Travail écrit 1: Système Numérique 2 2017/2018

Filière : Télécommunication
Classe : T-2a, T-2d
Date : 10 avril 2018, 15:00 à 16:35
Professeur : Fabio Cunha
Nom et prénom : Zamboni Yannick

Points : 19.75 /20 **Note :** 5.9

Problème 1 : Théorie sur les systèmes numériques (7 pts) 7

a) Dans le monde des systèmes numériques, il existe plusieurs familles. Citez ces familles et expliquez quelles sont leurs avantages/inconvénients.

- Circuit standards : circuit/chip basé uniquement sur les portes logiques de base. Ne permet que de concevoir efficacement des algorithmes très simples et peut rapidement prendre beaucoup de places. ✓
- PLD : circuits programmables allant de la ROM à la FPGA. Les "unités" sont déjà en place et on programme les liens entre celles-ci. Adapté pour des petits projets peu complexes, mais peut devenir très vite limitant si on a besoin de ~~[illegible]~~ concevoir des circuits un peu plus fonctionnalités. ✓
- Asic : Circuit où la liberté de conception est bien plus grande et où on travaille sur plusieurs "couches" d'implémentation tout en permettant son propre mapping/routing. ✓ C'est optimal pour les gros projets complexes car on ne fait aucun "gaspillage" et que le circuit est "unique".

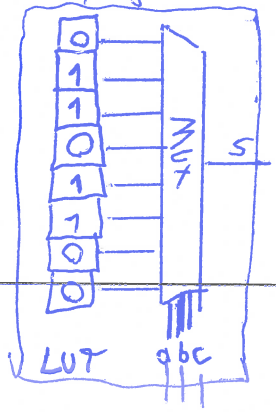
2/ 2 pts

b) Expliquez le fonctionnement et l'utilité du composant look-up-table (LUT) dans une FPGA. Illustrez vos propos par un schéma si nécessaire.

Mais le développement est très long et coûteux

1/ 1 pt

Plutôt qu'utiliser des portes logiques pour implémenter une "table de vérité", on utilise une LUT qui utilise un multiplexeur pour associer des entrées à des sorties et des 'mémoires' programmables :



est équivalent à la table de vérité

a	b	c	S
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

c) Expliquez à quoi sert le fichier .ucf dans un projet VHDL. Quels types d'informations peut-il contenir? Donnez deux exemples d'informations.

Le fichier de contrainte utilisateur permet entre autre d'assigner un I/O à un pin de la FPGA: On peut donc y trouver, par exemple : — A quel pin est associé un IO
— Quelle contrainte est forcée sur tel ou tel I/O (Buffer, type de pin, ...)

1/ **1 pt**

d) « La technologie FPGA est venu combler un manque dans le monde des systèmes numériques ». Expliquez cette affirmation.

Les PLDs sont adaptés pour les petits projet simples
Les Asic sont adaptés pour les gros projets complexes.

Entre deux, avant la FPGA, il n'y avait pas grand chose !

1/ **1 pt**

e) Quand est-ce qu'un glitch statique se produit ? Quelles solutions proposez-vous pour supprimer ce phénomène ?

Il se produit quand une entrée a momentanément une valeur et sa valeur inverse en même temps ?

1/ **1 pt**

Pour supprimer ce problème, on doit parfois utiliser des portes "redondantes" qui ~~[illegible]~~ ne changent pas lors des passages de valeurs critiques.
Dans une table de Karnaugh, cela revient à faire des regroupements :

f) Pourquoi est-il judicieux de faire une simulation post-routage ? Quelles sont les informations supplémentaires fournies avec cette simulation en comparaison avec la simulation fonctionnelle?

Cette simulation prend en compte le temps que mette les signaux pour "réagir" à un changement en fonction de leur distance géographique sur la fpga et de la distance à parcourir sur les routes implémentées. temps de propagation

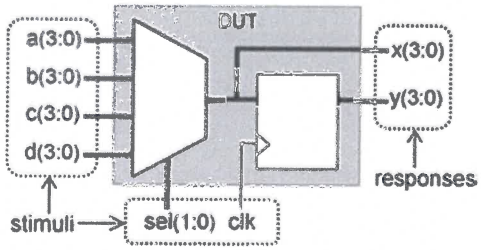
1/ **1 pt**

Cela peut mettre à jour des glitchs qui sont difficilement discernables à la synthèse.

Problème 2: Testbench (7 pts)

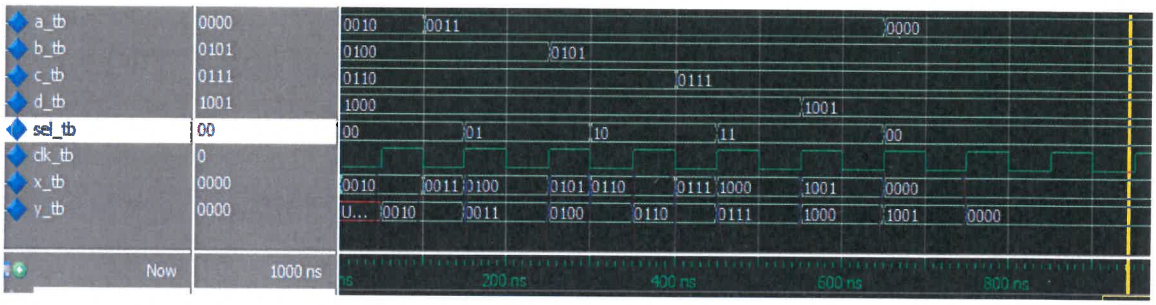
6.75

On veut tester le circuit ci-dessous, composé d'un registre flip-flop et d'un multiplexeur. Le code VHDL du circuit est décrit ci-contre :




```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
-------------------------------------------------------
ENTITY reg_mux IS
    PORT (  a, b, c, d: IN STD_LOGIC_VECTOR(3 DOWNTO 0);
            sel: IN STD_LOGIC_VECTOR(1 DOWNTO 0);
            clk: IN STD_LOGIC;
            x, y: OUT STD_LOGIC_VECTOR(3 DOWNTO 0));
END ENTITY;
-------------------------------------------------------
ARCHITECTURE reg_mux OF reg_mux IS
    SIGNAL mux: STD_LOGIC_VECTOR(3 DOWNTO 0);
BEGIN
    mux <=  a WHEN sel="00" ELSE
            b WHEN sel="01" ELSE
            c WHEN sel="10" ELSE
            d;
    x <= mux;
    PROCESS (clk)
    BEGIN
        IF (rising_edge(clk)) THEN
            y <= mux;
        END IF;
    END PROCESS;
END ARCHITECTURE;
```

Et la simulation que l'on veut effectuer est illustrée sur l'image ci-dessous :



- Écrivez le code VHDL du testbench correspondant à cette simulation. Ce modèle n'est pas paramétrable (aucun générique).

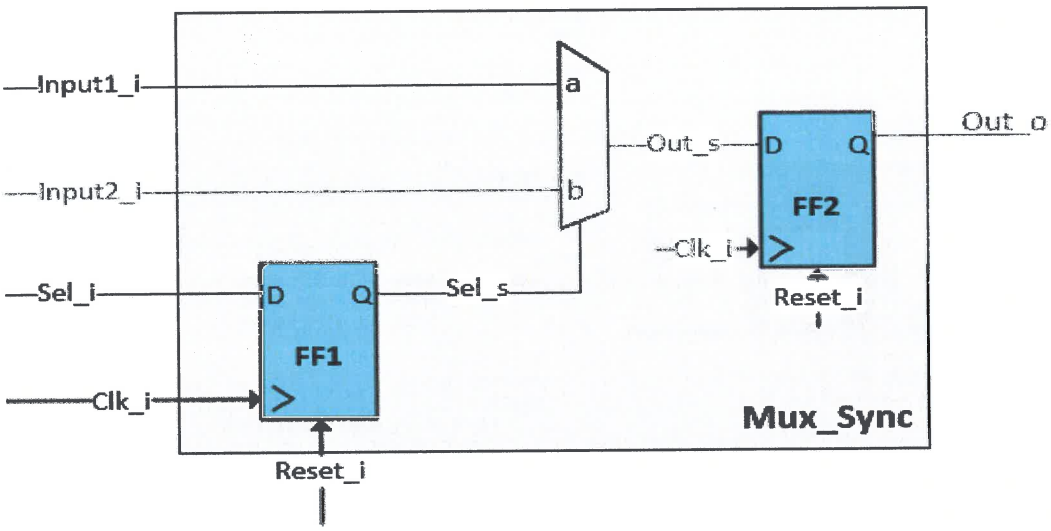
4.75/ **5 pts**

- Utilisez les instructions "assert" et "report" pour vérifier l'état des sorties

2/ **2 pts**

Problème 3 : Testbench (6 pts) 6

Soit le circuit suivant:



Vous devez :

a) Définir, à l'aide d'un chronogramme précis, le testbench que vous voulez réaliser afin de vérifier le bon fonctionnement du circuit de manière complète (toutes les possibilités seront vérifiées)
NB : Votre testbench commence par activer le reset_i des flip-flops D.

2/ **2 pts**

b) Écrire le code VHDL du testbench que vous avez défini au point b).
NB : Vous devez utiliser les instructions assert et report afin de vérifier que la sortie Out_o possède la bonne valeur.

4/ **4 pts**

Zanton Yannick

TEO3 Sys. Num. 2

Problème 2

```
library IEEE;
use IEEE.STD_logic_1164.all;

entity tb is
end tb;

architecture Behavioral of tb is

component reg_mux is
    Port ( a,b,c,d : in std_logic_vector (3 downto 0);
           sel : in std_logic_vector (1 downto 0);
           clk : in std_logic;
           x,y : out std_logic_vector (3 downto 0));
end component;

SIGNAL a_tb, b_tb, c_tb, d_tb : std_logic_vector (3 downto 0);
SIGNAL sel_tb : std_logic_vector (1 downto 0);
SIGNAL clk_tb : std_logic;
Constant clk_period : time := 100 ns;

begin

uut : reg_mux
    Port map ( a => a_tb, b => b_tb, c => c_tb, d => d_tb,
               sel => sel_tb, clk => clk_tb, x => x_tb, y => y_tb );

clk_signal : process
begin
    clk_tb <= '0';
    wait for clk_period/2;
    clk_tb <= '1';
    wait for clk_period/2;
end process;

a_signal : process
begin
    a_tb <= "0010";
    wait for clk_period;
    a_tb <= "0011";
    wait for 550 ns;
    a_tb <= "0000";
    wait;
end process;

b_signal : process
begin
    b_tb <= "0100";
    wait for 250 ns;
    b_tb <= "0101";
    wait;
end process;
```

```
c_signal : process
begin
    c_tb <= "0110";
    wait for 400 ns;
    c_tb <= "0111";
    wait;
end process;
```

```
d_signal : process
begin
    d_tb <= "1000";
    wait for 550 ns;
    d_tb <= "1001";
    wait;
end process;
```

```
sel_signal : process
begin
    sel_tb <= "00";
    wait for 150 ns;
    sel_tb <= "01";
    wait for 150 ns;
    sel_tb <= "10";
    wait for 150 ns;
    for i in 0 to 3 loop
        sel_tb <= std_logic_vector(to_unsigned(i,2));
        wait for 150 ns;
    end loop;
    sel_tb <= "00";
    wait;
end process;
```

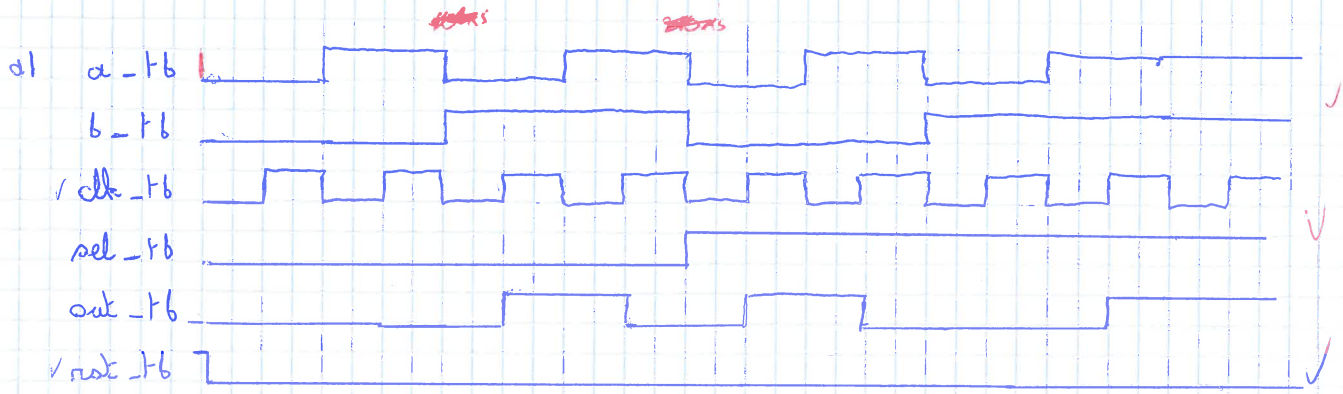
```
sortie : process
begin
    assert (x_tb = "0010" and y_tb = "0000") report "1";
    wait for clk_period/2;
    assert (x_tb = "0010" and y_tb = "0010") report "2";
    wait for clk_period/2;
    assert (x_tb = "0011" and y_tb = "0010") report "3!";
    wait for clk_period/2;
    assert (x_tb = "0100" and y_tb = "0011") report "4";
    wait for clk_period;
    assert (x_tb = "0100" and y_tb = "0011") report "5";
    wait for clk_period/2;
    assert (x_tb = "0101" and y = "0100") report "6";
    wait for clk_period/2;
    assert (x_tb = "0110" and y_tb = "0100") report "7";
    wait for clk_period/2;
    assert (x_tb = "0110" and y_tb = "0110") report "8";
    :
    assert (x_tb = "0000" and y_tb = "1001") report "11";
    wait for clk_period;
    assert (x_tb = "0000" and y_tb = "0000") report "12";
    wait;
end process;
end architecture;
```

Zambon Yaniek T2-a (3)

Problème 3

clk_période : 100 ns

a)



b)

```
library IEEE;
use IEEE.STD_Logic_1164.all;
entity tb is
end tb;
architecture behavioral of tb is
  component tb_3 is
    PORT ( input1_i : in std_logic;
           input2_i : in std_logic;
           sel_i, reset_i, clk_i : in std_logic;
           out_o : out std_logic);
  end component

  Signal a_tb, b_tb, clk_tb, sel_tb, out_tb, rst_tb : std_logic := '0';
  constant clk_period : time := 100 ns;
begin
  uut : tb_3
    PORTMAP ( input1_i => a_tb, input2_i => b_tb, sel_i => sel_tb, reset_i => rst_tb,
              clk_i => clk_tb, out_o => out_tb);

clk_signal : process
begin
  clk_tb <= '0'; wait for clk_period/2;
  clk_tb <= '1'; wait for clk_period/2;
end process;

rst_signal : process
begin
  rst_tb <= '1'; wait for 10 ns;
  rst_tb <= '0'; wait;
end process;
```

```
a_signal : process
begin
    for i in 0 to 3 loop
        a_tb <= '0';
        wait for clk_period;
        a_tb <= '1';
        wait for clk_period;
    end loop;
    wait;
end process;
```

```
b_signal : process
begin
    for i in 0 to 1 loop
        b_tb <= '0';
        wait for (2 * clk_period);
        b_tb <= '1';
        wait for (2 * clk_period);
    end loop;
    wait;
end process;
```

```
sel_signal : process
begin
    sel_tb <= '0';
    wait for 4000 ns;
    sel_tb <= '1';
    wait;
end process;
```

```
output : process
begin
    for i in 0 to 4 loop
        assert (out_tb = '0') report "1";
        wait for clk_period/2;
    end loop;
    for i in 0 to 1 loop
        assert (out_tb = '1') report "2";
        wait for clk_period;
        assert (out_tb = '0') report "3";
        wait for clk_period;
    end loop;
    assert (out_tb = '0') report "4";
    wait for clk_period;
    assert (out_tb = '1') report "5";
    wait for clk_period;
    assert (out_tb = '1') report "6";
end process;
end architecture;
```