Nom : _____ Matricule : _____

INF3500 : Conception et réalisation de systèmes numériques

Examen intra #2 – jeudi 6 novembre 2014

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.

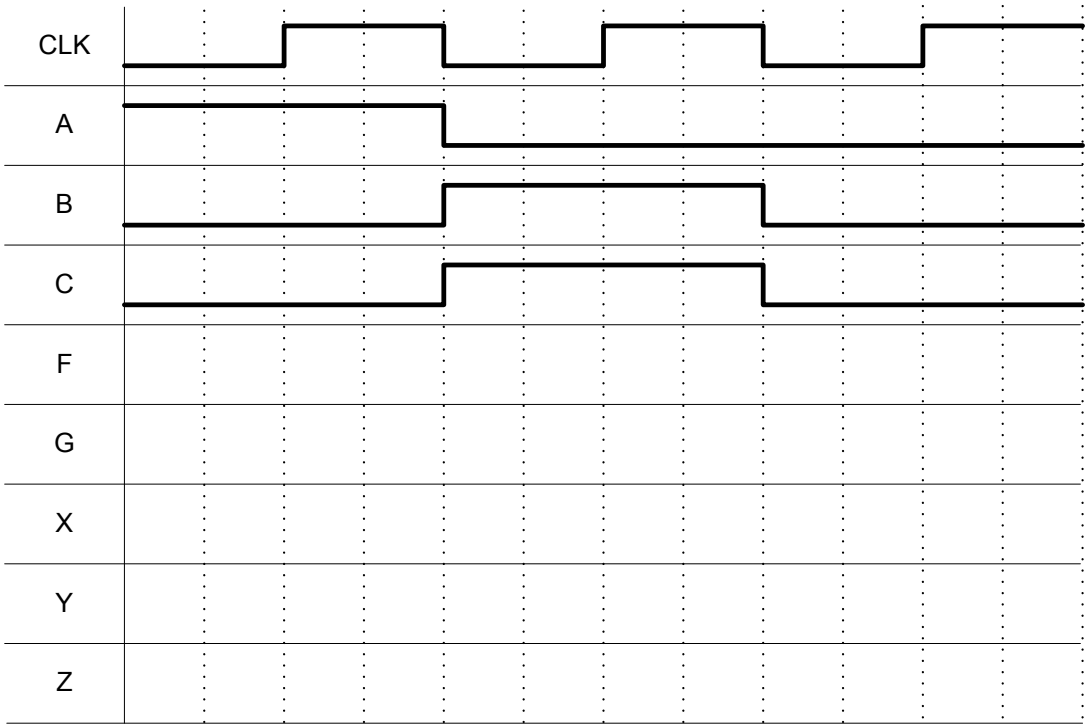
Question 1. (3 points)

Considérez le code VHDL suivant.

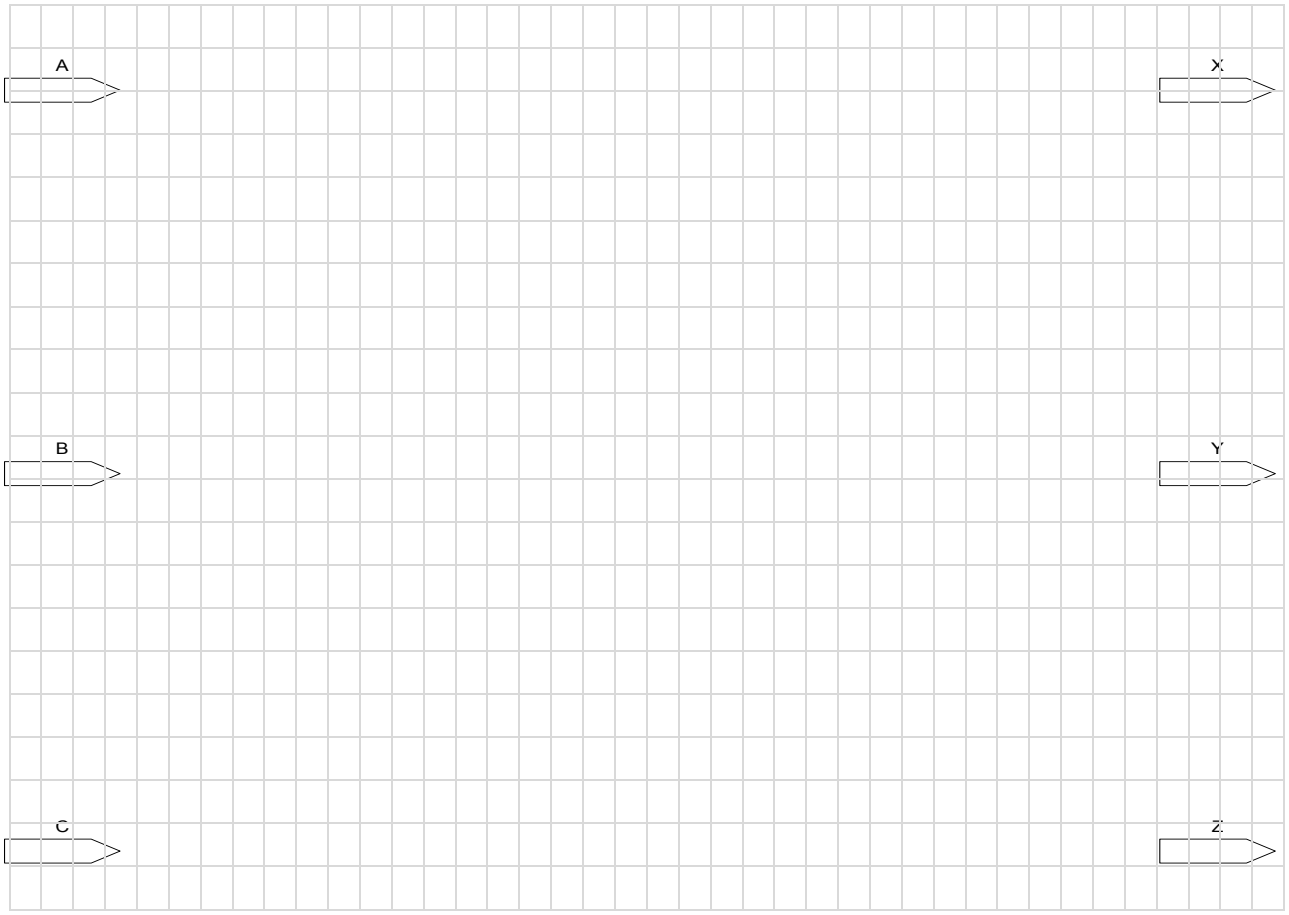
```
library ieee;
use ieee.std_logic_1164.all;

entity module9 is
   port (
      clk: in std_logic;
      A, B, C: in std_logic;
      X, Y, Z: out std_logic
   );
end module9;

architecture arch of module9 is

signal F, G : std_logic;
```

```
Begin

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

   X <= F or A;

   process(F, G)
   begin
      Y <= F and G;
   end process;

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

end arch;
```

a. (1.5 point) Complétez le chronogramme suivant pour ce module.



b. (1.5 point) Donnez un schéma d'éléments à mémoire et de portes logiques correspondant à ce code.



Question 2. (3 points)

La méthode de division par multiplications successives, ou méthode de Goldschmidt, consiste à multiplier le dividende N et le diviseur D par une suite de nombres X_k. Les nombres X_k sont choisis afin que le produit $D \times X_1 \times X_2 \times X_3 \ldots$ converge vers 1. Le produit $N \times X_1 \times X_2 \times X_3 \ldots$ converge alors vers Q.

$$Q = \frac{N}{D} \times \frac{X_1}{X_1} \times \frac{X_2}{X_2} \times \frac{X_3}{X_3} \times \ldots \times \frac{X_k}{X_k} = \frac{Q}{1}$$

À chaque étape on calcule une nouvelle valeur du numérateur *N* et du dénominateur *D.* Avec D_0 dans l'intervalle [0.5, 1] (suite à une normalisation préalable de N et D), on a, à chaque étape :

$$\frac{N_{k+1}}{D_{k+1}} = \frac{N_k}{D_k} \times \frac{X_{k+1}}{X_{k+1}} \qquad X_{k+1} = 2 - D_k$$

On peut décrire l'algorithme avec les micro-opérations suivantes où N0, D0 et init sont des ports d'entrée du circuit. On suppose que la valeur D0 est déjà dans l'intervalle [0.5, 1].

init : k ← 0; init' ET k ≠ 6: k ← k + 1;

init : N ← N0; init' ET k ≠ 6: N ← N × (2 – D)

init : D ← D0; init' ET k ≠ 6: D ← D × (2 – D)

Donnez le diagramme d'un chemin des données correspondant à ces micro-opérations.

Question 3. (2 points)

Considérez le code VHDL suivant. Pour chacun des signaux en sortie, estimez le nombre de ressources nécessaires en termes de tables de conversion (LUT), bascules (FF) et tranches DSP48 pour l'implémenter sur un FPGA de la famille Virtex 5 de Xilinx. Justifiez complètement votre réponse.

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

entity ual_ressources is
   generic (
      W : positive := 8
   );
   port (
      clk : in std_logic;
      A, B : in signed(W - 1 downto 0);
      F, H, J : out signed(W - 1 downto 0);
      K : out signed(W + 2 downto 0);
      L : out signed(2 * W - 1 downto 0)
   );
end ual_ressources;
```

```
architecture arch of ual_ressources is
begin

   process(clk)
   begin
      if rising_edge(clk) then
         F <= A + B;
         H <= abs(A);
         J <= A / 4;
      end if;
   end process;

   K <= A * 5;
   L <= A * B;

end arch;
```

Port	#LUT	#FF	#DSP48E	Justification
F				
H				
J				
K				
L				

Question 4. (2 points)

Considérez le code VHDL suivant.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity machineAEtats is
   port (
      reset, CLK : in STD_LOGIC;
      x : in STD_LOGIC_VECTOR(1 downto 0);
      sortie : out STD_LOGIC
   );
end machineAEtats;

architecture arch2 of machineAEtats is
type type_etat is (S1, S2, S3, S4);
signal etat : type_etat := S1;
begin

   process(etat)
   begin
      case etat is
         when S1 | S3 =>
            sortie <= '1';
         when S2 | S4 =>
            sortie <= '0';
      end case;
   end process;
```

```
   process(CLK, reset) is
   begin
       if (reset = '0') then
          etat <= S1;
       elsif (rising_edge(CLK)) then
          case etat is
             when S1 =>
                if x = "00" then
                   etat <= S3;
                else
                   etat <= S2;
                end if;
             when S2 =>
                   etat <= S1;
             when S3 =>
                if x = "11" then
                   etat <= S2;
                else
                   etat <= S4;
                end if;
             when S4 =>
                etat <= S1;
          end case;
       end if;
   end process;

end arch2;
```

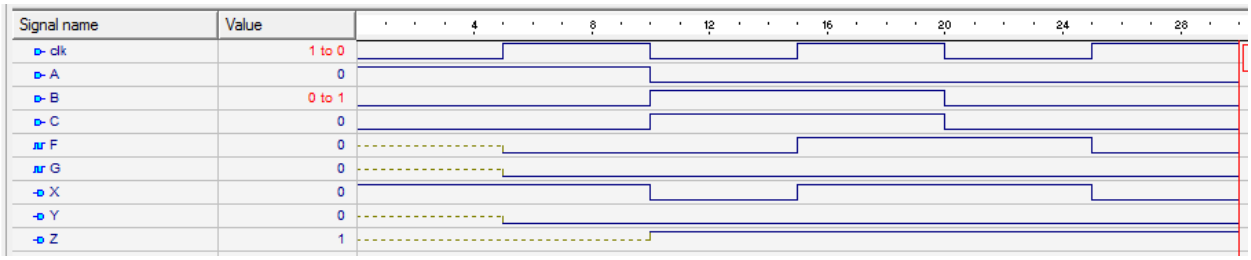
a. (0.5 point) Combien de vecteurs de test seraient nécessaires pour effectuer un test exhaustif ? Justifiez complètement votre réponse et montrez tous vos calculs.

b. (1.5 point) Proposez un ensemble de vecteurs de test à appliquer au port x qui maximise le nombre d'énoncés couverts. L'ensemble devrait être le plus petit possible. Justifiez votre réponse.

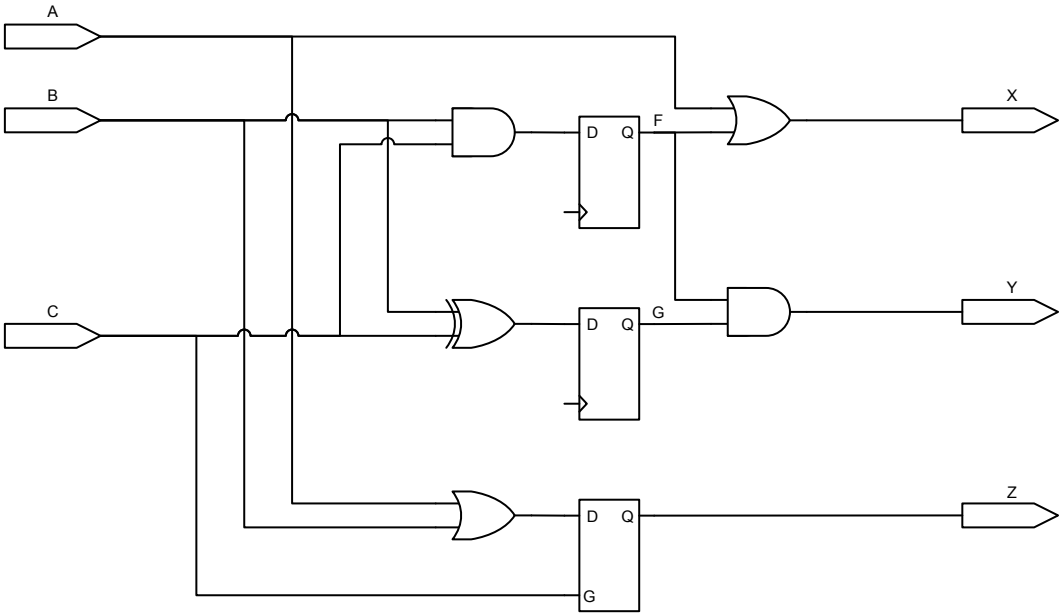
Solutions

Q1.

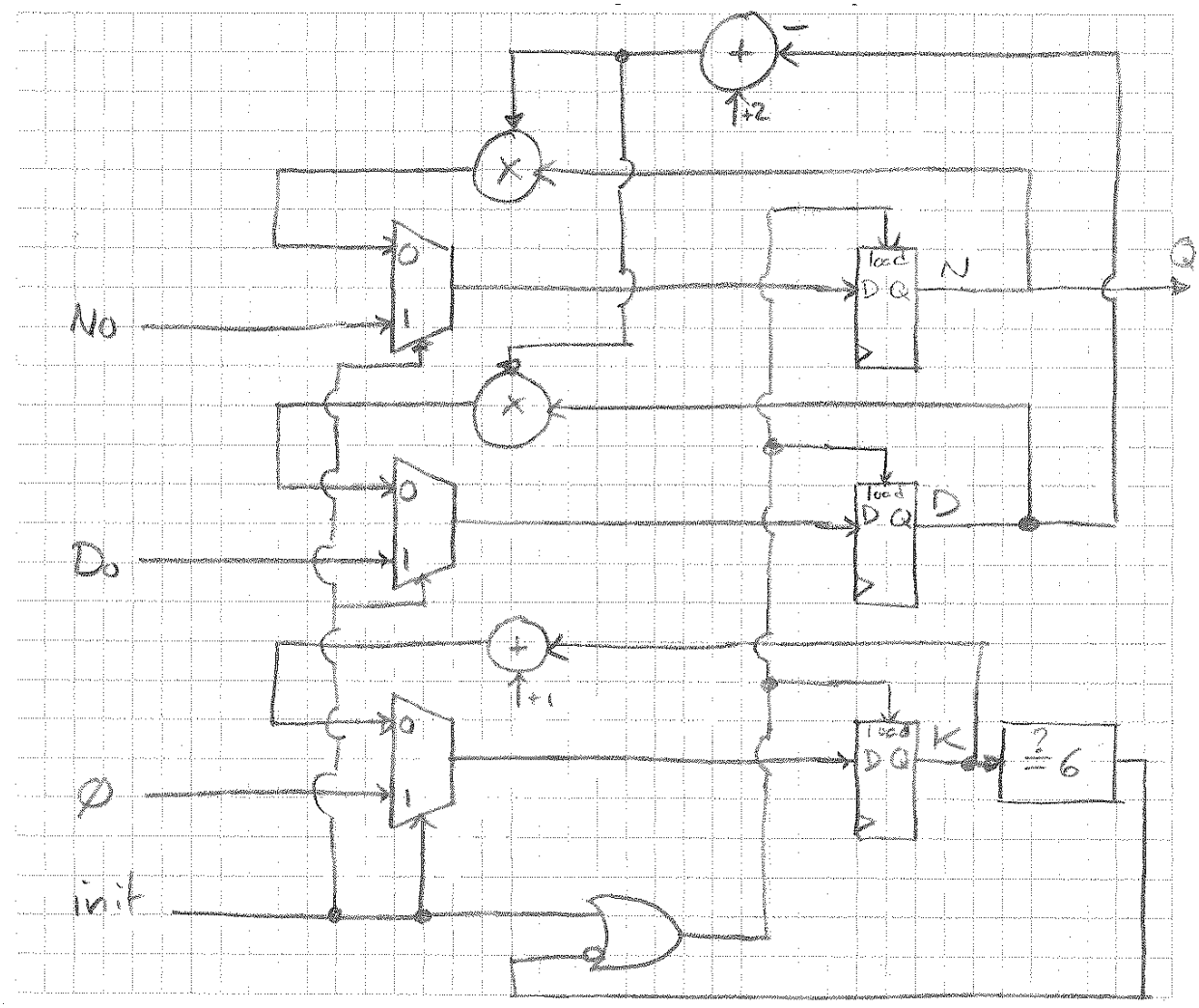
a.



b.



Q2. Diagramme



Q3.

Port	#LUT	#FF	#DSP48E	Justification
F	8	8	0	• 1 LUT par paire de bits additionnés = 8 LUT • Registre pour F
H	8	8	0	• Valeur absolue est comme l'addition : • 1 LUT par paire de bits additionnés = 8 LUT • Registre pour H
J	0 ou 8	8	0	• Division par 4 = décalage vers la droite de 2 positions, pas de calculs nécessaires • Registre pour J
K	8 ou 11	0	0	• $A \times 5 = A \times 4 + A \times 1$ • Donc une addition est requise, à proprement parler sur 8 bits seulement; le décalage est gratuit
L	0	0	1	• Multiplication générale, utilisation d'une tranche DSP48E

Q4.

a. 4 états × 2^2 entrées possibles = 4 × 4 = 16 vecteurs au minimum pour un test exhaustif

b. La séquence suivante parcourt tous les chemins de la machine à états au moins une fois, donc toutes les conditions de sa description.

temps	0	1	2	3	4	5	6	7	8
Etat	S1 (reset)	S2	S1	S3	S2	S1	S3	S4	S1
Entrée X	01 ou 10 ou 11	--	00	11	--	00	00 ou 01 ou 10	--	fini

Les 'dont'care' (--) peuvent être remplacés par n'importe quel vecteur de deux bits.

Pour parcourir les énoncés du deuxième processus, il suffit d'avoir passé à travers chacun des états, ce qui est fait.