Tronc commun 3A - Durée : 90 mn **Examen du** **Cours EE341** **NOM :**	2ème session Année 2011 – 2012 Un seul document autorisé : « VHDL résumé de syntaxe » Calculatrice interdite **PRENOM :**
1 Les points donnés dans l'énoncé entre crochets [X] après chaque question représentent le barème et indiquent le temps à passer en minutes sur chaque question *2 La qualité de la rédaction (lisibilité, orthographe) sera prise en compte*	

I Partie cours [20/90]

1.1 Que signifie les initiales de l'acronyme HDL ? [1]

1.2 Pourquoi ne pas utiliser un langage informatique comme le C plutôt que VHDL ? [2]

1.3 Quels sont les avantages (en citer au moins 4) d'utiliser le VHDL pour la conception des systèmes électroniques ? [2]

1.4 Expliquer succinctement les deux termes : description comportementale et description structurelle [4]

1.5 Pourquoi est-il recommandé d'utiliser le type *std_logic* plutôt que le type *bit* ? [2]

1.6 Citer 2 différences entre les instructions séquentielles et les instructions concurrentes ? [2]

1.7 Quand est-ce que les processus d'une description VHDL sont exécutés ? En particulier, dans quel ordre sont exécutées les instructions concurrentes ? [2]

1.8 Donner le code permettant de décrire un registre de 64 bascules Flip-Flop sur front montant avec reset actif au niveau bas [5]

II Partie exercices : [70/90]

2.1 Démultiplexeur [20]

Ci dessous l'entité d'un bloc de démultiplexage :

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

entity reg_in_i_dct is
  port (
    rstn    : in  std_logic; --reset
    clk     : in  std_logic; --horloge
    stall   : in  std_logic; --commande de freeze
    cpt_reg : in  std_logic_vector(2 downto 0); --commande
    data_in : in  std_logic_vector(15 downto 0); --bus input
    x0      : out std_logic_vector(15 downto 0); --sortie x0
    x1      : out std_logic_vector(15 downto 0); --sortie x1
    x6      : out std_logic_vector(15 downto 0); --sortie x6
    x7      : out std_logic_vector(15 downto 0)); --sortie defaut
```

```
end reg_in_i_dct;
```

A la mise sous tension, le reset asynchrone est actif (actif bas), les sorties sont toutes à '0'. Si le signal de stall est inactif (actif haut), alors l'affectation est la suivante :

- X0 reçoit l'entrée data_in si la commande cpt_reg=0,
- X1 reçoit l'entrée data_in si la commande cpt_reg=1,
- X6 reçoit l'entrée data_in si la commande cpt_reg=6,
- X7 reçoit l'entrée data_in dans les autres cas.

Si le signal de stall est actif, les sorties ne sont pas mises à jour.

2.1.1 Coder le corps d'architecture en synchronisant les sorties sur les fronts montants de clk. [20]

2.2 Analyse de code [50]

2.2.1 Le circuit correspondant à la description ci-dessous affecte-t-il les sorties de manière synchrone, asynchrone, ou les deux ? Expliquer. [4]

2.2.2 Quelles sont les valeurs maximales et minimales pour *entree* ? [2]

2.2.3 Quel est le rôle du signal *init* ? Comment opère t-il ? Quand doit-il être activé ? [1+1+1]

2.2.4 Quel est le rôle du signal *pret* ? Quelle est la durée minimale pendant laquelle il doit être actif? Quand doit-il être activé ? [2+1+1]

2.1.5 Qu'indique la variable *front* ? [2]

2.1.5 Qu'indique le signal *ok* ? [2]

2.1.6 Faire un chronogramme d'une séquence complète de calcul représentant tous les signaux d'entrées-sorties et les variables internes [20]. Représenter également l'enchaînement avec une autre séquence. [4]

2.1.7 Estimer le temps de calcul minimum total (en périodes d'horloge)? [5]

2.1.8 Combien de bascules seront générées par la synthèse de ce code ? [5]

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY minimum IS
GENERIC ( Nombre_mots : natural := 10);
PORT (  init : IN std_ulogic;
        h : IN std_ulogic;
        pret : IN std_ulogic;
        entree : IN unsigned(7 DOWNTO 0);
        sortie : OUT unsigned(7 DOWNTO 0);
        ok : OUT std_ulogic);
END minimum;

ARCHITECTURE synthetisable OF minimum IS
BEGIN -- synthetisable
p1 : PROCESS
VARIABLE min : unsigned(7 DOWNTO 0);
VARIABLE c : natural := 0;
VARIABLE front : boolean;
BEGIN -- PROCESS p1 totalement sequentiel
        WAIT ON init, h;
        IF init = '1' THEN
                sortie <= "00000000";
                ok <= '0';
                c := 0;
                min := "11111111";
                front := false ;
        ELSIF rising_edge(h) THEN -- front montant de h
                IF ( pret = '1') AND (NOT front) THEN
                        front := true ;
                        IF entree < min THEN
                                min := entree;
                        END IF;
                ELSIF pret = '0' AND front THEN
                        front := false ;
                        c := c +1;
                        ok <= '0';
                        IF c = Nombre_mots THEN
                                sortie <= min;
                                ok <= '1';
                                c := 0;
                                min := "11111111" ;
                        END IF;
                END IF;
        END IF;
END PROCESS p1;
END synthetisable;
```