AVIGNON UNIVERSITÉ — CERI

Évaluation de connaissances
Durée max : **1h30**

Partie A -Questions de cours (**15'-5.5pts**).
Généralités sur les HDL, les SoC, les FPGA, et le Cyclone V.

A.1) [0,25pt] Comment appelle-t-on la phase de conception de puces électroniques reconfigurables ou non ?
Réponse attendue.
La synthèse

A.2) [0,25pt] Comment appelle-t-on la phase de mise au point d'un modèle de simulation numérique ?
Réponse attendue.
La modélisation.

A.3) [0,5pt] Qu'appelle-t-on unité de conception en VHDL ?
Réponse attendue.
Le couple "entity-architecture".

A.4) [0,5pt] Comment appelle-t-on en VHDL l'opération qui consiste à l'appel d'un composant dans une description structurelle ?
Réponse attendue.
L'instanciation.

A.5) [1,25pts] Citer les 5 constituants de base d'un SoC.
Réponse attendue.
CPU-Contrôleur mémoire-Interface de communication-IPs spécialisées (vidéo,Audio, réseau, etc) et bus d'interconnexion.

A.6) [0,5pt] D'une façon générale, sans tenir compte du facteur "taux d'activité" α liée à une application et de la capacité **C** équivalente de commutation des portes logiques, de quelles grandeurs dépend la puissance **P** dissipée d'un circuit électronique numérique ?
Réponse attendue.
De la fréquence f et du carré de la tension d'alimentation V_{DD}.

A.7) [0,5pt] Quel est l'avantage principal d'intégrer un processeur hardcore (ex un ARM) dans un FPGA plutôt que d'y implémenter un processeur softcore (ex un NiosII) ?
Réponse attendue.
On économise les ressources matérielles (LE) du FPGA. Ce dernier pouvant servir à implémenter d'autre fonctionnalités matérielles.

A.8) [0,25pt] Donner la signification du sigle "**HPS**" du **Cyclone V** d'Intel/Altera?
Réponse attendue.
"Hard Processing/Processor System".

A.9) [0,5pt] Le **Cyclone V** d'Intel/Altera de la carte **DE1-SoC** est constitué d'un "bloc FPGA" et d'un "bloc HPS". Comment sont réalisées les communication(échanges) entre ces 2 "blocs" ?
Réponse attendue.
Les communications FPGA←→ HPS sont réalisée au travers de ponts (bridges) ou bus unidirectionnels AXI spécialisés.

A.10) [0,5pt] Sur la carte **DE1-SoC** comment les périphériques (switches, buttons, LED, etc) du "bloc FPGA" sont-ils accessibles par le "bloc HPS" ?

Réponse attendue.

Par leurs adresses physiques via un mapping mémoire "figé".

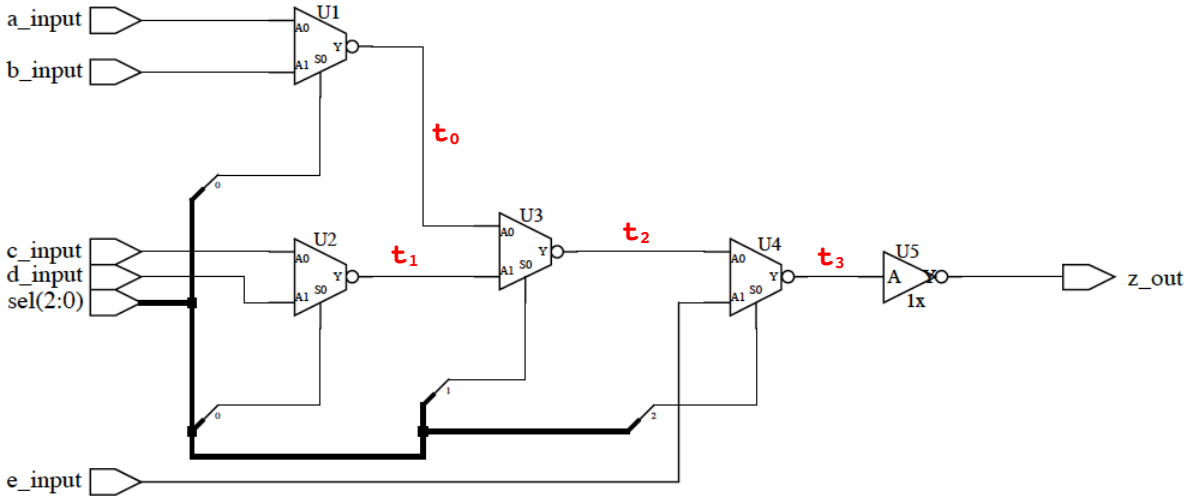
A.11) [0,5pt] Compléter la phrase suivante :

Une application **bare-metal** peut s'interfacer directement avec le matériel du système et s'exécuter **sans système d'exploitation**.

Partie B -Exercices d'application (1h05-14,5 pts)

B.1)

a) [1,5pts] **Réponse attendue.**

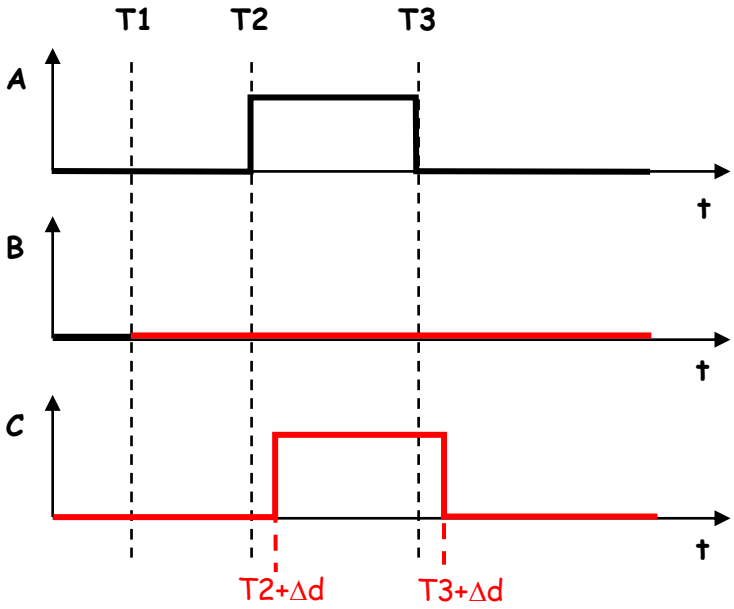


b) [0,5pt] **Réponse attendue.**

Il s'agit d'un multiplexeur 5 vers 1.

B.2)

a) **[2pts]**



Explications :

Les signaux A, B, C sont stockées dans des variables tampons a, b, c.
Au temps T2 **A** a changé. Comme le processus est sensible à **A**, le simulateur (re)démarre l'exécution du processus. Le **tampon c (pas C)** reçoit **A**, et la condition (**C**='1') est testée.
Le résultat du test sera **false**, puisque **C** est toujours égal à '0': la mise à jour à la valeur '1' n'aura lieu qu'au temps **T2+ Δd** , c'est-à-dire à la fin du processus.
A la fin de la première exécution du processus, **B** sera donc toujours égal à '0'.
Au temps **T3**, **A** passe à '0', ce qui fait redémarrer l'exécution du processus. Comme la valeur de **C** est mise à jour seulement au temps **T3+ Δd**, **C** garde sa valeur précédente ('1'). La condition (C='1') est maintenant **true**, et **B** reçoit **A** à la fin du processus, c'est-à-dire à '0'. En conclusion, **B** reste toujours à '0' pendant la simulation.

B.3)

a) **[1,5pts]**

Réponse attendue.

$$Q_0^+ = \overline{W}\,Q_0 + W\,\overline{Q_0}$$
$$Q_1^+ = \overline{W}\,Q_1 + Q_1\,\overline{Q_0} + W\,\overline{Q_1}\,Q_0$$
$$Q_2^+ = \overline{W}\,Q_2 + Q_2\,\overline{Q_0} + Q_2\,\overline{Q_1} + W\,\overline{Q_2}\,Q_1 Q_0$$

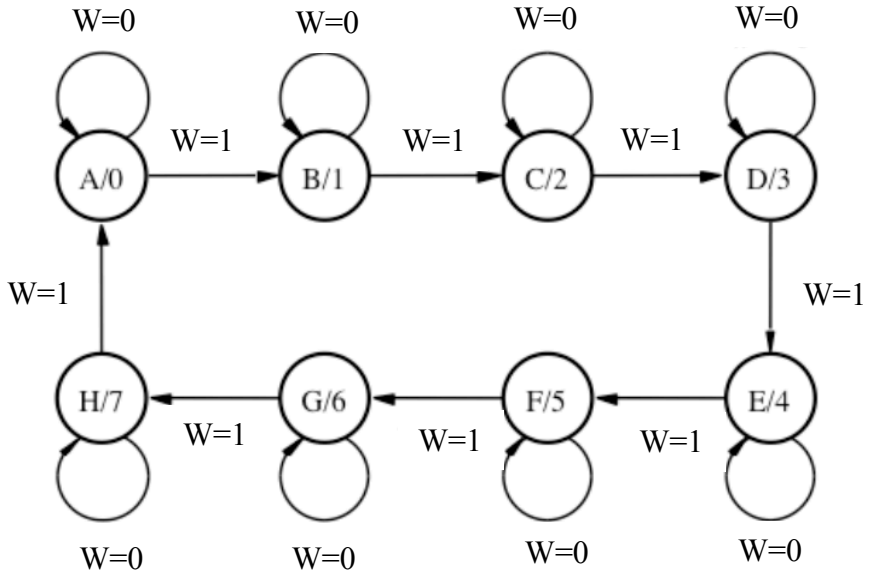
Etats Présents	Etats Futurs		Sorties
	W=0	W=1	
$Q_2\,Q_1\,Q_0$	$Q_2^+\,Q_1^+\,Q_0^+$	$Q_2^+\,Q_1^+\,Q_0^+$	$Q_2\,Q_1\,Q_0$
0 0 0	0 0 0	0 0 1	0 0 0
0 0 1	0 0 1	0 1 0	0 0 1
0 1 0	0 1 0	0 1 1	0 1 0
0 1 1	0 1 1	1 0 0	0 1 1
1 0 0	1 0 0	1 0 1	1 0 0
1 0 1	1 0 1	1 1 0	1 0 1
1 1 0	1 1 0	1 1 1	1 1 0
1 1 1	1 1 1	0 0 0	1 1 1

b) **[1,5pts]**

c) **[1pt]**

Réponse attendue.

En utilisant les lettres de A à H pour les états de 0 à 7



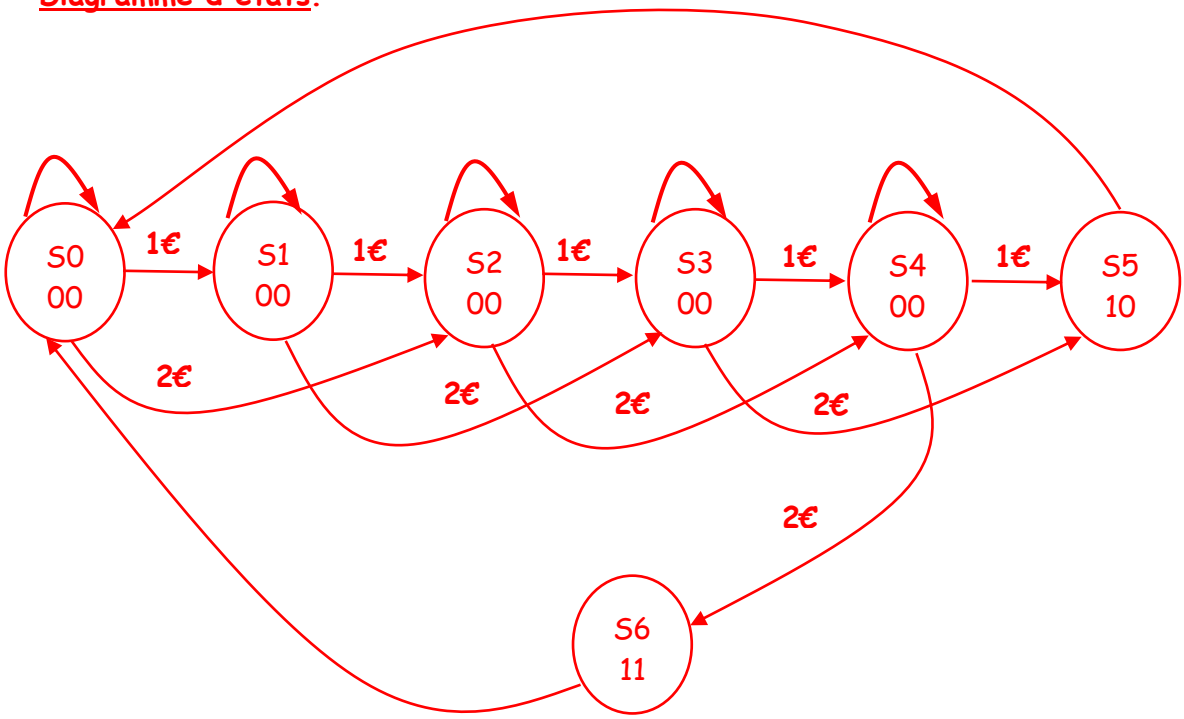
d) **[0,5pt]**
Réponse attendue.
Compteur octal de 0 à 7 tant que W=1 mais qui freeze si W=0.

B.4) Synthèse VHDL d'une machine à états finis à 2 processus.

a) [1pt] Réponse attendue.
Un processus sera synchrone (séquentiel-horloge) qui met à jour l'état présent et l'autre sera asynchrone (combinatoire) qui met à jour l'état futur en fonction des conditions sur les entrées. Il permet également de mettre à jour les sorties.

Cahier des charges.(Machine de Moore)

b) **[3pts]** Dessiner le diagramme d'états de la machine.
Diagramme d'états.



c) **[2pts]** Compléter le code VHDL ci-dessous à partir de votre diagramme d'état.

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY machine IS
PORT (
   clk              : IN STD_LOGIC;
   un_Euro          : IN STD_LOGIC;
   deux_Euros       : IN STD_LOGIC;
   film             : OUT STD_LOGIC;
   monnaie          : OUT STD_LOGIC);
END machine;
```

```
ARCHITECTURE rtl OF machine IS
TYPE etats IS (S0,S1,S2,S3,S4,S5,S6);
SIGNAL etat_present : etats;
SIGNAL etat_futur : etats;
BEGIN
   --Processus asynchrone

   PROCESS (etat_present, un_Euro, deux_Euros)
   BEGIN
        CASE etat_present IS
        WHEN S0 =>
             film <= '0';
             monnaie <= '0';
                  IF un_Euro = '1' THEN
                  etat_futur <= S1;
                  ELSIF deux_Euros = '1' THEN
                  etat_futur <= S2;
                  ELSE
                  etat_futur <= S0;
                  END IF;
        WHEN S1 =>
             film <= '0';
             monnaie <= '0';
                  IF un_Euro = '1' THEN
                  etat_futur <= S2;
                  ELSIF deux_Euros = '1' THEN
                  etat_futur <= S3;
                  ELSE
                  etat_futur <= S1;
                  END IF;
        WHEN S2 =>
             film <= '0';
             monnaie <= '0';
                  IF un_Euro = '1' THEN
                  etat_futur <= S3;
                  ELSIF deux_Euros = '1' THEN
                  etat_futur <= S4;
                  ELSE
                  etat_futur <= S2;
                  END IF;
        WHEN S3 =>
             film <= '0';
             monnaie <= '0';
                  IF un_Euro = '1' THEN
                  etat_futur <= S4;
                  ELSIF deux_Euros = '1' THEN
                  etat_futur <= S5;
                  ELSE
                  etat_futur <= S3;
                  END IF;
        WHEN S4 =>
             film <= '0';
             monnaie <= '0';
                  IF un_Euro = '1' THEN
                  etat_futur <= S5;
```

```
                    ELSIF deux_Euros = '1' THEN
                    etat_futur <= S6;
                    ELSE
                    etat_futur <= S4;
                    END IF;
            WHEN S5 =>
                 film <= '1';
                 monnaie <= '0';
                 etat_futur <= S0;
            WHEN S6 =>
                 film <= '1';
                 monnaie <= '1';
                 etat_futur <= S0;
            END CASE;

END PROCESS;



   -- Processus synchrone
PROCESS (clk)
BEGIN
     IF clk'EVENT AND clk = '1' THEN
     etat_present <= etat_futur;
     END IF;
END PROCESS;
END;
```