Nom : _____ Matricule : _____

INF3500 : Conception et réalisation de systèmes numériques

Examen intra #2 – jeudi 12 mars 2015

Durée: 1 heure. Documentation: Une feuille recto verso 8.5"×11" ou A4 permise.
Pondération: 10%. Calculatrice: Programmable permise.
Directives particulières:

- Ordinateurs interdits. Appareils mobiles interdits.
- Répondre à toutes les questions, la valeur de chaque question est indiquée.
- Répondre sur le questionnaire et le remettre.
- 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)

a. (1 point) Combien de tranches de type L/M d'un FPGA Virtex-5 sont-elles nécessaires pour implémenter les opérations suivantes? Pourquoi?

Opération	# Tranches	Brève explication
Soustraction de deux nombres de 14 bits		
Multiplication par 7 d'un nombre de 8 bits		
Valeur absolue d'un nombre (signé) de 24 bits		
Modulo 16 d'un nombre de 32 bits		
Division par 16 d'un nombre de 64 bits		

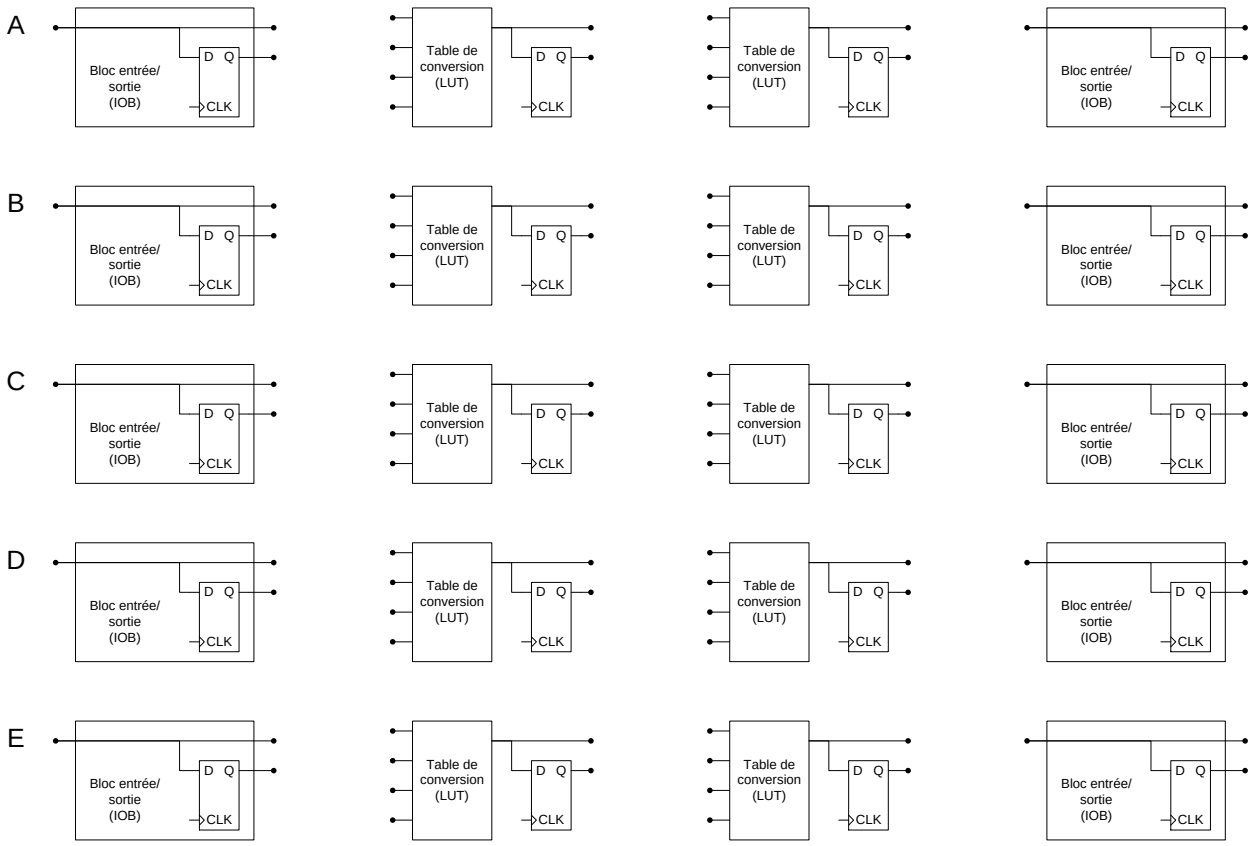
b. (1 point) Donnez la représentation binaire signée des quantités suivantes.

Quantité	Partie entière (4 bits)	Partie fractionnaire (8 bits)
–0,1	,	
37 / 7	,	
–1 / 6	,	
2π	,	
√20	,	

Question 2. (3 points)

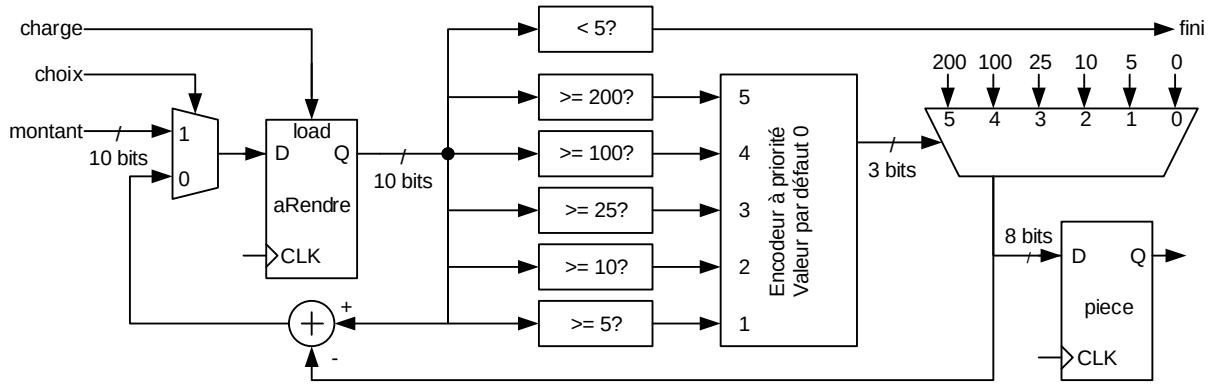
Considérez le code VHDL et le modèle de FPGA suivants. Montrez, sur le modèle du FPGA, un résultat possible de la synthèse et de l'implémentation de ce code. Indiquez directement sur le dessin où chaque signal et port de sortie se situe ainsi que les interconnexions entre les blocs. Les interconnexions peuvent contourner les blocs. Indiquez quand une bascule doit être utilisée. Indiquez par une équation la fonction logique réalisée par chaque LUT que vous utilisez. Respectez l'assignation donnée pour les ports d'entrée.

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

entity module10 is
   port (
      clk, A, B, C, D, E: in std_logic;
      X, Y, Z: out std_logic
   );
end module10;

architecture arch of module10 is

signal F, G, H : std_logic;

begin

   X <= not(A and B and E);
   Y <= G xor H;

   process(clk) is
   begin
      if rising_edge(CLK) then
         F <= A and B and C and D;
         G <= F xor E;
         H <= B or C or D;
      end if;
   end process;

   process(A, B, C)
   begin
      if A = '1' then
         Z <= B or C;
      else
         Z <= B and C;
      end if;
   end process;

end arch;
```



Question 3. (3 points)

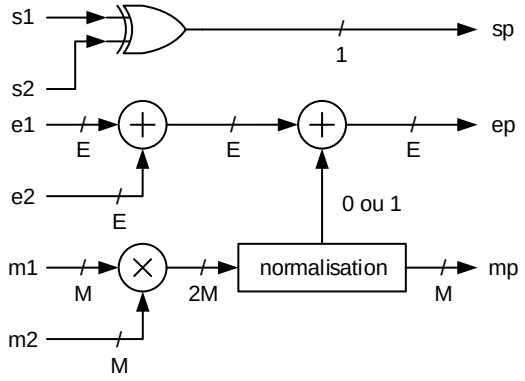
Le chemin des données suivant fait partie du processeur d'une machine distributrice pour rendre la monnaie en pièces de 2$, 1$, 25¢, 10¢ et 5¢. Au départ, on charge le registre avec le montant à rendre. Ensuite, à chaque coup d'horloge on compare le montant restant avec les valeurs des pièces. La plus grande valeur de pièce possible est choisie. Sa valeur est soustraite du montant restant. Quand la valeur est inférieure à 5¢, aucune pièce n'est rendue et le signal fini est activé. Donnez une déclaration d'entité et une architecture en VHDL synthétisable correspondant à cette entité et à ce diagramme. Énoncez clairement vos suppositions.



(*Utilisez le verso si nécessaire*)

Question 4. (2 points)

Le diagramme et le code VHDL suivants décrivent un circuit permettant d'effectuer la multiplication de nombres à virgule flottante. Considérez le cas E = 7 et M = 8.




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

entity mul_uvf is
   generic (
      E : natural := 7;
      M : natural := 8
   );
   port(
      s1, s2 : in std_logic;
      sp : out std_logic;
      e1, e2 : in signed(E - 1 downto 0);
      ep : out signed(E - 1 downto 0);
      m1, m2 : in unsigned(M - 1 downto 0);
      mp : out unsigned(M - 1 downto 0)
   );
end mul_uvf;
```

a. (0.5 point) Combien de vecteurs de test sont nécessaires pour une vérification exhaustive de cette entité?

b. (1 point) Pour chacune des entrées, donnez les valeurs à tester selon l'analyse des valeurs limites.

Entrées	Valeurs à tester
s1 et s2	
e1 et e2	
m1 et m2	

c. (0.5 point) Combien de vecteurs de tests seraient nécessaires pour effectuer une vérification selon l'analyse des valeurs limites?

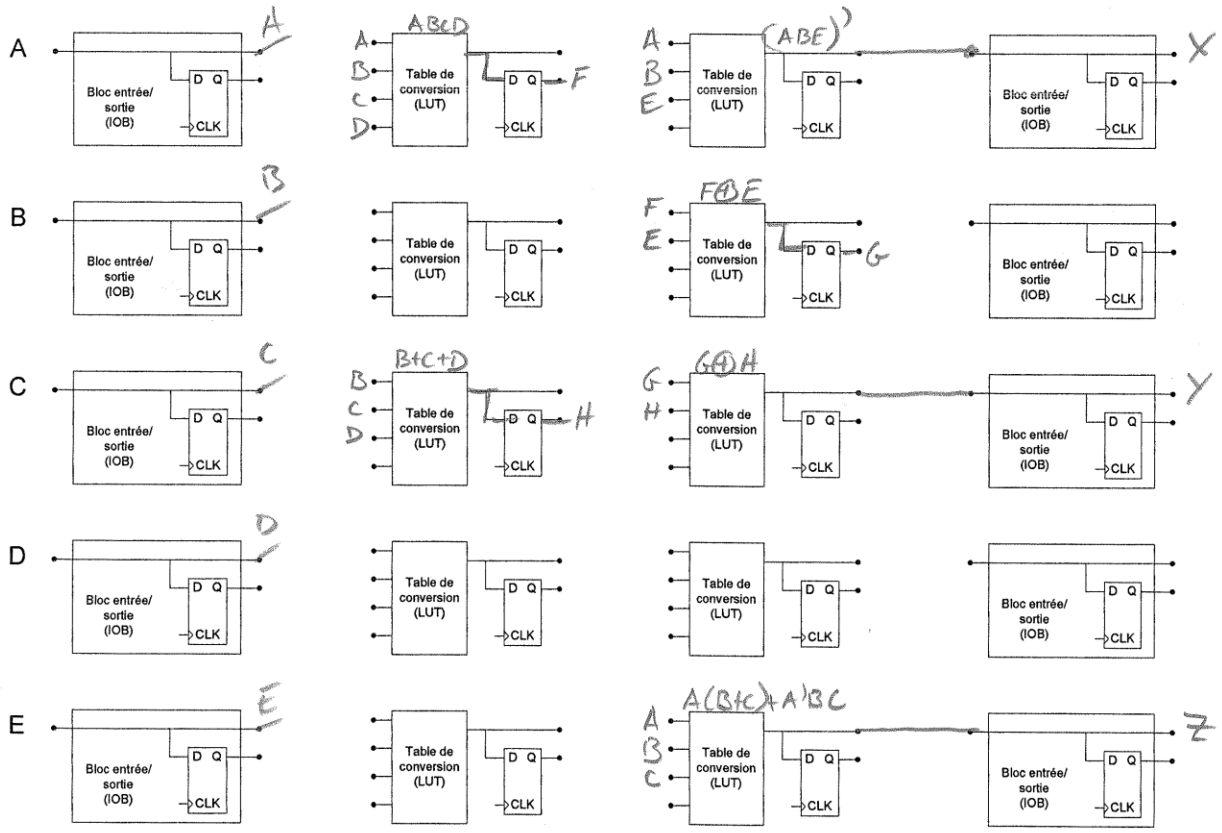
Solutions

#1 Réponse

Opération	# Tranches	Brève explication	
Soustraction de deux nombres de 14 bits	14 / 4 = 3.5	Une LUT et le circuit d'addition qui lui est joint peut faire l'addition ou la soustraction de deux nombres de 1 bit. 4 LUT par tranche. Donc 14 / 4 = 3.5 tranches.	
Multiplication par 7 d'un nombre de 8 bits	11 / 4 = 2,75	On a B × 7 = B << 3 – B On constate que la soustraction se fait sur 11 bits.	$\begin{array}{r} 1 \\ b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0 000 \\ \underline{+\,111 \bar{b}_7 \bar{b}_6 \bar{b}_5 \bar{b}_4 \bar{b}_3 \bar{b}_2 \bar{b}_1 \bar{b}_0} \end{array}$
Valeur absolue d'un nombre (signé) de 24 bits	24 / 4 = 6	La valeur absolue peut nécessiter l'inversion du signe, qui correspond à une inversion des bits et l'ajout de 1.	
Modulo 16 d'un nombre de 32 bits	0	Le modulo 16 consiste à extraire les 4 bits de poids faible du nombre. Aucun calcul nécessaire. Sortie <= entrée(3 downto 0);	
Division par 16 d'un nombre de 64 bits	0	La division par 16 consiste à décaler le nombre de 4 positions vers la droite. Aucun calcul nécessaire. Sortie <= "0000" & entrée(63 downto 4).	

Quantité	Partie entière (4 bits)	Partie fractionnaire (8 bits)
–0,1	1111,	1110 0110 (-26 / 256)
37 / 7	0101,	0100 1001 (73 / 256)
–1 / 6	1111,	1101 0101 (-43 / 256)
2π	0110,	0100 1000 (72 / 256)
$\sqrt{20}$	0100,	0111 1001 (121 / 256)

#2 Réponse


A
B
C
D
E
Bloc entrée/ sortie (IOB)
D Q
CLK
Table de conversion (LUT)
A
B
C
D
ABCD
F
A
B
E
(ABE)'
X
F
E
FBE
G
B
C
D
B+C+D
H
G
H
GBH
Y
A
B
C
A(B+C)+A'BC
Z

#3 Solution

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

entity monnaie is
   port (
      clk, reset : in std_logic;
      charge, choix : in std_logic;
      montant : in unsigned(9 downto 0);
      fini : out std_logic;
      piece: out unsigned(7 downto 0)
   );
end monnaie;

architecture arch of monnaie is

begin

   process (clk, reset)
   variable aRendre : unsigned(9 downto 0);
   begin
      if reset = '0' then
         aRendre := to_unsigned(0, 10);
         piece <= to_unsigned(0, 8);
      elsif rising_edge(clk) then
         if charge = '1' then
            if choix = '1' then
               aRendre := montant;
            else
               if aRendre >= 200 then
                  aRendre := aRendre - 200;
                  piece <= to_unsigned(200, 8);
               elsif aRendre >= 100 then
                  aRendre := aRendre - 100;
                  piece <= to_unsigned(100, 8);
               elsif aRendre >= 25 then
                  aRendre := aRendre - 25;
                  piece <= to_unsigned(25, 8);
               elsif aRendre >= 10 then
                  aRendre := aRendre - 10;
                  piece <= to_unsigned(10, 8);
               elsif aRendre >= 5 then
                  aRendre := aRendre - 5;
                  piece <= to_unsigned(5, 8);
               else
                  aRendre := aRendre - 0;
                  piece <= to_unsigned(0, 8);
               end if;
            end if;
         end if;
         if aRendre < 5 then
            fini <= '1';
         else
            fini <= '0';
         end if;
      end process;

      end arch;
```

#4

a. Il y a 2 × 16 = 32 bits en entrée, donc 2^{32} vecteurs de test possibles.

b. Réponses :

Pour les entrées s1 et s2, {0, 1}.

Pour les entrées m1 et m2, {0, 1, 128, 254, 255} ou bien : {00000000, 00000001, 10000000, 11111110, 11111111} ; (si on respecte l'esprit de IEEE 754 :{1.0000000, 1.0000001, 1.1000000, 1.1111110, 1.1111111}).

Pour les entrées e1 et e2, {-64, -63, 0, 62, 63} exprimés sur 7 bits.

c. Il faudrait prendre 5 (pour e1) × 5 (pour e2) × 5 (pour m1) × 5 (pour m2) × 2 (pour s2) × 2 (pour s1) = 2500 vecteurs de test.