INF3500 : Conception et réalisation de systèmes numériques

Examen final

19 août 2009

Durée: 2h30.
Pondération: 40%.
Documentation: Une feuille recto verso 8.5”×11” ou A4 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.

Les pages réponses à remplir et à remettre sont à la fin de l’énoncé. N’oubliez pas de préciser votre nom et numéro de matricule sur chacune des pages réponse.

Question 1. (12 points) – Processeur à but spécifique

Nous désirons concevoir un processeur à but spécifique qui gère une machine à café. L'entité du gestionnaire de la machine à café est comme suit :

```
entity gest_machine_a_cafe is

  generic (
    prix_cafe1 : positive;
    prix_cafe2 : positive
  );

  port (
    reset             : in std_logic;
    clk               : in std_logic;
    piece_introduite  : in unsigned(6 downto 0);
    piece_introduite_valide      : in std_logic;
    choix_cafe : in std_logic;
    fermer_entree_piece : out std_logic;
    annuler           : in std_logic;
    servir_cafe : out std_logic;
    cafe_servi : in std_logic;
    activer_cafe1 : out std_logic;
    activer_cafe2 : out std_logic;
    piece_a_rendre : out unsigned(6 downto 0);
    piece_a_rendre_valide : out std_logic
  );
end gest_machine_a_cafe;
```

Le client a le choix entre café1 (*choix_cafe* = '0') ou café2 (*choix_cafe* = '1'). Les prix des cafés sont des paramétres génériques : prix_cafe1 et prix_cafe2.

La partie mécanique de la machine à café comprend la fente dans laquelle le client introduit des pièces, la sortie par laquelle les pièces de la monnaie sont rendues et la structure qui sert les cafés. Ces parties mécaniques envoient et/ou recoivent des requêtes au gestionnaire. Ce dernier attend que le client fasse son choix et qu'il glisse des pièces dans la fente jusqu'à ce que le total des pièces introduites soit supérieur ou égal au montant du café à servir. Lorsque ce montant est atteint alors un ordre sera émis à la partie mécanique pourque le café soit servi (*servir_cafe* = '1'. Ensuite, le gestionnaire calculera le montant de la monnaie à rendre.

La machine ne peut recevoir ou retourner que des pièces de 1$, 25 cents, 10 cents ou 5 cents.

Le montant de la piece introduite est en cents donc lorsqu'une pièce de 1$ est introduite l'entrée piece_introduite sera égale à 100.

Pour éviter de considérer la même pièce plusieurs fois, le gestionnaire devra attendre que le signal *pièce_introduite_valide* transite de '0' à '1' avant de considérer la nouvelle pièce.

Après avoir reçu l'avis que le café est servi (*cafe_servi* = '1'), le gestionnaire procédera à la remise de la monnaie. Par rapport au montant introduit, le gestionnaire devra émettre des requêtes de remise de pièces. Donc, par exemple, si le gestionnaire doit remettre 35 cents, il devra émettre l'ordre de remettre une pièce de 25 cents puis d'une de 10 cents. Les ordres de remise de pièce doit être séparée à savoir que le port *piece_a_rendre_valide* doit transiter de '1' à '0', pour que le second ordre soit considéré.

Pendant que les pièces sont entrées, le client peut décider d'annuler sa commande. Dans ce cas, le port *annuler* sera égal à '1' et le gestionnaire devra émettre les ordres de remise de pièces afin de rendre au client le montant de pièces qu'il avait introduit avant l'annulation.

Pour éviter que le client introduise des pièces pendant que le café est servi ou que la monnaie est remise, la fente des pièces entrées doit est fermée (*fermer_entree_piece* = '1').

Les interfaces du gestionnaire de la machine à café se décrivent comme suit :

Port	Description
reset	Il réintialise les ports et bascules internes du gestionnaire.
clk	L'horloge du processeur.
piece_introduite	Le montant de la pièce introduite en cents.
piece_introduite_valide	La pièce introduite doit être considérée que lorsque ce port est égal à '1'. Ce port doit transiter de '0' à '1' entre l'introduction de deux pièces.
choix_cafe	Lorsque ce port a la valeur '0' alors le prix à charger devra correspondre à prix_cafe1 sinon c'est prix_cafe2 qui sera utilisé.
fermer_entree_piece	Le gestionnaire doit mettre ce port à '1' pour éviter que le client continue à introduire des pièces pendant que le café est servi ou que la monnaie est remise. En attendant que le client introduise une nouvelle pièce et que le montant d'un café soit introduit, le gestionnaire mettra ce port à '0'.
annuler	Lorsque ce port à la valeur '1', cela signifie que le client veut annuler sa commande et reprendre ses pièces. L'ordre d'annulation ne sera pas considéré pendant qu'un café est servi ou que la monnaie est rendue.
servir_cafe	Ce port est égal à '1' lorsque le total des pièces introduites est supérieur ou égal au montant du café à servir. Ce port sera égal à '1' jusqu'à ce que le café soit servi (cafe_servi='1'), ensuite il retombera à '0'.
activer_cafe1	Ce port est égal à '1' quand le montant du café a été introduit et que *choix_cafe* est égal a '1'.
activer_cafe2	Ce port est égal à '1' quand le montant du café a été introduit et que *choix_cafe* est égal à '0'.
cafe_servi	Ce port est égal à '1' pour signifier que le café a été servi au client. Le gestionnaire devra remettre la monnaie du client s'il y a lieu.
piece_a_rendre	Le montant de la pièce à rendre.
piece_a_rendre_valide	Lorsque ce port est égal à '1', la partie mécanique sortira la pièce équivalente au montant sur le port "piece_a_rendre".

Pour concevoir le processeur à but spécifique, vous devez :

a) Compléter le diagramme de l'architecture simplifiée de votre circuit en montrant les signaux et ports qui sont utilisés par le chemin de données et l'unité de contrôle. (4 points)

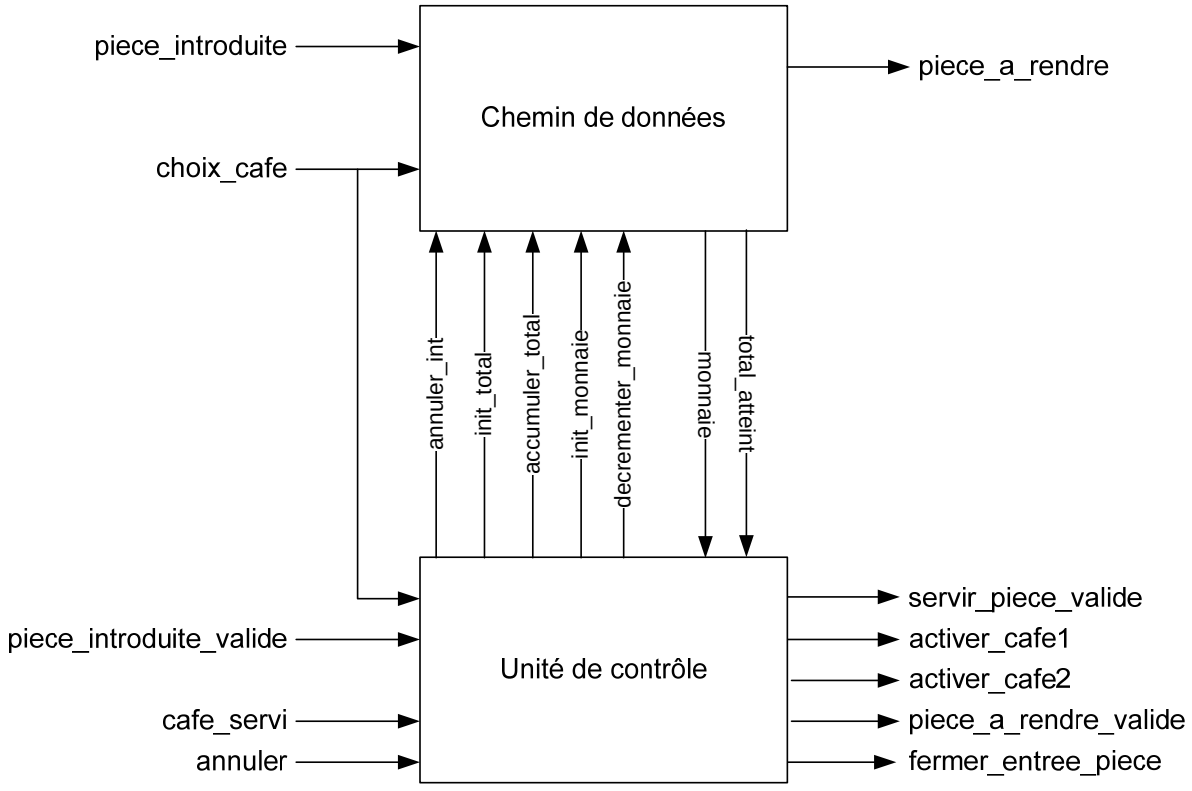
Détachez et remettez la page réponse de la question 1.a avec votre cahier d'examen.

b) Donner le diagramme de la machine à états de l'unité de contrôle. (4 points)

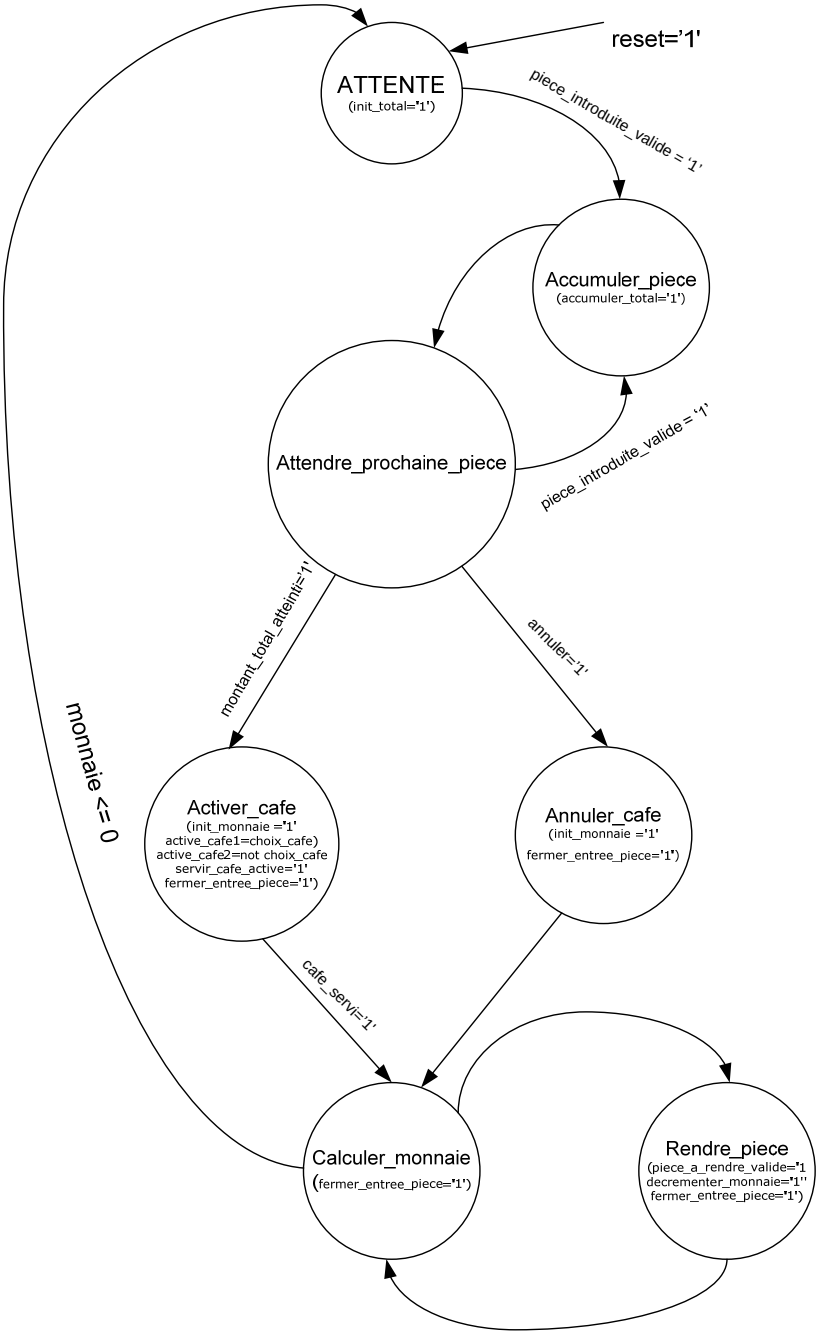
c) Donner le diagramme du chemin de données de votre circuit. (4 points)

Solution

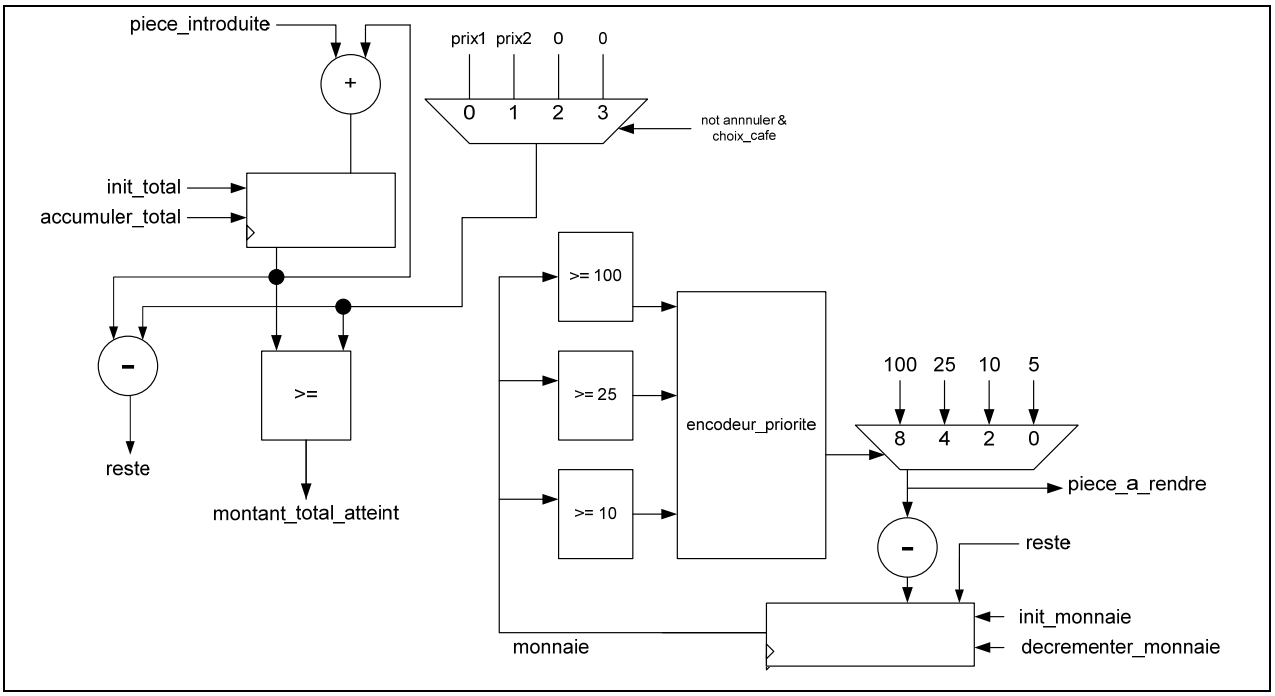
a. le diagramme de l'architecture simplifiée



b. le diagramme de la machine à états de l'unité de contrôle



c. Le diagramme du chemin de données


piece_introduite
+
prix1 prix2 0 0
0 1 2 3
not annnuler &
choix_cafe
init_total
accumuler_total
>= 100
>= 25
>= 10
encodeur_priorite
100 25 10 5
8 4 2 0
piece_a_rendre
-
reste
>=
montant_total_atteint
-
reste
init_monnaie
decrementer_monnaie
monnaie

Question 2. Questions brèves (4.5 points)

a. Citez 4 technologies de logique programmable (1 point).

b. Citez 4 composantes (autres que les LUT, multiplexeurs et bascules) qu'on peut retrouver dans un FPGA (1 points).

c. Énumérez 2 conditions d'opérations qui ont un impact sur le délai de propagation d'une composante (0.5 point)

d. Quelle technique de génération de vecteurs de tests serait la plus appropriée pour tester une machine à états? Expliquez pourquoi. (1 point)

e. Comparez les SRAM et les DRAM? (un avantage et un inconvénient). (1 point)

Solution

a. ROM, PLA, PAL, FPGA

b. CLB, arbre d'horloge, IOB port, block RAM, processeur, bloc DSP tel que des MAC, logique d'interconnexion.

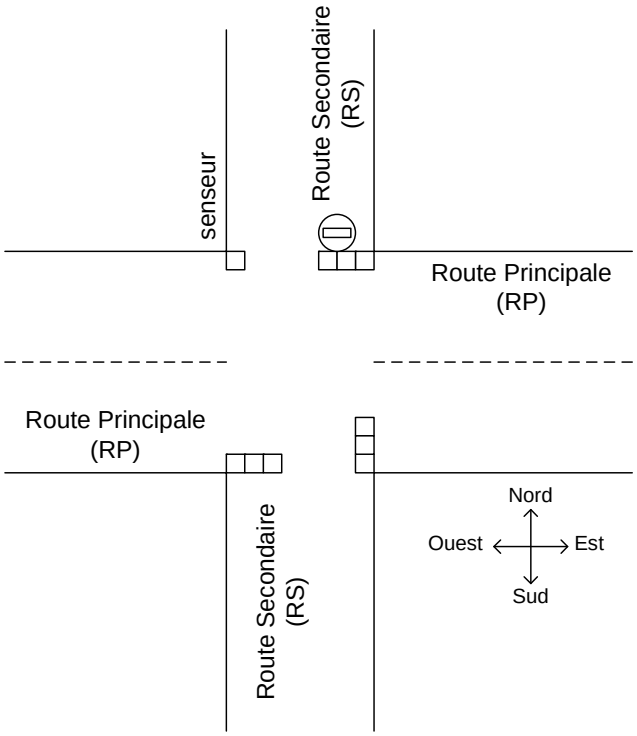
c. Temperature, tension d'alimentation, la charge menée par la composante.

d. Les circuits de machine à états devraient être vérifiés par des vecteurs exhaustifs afin de s'assurer que tous les états de la machine ont un comportement attendu.

e. SRAM : vitesse d'accès plus rapide mais elles nécessitent plus de portes logiques
DRAM : elles nécessitent un rafraichissement périodique sinon les données seront perdues, mais elles nécessitent moins de portes logiques.

Question 3. Circuit séquentiel (9 points)

Soit le circuit gestionnaire_feux dont l'entité est décrite comme suit. Ce contrôleur de feux gère la circulation à une intersection de voies. Les voies sont illustrées par la figure suivante.



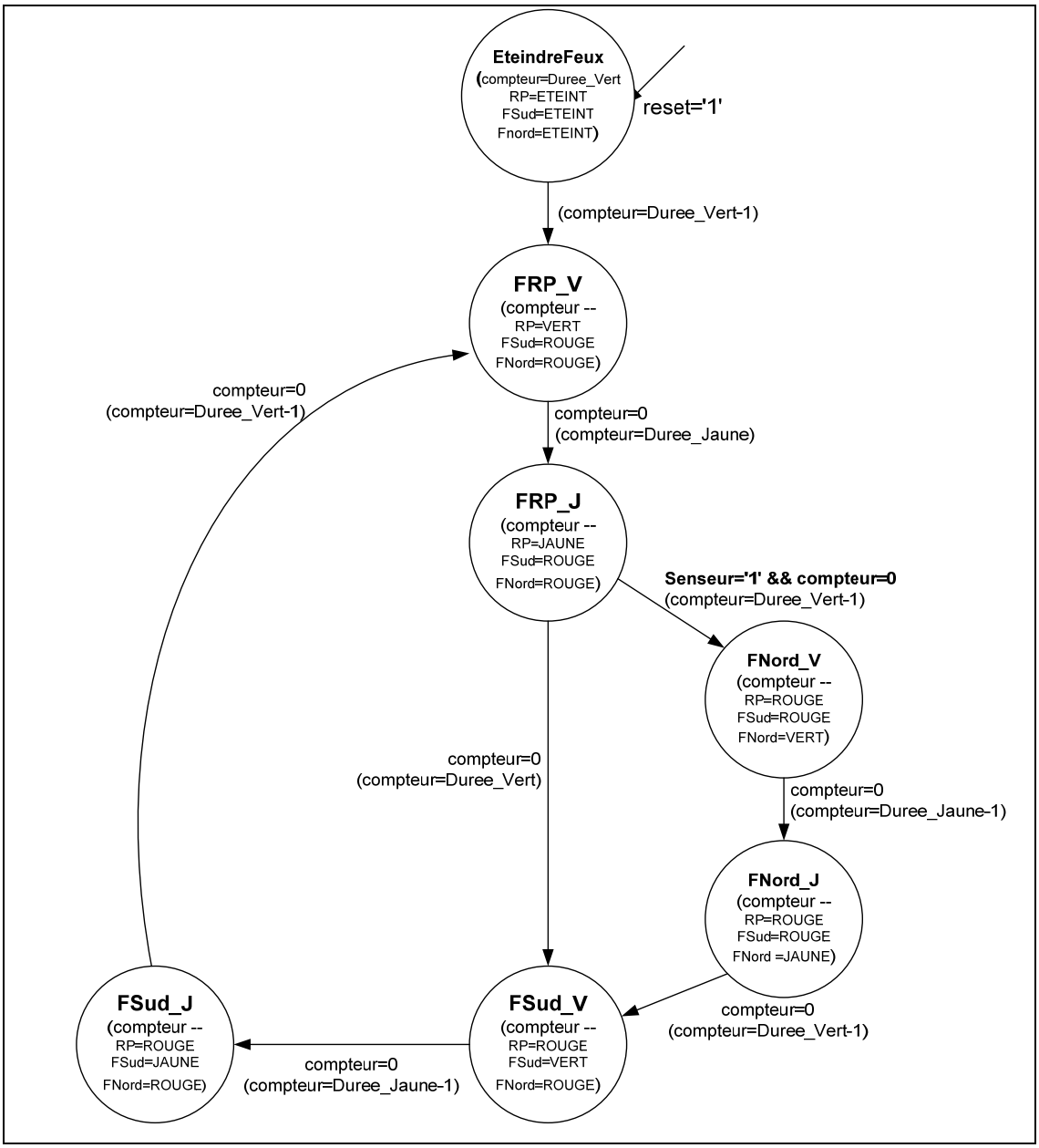
À cette intersection, les voitures de la voie secondaire ne peuvent que tourner vers la gauche ou vers la droite. Étant donné que le flôt de voitures venant du Nord de la voie secondaire est faible, un senseur a été placé afin que le feu autorisant les voitures venant du Nord ne soit vert que lorsqu'au moins une voiture y est présente (senseur='1').

Les voitures de la voie principale circuleront lorsque leurs feux seront vert ou jaune. Ces derniers ne peuvent circuler que d'Est en Ouest.

L'entité du gestionnaire de feux gest_feux est décrite comme suit :

```
entity gest_feux is
  generic (
    Duree_Vert : positive;
   Duree_Jaune : positive
    );
  port (
    reset : in std_logic;
    clk : in std_logic;
    senseur : in std_logic;
    FRP : out std_logic_vector (2 downto 0);
    FSud : out std_logic_vector (2 downto 0);
    FNord : out std_logic_vector (2 downto 0)
    );
end gest_feux;
```

La machine à états qui décrit le comportement du gestionnaire de feux est comme suit :



Donnez le code VHDL de l'architecture de la machine à états du gestionnaire de feux *gest_feux*. Le reset du circuit est asynchrone.

En sachant que les valeurs des feux sont énumérées comme suit :

- ROUGE = "100";
- VERT = "001";
- JAUNE = "010".

Solution

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity gest_feux is
  generic (
    Duree_Vert : positive;
   Duree_Jaune : positive
    );

  port (
    reset : in std_logic;
    clk : in std_logic;
    senseur : in std_logic;
    FRP : out std_logic_vector (2 downto 0);
    FSud : out std_logic_vector (2 downto 0);
    FNord : out std_logic_vector (2 downto 0)
    );

end gest_feux;

architecture beh of gest_feux is
  type etat_t is (eteindrefeux, FRP_V,  FRP_J,  FSUD_V, FSUD_J,  FNORD_V, FNORD_J);
  signal etat : etat_t;
  signal compteur : integer;
begin  -- beh
  process (clk, reset)
  begin  -- process
    if reset = '1' then                 -- asynchronous reset (active low)
      etat <= eteindrefeux;
      compteur <=  Duree_Vert-1;
      FRP <= "000";
      FSud <= "000";
      FNord <= "000";
    elsif clk'event and clk = '1' then  -- rising clock edge
      FRP <= "000";
      FSud <= "000";
      FNord <= "000";

      -- le compteur sera reinitialise lors de la transition entre 2 etats
      compteur <= compteur - 1;
      case etat is
        when eteindrefeux =>
          compteur <= Duree_Vert-1;
          etat <= FRP_V;
          -- feux de la route principale sont vert
        when FRP_V =>
          FRP <= "001";
          FSUD <= "100";
          FNord <= "100";
          if compteur = 0 then
```

```
      etat <= FRP_J;
      compteur <= Duree_Jaune - 1;
    end if;
    -- feux de la route principale sont jaune
  when FRP_J =>
    FRP <= "010";
    FSUD <= "100";
    FNord <= "100";
    if compteur = 0 then
      if senseur = '1' then
        etat <= FNORD_V;
      else
        etat <= FSUD_V;
      end if;
      compteur <= Duree_Vert - 1;
    end if;
    -- feu des voitures venant du sud est vert
  when FSUD_V =>
    FRP <= "100";
    FSUD <= "001";
    FNord <= "100";
    if compteur = 0 then
      etat <= FSUD_J;
      compteur <= Duree_Jaune-1;
    end if;
    -- feu des voitures venant du sud est jaune
  when FSUD_J =>
    FRP <= "100";
    FSUD <= "010";
    FNord <= "100";
    if compteur = 0 then
      etat <= FRP_V;
      compteur <= Duree_Vert - 1;
    end if;
    -- feu des voitures venant du nord est vert
  when FNORD_V =>
    FRP <= "100";
    FSUD <= "100";
    FNord <= "001";
    if compteur = 0 then
      etat <= FNORD_J;
      compteur <= Duree_Jaune - 1;
    end if;
    -- feu des voitures venant du nord est jaune
  when FNORD_J =>
    FRP <= "100";
    FSUD <= "100";
    FNord <= "010";
    if compteur = 0 then
      etat <= FSUD_V;
      compteur <= Duree_Vert-1;
```

```
        end if;
      when others =>
        etat <= eteindrefeux;
    end case;
  end if;
end process;

end beh;
```

Question 4. Processeur à but général (7 points)

Soit le processeur à but général présenté dans les notes de cours au chapitre 8.

On désire ajouter une seconde mémoire de données dans l'architecture du processeur. Cette mémoire sera une mémoire de 1024 cases contenant des mots de 16 bits. L'adresse d'un mot à lire ou à écrire d'une des mémoire de données sera maintenant contenue sur 11 bits en sachant que le bit 10 de l'adresse précise quelle mémoire de données est concernée par l'accès. Quand le bit est égal à '0' l'accès concerne la mémoire de données initiale, sinon l'accès concerne la seconde mémoire (bit 10 = '1'). Les bits 9 à 0 précisent la case à accéder selon la mémoire sélectionnée.

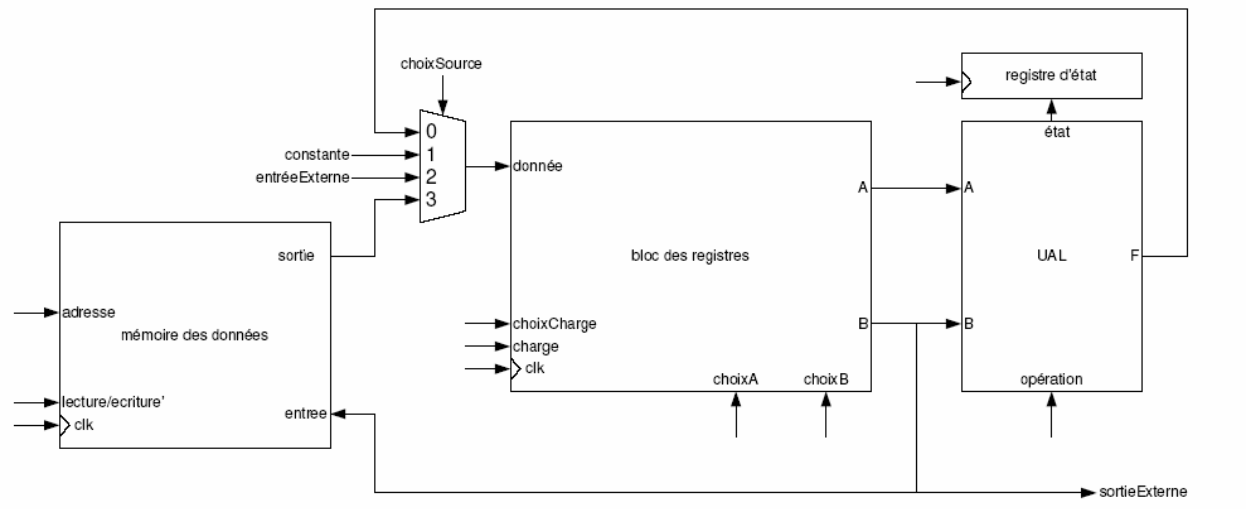
Nous rappelons que le registre d'instructions comporte 16 bits dont 4 bits alloués au code de l'instruction, 4 bits pour l'adresse du registre dans lequel la donnée sera lue ou écrite.

a. Donner le nouveau diagramme du chemin de données et préciser les composantes à rajouter ou à modifier. (3 points)

b. Décrivez les nouveaux signaux ou les modifications effectuées sur certains signaux du chemin de données. (1 points).

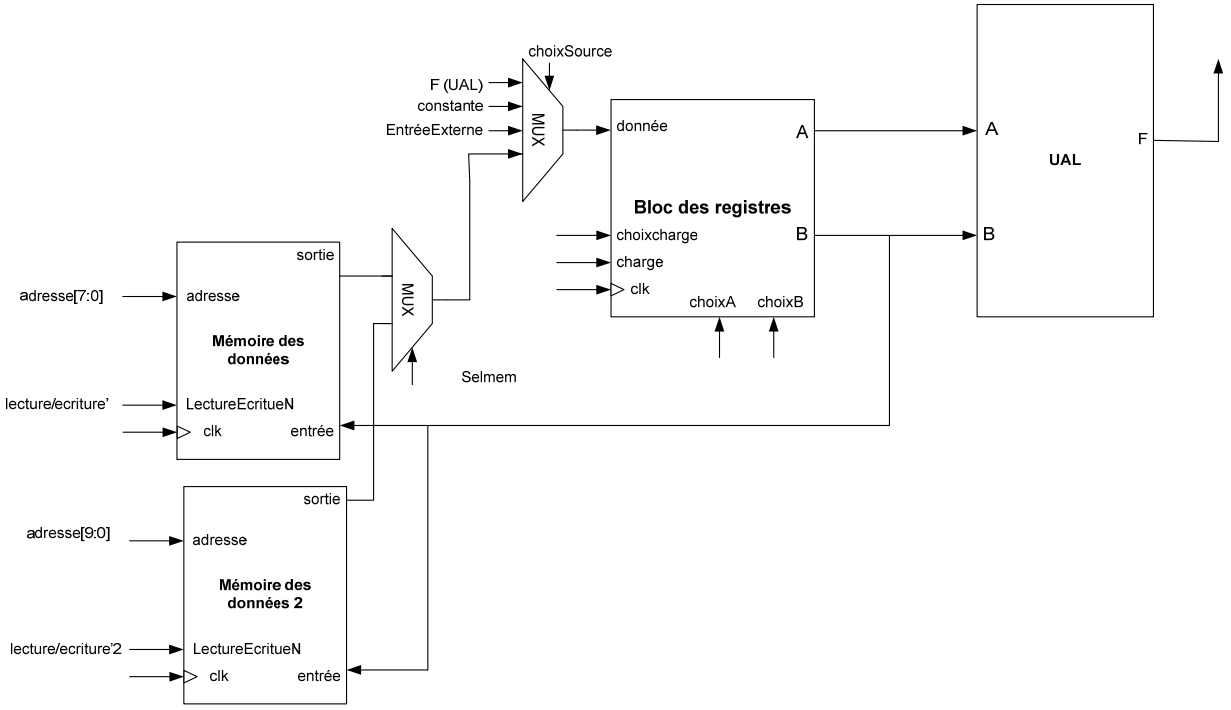
c. Quel sera l'impact de la nouvelle taille d'une adresse de mémoire de données sur l'unité de contrôle du processeur? Modifiez le diagramme de la machine à états. (3 points)
Détachez et remettez la page réponse de la question 4.c avec votre cahier d'examen.

À titre d'exemple, le diagramme initial du chemin de données du processeur à but général au chapitre 8 des notes de cours est présenté ci-après.



Solution

a.

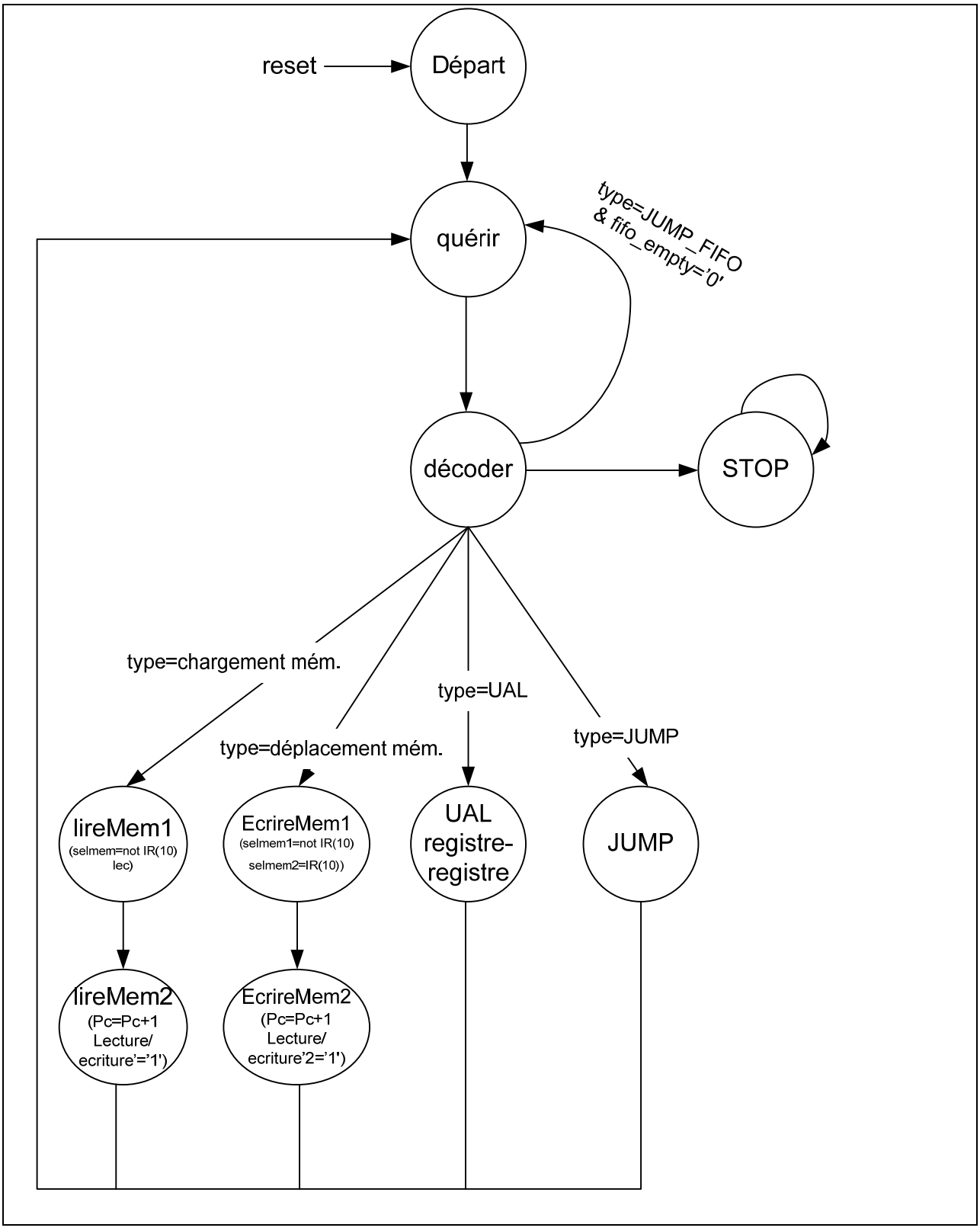


b. 2 signaux ont été rajoutés :

selmem : sélectionne la donnée qui devra être écrite dans un registre.

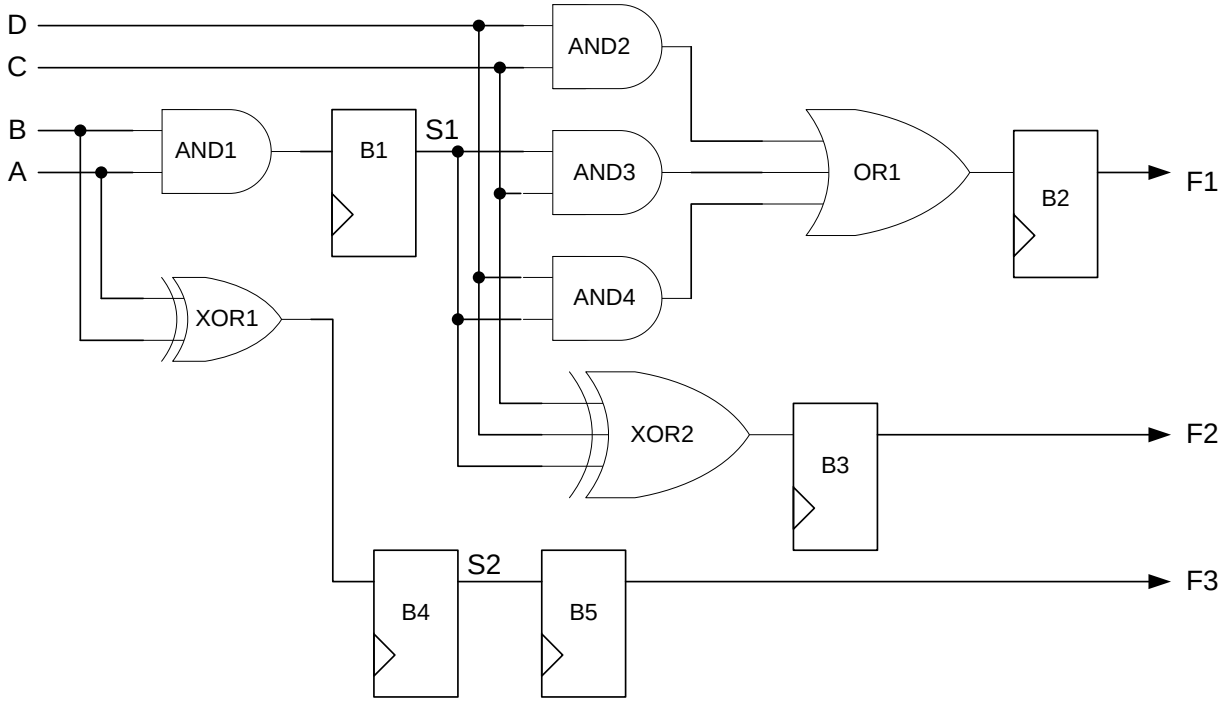
lecture/ecriture'2 : active la lecture ou l'écriture d'une donnée dans la seconde mémoire.

c. il faudra un cycle d'horloge de plus pour exécuter les instructions de chargement mémoire.


reset
Départ
quérir
type=JUMP_FIFO
& fifo_empty='0'
décoder
STOP
type=chargement mém.
type=déplacement mém.
type=UAL
type=JUMP
lireMem1
(selmem=not IR(10)
lec)
EcrireMem1
(selmem1=not IR(10)
selmem2=IR(10))
UAL
registre-
registre
JUMP
lireMem2
(Pc=Pc+1
Lecture/
ecriture'='1')
EcrireMem2
(Pc=Pc+1
Lecture/
ecriture'2='1')

Question 5. Simulation et Synthèse (7.5 points)

Soit le circuit suivant :



En sachant que les portes ont les délais suivants :

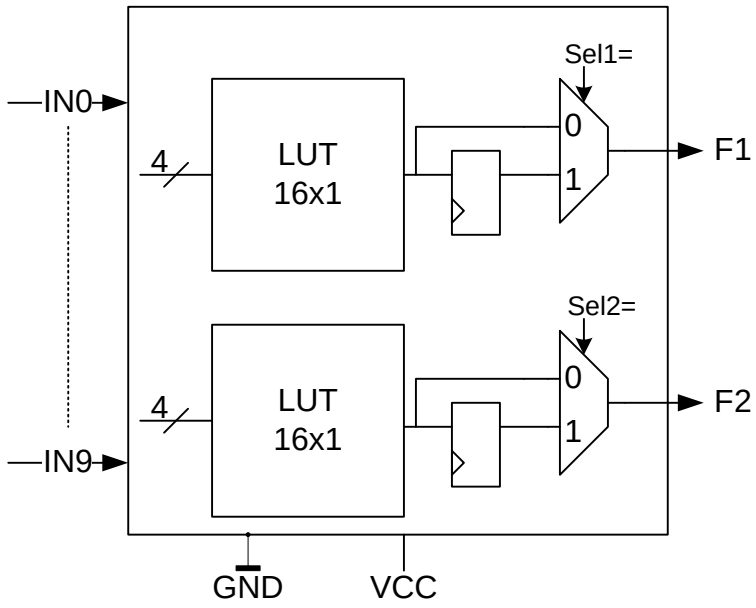
- Les bascules ont un temps de préparation de 1 ns, un temps de maintien de 1 ns et un délai de propagation de 2 ns.
- Les portes logiques ET et OU exclusif à 2 entrées ont un délai de propagation de 2 ns
- Les portes logiques OU et OU-exclusif à 3 entrées ont un délai de propagation de 3 ns.

a. En tenant compte des délais, compléter le chronogramme donné à la page suivante afin de fournir les variations de S1, S2, F1, F2 et F3 jusqu'au temps t = 60 ns (Remettre la page du chronogramme avec votre cahier d'examen) (2.5 points).

Détachez et remettez la page réponse de la question 5.a avec votre cahier d'examen.

b. Donnez la période minimale du circuit. Explicitez vos calculs. (1 point)

c. Soit le bloc de logique programmable (CLB) suivant :

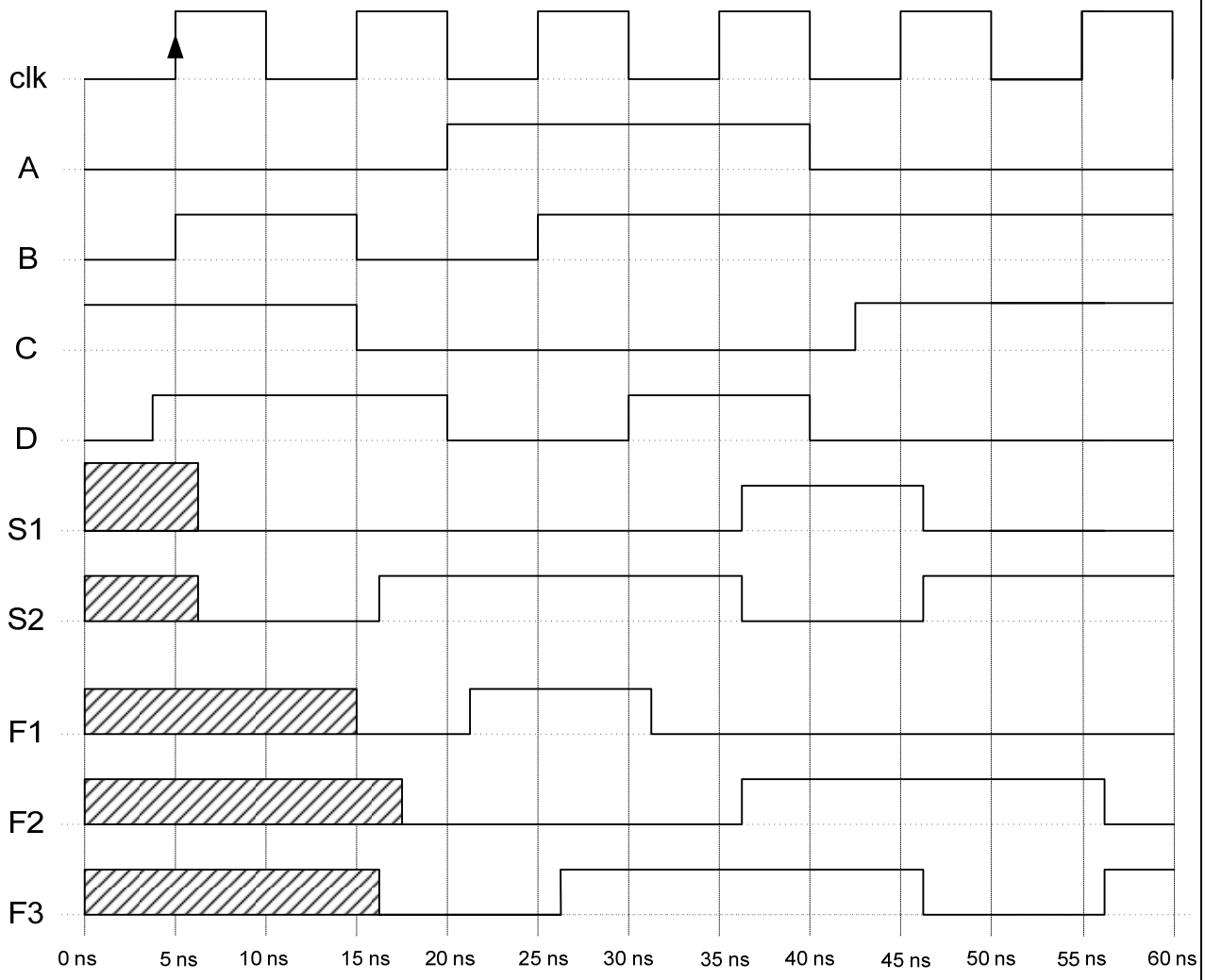


Dans ce dernier, on peut programmer 2 tables de conversion, 2 multiplexeurs. Les entrées IN0 à IN9, GND et VCC peuvent être connectées à tous éléments du CLB. L'entrée GND permet de propager la valeur logique '0' et l'entrée VCC la valeur '1'. Les sorties des tables de conversion ou celles des multiplexeurs peuvent être connectées à une des sorties F1 et F2.

Proposez un routage possible du circuit dans un FPGA. Vous pouvez instancier autant de CLB que nécessaire. Montrez les connexions entre les CLB, la valeur des ports de sélection Sel1 et Sel2 des multiplexeurs, et la fonction logique réalisée par les tables de correspondance (LUTs). (4 points)

Solution

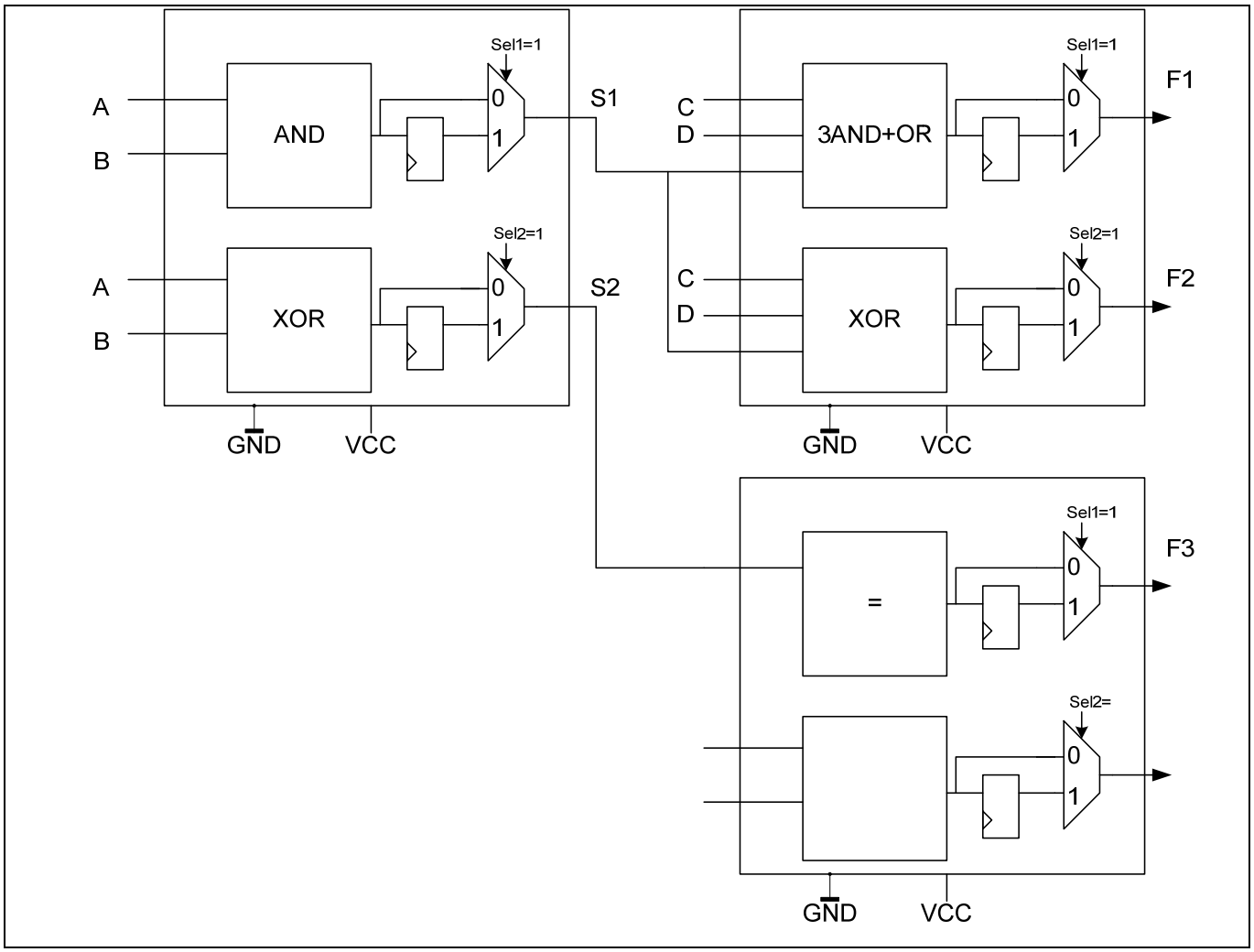
a.



b. la période minimale

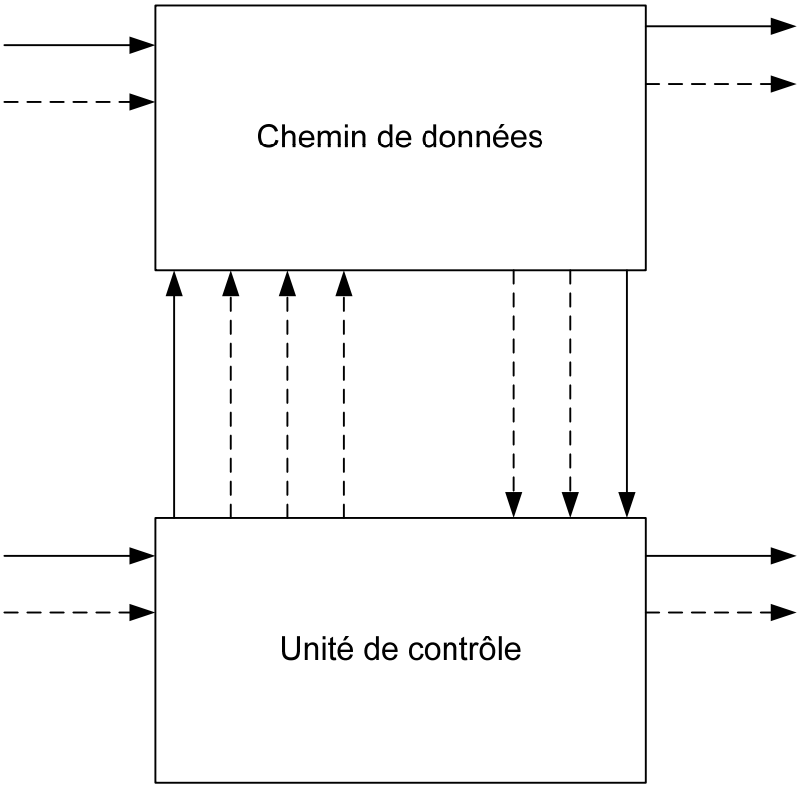
B1-AND3-OR1-B3->periode = 2+2+3+1 = 8 ns

c.


A
B
AND
Sel1=1
0
1
S1
A
B
XOR
Sel2=1
0
1
S2
GND
VCC
C
D
3AND+OR
Sel1=1
0
1
F1
C
D
XOR
Sel2=1
0
1
F2
GND
VCC
=
Sel1=1
0
1
F3
Sel2=
0
1
GND
VCC

INF3500-final H2009 – Question 1.a

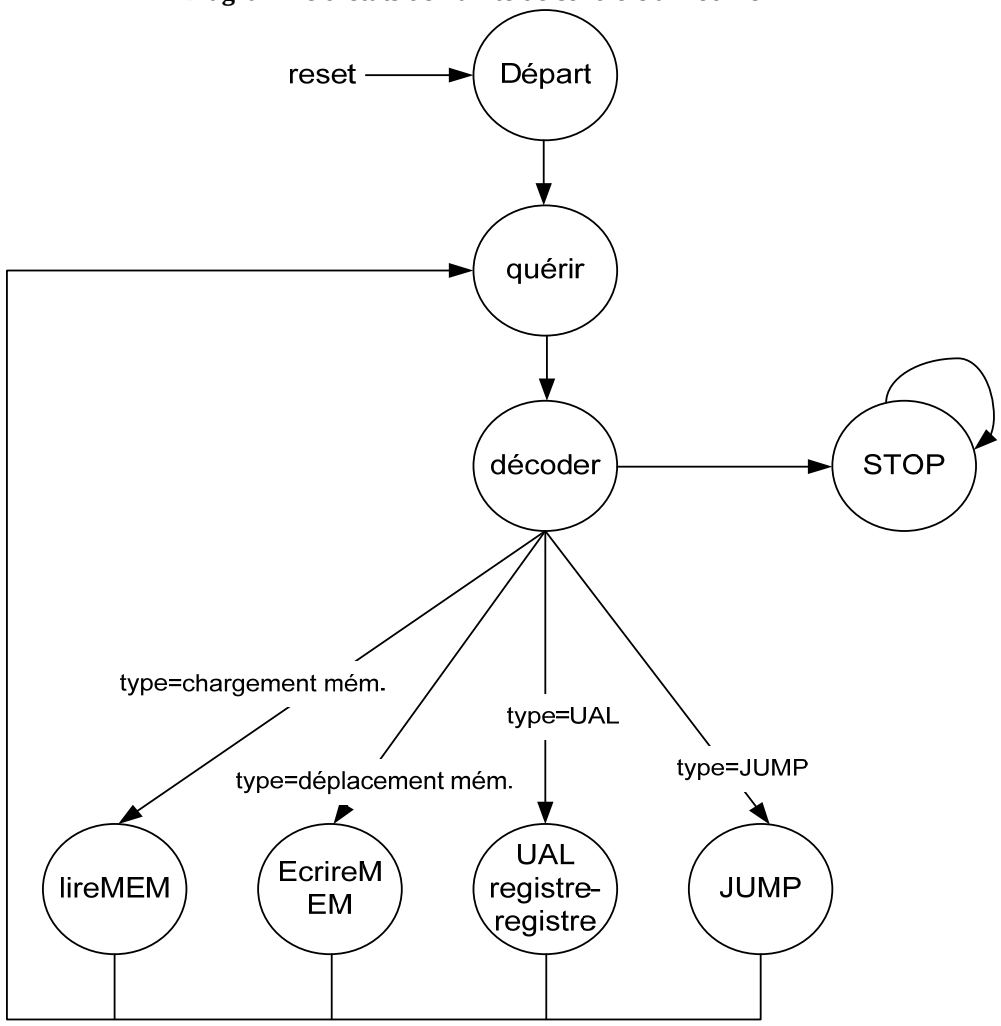
Nom : **Matricule :**



INF3500-final H2009 – Question 4.c

Nom : **Matricule :**

Diagramme d'états de l'unité de contrôle à modifier



INF3500-final H2009 – Question 5.a

Nom : **Matricule :**