Correction

SCIENCES ET MONTAGNE
UNIVERSITÉ SAVOIE MONT BLANC

UNIVERSITÉ SAVOIE MONT BLANC

L3 STIC ESET
FPGA AND VHDL (ETRS 501-ETRS507)

Date : 14/12/2021
Durée : 2h

Règles pour l'épreuve :
- Tous les documents papier sont autorisés

Exercice 1 : Cherchez l'erreur !
Un étourdi a écrit le code VHDL suivant et le compilateur lui signale au moins 11 erreurs.
Retrouvez au moins 8 erreurs dont vous êtes surs et expliquez pourquoi la syntaxe est fausse.

fichier-exo1.vhd

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

entity fichier-exo1 is
port(           clk : in std_logic;
                A : in std_logic;
                B : out integer (0 to 64);
                C : out std_logic_vector(8 downto 0);
);
end fichier-exo1;

architecture arch_ of fichier-exo1 is
begin
                process()
                signal S : integer;
                begin
                    if rising_edge(clock)=1 then
                        S:=S-1;
                        B<=S;
                end if;

                C<=11110000 when A='1' ;
                else C<=00001111 when others;
end arch_;
```

Exercice 2 :
Vrai ou faux (**n'oubliez pas de justifier**) :

a - je peux utiliser un FPGA sans horloge externe

b - un FPGA est aussi performant qu'un ASIC

c - je peux configurer un FPGA pour l'utiliser comme carte graphique

d - un signal en sortie de compteur peut être considéré comme un signal d'horloge

e - si je fais une simulation temporelle de mon circuit, je suis certain de tester mon circuit dans les conditions d'utilisation

f - un FPGA est forcément volatile

g - si le temps de propagation le plus long dans mon circuit est de 30 ns, alors celui-ci peut fonctionner à une fréquence d'horloge de 27 MHz

h - un FPGA n'a pas de patte d'alimentation

i - si l'entrée d'un circuit 2 est connectée à la sortie d'un circuit 1 et si la sortie du circuit 1 est déclarée en « integer », alors l'entrée du circuit 2 doit être forcément déclarée en « integer »

j - il est déconseillé d'utiliser le mode « buffer » pour un port. Il faut s'en tenir aux modes « in », « out » ou « inout »

Exercice 3 :

On étudie le code de l'annexe 1.

Q1. Ce composant est décrit en utilisant un paramètre générique. Pourquoi utiliser un tel paramètre ?

Q2. Quelle est la valeur par défaut du paramètre générique ?

Q3. A quoi sert la ligne « use ieee.numeric_std.all; » dans cette description ?

Pour comprendre le fonctionnement de ce bloc, on propose de le simuler. Pour cette simulation, on fixera le paramètre générique à 3.

Q4. Ecrivez le fichier vhdl permettant de simuler le composant. On appliquera en entrée un signal périodique carré de fréquence 100 MHz. On choisir une durée de simulation de 320 ns.

Q5. Dessinez le chronogramme que l'on peut attendre en sortie après une simulation fonctionnelle. Indiquez également la valeur du signal « compteur » tout au long de la simulation.

Q6. Quel autre type de simulation existe-il ? Quelle est la différence avec une simulation fonctionnelle ?

Q7. Le nom de ce composant est « diviseur ». Que divise-t-il et par combien?

Q8. Que faudrait-il modifier dans le code pour diviser par 16 ?

Q9. Quel est l'inconvénient de la méthode employée ici pour décrire un tel composant ?

Exercice 4 : Etude d'un circuit

On donne en annexe les fichiers vhdl d'un projet décrivant un circuit.

1. Etude du Top Level

Q1. Quel type de description est utilisé dans ce fichier ?

Q2. Dessinez de façon claire le schéma décrit par ce fichier. Faites apparaitre les entrées et sorties du circuit et des composants. Précisez bien les tailles des bus utilisés et nommez tous les signaux.

2. Etude du composant clk_div

On utilise l'horloge de la carte DE1 à 50 MHz.

Q3. Par combien faut-il diviser la fréquence de l'horloge pour avoir une fréquence de 9600 Hz en sortie du composant ?

Q4. En déduire la valeur que doit prendre la constante « limite ».

3. Etude du composant FSM

Q5. Quels sont les entrées/sorties et les états de cette machine à états ?

Q6. Dessinez le diagramme d'état.

Q7. A quoi sert l'entrée « enable » ?

Q8. Quelle opération est décrite par la ligne ci-dessous :

```
data_bus <= '0'&data_bus(8 downto 1);
```

Q9. Si data_bus vaut « 101010001 » avant cette ligne, que vaut-il une fois cette ligne exécutée ?

Q10. Comment faudrait-il modifier l'architecture pour ajouter au composant un reset synchrone ?

4. Etude du composant ROM

Comme son nom l'indique, ce composant est une mémoire ROM.

Q11. Quelle est la taille de la mémoire décrite ?

Q12. Si on envoie en entrée l'adresse 5, quelle donnée obtient-on en sortie ?

Q13. Quel est la différence entre une ROM et une RAM ?

Q14. Que faudrait-il modifier dans le code pour transformer la ROM en RAM. Ne pas prendre en compte ces modifications pour la suite de l'exercice.

5. Fonctionnement du circuit

Q15. Complétez les chronogrammes de l'annexe 6. Pour le composant clk_div, on prendra limite=2.

Q16. Expliquez le fonctionnement de ce circuit.

6. Compilation

Q17. Expliquez à quoi correspondent ces 3 étapes de la compilation :

Q18. Expliquez à quoi sert l'étape d'assignation des pattes.

Le rapport de compilation est le suivant :

Flow Summary	
Flow Status	Successful - Mon Dec 13 16:43:47 2021
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Web Edition
Revision Name	exam
Top-level Entity Name	dk_div
Family	Cyclone IV GX
Total logic elements	2 / 14,400 (< 1 %)
Total combinational functions	1 / 14,400 (< 1 %)
Dedicated logic registers	2 / 14,400 (< 1 %)
Total registers	2
Total pins	2 / 81 (2 %)
Total virtual pins	0
Total memory bits	0 / 552,960 (0 %)
Embedded Multiplier 9-bit elements	0
Total GXB Receiver Channel PCS	0 / 2 (0 %)
Total GXB Receiver Channel PMA	0 / 2 (0 %)
Total GXB Transmitter Channel PCS	0 / 2 (0 %)
Total GXB Transmitter Channel PMA	0 / 2 (0 %)
Total PLLs	0 / 3 (0 %)
Device	EP4CGX15BF14C6
Timing Models	Final

Q19. Trouvez dans le rapport de compilation les lignes qui expliquent pourquoi
- il n'est pas possible de télécharger ce projet dans une carte DE1
- même si le projet était téléchargé, le circuit ne fonctionnerait pas comme prévu

ANNEXE 1 : diviseur.vhd

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

entity diviseur is
generic (n : integer := 3);
port(       clk : in std_logic;
              S : out std_logic);
end diviseur;

architecture arch of diviseur is
begin
            process(clk)
            variable compteur : unsigned(9 downto 0);
            begin
            if clk'event and clk='1' then
                  compteur := compteur + 1;
            end if;
            S <= compteur(n);
            end process;
end arch;
```

ANNEXE 2 : topevel.vhd

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

entity toplevel is
port(  clk : in std_logic;
     D_out : out std_logic);
end toplevel;

architecture toplevel_arch of toplevel is

signal enable : std_logic;
signal code_lettre : std_logic_vector(6 downto 0);
signal adresse_lettre : integer range 0 to 25;

component clk_div
port(  clk : in std_logic;
          S : out std_logic);
end component;

component FSM
port(H, clk_enable : in std_logic;
  lettre_a_envoyer : in std_logic_vector(6 downto 0);
                data : out std_logic;
             adresse : out integer range 0 to 25);
end component;

component ROM
port   (addr : in integer range 0 to 25;
          clk  : in std_logic;
            q : out std_logic_vector(6 downto 0));
end component;

begin

comp1: clk_div
port map (clk, enable);

comp2: ROM
port map(adresse_lettre, clk,code_lettre);

comp3: FSM
port map (clk,enable, code_lettre, D_out,adresse_lettre);
end toplevel_arch;
```

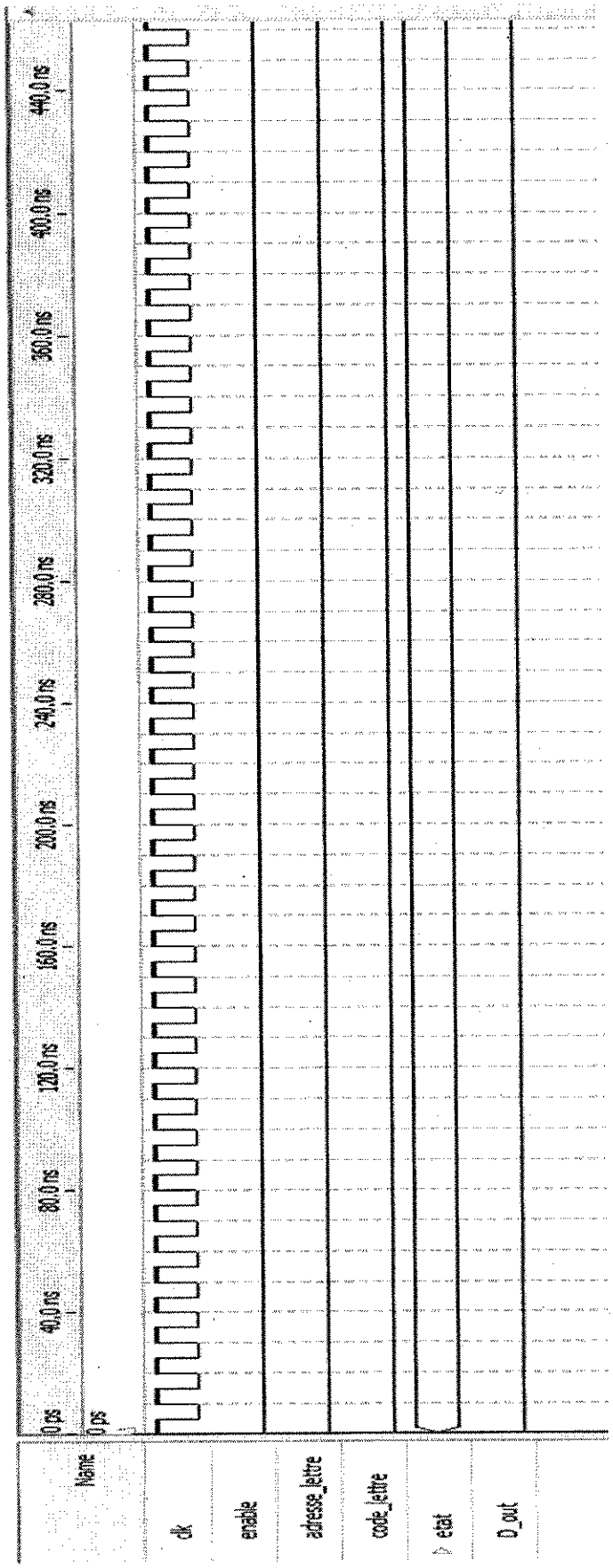
ANNEXE 3 : clk_div.vhd

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

entity clk_div is
port( clk : in std_logic;
      S : out std_logic);
end clk_div;

architecture arch of clk_div is
constant limite : integer := ?;
begin
    process(clk)
    variable compteur : integer range 0 to limite-1;
    begin
        if clk'event and clk='1' then
            if compteur = limite -1 then
                compteur := 0;
                S<='1';
            else
                compteur := compteur + 1;
                s<='0';
            end if;
        end if;
    end process;
end arch;
```

ANNEXE 4 : FSM.vhd

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

entity FSM is
port (H, clk_enable : in std_logic;
   lettre_a_envoyer : in std_logic_vector(6 downto 0);
               data : out std_logic;
           adresse : out integer range 0 to 25);
end FSM;


architecture FSM_arch of FSM is

type etat_type is (debut, envoi_serie_lettre, fin_lettre);

signal etat : etat_type;
signal compteur : integer range 0 to 8;
signal data_bus : std_logic_vector(8 downto 0);
signal lettre : integer range 0 to 25 := 0;

begin
  adresse<=lettre;
     process (H)
     begin
        if H'event and H='1' then
            if clk_enable='1' then
                case etat is
                    when debut =>  compteur <=0;
                                        data <= '1';
                                   data_bus <= '1' & lettre_a_envoyer & '0';
                                        etat <= envoi_serie_lettre;

                     when envoi_serie_lettre =>  compteur <= compteur+1;
                                                      data <= data_bus(0);
                                                 data_bus <= '0'&data_bus(8 downto 1);
                                                 if compteur = 8 then
                                                     etat <= fin_lettre;
                                                 else
                                                     etat <= envoi_serie_lettre;
                                                 end if;

                      when fin_lettre => lettre <= lettre+1;
                                         etat <= debut;
                   end case;
                end if;
            end if;
     end process;
end FSM_arch;
```

ANNEXE 5 : ROM.vhd

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

entity rom is

        port ( addr : in integer range 0 to 25;
                clk : in std_logic;
                  q : out std_logic_vector(6 downto 0));
end rom;

architecture arch of rom is

type memory_t is array(0 to 25) of std_logic_vector(6 downto 0);

constant memoire : memory_t := (0=>"1000001", 1=>"1000010", 2=>
"1000011", 3=>"1000100", 4=>"1000101", 5=>"1000110",6=>"1000111",
7=>"1001000", 8=>"1001001", 9=>"1001010", 10=> "1001011",
11=>"1001100", 12=>"1001101", 13=>"1001110", 14=>"1001111",
15=>"1010000",16=>"1010001", 17=>"1010010", 18=> "1010011",
19=>"1010100", 20=>"1010101", 21=>"1010110", 22=>"1010111",
23=>"1011000",24=>"1011001", 25=>"1011010");

begin
      process(clk)
      begin
          if (rising_edge(clk)) then
             q <= memoire(addr);
          end if;
      end process;
end arch;
```

ANNEXE 6 : Chronogramme