efrei PARIS
Ecole d'ingénieurs du numérique

20180932: WIECLAW Manon
L3: TD B / Com G8
TE605-DE (08/03/2021)
A001

Ne rien inscrire dans ce cadre

Prénom Manon
Nom WIECLAW
Promotion 2023
Groupe E01

18,5/20

Promotion : L3

Module VHDL

TE605

Devoir Ecrit Horaire : 14h00-15h45

Examen sur table

Sujet rédigé par : ILKI (Didier MEIER)

Supports autorisés :

Calculatrice autorisée : **non**

Documents autorisés : **non**

Traducteur électronique : **non**

Dictionnaire : **non**

Consigne :

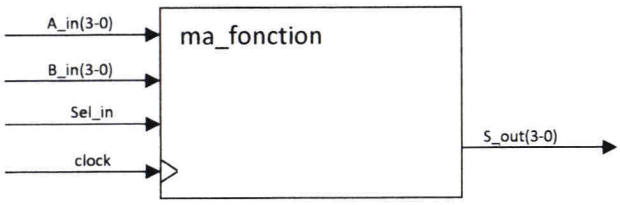
Merci de restituer : **le sujet avec votre copie quadrillée**

Rappel :

- Tous les appareils électroniques (téléphones portables, ordinateurs, tablettes, montres connectées, accès à internet ...) doivent être éteints et rangés.
- Il est interdit de communiquer.
- Toute fraude ou tentative de fraude fera l'objet d'un rapport de la part du surveillant et sera sanctionnée par la note zéro, assortie d'une convocation devant le conseil de discipline. Aucune contestation ne sera possible. Tous les documents et supports utilisés frauduleusement devront être remis au surveillant.
- Aucune sortie de la salle d'examen ne sera autorisée avant la moitié de la durée de l'épreuve.

Ne rien inscrire dans ce cadre

3. On souhaite décrire en VHDL le fonctionnement « hardware » du bloc suivant :



Le bloc fonctionnel réalise un multiplexeur 2 vers 1 synchrone. Son fonctionnement est le suivant :

- A chaque front descendant de l'horloge clock :
  - si l'entrée Sel_in est égale à '0' : la sortie S_out est égale A_in + 1 (addition)
  - si l'entrée Sel_in est égale à '1' : la sortie S_out est égale B_ in - 1 (soustraction)

Pour la synthèse VHDL, les types de signaux utilisés sont STD_LOGIC et STD_LOGIC_VECTOR.

3.1.1. Complétez alors la description ci-dessous de l'entité.

entity ma_fonction is

Port (

A_in : in std_logic_vector (3 downto 0);
B_in : in std_logic_vector (3 downto 0);
sel_in : in std_logic ;
clock : in std_logic ;
S_out : out std_logic_vector (3 downto 0)

+ 2

);

end ma_fonction;

3.1.2. Complétez l'architecture nommée ma_fonction_arch **de l'entité** ma_fonction **permettant de décrire puis synthétiser ce bloc fonctionnel.**

architecture ma_fonction_arch of ma_fonction is

-- définition des signaux (si nécessaire) :

my_S_out : std_logic_vector (3 down to 0) := "0000";

Réservé à la correction :	+2

begin

-- début de la description de l'architecture

```
myprocess : process (clock)
  begin
  when (clock'event and clock = '0') then
    if (sel_in = '0') then
      my_S_out <= A_in + 1;
    else
      my_S_out <= B_in - 1;
    end if;
end process;

S_out <= my_S_out;
```

end ma_fonction_arch;

Réservé à la correction :	+3,5

NOM :	WIECLAW
PRENOM :	Manon

VHDL

18,5

Introduction :

Le devoir surveillé de programmation VHDL se décompose en trois parties distinctes et indépendantes :

1. Questions de cours (7 points)
2. Analyses de synthèses en VHDL (7 points)
3. Réalisation d'une synthèse en VHDL (6 points)

Vos réponses aux différentes questions sont à indiquer directement sur l'énoncé. L'ensemble du feuillet est à remettre obligatoirement au surveillant à la fin de l'examen.

La durée impartie à la réalisation de ce devoir surveillé est de 1h45min.

Bon travail !

Réservé à la correction :	✓

1. Questions de cours

1.1. Que signifie l'acronyme VHDL ?

VHDL signifie VHSIC Hardware Description Language.

+0,5

VHSIC = very high speed integrated circuit.

1.2. Quelles sont les 5 unités de base permettant de décrire la structure du langage VHDL ?

Les 5 unités sont :

① Les bibliothèques ✓

② Les unités de conception VHDL (entité, architecture, package ...)

③ Les unités d'instruction VHDL (séquentielles et concurrentielles)

④ Le support de l'information (variable, signaux ...)

⑤ Le typage des objets (std_logic_vector ...)

+1

1.3. L'entité « entity » permet-elle de décrire la vue externe ou interne d'une description matérielle ?

L'entité permet de décrire la vue externe.

+0,5

1.4. L'architecture « architecture » permet-elle de décrire la vue externe ou interne d'une description matérielle ?

L'architecture permet de décrire la vue interne.

+0,5

Réservé à la correction :	+2,5

2.1.1. L'automate est-il initialisé (reset) de manière synchrone ou asynchrone (justifiez votre réponse) ?

L'automate est initialisé de manière asynchrone car il n'est pas encapsulé dans la condition (clock'event and clock = '0').

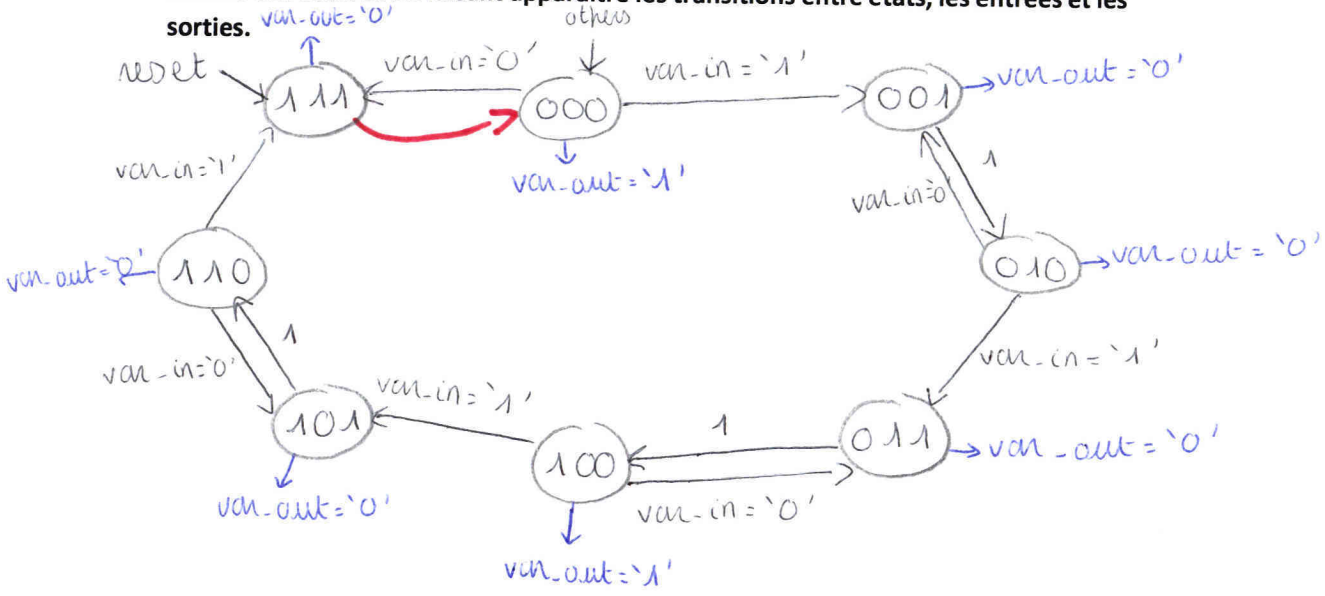
+0,5

2.1.2. L'évolution de l'automate s'effectue-t-elle sur front montant ou descendant de l'horloge d'entrée (clock) ?

L'évolution de l'automate s'effectue sur front descendant de l'horloge : clock = '0'.

+0,5

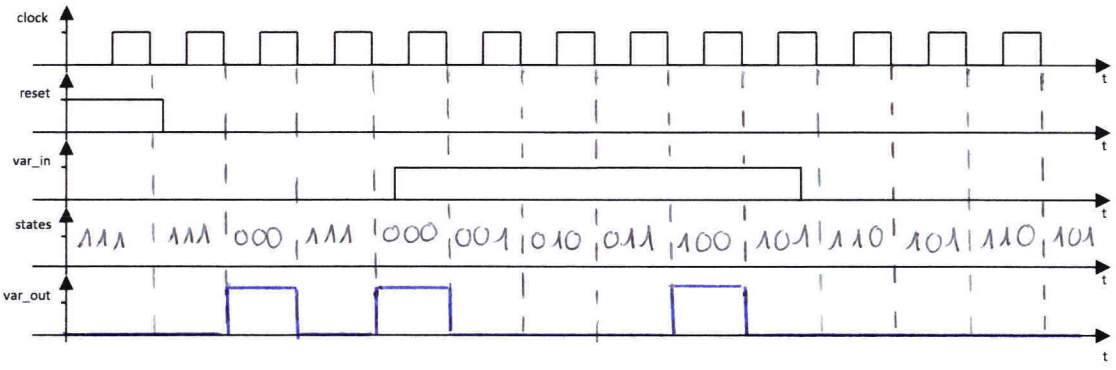
2.1.3. A l'aide d'un schéma, décrivez l'automate du code ci-dessus en respectant le codage binaire des états et en faisant apparaître les transitions entre états, les entrées et les sorties.



+2,8

Réservé à la correction :	+3,8

2.1.4. Complétez, à l'aide du code précédent, le chronogramme suivant pour les sorties *var_out* et *states* en fonction des évolutions des différentes entrées.



+0,5

+0,5

2.2. On considère les extraits de codes VHDL du tableau suivant. Indiquez les fonctions logiques décrites par ces extraits (soyez précis dans vos descriptions).

Tous les signaux utilisés sont de types STD_LOGIC ou STD_LOGIC_VECTOR.

Extraits de codes VHDL	Fonctions logiques décrites ?

Myprocname1 : process (clock, reset, preset)
Begin
if (preset = '1') then
Qout <= '1';
elsif (reset = '1') then
Qout <= '0';
elsif (clock'event and clock = '1') then
Qout <= Din;
end if;
End process ;
	Représente une bascule D synchrone avec un preset et un reset asynchrone.
	Fct ?

Myprocname2 : process (Ain, Bin)
Begin
if (Bin < Ain) then
gre <= '1'; equ <='0'; low <= '0';
elsif (Bin = Ain) then
gre <= '0'; equ <='1'; low <= '0';
else
gre <= '0'; equ <='0'; low <= '1';
end if;
End process ;
	C'est un comparateur asynchrone entre deux entrées.

Myprocname3 : process (clock, reset)
Begin
If (reset = '1') then
Cntout <= "0000";
elsif (clock'event and clock = '1') then
If (Cntout = "1100") then
Cntout <= "0000";
else
Cntout <= Cntout + 1;
end if;
end if;
End process ;
	C'est un compteur synchrone sur 4 bits modulo 13 avec un reset asynchrone, compteur actif sur front montant.

Myprocname4 : process (clock)
Begin
if (clock'event and clock = '1') then
Sout <= not Sout;
end if;
End process ;
	C'est un inverseur synchrone, sur front montant.

+0,5

+0,5

+0,5

+0,5

Réservé à la correction :	+?,9

1.5. Pourquoi peut-on dire « au sein d'une architecture, tout est process ! » ?

Les process sont les éléments calculatoire de base au simulateur. Il y a les process explicites avec le mot clé "process" et ceux implicites qui sont les instructions concurrentielles. Donc du point de vue de l'exécution tout est process.

~ oh...

+1

1.6. Quelles sont les principales différences entre un signal et une variable ?

Une variable est affectée post-~~compilation~~ ? alors que les signaux doivent attendre la fin de l'exécution de quoi ? pour que la valeur dans leur pilote leur soit affectée.

TOP

Aucune de précision...

= portée ?

= ...

Réservé à la correction :	+ 1,3

1.7. Le langage VHDL permet-il d'effectuer des descriptions de schémas hiérarchiques ? Si oui, décrivez-en le principe.

Oui il permet d'effectuer des descriptions de schémas hiérarchique.

On déclare plusieurs composants qui seront tous reliés entre eux par des signaux. Il y aura un composant "principal" qui possèdera les entrées et sorties nécessaires aux composants internes.

+0,5

1.8. Les corps des fonctions et les procédures contiennent-ils des instructions séquentielles ou concurrentielles ?

Les corps des fonctions et des process contiennent des instructions séquentielles.

+0,5

1.9. En VHDL, à quoi correspond le driver d'un signal ? Donnez un exemple de sa composition.

Le driver d'un signal ou pilote est la mémoire du signal, c'est un couple état - heure :

+0,5

S ←

'1'	0ns	10ns	20ns
	'0'	'1'	'0'

Réservé à la correction :	+1,5

1.10. Décrivez, de manière synthétique, le principe de fonctionnement d'une simulation en VHDL (penser à la « roue temporelle »).

Une simulation VHDL fonctionne selon le principe d'une roue temporelle :

① Tous les évènements non mûrs présents dans les pilotes sont classés par ordre croissant des dates futures.

② Le temps est avancé jusqu'au premier évènement mûr de la liste et va être supprimé de celle-ci. On affecte au signal la valeur présente dans le pilote.

③ Cette affectation va déclencher un processus qui va affecter de nouvelles valeurs dans les pilotes des autres signaux ce qui va contribuer à une nouvelle liste.

On revient ensuite au ①.

+1

+ 1h

Réservé à la correction :	+1

2. Analyses de synthèses en VHDL

2.1. On considère le code VHDL suivant représentant un automate :

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity DS_Auto_1 is
   Port (    reset : in  STD_LOGIC;
             clock : in  STD_LOGIC;
             var_in : in  STD_LOGIC;
             states : out  STD_LOGIC_VECTOR (2 downto 0);
             var_out : out  STD_LOGIC);
end DS_Auto_1;

architecture my_auto_arch of DS_Auto_1 is
          signal my_actual_state : STD_LOGIC_VECTOR (2 downto 0) := "000";
          signal my_next_state : STD_LOGIC_VECTOR (2 downto 0) := "000";
begin

          my_state_evolution_process : process (reset, clock)
          begin
                    if (reset = '1') then
                              my_actual_state <= "111";
                    elsif (clock'event and clock = '0') then
                              my_actual_state <= my_next_state;
                    end if;
          end process;

          my_state_calulation_process : process (my_actual_state, var_in)
          begin
                    case (my_actual_state) is
                              when "000" =>
                                        if (var_in = '0') then my_next_state <= "111";
                                        else my_next_state <= "001";
                                        end if;
                              when "001" =>
                                        my_next_state <= "010";
                              when "010" =>
                                        if (var_in = '0') then my_next_state <= "001";
                                        else my_next_state <= "011";
                                        end if;
                              when "011" =>
                                        my_next_state <= "100";
                              when "100" =>
                                        if (var_in = '1') then my_next_state <= "101";
                                        else my_next_state <= "011";
                                        end if;
                              when "101" =>
                                        my_next_state <= "110";
                              when "110" =>
                                        if (var_in = '1') then my_next_state <= "111";
                                        else my_next_state <= "101";
                                        end if;
                              when others =>
                                        my_next_state <= "000";
                    end case;
          end process;

          var_out <= '1' when (my_actual_state(1 downto 0) = "00") else '0';

          states <= my_actual_state;

end my_auto_arch;
```


Réservé à la correction :