INF3500 : Conception et réalisation de systèmes numériques

Examen intra

24 février 2009

Durée: 2h.
Pondération: 20%.
Documentation: Une feuille recto-verso 8.5"×11" ou A4 manuscrite 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.

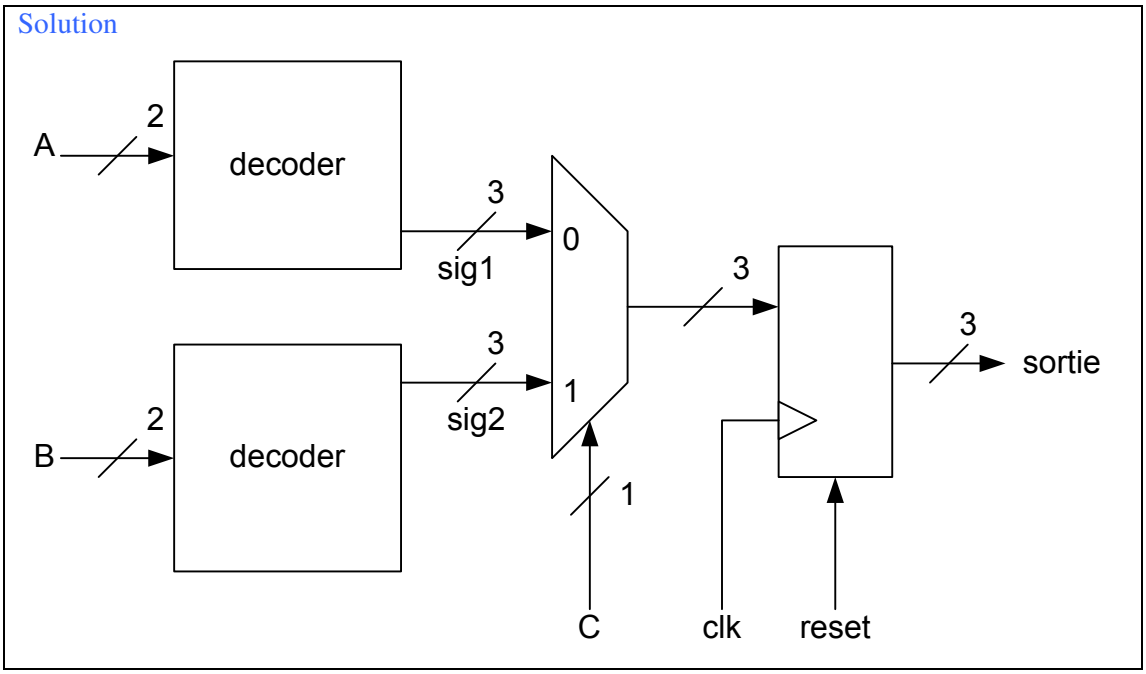
Question 1. (4 points)-exercice sur la synthèse

Donnez le diagramme du circuit correspondant au code VHDL suivant. Indiquez la largeur en bits des ports et des signaux en sachant que N est égal à 2. Vous pouvez utiliser des opérations logiques et arithmétiques, la comparaison, des multiplexeurs, décodeurs et encodeurs, éléments mémoire, etc.

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

entity entite1 is
   generic (N : positive)
   port(
       reset : in std_logic;
      clk : in std_logic;
       A : in std_logic_vector(n - 1 downto 0);
       B : in std_logic_vector(n - 1 downto 0);
       C: in std_logic;
       sortie : out std_logic(2 downto 0)
   );

end entite1;

architecture comportementale of entite1 is
       signal sig1: std_logic_vector(2 downto 0);
       signal sig2: std_logic_vector(2 downto 0);
       signal sig3: std_logic_vector(2 downto 0);

begin
   process(A)
   begin
       sig1 <= (others => '0');
       sig1(to_integer(unsigned(A))) <= '1';
       sig2 <= (others => '0');
       sig2(to_integer(unsigned(B))) <= '1';
   end process;

   process(clk, reset)
      begin
    if(reset='1') then
       sortie <= (others => '0');
    elsif (clk'event and clk='1') then
```

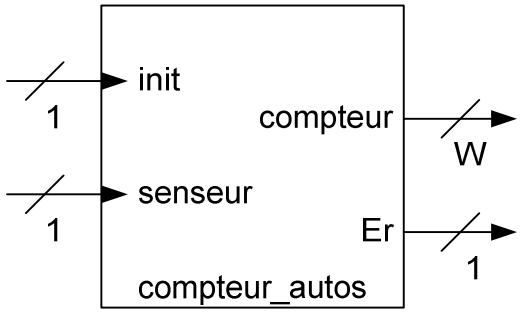
```
        if (C = '0') then
           sortie <= sig1;
        else
           sortie <= sig2;
        end if;
    end if;
       end process;
end comportementale;
```

Solution



Question 2. (4 points)

Soit un circuit de compteur de voitures *compteur_autos* destiné à la gestion de la circulation routière d'une ville, dont le symbole est :



Les ports de l'entité compteur_autos sont décrits comme suit :

- Quand le port *init* est égal à '1' le port *compteur* est remis à 0 et le port *Er* aussi est mis à '0'.
- À chaque fois que la valeur du port *senseur* passe de '0' à '1', *compteur* doit être incrémenté;

- Le port *compteur* correspond au nombre de voitures qui sont passées depuis la dernière initialisation;
- Le port *Er* est égal à '1' quand il y a eu un débordement sur *compteur*. En effet, il y aura un débordement quand le port *compteur* atteint la valeur maximale qu'il peut contenir et une nouvelle incrémentation est effectuée. Dans ce cas, l'incrémentation fera que la nouvelle valeur de *compteur* sera égale à 0. Pour éviter qu'un employé lise une mauvaise valeur, le port *Er* lui permettra de savoir qu'il doit ignorer la valeur de compteur Y. Le port *Er* restera égal à '1' jusqu'à la prochaine initialisation (int='1').

Les ports de 1 bit sont des std_logic alors que le port ***compteur*** **est un vecteur de W bits non-signés** (unsigned).

Donnez l'entité et l'architecture synthétisable du compteur d'autos *compteur_autos*.

Solution

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

entity compteur_autos is
  generic (
    W : positive);

  port (
    init     : in  std_logic;
    senseur  : in  std_logic;
    compteur : out unsigned(W-1 downto 0);
    Er  : out std_logic);
end compteur_autos;

architecture comportementale of compteur_autos is
  signal sig_compteur : integer range 2**W-1 downto 0;
  signal sig_Er : std_logic;
begin  -- comportementale

  compteur <= to_unsigned(sig_compteur, W);
  Er <= sig_Er;

  process (init, senseur)
  begin  -- process
    if (init='1') then
      sig_compteur <= 0;
      sig_Er <= '0';
    elsif (senseur'event and senseur = '1') then
      sig_Er <= '0';

      if (sig_Er = '1' or sig_compteur = 2**W-1) then
        sig_Er <= '1';
      end if;

      sig_compteur <= sig_compteur + 1;
    end if;
  end process;
end comportementale;
```

Question 3. (2 points)

Soit le module *majorite* dont l'entité et l'architecture sont décrits comme suit :

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity majorite is
  port (
    X: in std_logic_vector(7 downto 0);
    F : out std_logic
    );
end majorite;

architecture comportementale of majorite is
begin
  process (I)
    variable compte : integer := 0;
  begin
    compte := 0;
    for k in W - 1 downto 0 loop
      if I(k) = '1' then
        compte := compte + 1;
      end if;
    end loop;
    if (compte > W / 2) then
      F <= '1';
    else
      F <= '0';
    end if;
  end process;
end comportementale;
```

Écrivez en VHDL un banc d'essai qui teste le module majorité. Le banc d'essai doit générer aléatoirement 10 vecteurs de test pour l'entrée X de l'unité à tester, en sachant que la valeur générée de X doit être comprise entre 55 et 128. Les vecteurs seront générés à chaque 10 ns.

Rappel :

La procédure **`uniform(seed1, seed2, resultat)`** génère aléatoirement un nombre réel compris entre 0.0 et 1.0. Cette procédure a trois paramètres: deux valeurs « graines » **`seed1`** et **`seed2`** pour guider la génération de nombres pseudo-aléatoires, et le nombre pseudo-aléatoire retourné dans le paramètre **`resultat`**. La procédure **`uniform`** est définie dans le package `math_real`.

Solution

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

entity  majorite_tb is
end majorite_tb;

architecture arch of majorite_tb is
  component majorite
    port (
      X : in unsigned(7 downto 0);
      F :  out std_logic)  ;
  end component;


  signal X : unsigned(7 downto 0);
  signal F : std_logic;

begin
  -- instancier l'unitie a teste
  UUT : majorite
    port map (
      X =>X, F => F);

  process
    variable seed1 : positive := 1;
    variable seed2 : positive := 2;
    variable aleatoire : real;
    variable t : integer := -1;

  begin  -- process
    for i in 1 to 10 loop
      uniform(seed1, seed2, aleatoire); -- 0.0 < aleatoire < 1.0
      aleatoire := 55.0 + floor(aleatoire * 73.0); -- 55.0 <= aleatoire <= 128.0
      t := integer(aleatoire); -- 55 <= t <= 128
      X <= to_unsigned(t, 8);

      wait for 10 ns;
    end loop;  -- i
  end process;

end arch;
```

Question 4. (4 points)

Soit l'entité module1 ainsi que le banc d'essai module1_tb permettant de le vérifier. Les codes VHDL de ces modules sont comme suit:

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

entity module1 is
  port (
    A, B : in std_logic;
    F : out std_logic
    );
end module1;
architecture arch of module1 is
```

```
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;
```

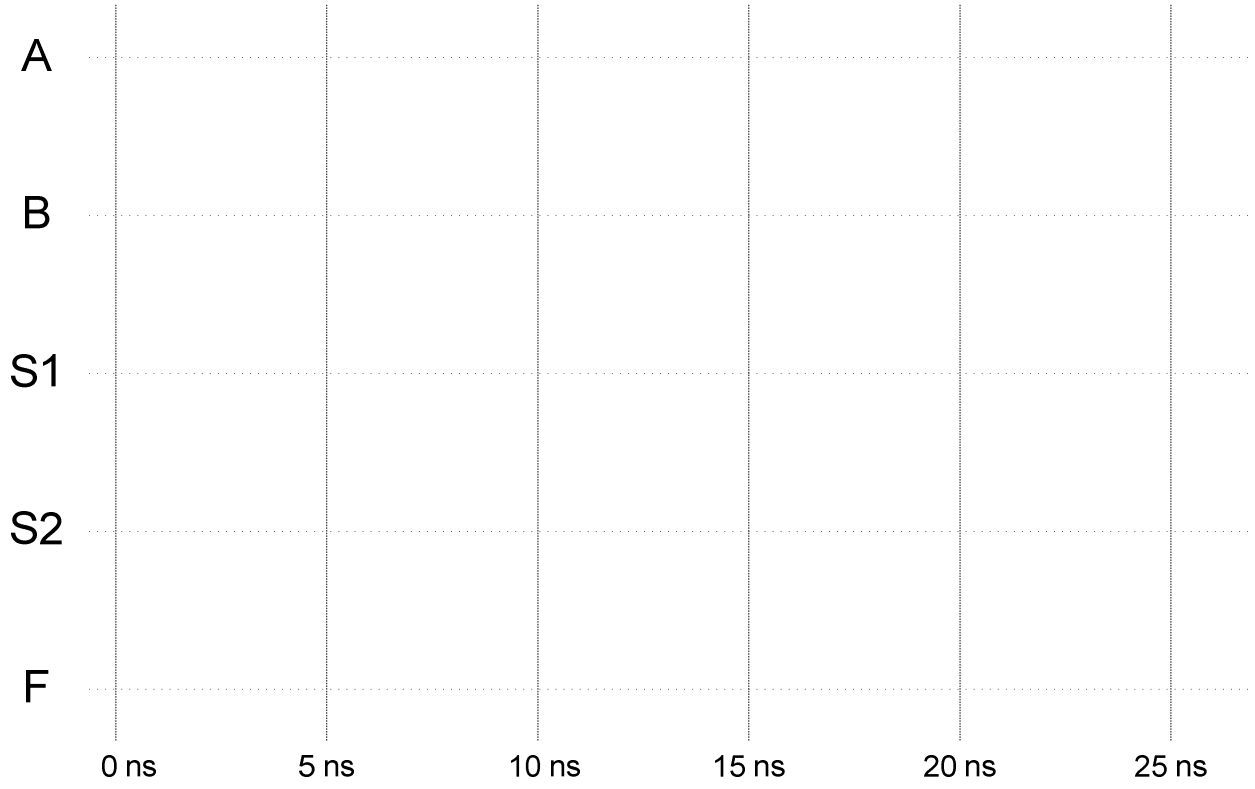
```
  signal S1, S2: std_logic;
begin
  S1 <= A and B;
  S2 <= not(B);
  process (S1, S2)
  begin
    F <= S1 xor S2;
  end process;
end arch;
```

```
  signal A,B,F : std_logic;
begin
  UUT : module1 port map (A, B, F);
  A <= '0' after 0 ns, '1' after 20 ns;
  B <= '1' after 0 ns, '0' after 10 ns,
'1' after 15 ns;
end arch;
```

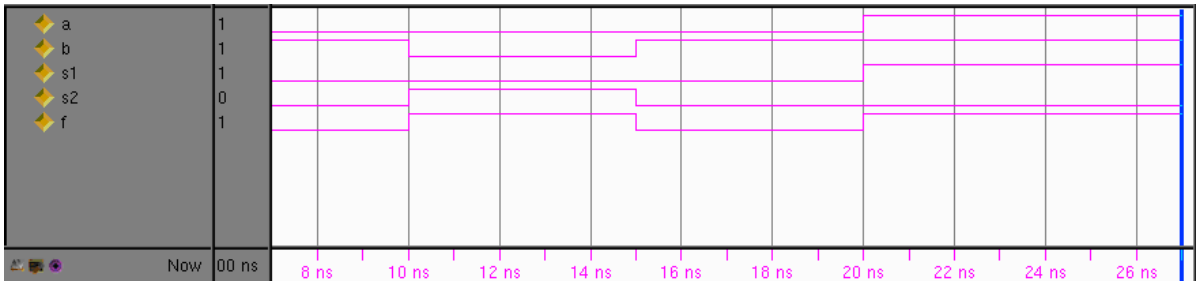
Donnez le chronogramme des ports A, B, F et des signaux internes S1 et S2 durant une simulation comprise entre les temps 0 ns jusqu'à 25 ns. Remplissez le gabarit de la page suivante. N'oubliez pas de remettre la page dans votre cahier d'examen.

Nom :

Matricule :



Solution



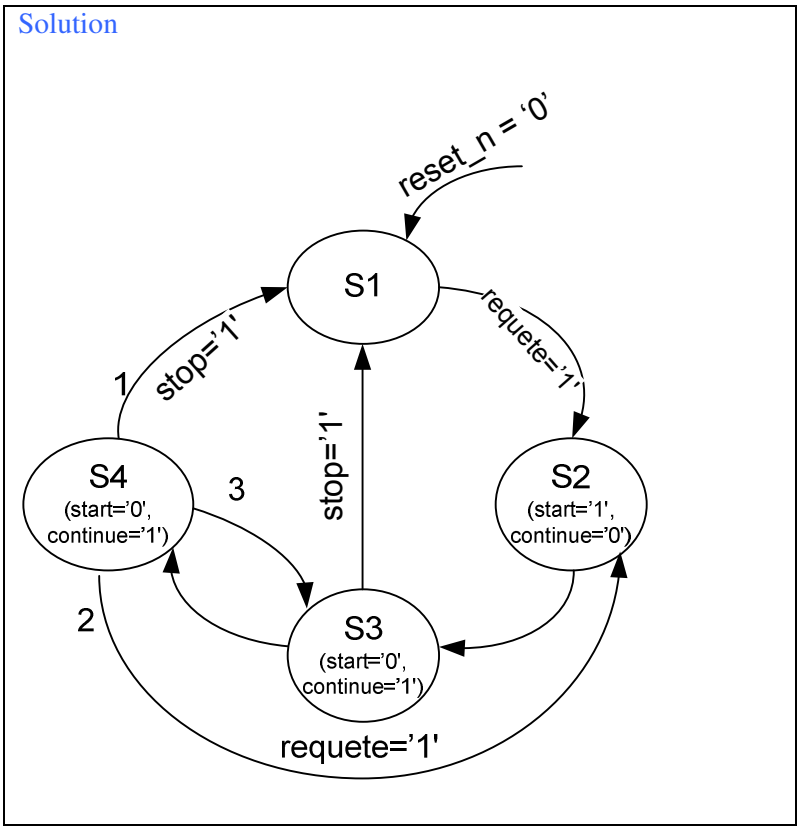
Question 5. (4 points)

Donnez le diagramme d'états correspondant au code VHDL suivant. Identifiez clairement les états, les transitions entre états et les conditions correspondantes, et la valeur des signaux de sortie.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity controleur is

  port (
    reset_n : in  std_logic;
    clk : in  std_logic;
    requete  : in  std_logic;
    stop     : in std_logic;
    start : out std_logic;
    continue : out std_logic);
end controleur;

architecture beh of controleur is
  type type_etat is (S1, S2, S3, S4);
  signal etat_courant : type_etat := S1;
  signal etat_prochain : type_etat := S1;
begin  -- beh
  -- mise a jour de l'etat courant
  process (clk, reset_n)
  begin  -- process
    if reset_n = '0' then               -- asynchronous reset (active low)
      etat_courant <= S1;
    elsif clk'event and clk = '1' then  -- rising clock edge
      etat_courant <= etat_prochain;
    end if;
  end process;

  -- mise aj our de l'etat prochain
  process (etat_courant, requete, stop)
  begin  -- process
    etat_prochain <= etat_courant;

    case etat_courant is
      when S1 =>
        if requete='1' then
          etat_prochain <= S2;
        end if;
```

```
      when S2 =>
        etat_prochain <= S3;
      when S3 =>
        if stop='1' then
          etat_prochain<= S1;
        else
          etat_prochain <= S4;
        end if;
      when S4 =>
        if stop='1' then
          etat_prochain<= S1;
        elsif requete='1' then
          etat_prochain <= S2;
        else
          etat_prochain <= S3;
        end if;
      when others => null;
    end case;
  end process;

  -- mise a jour des sorties
  process (etat_courant)
  begin  -- process
    start <= '0';
    continue <= '0';

    case etat_courant is
      when  S2=>
        start <= '1';
        continue <= '0';
      when S3 | S4 =>
        start <= '0';
        continue <= '1';
      when others => null;
    end case;
  end process;


end beh;
```

Solution



Question 6. (2 points)

Dans le code VHDL suivant, indiquez les quatre erreurs détectées au moment de la compilation. Pour chaque ligne comportant une erreur, donnez le code corrigé correspondant.

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

entity unitearithmetique is
  generic (
    W : positive := 8 -- largeur des opérandes
    );
  port(
    A, B : in integer(W - 1 downto 0); -- les operandes de type signé
    choix : in std_logic_vector(2 downto 0); -- le sélecteur d'opération
    F : out std_logic_vector(W - 1 downto 0) -- le résultat
    );
end unitearithmetique;

architecture arch of unitearithmetique is
begin
  process(A, B, choix)
    variable t : integer range 0 to 2**W-1;
  begin
    t <= to_integer(A * B);
    case to_integer(unsigned(choix)) is
      when 4 => F <= std_logic_vector(abs(A));
      when 6 => F <= t;
      when others => F <= 'X';
```

```
    end case;
  end process;
end arch;
```

Solution

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity unitearithmetique is
  generic (
    W : positive := 8 -- largeur des opérandes
    );
  port(
    A, B : in signed(W - 1 downto 0); -- les opérandes
    choix : in std_logic_vector(2 downto 0); -- le sélecteur d'opération
    F : out std_logic_vector(W - 1 downto 0) -- le résultat
    );
end unitearithmetique;
architecture arch of unitearithmetique is
begin
  process(A, B, choix)
    variable t : integer range 0 to 2**W-1;
  begin
    t := to_integer(A * B);
    case to_integer(unsigned(choix)) is
      when 4 => F <= std_logic_vector(abs(A));
      when 6 => F <=std_logic_vector(to_unsigned(t,W)) ;
      when others => F <= (others => 'X');
    end case;
  end process;
end arch;
```