INF3500 : Conception et réalisation de systèmes numériques
Examen final
12 août 2008

Durée: 2h30.
Pondération: 40%.
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. (8 points)

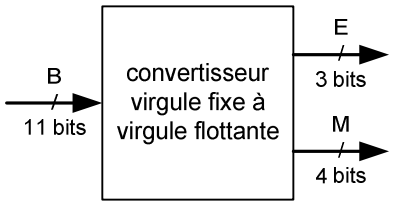
Les deux systèmes de numération les plus populaires sont à virgule fixe et à virgule flottante. Le système à virgule fixe correspond aux types `char`, `short`, `int`, et `long`, et le système à virgule flottante correspond aux types `float` et `double` du langage C. Le système à virgule flottante peut représenter une plus grande gamme de nombres avec moins de bits, mais avec moins de précision.

Par exemple, pour la représentation d'un nombre *B* non signé à virgule fixe avec 11 bits $b_{10}b_9b_8 \ldots b_1b_0$ la valeur du nombre est égale à $\sum_{k=0}^{k=10} b_k \cdot 2^k$. Le plus grand nombre pouvant être représenté est alors $2^{11} - 1 = 2047$. La même gamme de nombres peut être représentée approximativement avec un système à virgule flottante de 7 bits seulement, avec une mantisse *M* de 4 bits et un exposant *E* en base 2 de 3 bits. La valeur du nombre est égale à $M \cdot 2^E = \sum_{k=0}^{k=3} m_k \cdot 2^{\sum_{p=0}^{p=2} e_p 2^p}$. Dans ce système simplifié, les nombres et leurs exposants sont positifs ou nuls. Le tableau suivant donne des exemples d'équivalences entre les deux systèmes. Il indique aussi l'erreur d'approximation de la représentation en virgule flottante.

Nombre *B* (base 10)	Représentation en virgule fixe (11 bits)	Représentation en virgule flottante		
		Exposant *E* (3 bits)	Mantisse *M* (4 bits)	Erreur (base 10)
0	00000000000	000	0000	0
1	00000000001	000	0001	0
13	00000001101	000	1101	0
49	00000110001	010	1100	1
633	01001111001	110	1001	57
2041	11111111001	111	1111	121

On observe que l'exposant correspond à l'indice du '1' le plus significatif du nombre, moins 3. S'il n'y a pas de '1' aux indices 4 à 10, l'exposant est 0. On observe aussi que la mantisse est donnée par les 4 bits à partir du '1' le plus significatif, ou par les 4 bits les moins significatifs du nombre si il n'y a pas de '1' aux indices 4 à 11.

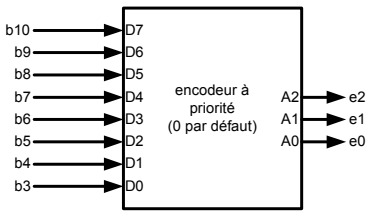
Vous devez concevoir un circuit numérique combinatoire qui accepte en entrée un nombre non signé représenté en format virgule fixe sur 11 bits et qui retourne sa valeur en format virgule flottante avec 4 bits pour représenter la mantisse *M* et 3 bits pour représenter l'exposant *E* en base 2. Vous pouvez utiliser des blocs préconçus comme des multiplexeurs, décodeurs, encodeurs, additionneurs, décaleurs, etc. Votre circuit doit être purement combinatoire, il ne doit pas inclure d'éléments à mémoire.



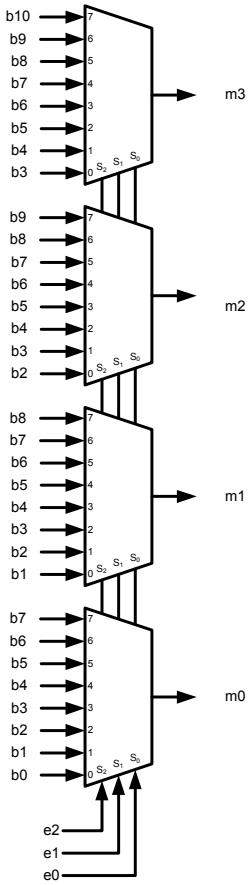
a. Donnez un diagramme de votre circuit.

b. Donnez sa description en VHDL.

a. Solution :



L'encodeur à priorité a une sortie A = 0 quand toutes ses entrées sont égales à 0.



b. Solution :

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

entity vfixeavflottante is
	port (
		B : in unsigned(10 downto 0);
		E : out unsigned(2 downto 0);
		M : out unsigned(3 downto 0)
	);
end vfixeavflottante;

architecture arch1 of vfixeavflottante is
begin
	process(B)
	begin
		E <= to_unsigned(0, E'length);
		M <= B(3 downto 0);
		for k in B'length - 1 downto 3 loop
			if B(k) = '1' then
				E <= to_unsigned(k - 3, E'length);
				M <= B(k downto k - 3);
				exit;
			end if;
		end loop;
	end process;
end arch1;
```

Question 2. (14 points)

Concevez un processeur qui effectue la traduction d'adresses dans un routeur. La traduction d'adresses consiste à faire correspondre une adresse IP d'un intranet à une adresse IP publique visible sur le web. Un des buts de cette technique de sécurité est de rendre le réseau intranet invisible du web et donc moins susceptible aux attaques de pirates informatiques. Pour ce faire, des tables de correspondances sont implémentées telles que :

- table_adresse_ip_interne : liste des adresses internes valides
- table_adresse_ip_externe : liste des adresses externes associées correspondantes.

La Figure 1 illustre un cas de correspondance entre plusieurs adresses. L'adresse IP 0 (0.0.0.0) n'est allouée à aucune machine.

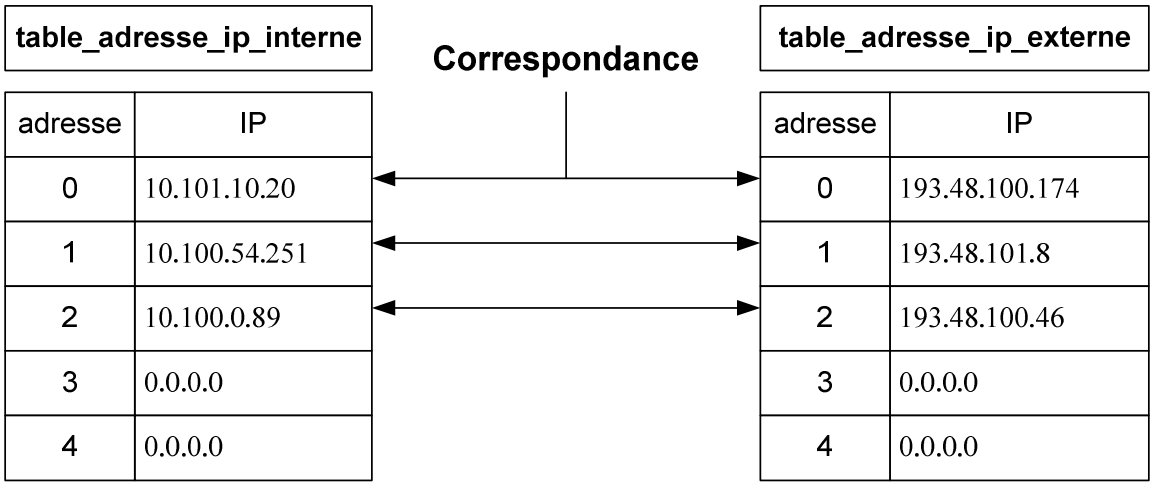


Figure 1. Exemple de correspondance entre deux tables.

En bref, le circuit fonctionnera comme suit :

- Remise à zéro des tables : le circuit doit être capable d'initialiser les tables en mettant toutes leurs cases à 0.

- Mise à jour des tables : le circuit doit pouvoir mettre à jour les cases une par une selon des requêtes entrantes. Une requête de mise à jour précisera les cases à mettre à jour dans les deux tables. Pour cela, l'adresse des cases sera transmise ainsi que les nouvelles valeurs à écrire dans celles-ci.

- Recherche d'adresses : le circuit doit être capable de répondre à des requêtes de recherche. Il recevra une adresse IP qui peut être une adresse interne ou externe et il devra retourner l'adresse externe ou interne correspondante. Si aucune correspondance n'est trouvée alors une erreur sera émise.

- Une mise à jour ne peut pas s'effectuer en même temps qu'une requête de recherche.

La Figure 2 précise les interfaces du circuit NAT.

```
entity nat is
generic (tab_size : integer := 10);
port (
        rst_n : in std_logic;
        clk : in std_logic;
        maj_req : in std_logic;
        maj_adr : in std_logic_vector(tab_size-1 downto 0);
        maj_ip_prive : in std_logic_vector(31 downto 0);
        maj_ip_publique : in std_logic_vector(31 downto 0);
        recherche_req : in std_logic;
        ip_type : in std_logic;
        ip_recherche : in std_logic_vector(31 downto 0);
        recherche_resp : out std_logic;
        erreur: out std_logic;
        ip_trouve : out std_logic_vector(31 downto 0)
);
end nat;
```

Figure 2. Déclaration de l'entité `nat`

Les interfaces se décrivent comme suit :

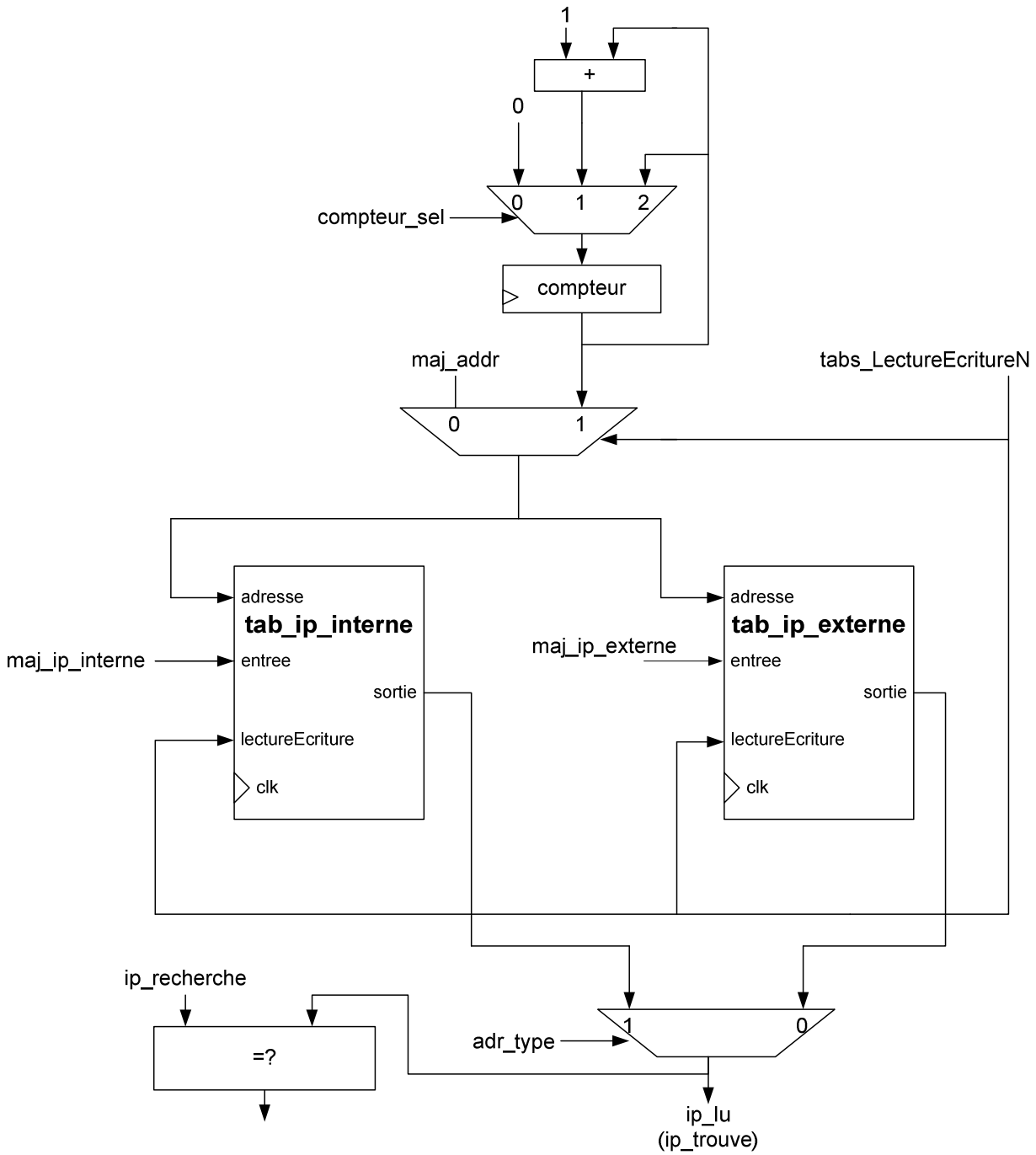
rst_n	Le circuit est réinitialisé lorsque ce signal est égal à '0'. Toutes les cases des tables seront mises à 0.
clk	L'horloge du circuit.
maj_req	Le signal d'activation d'écriture.
maj_adr	L'adresse mémoire dans laquelle les adresses IP transmises interne et externe seront écrites.
maj_ip_prive	L'adresse IP privée (interne) à écrire.
maj_ip_publique	L'adresse IP publique (externe) à écrire.
recherche_req	Ce signal est activé lorsqu'une requête de recherche est transmise.
ip_type	Lorsque ce port est égal à '1' alors c'est l'adresse IP privée qui est transmise et le circuit doit fournir l'adresse IP publique correspondante. Lorsque ce port est égal à '0' alors c'est l'adresse IP publique qui est transmise et le circuit doit fournir l'adresse IP privée correspondante.
ip_recherche	L'adresse IP dont on recherche la correspondance.
recherche_resp	Ce signal est activé par le circuit lorsque la réponse de recherche est prête.
erreur	Ce signal est activé lorsque l'adresse transmise en requête n'est pas contenue dans une des tables.
ip_trouve	L'adresse de sortie correspondante à l'adresse IP recherchée.

Vous devez :

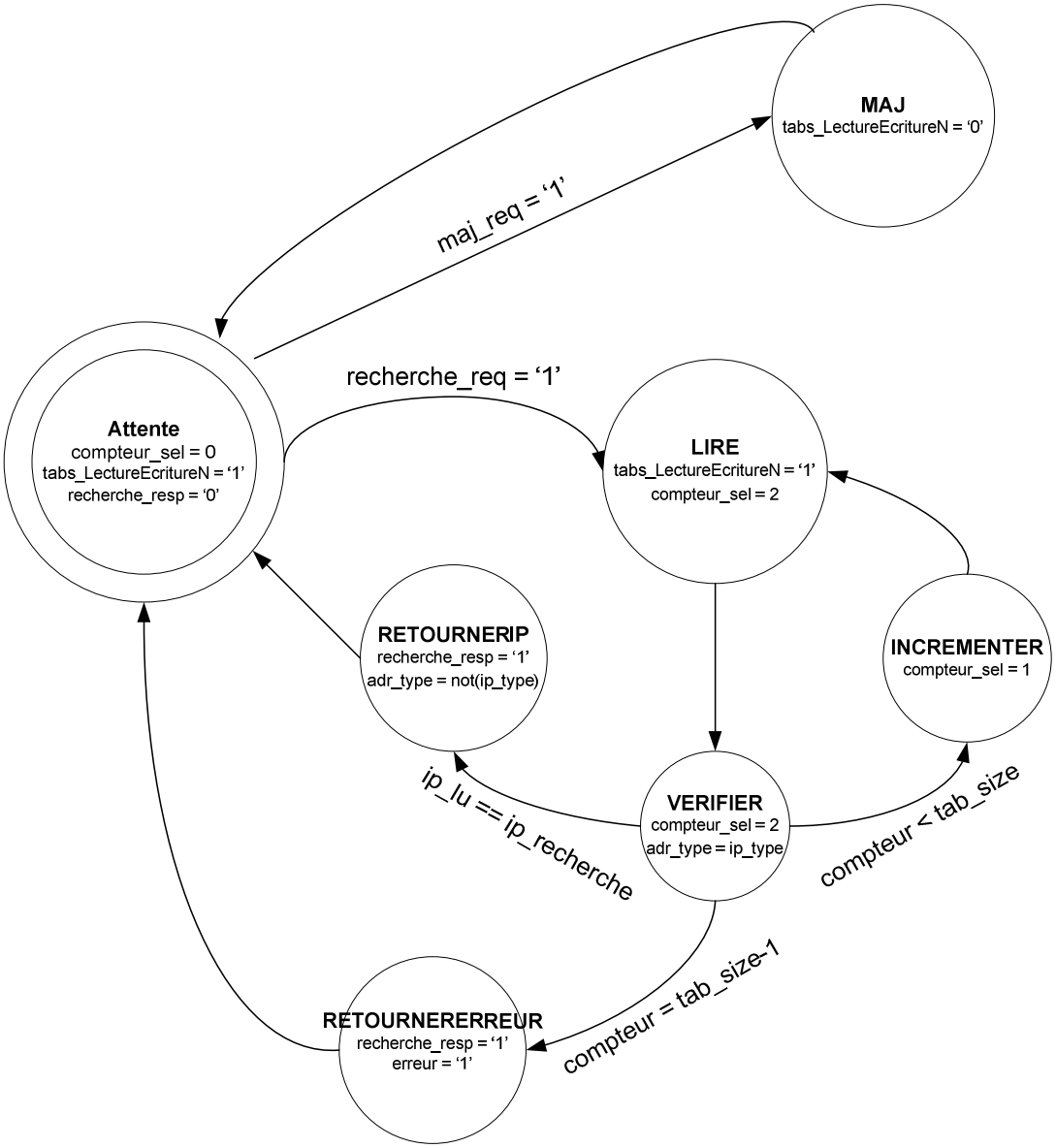
a) donner le diagramme du chemin de données du NAT (5 points).
b) donner le diagramme de la machine à états de l'unité de contrôle du NAT. Identifier clairement les états ainsi que les valeurs des signaux de contrôle en fonction des états de la machine (5 points).
c) écrire le code VHDL du chemin des données seulement (4 points).

a. Solution

(Note: On pourrait considérer que le compteur et la logique qui lui est associée ne font pas partie du chemin des données mais plutôt de l'unité de contrôle.)



b. Solution


MAJ
tabs_LectureEcritureN = '0'
maj_req = '1'
Attente
compteur_sel = 0
tabs_LectureEcritureN = '1'
recherche_resp = '0'
recherche_req = '1'
LIRE
tabs_LectureEcritureN = '1'
compteur_sel = 2
RETOURNERIP
recherche_resp = '1'
adr_type = not(ip_type)
INCREMENTER
compteur_sel = 1
ip_lu == ip_recherche
VERIFIER
compteur_sel = 2
adr_type = ip_type
compteur < tab_size
RETOURNERERREUR
recherche_resp = '1'
erreur = '1'
compteur = tab_size-1

c. solution (toute l'entité, chemin des données et machine à états)

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

entity nat is

    generic (tab_size : integer := 10);

    port (rst_n : in std_logic;
          clk : in std_logic;
          maj_req : in std_logic;
          maj_adr : in std_logic_vector(tab_size-1 downto 0);
          maj_ip_prive : in std_logic_vector(31 downto 0);
          maj_ip_publique : in std_logic_vector(31 downto 0);
          recherche_req : in std_logic;
          ip_type : in std_logic;
          ip_recherche : in std_logic_vector(31 downto 0);
          recherche_resp : out std_logic;
          erreur: out std_logic;
          ip_trouve : out std_logic_vector(31 downto 0)
          );
end nat;


architecture beh of nat is
    signal operation : std_logic;
    signal compteur_sel : integer range 0 to 2;
    signal compteur : integer range tab_size-1 downto 0;
    type tab_type is array (tab_size-1 downto 0) of std_logic_vector(31 downto 0);
    signal tab_ip_interne : tab_type;
    signal tab_ip_externe : tab_type;
    signal tabs_LectureEcritureN : std_logic;
    signal tab_addr : std_logic_vector(tab_size-1 downto 0);
    signal ip_lu : std_logic_vector(31 downto 0);
    type etat_type is (ATTENTE, INIT, MAJ, LECTURE, VERIFIER,
                       INCREMENTER, RETOURNERERREUR, RETOURNERIP);
    signal etat : etat_type;
begin  -- beh


    unite_controle : process (clk, rst_n)

    begin  -- process unite_controle
       -- activities triggered by asynchronous reset (active low)
       if rst_n = '0' then
           etat <= INIT;
           -- activities triggered by rising edge of clock
       elsif clk'event and clk = '1' then
-- Mise a jour de la FSM
           case etat is
               when ATTENTE =>
                   if maj_req = '1' then
                       etat <= MAJ;
                   elsif recherche_req = '1' then
                       etat <= LECTURE;
                   end if;
               when INIT =>
                   etat <= ATTENTE;
               when MAJ =>
                   etat <= ATTENTE;
               when LECTURE =>
                   etat <= VERIFIER;
```

```
                when VERIFIER =>
                    if ip_lu = ip_recherche then
                        etat <= RETOURNERIP;
                    elsif compteur = tab_size-1 then
                        etat <= RETOURNERERREUR;
                    else
                        etat <= INCREMENTER;
                    end if;
                when INCREMENTER =>
                    etat <= LECTURE;
                when RETOURNERERREUR =>
                    etat <= ATTENTE;
                when RETOURNERIP =>
                    etat <= ATTENTE;
                when others =>
                    etat <= ATTENTE;
            end case;
        end if;
    end process unite_controle;

    -- chemin de donnees
    -- gestion des memoires
    process (clk, rst_n)

    begin  -- process
        -- activities triggered by asynchronous reset (active low)
        if rst_n = '0' then
            for i in 0 to tab_size-1 loop
                tab_ip_interne(i) <= (others => '0');
                tab_ip_externe(i) <= (others => '0');
            end loop;  -- i
            compteur <= 0;

            -- activities triggered by rising edge of clock
        elsif clk'event and clk = '1' then
            if tabs_LectureEcritureN = '0' then
                tab_ip_interne(to_integer(unsigned(tab_addr))) <= maj_ip_prive;
                tab_ip_externe(to_integer(unsigned(tab_addr))) <= maj_ip_publique;
            end if;
            if compteur_sel = 0 then -- init
                compteur <= 0;
            elsif compteur_sel = 1 then -- incrementer
                compteur <= compteur + 1;
            else
                compteur <= compteur;
            end if;
        end if;
    end process;

    -- lecture et multiplexage de l'IP lu selon le type de l'adresse a lire
    ip_lu <= tab_ip_interne(to_integer(unsigned(tab_addr))) when ip_type = '0'
             else  tab_ip_externe(to_integer(unsigned(tab_addr)));
    ip_trouve <= ip_lu;


    -- mise a jour des signaux de controle et de sortie selon l'etat courant
    recherche_resp <= '1' when etat = RETOURNERIP or etat = RETOURNERERREUR else '0';
    erreur <= '1' when etat = RETOURNERERREUR else '0';
    tabs_LectureEcritureN <= '0' when etat = MAJ else '1';
    compteur_sel <= 0 when etat = ATTENTE else 1 when etat = INCREMENTER else 2;
    tab_addr <= maj_adr when tabs_LectureEcritureN = '0' else
std_logic_vector(to_unsigned(compteur, tab_addr'length));

end beh;
```

Question 3. (6 points)

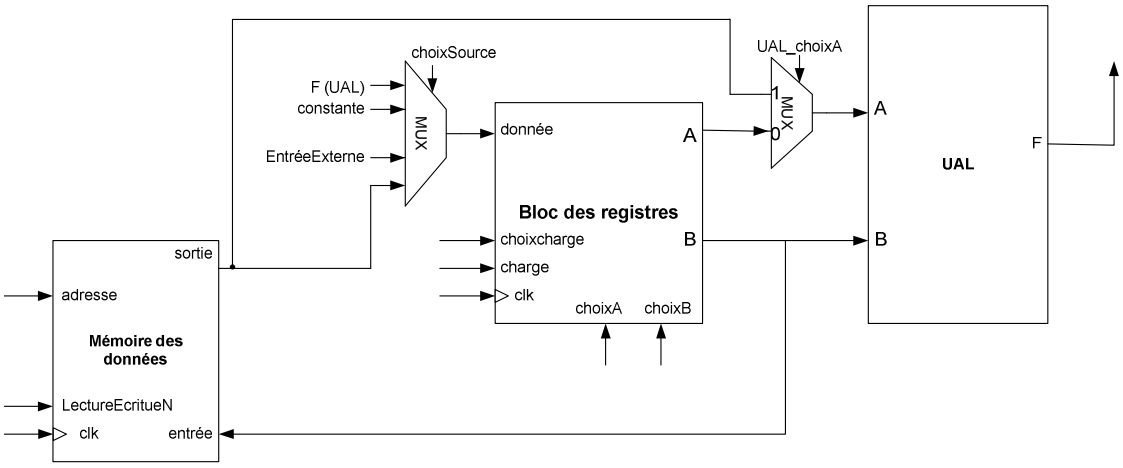
Considérez le processeur à usage général décrit au chapitre 8 des notes de cours. Nous désirons ajouter une nouvelle instruction :
Rk ← MEM[adresse] + Rk

Cette instruction lit dans la mémoire de données le contenu de la case MEM[adresse], et le met sur le signal A qui entre dans l'UAL. Pendant ce temps, le contenu du registre Rk est aussi lu et mis sur le signal B qui sera aussi utilisé par l'UAL. L'UAL effectuera l'addition puis le résultat transmis sur sa sortie F sera écrit dans le registre Rk. Donc, le résultat de l'addition sera écrit dans le registre source.

Vous devez :
a. donner le diagramme modifié du chemin de données du processeur (2 points).
b. donner la liste et une brève description du/des signal/signaux signaux que vous rajoutez au processeur (1 point).
c. donner l'encodage de l'instruction en conformité avec l'encodage décrit dans les notes de cours, section 8, tableau 8.5 (1 point).
d. donner la valeur des signaux de contrôle du processeur (les anciens et les nouveaux) si l'instruction à exécuter est R0 ← MEM[20] + R4. Suivez le format du tableau suivant :

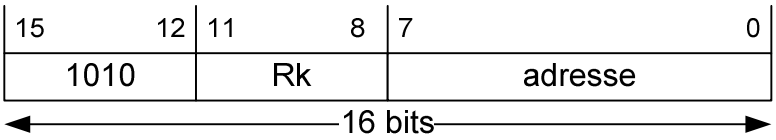
instruction	**choixSource**	**choixCharge**	**charge**	**choixA**	**choixB**	**opération**	**lectureEcritue'**	**….**	**….**
Rk ← MEM[adresse] + Rk									

a. solution



b. solution : Le signal UAL_choixA a été rajouté. Il sera actif lorsque l'instruction sera décodée, sinon par défaut il sera égal à '0'.

c. solution



d. solution

instruction	choixSource	choixCharge	charge	choixA	choixB	opération	lectureEcritue'	UAL_choixA
Rk <– MEM[] + Rk	0	0	1	-	4	0	1	1

Question 4. (8 points)

Soit le circuit suivant.

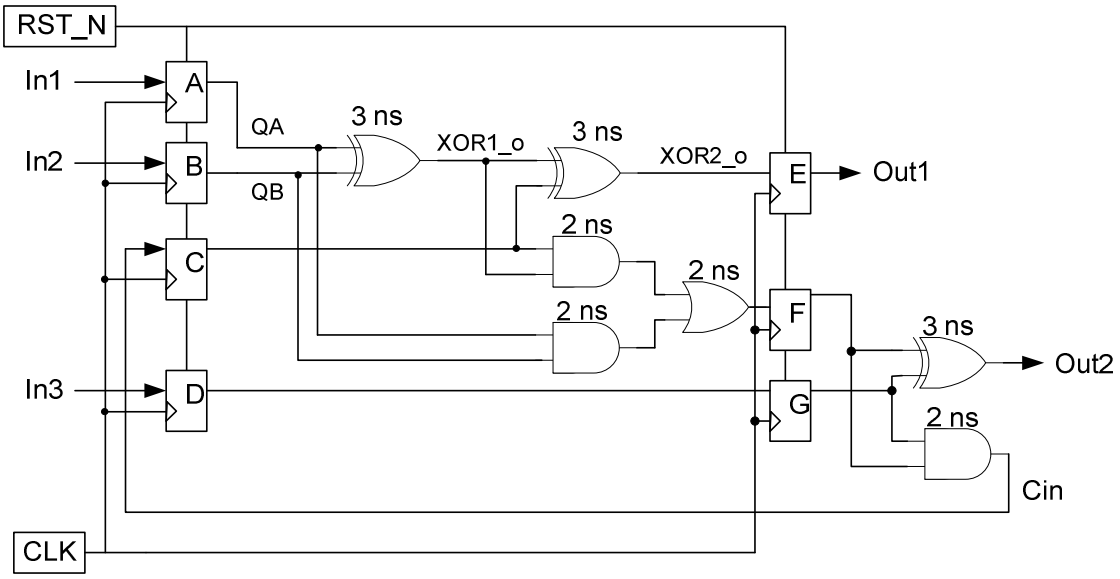


Figure 3. Circuit numérique.

En sachant que les contraintes de temps sont les suivantes :
- 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 ont un délai de propagation de 2 ns
- Les portes logiques XOR ont un délai de propagation de 3 ns.

Les bascules du circuit ont un reset asynchrone.

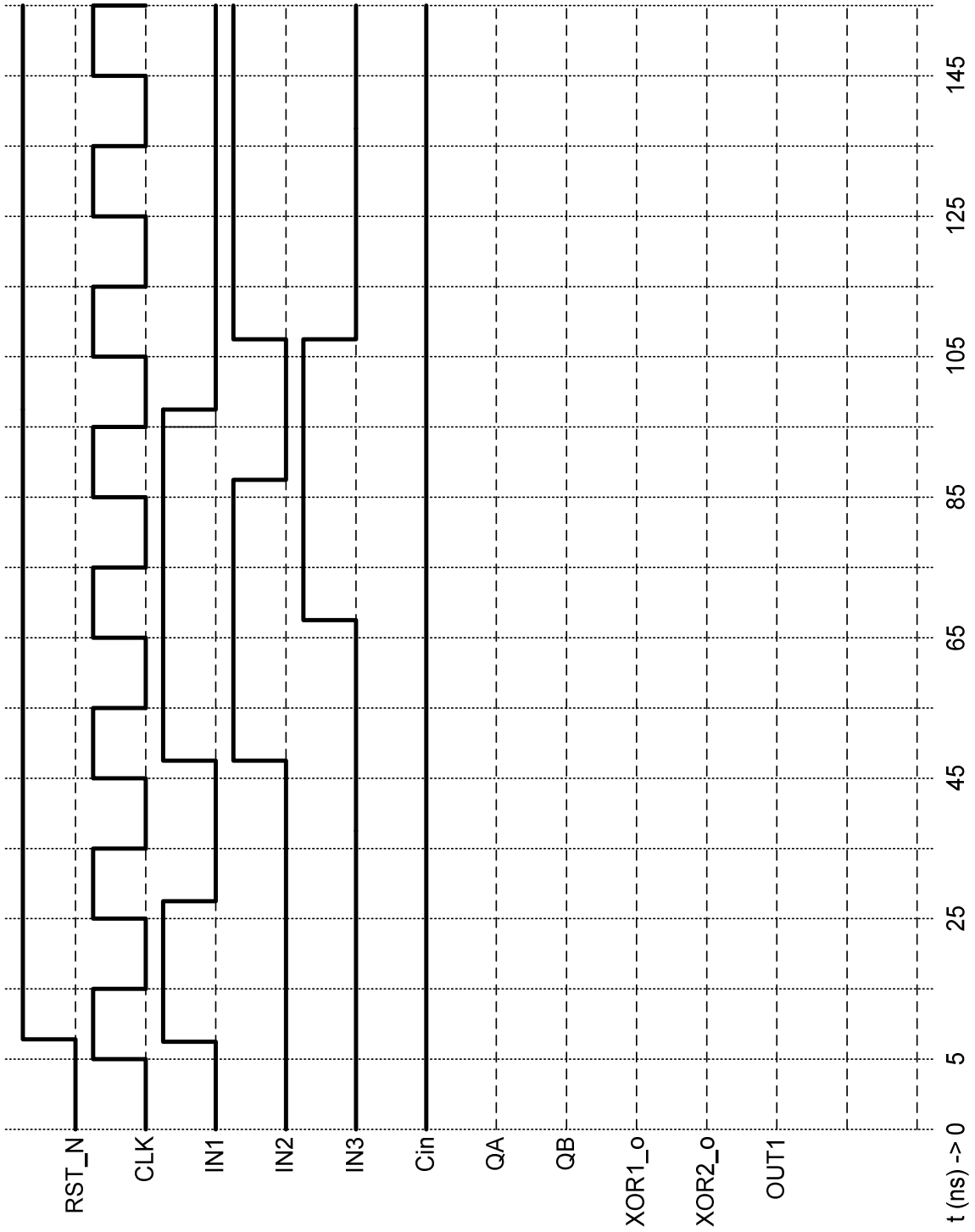
Vous devez :

a) En tenant compte des délais, compléter le chronogramme donné à la page suivante afin de fournir les variations de la sortie OUT1 jusqu'au temps t = 150 ns (Remettre la page du chronogramme avec votre cahier d'examen) (4 points).

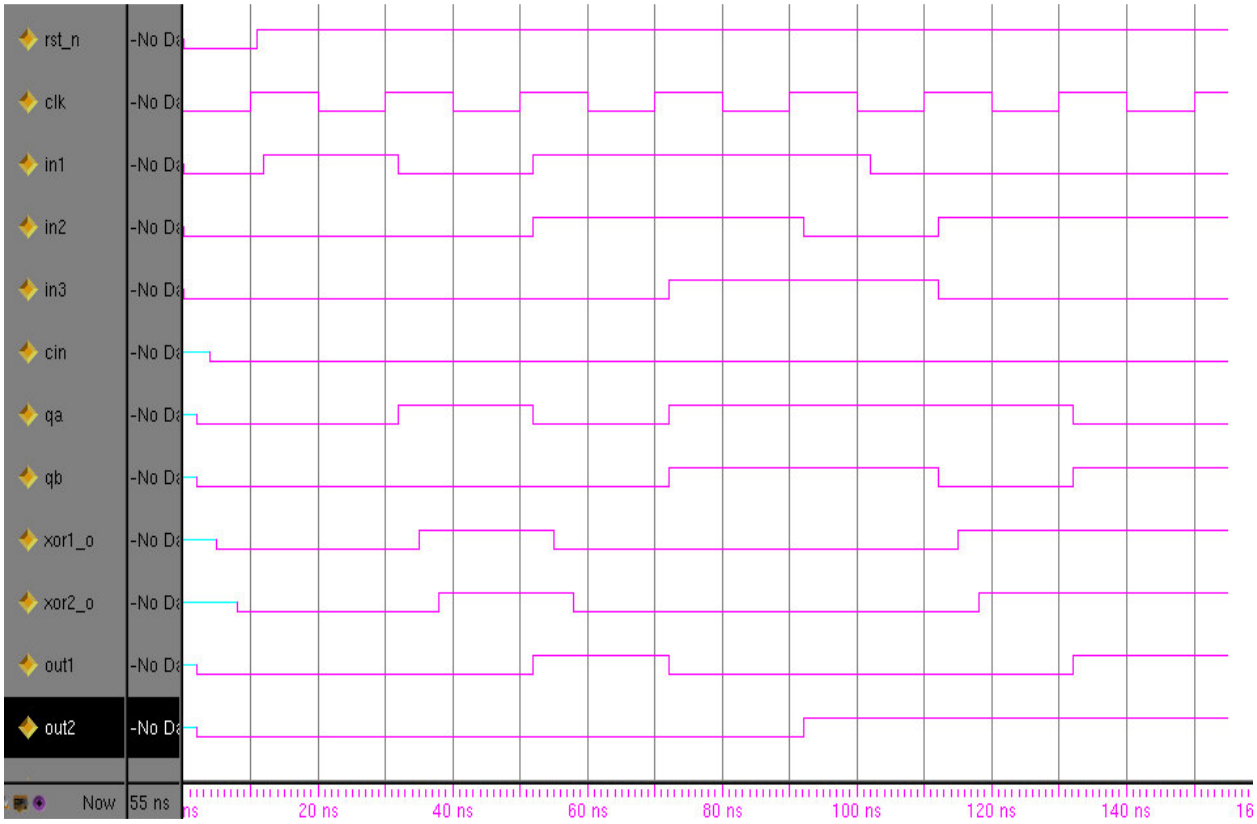
b) Identifier le chemin critique et calculer la fréquence d'horloge maximale (4 points).

Nom: _____ **Matricule:** _____

Question 4 a (compléter et remettre avec votre cahier)



Solution
a.



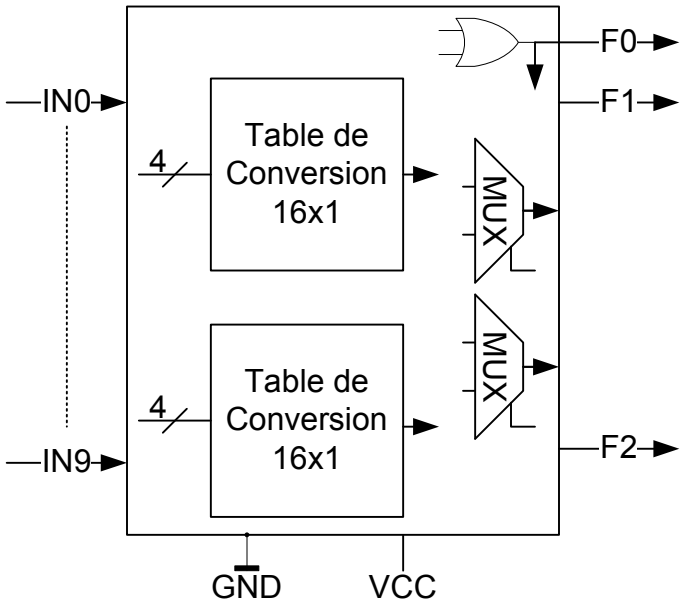
b.

Chemin critique : A/B -> XOR -> ET -> OR -> F

Fréquence d'horloge :
$T = t_dA + t_{comb}XOR + t_{comb}ET + t_{comb}OR + t_{setup}F = 2 + 3 + 2 + 2 + 1 = 10$ ns.
Donc, $Fmax = 1/10^{-9} = 100$ MHz.

Question 5. (4 points)

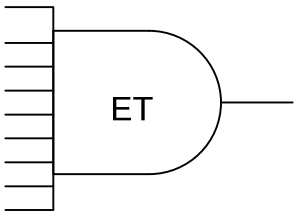
Soit le bloc de logique programmable (CLB) suivant :



Vue simplifiée d'un CLB.

Dans ce CLB, on peut programmer 2 tables de conversion, 2 multiplexeurs et une porte logique OU. La sortie F0 de la porte OU peut être connectée à l'entrée d'un autre CLB ou connectée aux autres sorties du CLB : F1 et F0. Les entrées IN0 à IN9, GND et VCC peuvent être connectées à tous les 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 et celles des multiplexeurs peuvent être connectées à une des sorties F1 et F2.

Utilisez ce patron de CLB pour concevoir une porte ET à 8 entrées illustrée par la figure suivante. Montrez les connexions internes et externes entres les différents éléments des CLB et entre les entrées/sorties des CLB que vous utiliserez. Précisez la fonction logique que chacune des tables de conversion effectuera.



Porte ET à 8 entrées.

Solution :

Solution proposée par Xilinx pour le Virtex II Pro. Plusieurs solutions sont possibles.