Question 1. (2.5 points)

Considérez le processeur d'usage général présenté dans les notes de cours. L'architecture de son chemin de données est illustrée à la Figure 1. Les extraits du code VHDL donnés ci-après rappellent le fonctionnement de l'ALU et de la mémoire.

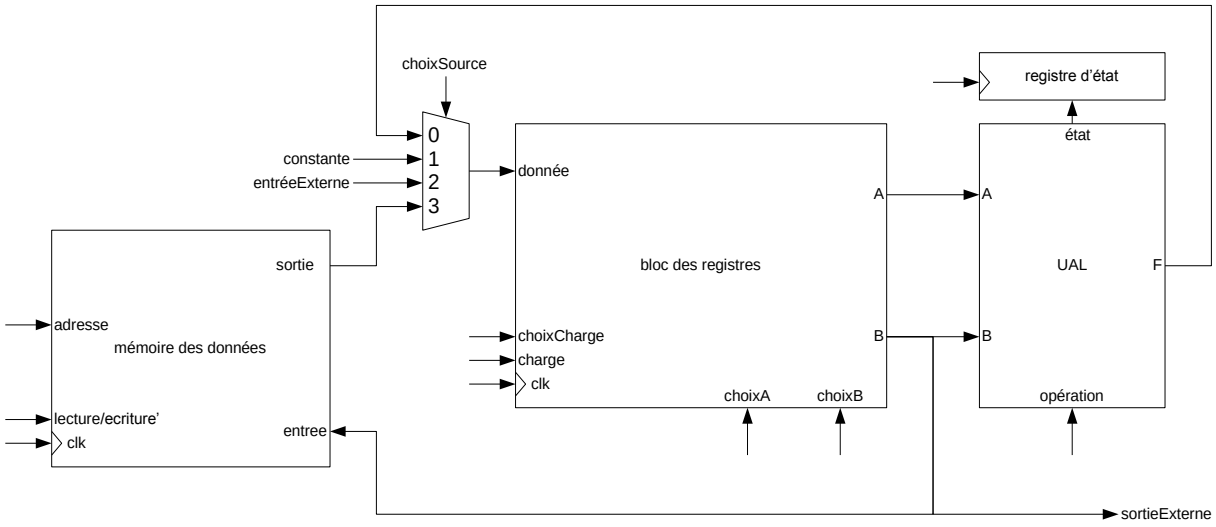


Figure 1

```
-- dans la partie déclarative
signal F : signed(Wd - 1 downto 0);
signal Z : std_logic;
signal N : std_logic;
signal op : integer range 0 to 7;

type memoireDonnees_type is array(0 to 2 ** Md - 1) of signed(Wd - 1 downto 0);
signal memoireDonnees : memoireDonnees_type;
signal sortieMemoireDonnees : signed(Wd - 1 downto 0);
signal adresseMemoireDonnees : integer range 0 to 2 ** Md - 1;
signal lectureEcritureN : std_logic;

-- dans le corps de l'architecture
process(A, B, op)
begin
    case op is
        when 0 => F <= A + B;
        when 1 => F <= A - B;
        when 2 => F <= shift_right(A, 1);
        when 3 => F <= shift_left(A, 1);
        when 4 => F <= not(A);
        when 5 => F <= A and B;
        when 6 => F <= A or B;
        when 7 => F <= A;
        when others => F <= (others => 'X');
    end case;
end process;


process (CLK)
begin
    if rising_edge(CLK) then
        if lectureEcritureN = '0' then
            memoireDonnees(adresseMemoireDonnees) <= B;
        end if;
    end if;
end process;


sortieMemoireDonnees <= memoireDonnees(adresseMemoireDonnees);
```

a. Complétez la table 1 suivante, en donnant les instructions selon les valeurs des signaux de contrôle indiquées.

Instruction	choixSource	choixCharge	charge	choixA	choixB	opération	Lecture/Écriture
R2<-R4	0	2	1	4	-	7	1
M[addr] <-R2	-	-	0	-	2	-	0
R3<-R1+R4	0	3	1	1	4	0	1
R4<-M[addr]	3	4	1	-	-	-	1

Table 1

Ajoutez une instruction de la forme Rk _ #WXYZ, permettant de charger le registre *k* avec une valeur immédiate de 16 bits de large (WXYZ représente 4 chiffres hexadécimaux). Cette instruction nécessite deux mots de 16 bits dans la séquence des instructions : le premier représente l'instruction (chargement de valeur immédiate de 16 bits) et le deuxième contient la valeur WXYZ.

b. Modifiez le diagramme d'états de l'unité de contrôle donné à l'**Annexe I** pour ajouter cette nouvelle instruction.

Après l'état de décodage il faut ajouter un état pour exécuter la nouvelle instruction.

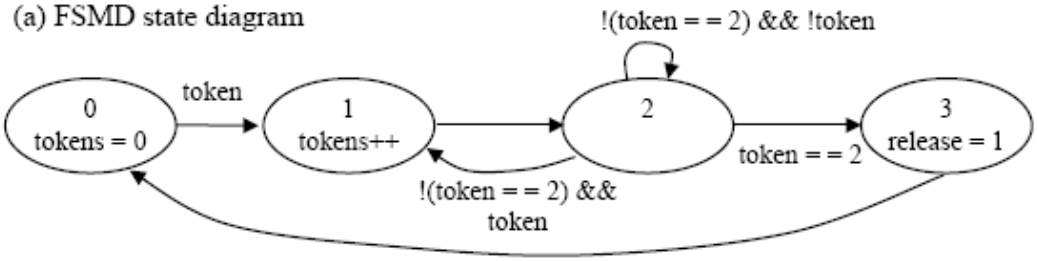
c. Donnez l'encodage de votre nouvelle instruction. Au besoin, inspirez-vous de la Table 2.

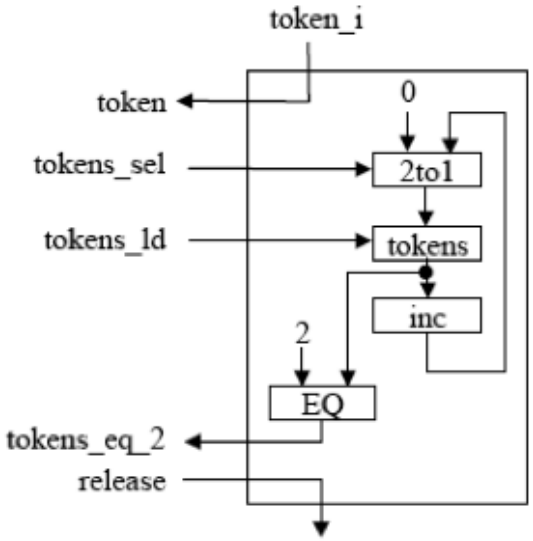
instruction	bits 15-12	bits 11-8	bits 7-4	bits 3-0
$R_{dest} \leftarrow R_{source1} \Diamond R_{source2}$	$\{0 \mid op_2 \mid op_1 \mid op_0\}$	destination	source1	source2
$R_{dest} \leftarrow$ MD[adresse]	1000	destination	adresse[7:4]	adresse[3:0]
MD[adresse] $\leftarrow R_{source}$	1001	source	adresse[7:4]	adresse[3:0]
JUMP adresse	1100	condition	adresse[7:4]	adresse[3:0]
STOP	1111	-	-	-

Table 2

Une solution : 1010 | destination | - | -

Question2. (3 points) Un système embarqué équipé d'un processeur à but spécifique contrôle l'entrée dans un garage. L'entrée est permise si deux jetons sont déposés par les conducteurs. Donnez le schéma du chemin de donné et la machine à états de l'unité de contrôle de ce système. Le système a une entrée qui indique la présence d'un jeton et un signal de sortie pour accorder l'accès.





Question 3. (4 points)

Soit le circuit représenté à l'**Annexe II**. Les bascules ont un temps de préparation de 1 ns, un temps de maintien de 0.5 ns et un délai de propagation de 2 ns. Le temps de propagation dans les portes ET, OU, XOR et INV de 3 ns, 3 ns, 4 ns, 1 ns respectivement. Négligez les délais des interconnexions.

a. Déterminez la fréquence maximale d'horloge.

Pour déterminer la période minimale de chaque chemin: T = tcomb+tsu+td
A vers F: T=3+3+2+1 = 9ns
B vers F: T=1+1+3+2+1= 8ns
B vers G: T=4+4+1+2+1= 12 ns
C vers G: T=3+3+4+1+2+1= 14 ns
D vers G: T=3+3+4+1+2+1= 14 ns
E vers G: T=3+3+4+1+2+1=14ns
F vers A: T=3+2+1=6ns
G vers A: T=2+3+1=6ns
Le chemin le plus long fait 14 ns. La fréquence maximale du circuit est donc 1/14ns= 71.4Mhz

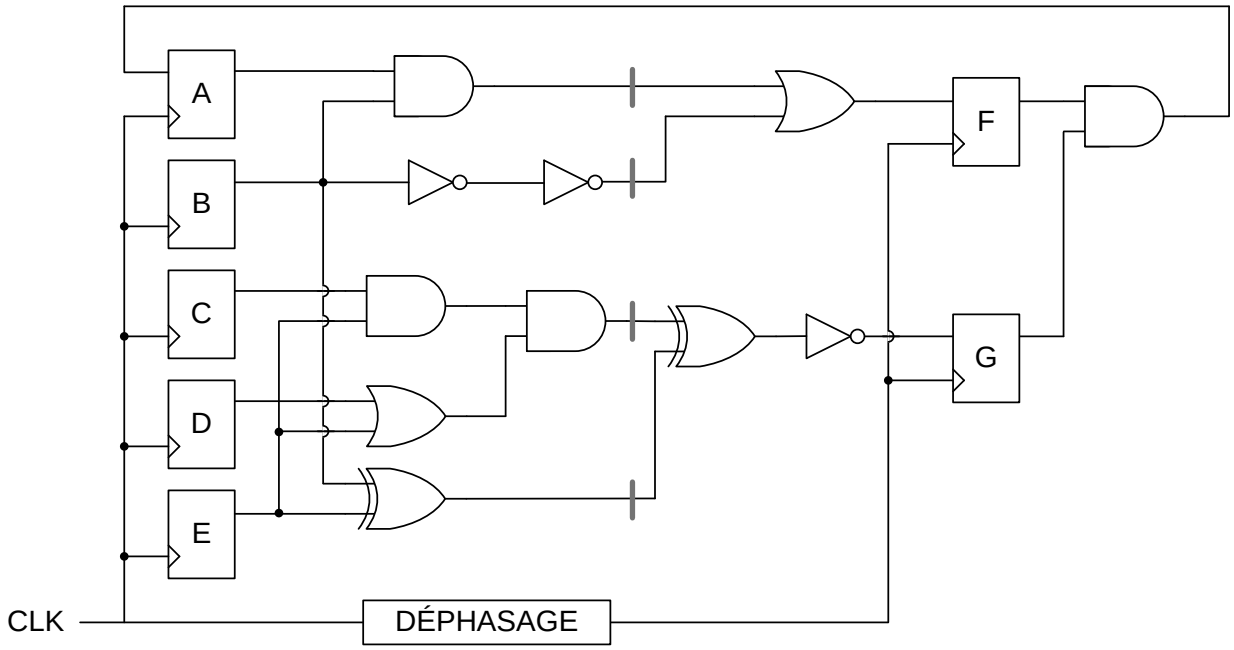
b. Déterminez la marge libre de préparation de chaque chemin.

A vers F: 5ns
B vers F: 6ns
B vers G: 2ns
C vers G: 0ns
D vers G: 0ns
E vers G: 0ns
F vers A: 8ns
G vers A: 8ns

c. Quelles sont les bornes minimales et maximales du déphasage d'horloge acceptables tout en garantissant cette fréquence maximale?

T>= tcomb+tsu+td-tcs d'où la borne tcs >= tcomb+tsu+td-T
tcomb +td-tcs>th d'où la borne tcs < tcomb+td-th
A vers F: -5ns<=tcs<7.5ns
B vers F: -6ns<=tcs<6.5ns
B vers G: -2 ns <= tcs <10.5ns
C vers G: 0ns<=tcs<12.5ns
D vers G: 0ns<=tcs<12.5ns
E vers G: 0ns<=tcs<12.5ns
F vers A: -8ns<=tcs<4.5ns
G vers A: -8ns<=tcs<4.5ns
Pour que le circuit continue de bien fonctionner, les bornes de déphasage acceptables sont: 0ns<=tcs<4.5ns

d. En considérant un déphasage d'horloge nul, modifiez le circuit donné à l'**Annexe II** de sorte à ajouter un étage de pipeline permettant d'augmenter la fréquence maximale. Déterminez la nouvelle fréquence maximale.



Question 4. (2.5 points)

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

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

entity module3 is
      port (
            A, B, C : in std_logic;
            F, G : out std_logic
      );
end module3;

architecture arch of module3 is
signal S, T, U : std_logic;
begin
      S <= A or B;
      T <= A xor B;
      U <= A and C;

      process (S, T)
      begin
            F <= S and T;
      end process;

      process (C)
      begin
            G <= C or U;
      end process;
end arch;

library ieee;
use ieee.std_logic_1164.all;

entity module3_TB is
end module3_TB;

architecture arch of module3_TB is
      signal A, B, C, F, G : std_logic;
begin
      UUT : entity module3(arch) port map (A, B, C, F, G);
      A <= '1' after 0 ns;
      B <= '0' after 0 ns;
      C <= '0' after 0 ns, '1' after 10 ns;
end arch;
```

Donnez la liste des événements dressée par un simulateur qui exécute le banc d'essai.

À t = 0 + 0 Δ, initialisation de la simulation (tout à 'U')
À t = 0 + 1 Δ, assignation de 100 à (A, B, C)
À t = 0 + 2 Δ, évaluation des signaux S, T, U et G (provoquées par les changements sur A, B, C)
À t = 0 + 3 Δ, évaluation de F (provoquées par les changements sur S, T, U) ** mais pas d'évaluation de F parce que U n'est pas dans la liste de sensitivité du processus **
Aucun nouvel événement n'est ajouté à la liste d'événements.
À t = 10 ns + 0 Δ, assignation de 1 à C
À t = 10 ns + 1 Δ, évaluation de U (provoquée par le changement sur C),
Aucun nouvel événement n'est ajouté à la liste d'événements.

b. Discutez l'utilisation des boucles et conditions dans un code VHDL synthétisable.

•À l'intérieur d'un processus, on peut utiliser des boucles et des conditions pour modéliser le comportement d'un circuit.

•Les boucles sont une manière compacte de représenter plusieurs énoncés reliés logiquement entre eux.

•Les paramètres d'exécution de la boucle doivent prendre des valeurs statiques au moment de la synthèse.

•Les boucles sont implémentées en les déroulant: les énoncés d'assignation qu'elles contiennent sont répliqués, un pour chaque itération de la boucle.

•Pour les circuits combinatoires, les conditions permettent d'effectuer un choix.

–L'énoncé case à l'avantage de représenter des choix qui sont mutuellement exclusifs et qui ont la même préséance. Il correspond assez exactement à l'action d'un multiplexeur.

–L'énoncé if, est plus général avec les clauses elsif ainsi qu'une clause else. Il est possible de l'utiliser pour donner préséance à certaines conditions par rapport à d'autres. Cela peut résulter en un circuit plus complexe que nécessaire, parce que le comportement décrit peut être plus restrictif que ce que le concepteur a en tête.

Question 5 (4 points)

Au laboratoire, vous avez eu à concevoir un processeur à usage général dont les instructions tiennent sur 28 bits et dont la déclaration d'entité suit. Le jeu d'instructions de ce processeur est donné à l'**Annexe III**.

```
entity processeurv20 is
    generic (
        Nreg        : integer   := 32; -- nombre de registres
        Wd          : integer   := 16; -- largeur du chemin des données en bits
        Wi          : integer   := 28; -- largeur des instructions en bits
        Mi          : integer   := 12; -- nombre de bits d'adresse de la mémoire d'instructions
        Md          : integer   :=  8; -- nombre de bits d'adresse de la mémoire des données
        resetvalue  : std_logic := '1'
    );
    port(
        reset               : in  std_logic;
        clk                 : in  std_logic;
        entreeExterne       : in  signed(Wd - 1 downto 0);
        fifoInNotReady      : in  std_logic;
        reqEntreeExterne    : out std_logic;
        sortieExterne       : out signed(Wd - 1 downto 0);
        fifoOutNotReady     : in  std_logic;
        ecrireSortieExterne : out std_logic
    );
end processeurv20;
```

a. Quelle sera la valeur du registre `r0` à la fin de l'exécution de ce programme ?

```
constant memoireInstructions : memoireInstructions_type :=
(x"A080018", -- r8    <- +24               / addr x00
 x"A090001", -- r9    <- +1                / addr x01
 x"A000000", -- r0    <- +0                / addr x02
 x"9000000", -- M[r0] <- r0                / addr x03
 x"0000009", -- r0    <- r0 + r9           / addr x04
 x"9000000", -- M[r0] <- r0                / addr x05
 x"0000009", -- r0    <- r0 + r9           / addr x06
 x"10A0008", -- rA    <- r0 - r8           / addr x07
 x"C040012", -- jump to addr x12 si N = '0' / addr x08
 x"1000009", -- r0    <- r0 - r9           / addr x09
 x"8020000", -- r2    <- M[r0]             / addr x0A
 x"1000009", -- r0    <- r0 - r9           / addr x0B
 x"8010000", -- r1    <- M[r0]             / addr x0C
 x"0000009", -- r0    <- r0 + r9           / addr x0D
 x"0000009", -- r0    <- r0 + r9           / addr x0E
 x"0030102", -- r3    <- r1 + r2           / addr x0F
 x"9000003", -- M[r0] <- r3                / addr x10
 x"C000006", -- jump to addr x06           / addr x11
 x"1000009", -- r0    <- r0 - r9           / addr x12
 x"8000000", -- r0    <- M[r0]             / addr x13
others => (others => '1'));
```

Le registre r0 aura la valeur 28 657.

b. Quel est le nombre de cycles d'horloge nécessaire à l'exécution du programme ? Justifiez clairement votre réponse en énonçant vos hypothèses de travail.

D'abord nous devons évaluer le nombre d'instructions qui seront exécutées. Les instructions x01 à x05 seront exécutées exactement 1 fois chacune car elles sont en dehors de la boucle du programme. Il en est de même pour les instructions x13 et x14. Les instructions x09 à x11 seront exécutées 22 fois. Les instructions x06 à x08 seront exécutées 23 fois (une fois de plus que les instructions de la boucle, la dernière intervenant quand le programme sort de la boucle). Nous avons par conséquent un total de 5 + 2 + 22(9) + 23(3) = 274 instructions. Chaque instruction nécessite 3 cycles d'horloge, on obtient 822 cycles d'horloge.

c. Décrivez dans un pseudo-code le plus lisible possible l'algorithme précédent. On supposera que l'algorithme retourne la valeur du registre `r0`.

```
M[0] ← 0;
M[1] ← 1;
Pour i = 2 à 23
        M[i] ← M[i-1] - M[i-2];
Fin Pour
return M[23];
```

d. Il n'est pas nécessaire pour exécuter ce programme de recourir à la mémoire de données. Proposez un programme qui exécute le même algorithme sans utiliser la mémoire de données. Vous êtes libre d'utiliser tous les registres à votre disposition mais vous devez absolument vous assurer que le résultat de votre programme se retrouve au registre `r0` à la fin de son exécution.

```
constant memoireInstructions : memoireInstructions_type :=
(x"A080018", -- r8    <- +24               / addr x00
 x"A090001", -- r9    <- +1                / addr x01
 x"A010000", -- r1    <- +0                / addr x02
 x"A000001", -- r0    <- +1                / addr x03
 x"A030002", -- r3    <- +2                / addr x04
 x"10A0308", -- rA    <- r3 - r8           / addr x05
 x"C04000C", -- jump to addr x0C si N = '0' / addr x06
 x"7020000", -- r2    <- r0                / addr x07
 x"0000100", -- r0    <- r1 + r0           / addr x08
 x"7010200", -- r1    <- r2                / addr x09
 x"0030309", -- r3    <- r3 + r9           / addr x0A
 x"C000005", -- jump to addr x05           / addr x0B
others => (others => '1'));
```

Question 6 (4 points)

On vous demande de réaliser un circuit permettant de trouver la valeur médiane de trois nombres signés donnés en entrée : a, b et c. Ainsi, si (a, b, c) = (+3, +8, +5), la sortie devrait être z = +5. Si (a, b, c) = (+3, +8, +3), la sortie devrait être z = +3. Si (a, b, c) = (+8, +8, +3), la sortie devrait être z = +8.

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

        entity mediane3 is
        generic(
            w : positive := 8
        );
        port(
            a : in signed(w-1 downto 0);
            b : in signed(w-1 downto 0);
            c : in signed(w-1 downto 0);
            z : out signed(w-1 downto 0)
        );
        end mediane3;
```

a) Écrivez le code VHDL synthétisable de l'entité **mediane3**.

```
architecture arch of mediane3 is
begin

    process( a, b, c ) is
        variable max_1 : signed(w-1 downto 0);
        variable max_2 : signed(w-1 downto 0);
    begin

        if( a > b ) then
            max_1 := a;
            max_2 := b;
        else
            max_1 := b;
            max_2 := a;
        end if;

        if( c >= max_1 ) then
            z <= max_1;
        elsif( c < max_2 ) then
            z <= max_2;
        else
            z <= c;
        end if;

    end process;

end arch;
```

b) Quelle est la plus petite taille de l'ensemble de vecteurs de test permettant d'effectuer un test exhaustif de l'entité **mediane3** lorsque w=8 ? Justifiez votre réponse par un calcul.

Comme il y a 3 entrées, l'ensemble contient au minimum $2^{3w} = 2^{24} \geq 16$ millions vecteurs de test.

On vous propose trois classes pour chacune des entrées du circuit **mediane3** (on suppose w≥2) :

$\{ \{-2^{w-1}, -2^{w-1} + 1, \ldots -2^{w-2} - 1\},$

$\{-2^{w-2}, \ldots +2^{w-2} - 1\},$

$\{+2^{w-2}, +2^{w-2}+1, \ldots +2^{w-1} - 1\} \}$

c) Quelle est la taille minimale de l'ensemble de vecteurs de test permettant d'effectuer un test faible sur l'entité **`mediane3`** et ce, en recourant aux classes identifiées précédemment. Justifiez votre réponse et proposez un exemplaire d'un tel vecteur de test pour w = 8.

Pour effectuer un test faible, il faut qu'au moins un représentant de chaque classe apparaisse une fois, et ce pour l'ensemble des entrées. Comme nous avons trois classes pour chaque entrée, l'ensemble permettant d'effectuer un test faible peut contenir au minimum 3 vecteurs tests.

Un exemple d'un tel ensemble est {{-127, 0, +127}, {0, +127, -127}, {+127, -127, 0}}

d) Quelle est la taille minimale de l'ensemble de vecteurs de test permettant d'effectuer un test fort sur l'entité **`mediane3`** et ce, en recourant aux classes identifiées précédemment. Justifiez votre réponse et proposez un exemplaire d'un tel vecteur de test pour w = 8.

Pour effectuer un test fort, il faut que toutes les combinaisons des classes de chaque entrée soient considérées au moins une fois. Comme nous avons trois classes pour chaque entrée, l'ensemble permettant d'effectuer un test fort peut contenir au minimum $3^3 = 27$ vecteurs de test.

Un exemple d'un tel ensemble est :

{{-127, -127, -127}, {-127, -127, 0}, {-127, -127, +127}, {-127, 0, -127}, {-127, 0, 0},
{-127, 0, +127},{-127, +127, -127}, {-127, +127, 0}, {-127, +127, +127}, {0, -127, -127}, {0, -127, 0},
{0, -127, +127}, {0, 0, -127}, {0, 0, 0}, {0, 0, +127},{0, +127, -127}, {0, +127, 0},
{0, +127, +127},{+127, -127, -127}, {+127, -127, 0}, {+127, -127, +127}, {+127, 0, -127}, {+127, 0, 0},
{+127, 0, +127},{+127, +127, -127}, {+127, +127, 0}, {+127, +127, +127}}

e) Quelle est la couverture de branchement de l'implémentation proposée à la question (a) en utilisant l'ensemble de vecteurs de test proposé telle qu'à la question (d). Justifiez adéquatement votre réponse.

Comme on considère un test fort, la couverture de branchement sera de100 %.