnez son équivalent en matériel à l'aide d'un schéma de portes logiques.

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

entity monModule is
    port (
        A : in std_logic;
        B : in std_logic;
        C : in std_logic;
        F : out std_logic;
        G : out std_logic;
        H : out std_logic
    );
end monModule;

architecture arch of monModule is
begin
    F <= not(A and (B xor not(C)));
    G <= '0' when (A = '1' or B /= C) else '1';
    with A select H <= B when '0', C when others;
end arch;
```

Solution :

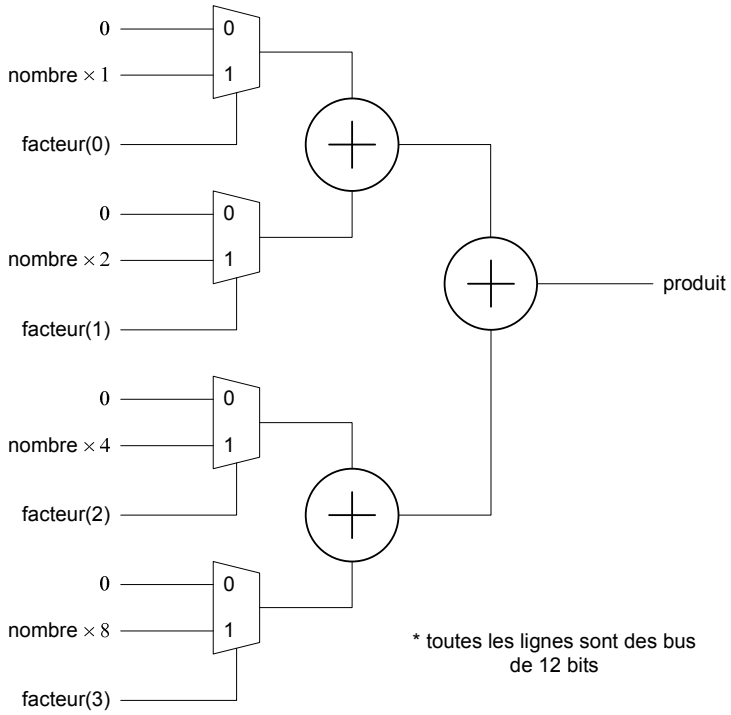


Question 2. (6 points)

Faites la conception d'un circuit numérique combinatoire qui accepte en entrée un nombre non-signé exprimé avec 8 bits ainsi qu'un facteur non-signé exprimé avec 4 bits. La sortie doit être un produit de 12 bits du nombre et de son facteur. Utilisez uniquement les opérations d'addition, soustraction et décalage. Votre circuit doit être purement combinatoire, il ne doit pas inclure d'éléments à mémoire.

a. Donnez un diagramme de votre circuit (3 points)

Solution :



b. Donnez sa description en VHDL (3 points)

Solution :

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

entity controleMultiplicateur is
    port (
        nombre : in unsigned(7 downto 0);
        facteur : in unsigned(3 downto 0);
        produit : out unsigned(11 downto 0)
    );
end controleMultiplicateur;

architecture arch of controleMultiplicateur is
signal add0, add1, add2, add3 : unsigned(11 downto 0);
begin
    add0 <= "0000" & nombre when facteur(0) = '1' else (others => '0');
    add1 <= "000" & nombre & "0" when facteur(1) = '1' else (others => '0');
    add2 <= "00" & nombre & "00" when facteur(2) = '1' else (others => '0');
    add3 <= "0" & nombre & "000" when facteur(3) = '1' else (others => '0');
    produit <= (add3 + add2) + (add1 + add0);
end arch;
```

Question 3. (4 points)

Considérez le code VHDL suivant pour un module combinatoire et son banc de test associé.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity module1 is
    port (
        A, B : in std_logic;
        F : out std_logic
    );
end module1;
architecture arch of module1 is
signal S1, S2: std_logic;
begin
    S1 <= A and B;
    S2 <= not(B);
    process (S1, S2)
    begin
        F <= S1 xor S2;
    end process;
end arch;
```

```
library IEEE;
use IEEE.std_logic_1164.all;
entity module1TB is
end module1TB;
architecture arch of module1TB is
component module1
port (
    A, B : in std_logic;
    F : out std_logic
);
end component;
signal A,B,F : std_logic;
begin
    UUT : module1 port map (A, B, F);
    A <= '0' after 0 ns;
    B <= '1' after 0 ns, '0' after 10 ns;
end arch;
```

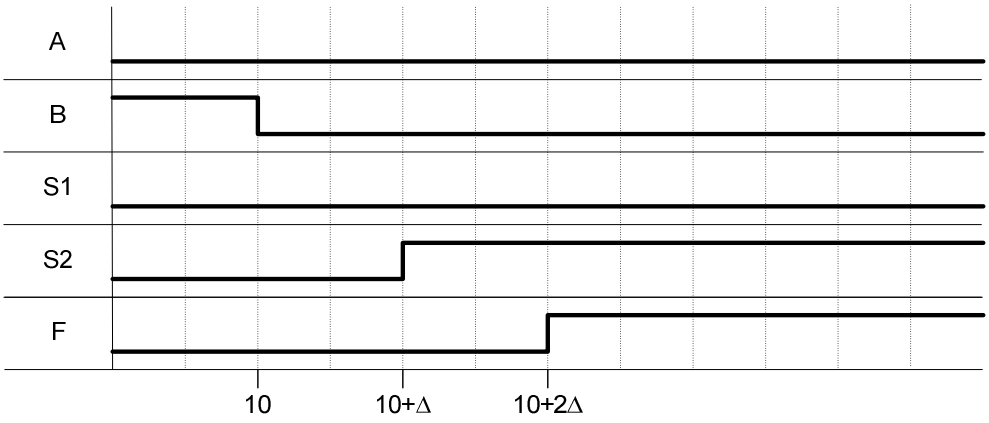
a. Donnez la liste des événements à partir du temps de simulation 10 ns, telle qu'elle pourrait être dressée par un simulateur qui exécuterait le banc d'essai. (2 points)

Solution :

énoncé	temps
B <= '0'	10 ns
S1 <= A and B	10 ns + Δ
S2 <= not(B)	10 ns + Δ
F <= S1 xor S2	10 ns + 2Δ

b. En vous basant sur votre liste des événements, donnez un chronogramme à partir du temps de simulation 10 ns, montrant la valeur des signaux internes et de sortie en tenant compte des délais deltas. (2 points)

Solution :

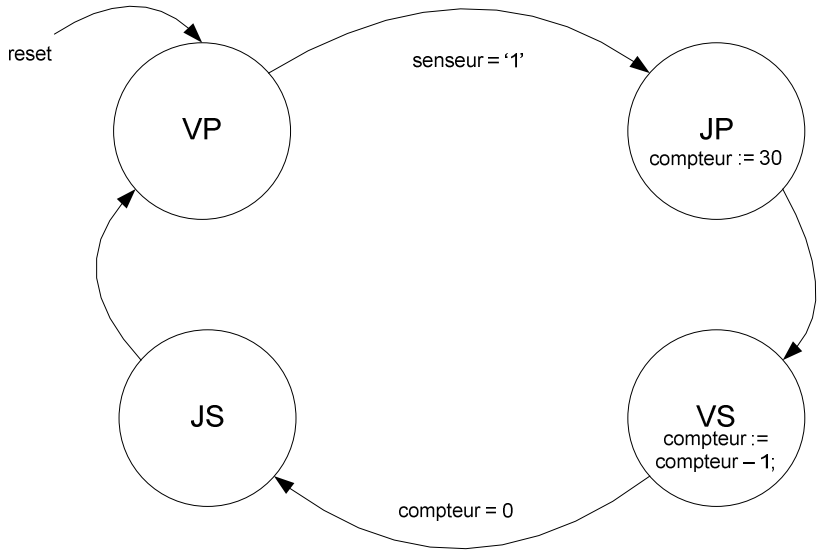


Question 4. (6 points)

Considérez le problème des feux de circulation à l'intersection d'une route secondaire et d'une route principale. Le feu doit débuter vert sur la route principale, sauf quand un senseur sous la chaussée de la route secondaire détecte la présence d'une voiture. Le feu de la route principale doit alors passer au jaune pour une seconde, puis au rouge pour 30 secondes. Le feu de la route secondaire doit être vert pendant ces trente secondes, puis passer au jaune pendant une seconde, puis passer au rouge. Le feu doit alors être vert sur la route principale jusqu'à ce que le senseur soit activé de nouveau.

a. Donner un diagramme d'états pour ce système. Ne pas indiquer la valeur des sorties (les signaux de contrôle des feux) sur le diagramme. (2 points)

Solution :



b. Considérez la déclaration d'entité suivante en VHDL. Donnez une architecture pour cette entité qui respecte les spécifications du problème. Supposez que le signal `clk` est une horloge de 1 Hz, et que le signal `reset` est utilisé lors de la maintenance du système pour le réinitialiser. (4 points)

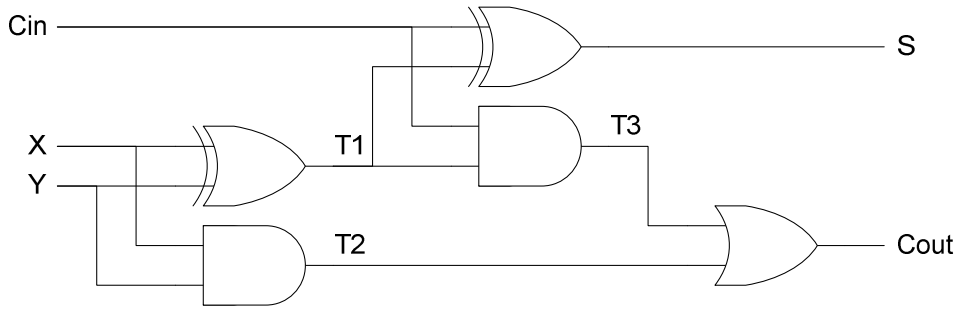
```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity feuxcirculation is
   port(
       clk, reset, senseur : in std_logic;
        -- feux rouge, jaune et vert, pour chacune des deux routes:
       feuxPrincipale : out std_logic_vector(2 downto 0);
       feuxSecondaire : out std_logic_vector(2 downto 0)
   );
end feuxcirculation;
```

Solution :

```
architecture arch of feuxcirculation is

type type_etat is (VP, JP, VS, JS);
signal etat : type_etat := VP;

-- durée du feu vert sur la route secondaire, en périodes d'horloge
constant dureeVertSecondaire : integer := 30;

begin

    process(clk, reset) is
    variable compteur : integer range 0 to dureeVertSecondaire;
    begin
        if (rising_edge(clk)) then
            if (reset = '0') then
                etat <= VP;
            else
                case etat is
                    when VP =>
                        if (senseur = '1') then
                                etat <= JP;
                        end if;
                    when JP =>
                        etat <= VS;
                        compteur := dureeVertSecondaire;
                    when VS =>
                        compteur := compteur - 1;
                        if (compteur = 0) then
                                etat <= JS;
                        end if;
                    when JS =>
                        etat <= VP;
                end case;
            end if;
        end if;
    end process;

    process(etat)
    begin
        case etat is
            when VP =>
                feuxPrincipale <= "001";
                feuxSecondaire <= "100";
            when JP =>
                feuxPrincipale <= "010";
                feuxSecondaire <= "100";
            when VS =>
                feuxPrincipale <= "100";
                feuxSecondaire <= "001";
            when JS =>
                feuxPrincipale <= "100";
                feuxSecondaire <= "010";
        end case;
    end process;

end arch;
```

Question 5. (2 points)

Considérez le circuit d'un additionneur à trois bits montré ici comme module de base.



En utilisant une seule instance de ce module, donnez un circuit séquentiel qui permet d'additionner des paires de nombres de *n* bits présentés de façon sérielle.

Solution :