

INF3500 : CONCEPTION ET RÉALISATION DE SYSTÈMES NUMÉRIQUES

Contrôle périodique #1 – 6 octobre 2023

Durée : 90 minutes.

Pondération: 20%.

Nom	
Prénom	
Matricule	

Directives

- Une feuille de note recto verso 8.5"×11" ou A4 permise.
- Calculatrice programmable permise.
- Ordinateurs interdits.
- Appareils mobiles interdits.
- Répondre à toutes les questions, la valeur de chaque question est indiquée.
- Répondre sur le questionnaire et le remettre. Au besoin, utilisez le verso des feuilles.
- Ne posez pas de questions. En cas de doute sur le sens d'une question, énoncez clairement vos suppositions.

Réservé au correcteur

Q1	3.7	/4
Q2	1.9	/2
Q3	2.2	/4
Q4	2.5	/4
Q5	1.8	/4
Q6	0.7	/2
Total	12.8	/20

Q1 (4 points)

Vous travaillez pour une organisation qui conçoit une unité de commande électronique principale d'une voiture technologiquement avancée et équipée de capteurs modernes.

Deux options d'implémentation sont considérées :

1. Une solution basée sur un FPGA (voir ci-dessous le tableau des dispositifs disponibles). On estime que la solution FPGA utilisera les ressources suivantes : 4000 CLB, 400k octets de mémoire, et 125 entrées/sorties (E/S).
2. Une solution sur une puce logique fixe, au coût unitaire de \$20.

Les efforts et frais de développement sont estimés comme suit :

	Nombre ingénieurs	Temps de conception	Coût des licences*	Frais de fronderie
FPGA	4	9 mois	\$12000	-
Logique fixe	6	14 mois	\$46000	\$1500K

* Coût total des licences pour la durée des projets

Le coût total d'ingénieur est fixé à \$150000 par année.

Le coût pour fabriquer et tester les cartes est de \$35 pour une carte utilisant la solution FPGA, et de \$25 pour la solution utilisant la logique fixe.

Les dispositifs FPGA disponibles sont ceux-ci :

	Nombre de CLB	Mémoire (Kbits)	E/S (I/O)	Coût unitaire
FPGA-1	2000	720	150	\$55
FPGA-2	5000	1440	250	\$75
FPGA-3	10000	4320	300	\$110
FPGA-4	30000	11520	500	\$150

Q1.1 Calculez les coûts fixes et variables pour chacune des solutions. Pour la solution FPGA, indiquez votre choix de dispositif. Montrez vos calculs et donnez vos réponses aux endroits indiqués (page suivante).

espace pour vos calculs

1. FPGA : coût fixes : $12000\$ + 4 \text{ ingénieurs} \times (9 \text{ mois} \times \frac{150000}{12 \text{ mois}}) = 462 \text{ K\$}$
 Coût variables : $N \times (35\$ + \text{coût unitaire})$: FPGA-1 : 90N\$ / FPGA-3 : 145N\$
 FPGA-2 : 110N\$ / FPGA-4 : 185N\$

2. Logique fixe : coût fixes : $46000\$ + 150000\$ + 6 \text{ ingénieurs} \times (14 \text{ mois} \times \frac{150000}{12 \text{ mois}}) = 2596 \text{ K\$}$
 Coût variables : $N \times (25 + 20) = 45N\$$

Choix de dispositif : 400K oct = 3200Kbits ; donc il nous faut 4000CLB, 3200kbits, 125E/S
 L'option la moins chère qui répond à nos critères est la FPGA-3.

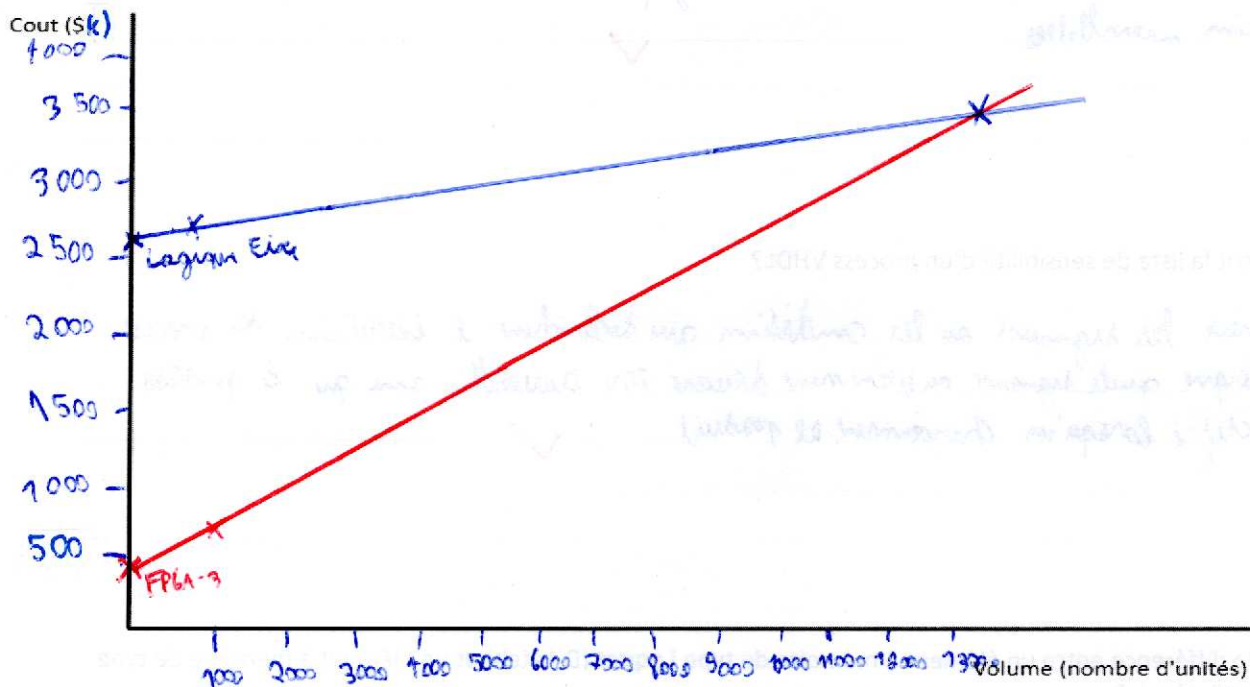
Dispositif FPGA retenu	FPGA-3 ✓
------------------------	----------

	Frais de développement (Frais Fixes)	Cout par unité (Frais variables)
Solution FPGA	462K\$ ✓	145\$ ✓
Solution Logique fixe	2596K\$ ✓	45\$ ✓

Q1.2 Rapportez sur le graphique ci-dessous les droites *couts versus volume* pour chacune des deux solutions. Indiquez clairement la solution associée à chacune des droites. Indiquez le volume correspondant au point de croisement des deux droites.

espace pour vos calculs

FPGA-3 : pour 0 unités : 462K\$; pour 1000 unités : 607K\$ -
 Logique Fixe : pour 0 unités : 2596K\$; pour 1000 unités : 2 641K\$ -



Expliquer à quoi correspondent les zones de part et d'autre du point de croisement des deux droites.

La droite rouge représente le coût pour les FPGA-3 non rapport aux nombre d'unités que l'on produit et la droite bleue représente de même pour la logique fixe, leur intersection représente le nombre d'unités qu'il faut dépasser pour que la solution fixe soit plus rentable que la FPGA soit : $462K + 145N = 2596K + 45N$

$$100N = 2134K$$

$$N = 21.34K \text{ unité produite}$$

$$21340$$

Q2 (2 points)

Q2.1 Une fois votre code VHDL complété et vérifié (par simulation), quelles sont les étapes à suivre avant de pouvoir programmer votre FPGA. Donnez les étapes dans l'ordre.

1. Synthèse 2. Implémentation 3. Génération des fichiers de configuration
4. Programmation sur le FPGA

Q2.2 Dans quelles conditions une boucle en VHDL (*for loop*) est-elle synthétisable dans un FPGA?

Il faut une boucle à itération statique avec un nombre défini et fini d'itérations.
La boucle doit être utilisée pour décrire une logique combinatoire ou une séquence d'opérations parallèles.

Q2.3 Que contient la liste de sensibilité d'un *process* VHDL?

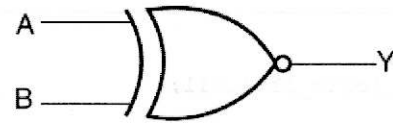
elle contient les signaux ou les conditions qui déclenchent l'exécution du *process*.
elle indique quels signaux ou événements doivent être surveillés pour que le *process* soit activé lorsqu'un changement se produit.

Q2.4 Expliquer la différence entre un élément à mémoire de type **Loquet** (D-Latch) et un élément à mémoire de type **Bascule** (D-FlipFlop) ?

un D-latch peut changer de valeur au temps réel tandis qu'un D-flipflop enregistre la valeur d'entrée à un moment précis généralement déterminé par un clock (CLK).

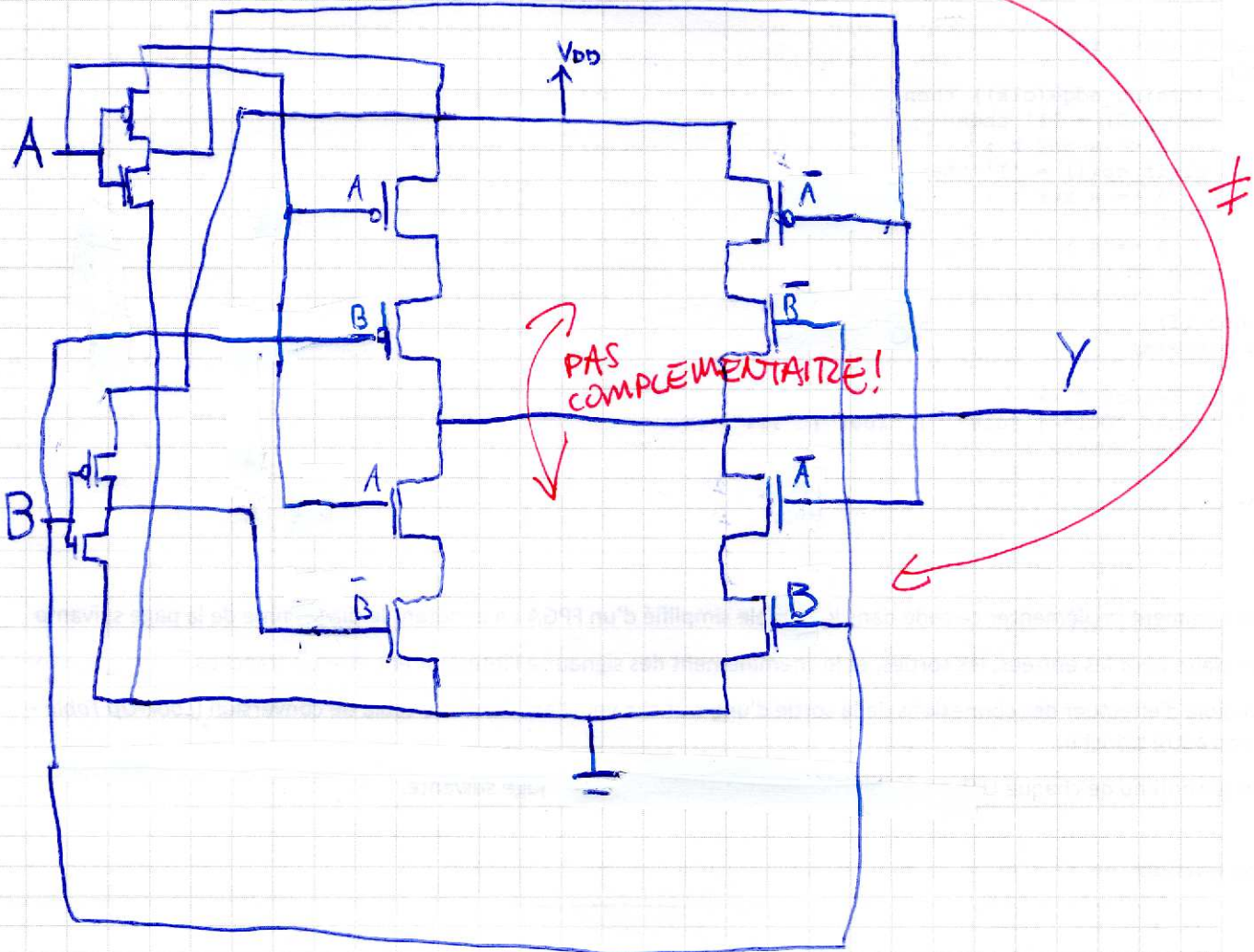
Q3 (4 points)

Faites la conception d'un circuit CMOS qui implémente la fonction booléenne ci-contre. Les entrées A et B inversées ne sont pas disponibles, vous devez donc prévoir les transistors nécessaires au besoin. Utilisez le moins de transistors possible.



$$Y = A \oplus B \\ = AB + \bar{A}\bar{B}$$

$$Y' = A \oplus B \\ = AB' + \bar{A}B$$



Q4 (4 points)

Considérez le code VHDL suivant :

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

entity cplq4 is
    port (clk      : in std_logic;
          s, t      : in std_logic;
          op        : in std_logic_vector(2 downto 0);
          A, B, C    : out std_logic);
end cplq4;

architecture ex of cplq4 is
begin

    process (clk) is
    begin
        if (rising_edge(clk)) then
            if op(0) = '1' then
                A <= s and t ;
            elsif op(1) = '1' then
                A <= s xor t ;
            else
                A <= s ;
            end if ;
            B <= (s xnor t) or op(2);
        end if;
    end process;

    with op select C <=
        '1' when "001" | "010" | "100" | "111",
        '0' when others ;

end ex;
```

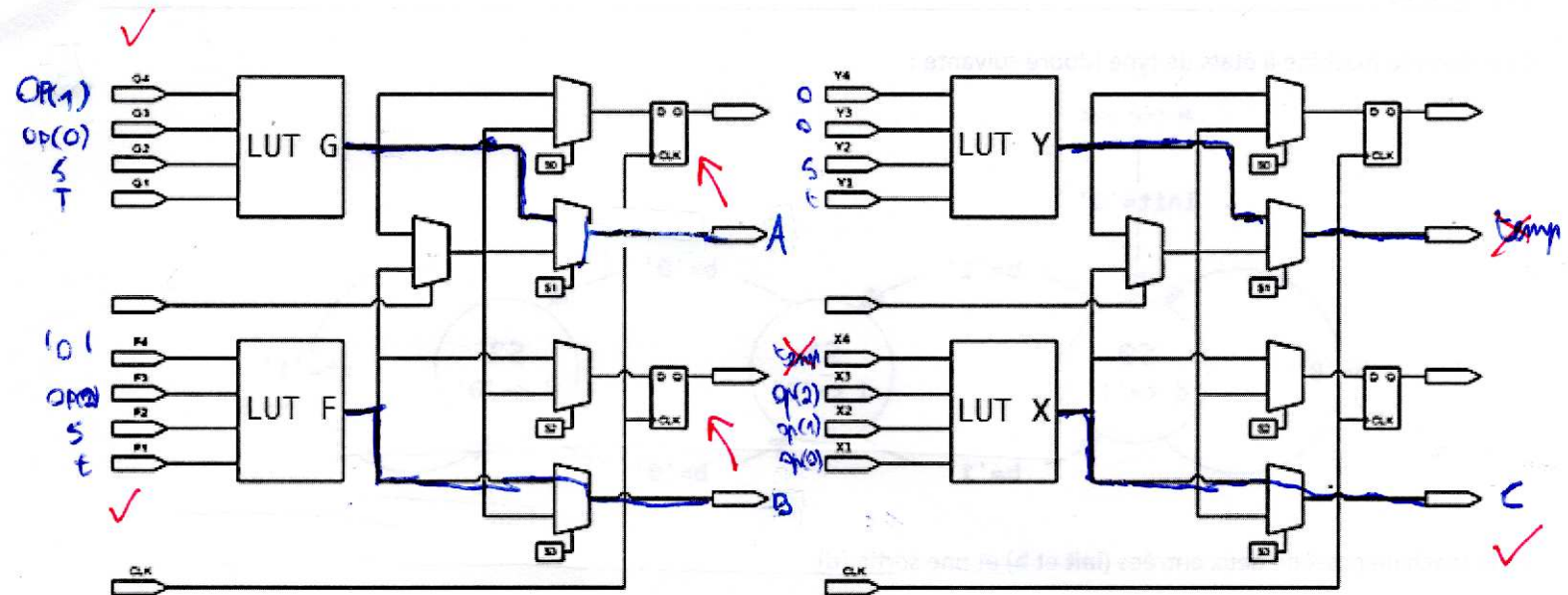
Montrez comment implémenter ce code dans le modèle simplifié d'un FPGA en annotant le diagramme de la page suivante.

Indiquez clairement les entrées, les sorties, et le cheminement des signaux à l'intérieur et entre les tranches.

Il est possible d'effectuer des connexions de la sortie d'une tranche vers l'entrée d'une table de conversion (*Look-Up Table – LUT*) d'une autre tranche.

Montrez le contenu de chaque LUT que vous utilisez dans le tableau de la page suivante.

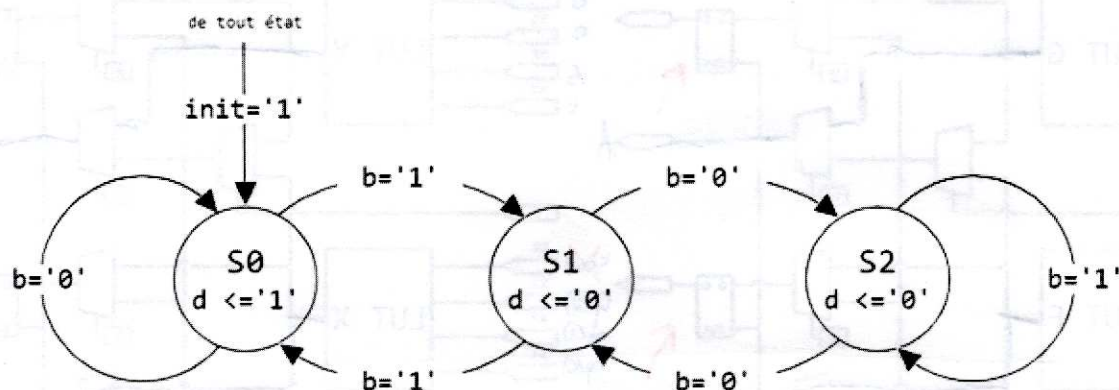
espace pour vos calculs



				B	A	C	X
4	3	2	1	LUT F	LUT G	LUT X	LUT Y
0	0	0	0	1	0	0	1
0	0	0	1	0	0	1	0
0	0	1	0	0	1	1	0
0	0	1	1	1	1	0	1
0	1	0	0	1	0	1	1
0	1	0	1	1	0	0	1
0	1	1	0	1	0	0	1
0	1	1	1	1	1	1	1
1	0	0	0	-	0	-	-
1	0	0	1	-	1	-	-
1	0	1	0	-	1	-	-
1	0	1	1	-	0	-	-
1	1	0	0	-	0	-	-
1	1	0	1	-	0	-	-
1	1	1	0	-	0	-	-
1	1	1	1	-	1	-	-

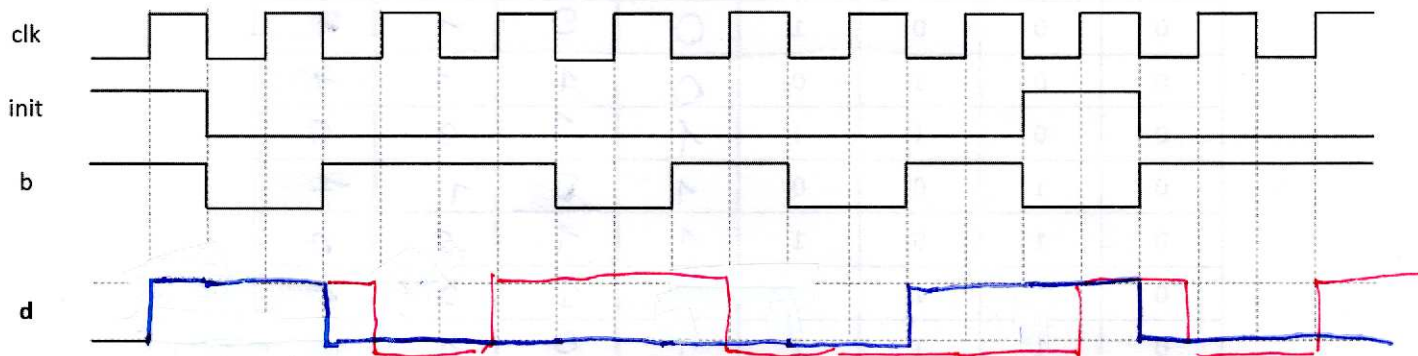
Q5 (4 points)

Considérez la machine à états de type Moore suivante :

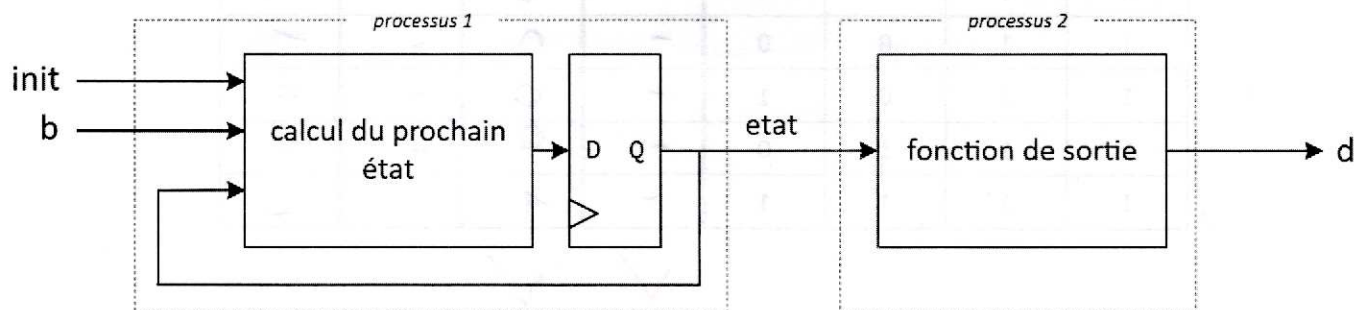


Cette machine possède deux entrées (**init** et **b**) et une sortie (**d**).

Q5.1 Complétez le chronogramme suivant en indiquant l'évolution de la sortie **d** en fonction du temps.
La machine à états est synchronisée sur les fronts montant de l'horloge **clk**.



Q5.2 Complétez le code VHDL suivant pour cette machine à états.
Utilisez une architecture à deux process tel qu'illustré ci-dessous.




```

library ieee;
use ieee.std_logic_1164.all;

entity cplq5 is
    port (clk : in std_logic ;
          init : in std_logic ;
          b : in std_logic ;
          d : out std_logic) ;
end cplq5 ;

```

```

architecture etudiant of cplq5 is

```

```

    type etat_type is (S0, S1, S2);
    signal etat : etat_type := S0 ;

```

```

process (CLK, init)

```

```

begin

```

```

    if (rising_edge (CLK)) then
        if (init = '1') then
            etat_type = S0 ;
        end if;
    end if;

```

sortied ?

```

process (CLK, b)

```

```

begin

```

```

    if (rising_edge (CLK)) then
        when etat_type is
        when S0 =>
            if (b = '1') then
                etat_type = S1 ;
            else
                etat_type = S0 ;
            end if;

```

```

        when S1 =>
            if (b = '0') then
                etat_type = S2 ;
            else
                etat_type = S0 ;
            end if;

```

```

        when S2 =>
            if (b = '1') then
                etat_type = S2 ;
            else
                etat_type = S1 ;
            end if;

```

```

end

```

```

end etudiant;

```

Q6 (2 points)


Q6.1 Pour chacun des cas suivants, indiquez l'élément séquentiel et les caractéristique(s) inféré(s) par le code VHDL.
Cochez toutes les cases pertinentes.

<pre>process (clk) is begin if (rising_edge(clk)) then if (reset = '1') then A <= '0'; else A <= A xor B; end if ; end if; end process;</pre>	<ul style="list-style-type: none"> <input type="checkbox"/> Loquet SR <input type="checkbox"/> Loquet D <input checked="" type="checkbox"/> Bascule D – front montant <input type="checkbox"/> Bascule D – front descendant ✗ <input checked="" type="checkbox"/> Reset synchrone <input type="checkbox"/> Reset asynchrone <input type="checkbox"/> Contrôle de charge
<pre>process (clk, reset) is begin if (reset = '1') then B <= '0'; elsif (CLK'event and CLK = '0') then if (ce = '1') then B <= A or not(B); End if ; end if; end process;</pre>	<ul style="list-style-type: none"> <input type="checkbox"/> Loquet SR <input type="checkbox"/> Loquet D <input type="checkbox"/> Bascule D – front montant ✗ <input checked="" type="checkbox"/> Bascule D – front descendant ✗ <input checked="" type="checkbox"/> Reset synchrone ✗ <input checked="" type="checkbox"/> Reset asynchrone ✗ <input checked="" type="checkbox"/> Contrôle de charge

Q6.2 Expliquez pourquoi le *process* de l'encadré suivant n'est pas purement combinatoire.

Les signaux p, q et r sont définis comme suit :

```
type p_type is (p1, p2) ;
signal p : p_type ;
signal q, r : std_logic ;
```

<pre>process (p, q) begin case p is when p1 => if (q = '1') then r <= '0'; end if; when p2 => r <= '1'; end case; end process;</pre>	
--	--