Nom :
Prénom :
Groupe TP :

/20

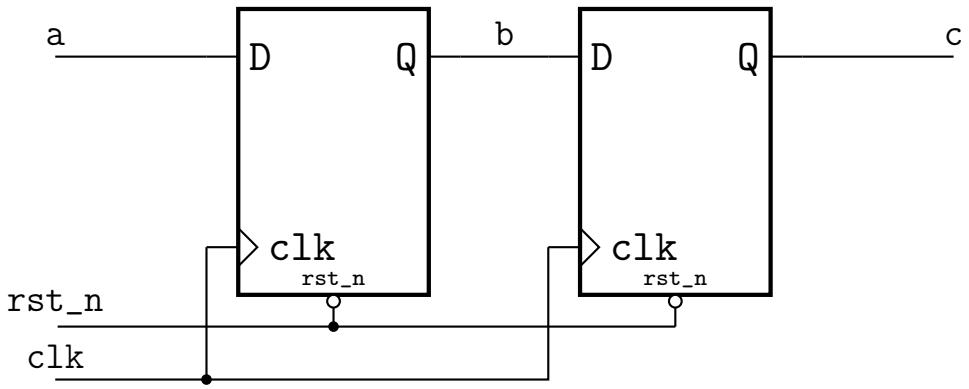
Examen Auto1
Lundi 15 janvier 2024

Aucun document autorisé. Calculatrice non autorisée. Durée : 1h30.
Les exercices sont indépendants et peuvent donc être traités dans n'importe quel ordre.
*Toutes les réponses doivent être **justifiées**.*

1 Questions de cours

/3

Soit le circuit numérique ci-dessous :

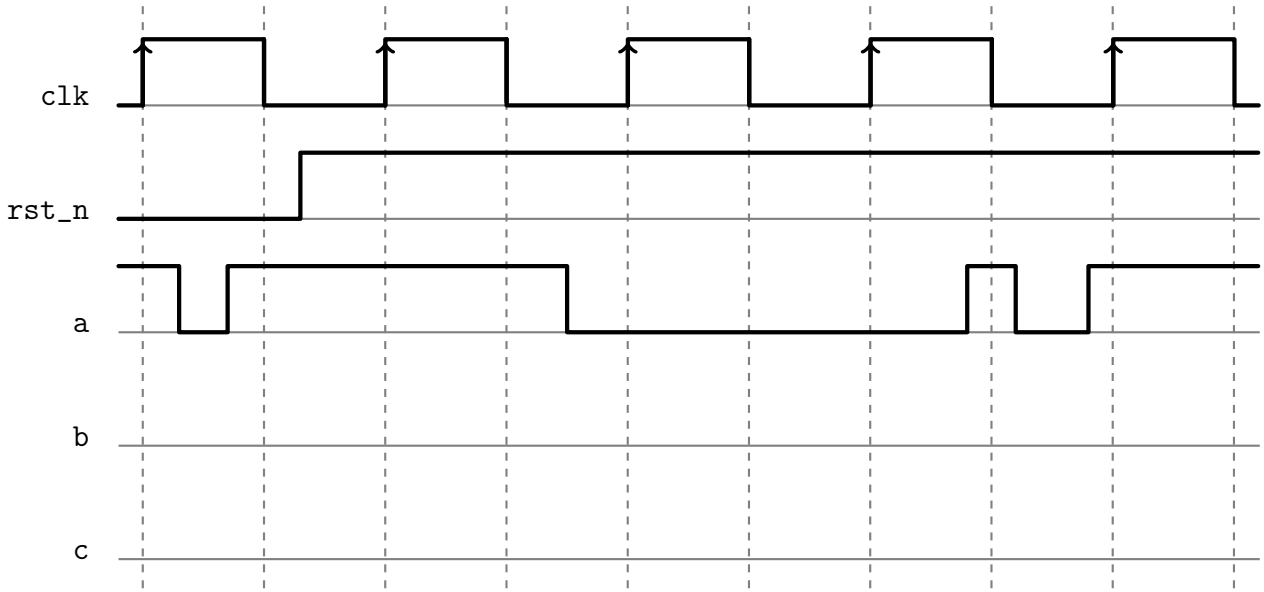


1. Ce circuit est-il synchrone ?

/0,5

2. Compléter dans le chronogramme suivant les valeurs des signaux `b` et `c` :

/1



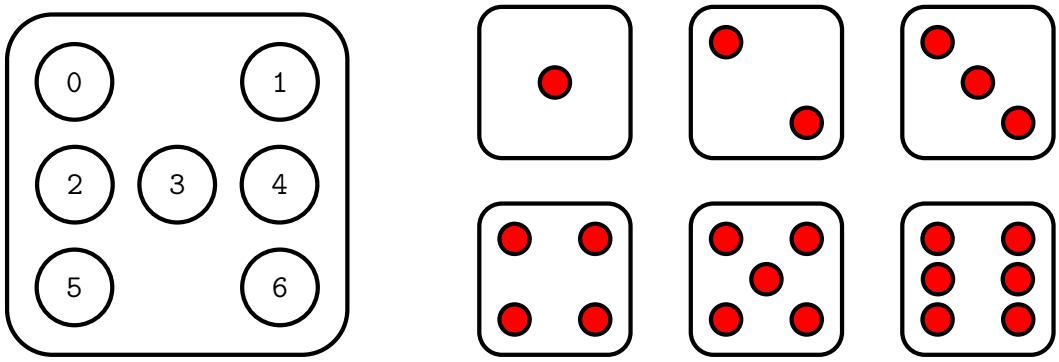
3. Dessiner le schéma correspondant au code VHDL suivant : /1,5

```
PROCESS (clk, rst_n)
BEGIN
  IF rst_n = '0' THEN
    s <= '0';
    f <= '0';
  ELSE
    IF rising_edge(clk) THEN
      f <= a;
      s <= f;
    END IF;
  END IF;
END PROCESS;

b <= f XOR s;
```

2 Simulation d'un lancer de dé /14

On souhaite concevoir un circuit permettant de simuler un lancer de dé. Ce circuit sera connecté à un dé lumineux, qui comprend 7 LED numérotées de 0 à 6 et disposées comme indiqué en Figure 1a. Une LED s'allume lorsqu'un niveau logique **haut** est appliqué sur l'entrée associée. Les six affichages possibles sont donnés en Figure 1b. Il est également possible de n'allumer aucune LED, pour un dé qui n'a pas encore été lancé.



(a) Dé lumineux avec les LED numérotées

(b) Six affichages possibles. Il est aussi possible d'éteindre toutes les LED.

FIGURE 1 – Afficheur de dé à 7 LED

Ce circuit sera composé :
— d'un compteur avec autorisation de comptage, dont la valeur correspondra à la valeur du dé,
— d'un décodeur, qui commandera l'allumage des LED.

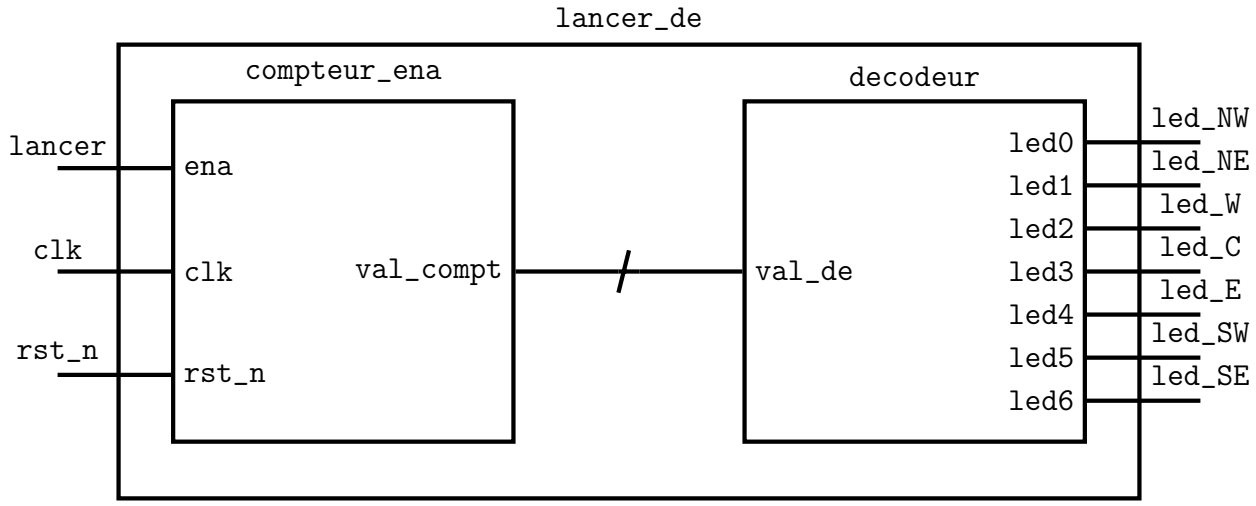


FIGURE 2 – Simulateur de lancer de dé

Pour utiliser le circuit, il faut appuyer pour une certaine durée sur un bouton, actif à l'état haut et connecté à l'entrée `lancer`. La fréquence d'horloge du circuit étant de 50 MHz, comme sur la carte Altera DE1 utilisée en TP, on considérera que la durée d'appui est aléatoire.

Le compteur sera traité dans la partie 2.1, le décodeur dans la partie 2.2, puis le système global dans la partie 2.3. Ces parties se suivent mais peuvent être traitées indépendamment.

2.1 Logique séquentielle /6,5

1. Déterminer la largeur du bus reliant le compteur au décodeur. /0,5

2. Compléter la partie `ENTITY` de la description en VHDL du compteur. /0,5

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

ENTITY compteur_ena IS

  PORT (
    clk     : IN STD_LOGIC;
    rst_n   : IN STD_LOGIC;
    ena     : IN STD_LOGIC;

    val_cpt :

    );

END compteur_ena;
```

3. Le signal de reset de la question suivante est-il synchrone ou asynchrone ? Comment cela est-il spécifié en VHDL ? /0,5

4. Compléter la partie ARCHITECTURE de la description en VHDL du compteur. /3,5

```
ARCHITECTURE archi OF compteur_ena IS


BEGIN

  PROCESS (clk, rst_n)
  BEGIN
    IF rst_n = '0' THEN


    ELSE
      IF rising_edge(clk) THEN









      END IF;
    END IF;
  END PROCESS;

END archi;
```

5. On souhaite ajouter une entrée permettant de remettre à zéro le système de manière **synchrone**, pour pouvoir remettre la valeur du dé à zéro et ne pas avoir à utiliser le signal de reset pour cela. La ligne suivante a été ajoutée à la partie `ENTITY` de la description en VHDL du compteur.

```
remise_a_zero : IN STD_LOGIC;
```

Donner les lignes à ajouter à la partie `ARCHITECTURE` de la description en VHDL du compteur permettant de gérer cette nouvelle entrée. Vous ajouterez une astérisque (*) dans le code que vous avez écrit à la question précédente pour indiquer où ces lignes doivent être ajoutées. /1,5

2.2 Logique combinatoire /5

1. Compléter ci-dessous la table de vérité du décodeur. Vous ajouterez autant de colonnes que nécessaire pour représenter tous les bits du bus d'entrée. /1

	`led0`	`led1`	`led2`	`led3`	`led4`	`led5`	`led6`

2. Par un tableau de Karnaugh, donner l'équation logique simplifiée de la sortie `led6`. /1,5

3. Donner la ligne de VHDL qui permettrait d'implanter directement cette équation logique, sous forme d'une affectation inconditionnelle. /0,5

4. L'affectation inconditionnelle ci-dessus n'est pas très explicite. Il faudrait procéder autrement pour décrire les autres sorties `led0` à `led5`. Complétez l'architecture de la description en VHDL du décodeur ci-dessous, en utilisant une affectation plus explicite, permettant de définir les valeurs prises par toutes les sorties de `led0` à `led6`. /2

```
END decodeur;

ARCHITECTURE archi OF decodeur IS


BEGIN
```

```
END archi;
```

2.3 Conception structurelle /2,5

1. Dans l'architecture ci-dessous, déclarer le ou les signaux internes nécessaires. /0,5

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

ENTITY lancer_de IS

  PORT (
    clk    : IN  STD_LOGIC;
    rst_n  : IN  STD_LOGIC;
    lancer : IN  STD_LOGIC;
    led_C  : OUT STD_LOGIC;
    led_E  : OUT STD_LOGIC;
    led_NE : OUT STD_LOGIC;
    led_N  : OUT STD_LOGIC;
    led_NW : OUT STD_LOGIC;
    led_W  : OUT STD_LOGIC;
    led_SW : OUT STD_LOGIC;
    led_S  : OUT STD_LOGIC;
    led_SE : OUT STD_LOGIC);
END lancer_de;

ARCHITECTURE archi OF lancer_de IS

  COMPONENT compteur_ena
    PORT (
      clk       : IN STD_LOGIC;
      rst_n     : IN STD_LOGIC;
      ena       : IN STD_LOGIC;
      val_cpt   :                          -- d'apres les questions precedentes
      );
  END COMPONENT;

  COMPONENT decodeur
    PORT (
      val_seg  :                           -- d'apres les questions precedentes
      led0     : OUT STD_LOGIC;
      led1     : OUT STD_LOGIC;
      led2     : OUT STD_LOGIC;
      led3     : OUT STD_LOGIC;
      led4     : OUT STD_LOGIC;
      led5     : OUT STD_LOGIC;
      led6     : OUT STD_LOGIC;
```

```
        led7      : OUT STD_LOGIC;
        led8      : OUT STD_LOGIC);
    END COMPONENT;
```

2. Terminer la description du système global, en instanciant les `COMPONENT` notamment. /2

```
BEGIN
```

```
END archi;
```

3 Machine d'états /3

L'architecture d'un fichier VHDL décrivant une machine d'états est donné ci-dessous, et se poursuit sur la page suivante.

```
1  ARCHITECTURE archi OF controller IS
2
3    TYPE etat IS (attente, initialisation, stop, traitement, final);
4    SIGNAL etat_actuel : etat;
5
6    SIGNAL ack_s       : STD_LOGIC;
7    SIGNAL arret_s     : STD_LOGIC;
8    SIGNAL done_s      : STD_LOGIC;
9    SIGNAL demarrage_s : STD_LOGIC;
10
11 BEGIN
12
13   PROCESS (clk, rst_n)
14   BEGIN
15     IF rst_n = '0' THEN
16       etat_actuel <= initialisation;
17     ELSE
18       IF rising_edge(clk) THEN
19         ack_s       <= ack;             --
20         arret_s     <= arret;
21         demarrage_s <= demarrage;
22         done_s      <= done;            --
23         CASE etat_actuel IS
24           WHEN attente =>
25             IF demarrage_s = '1' THEN
26               etat_actuel <= traitement;
27             END IF;
28           WHEN initialisation =>
29             etat_actuel <= attente;
30           WHEN stop =>
31             IF ack_s = '1' THEN
32               etat_actuel <= final;
33             END IF;
34           WHEN traitement =>
35             IF arret_s = '1' THEN
36               etat_actuel <= stop;
37             ELSE
38               IF done_s = '1' THEN
39                 etat_actuel <= attente;
40               END IF;
41             END IF;
42           WHEN final =>
43             etat_actuel <= attente;
44           WHEN OTHERS =>
45             etat_actuel <= attente;
46         END CASE;
47       END IF;
48     END IF;
49   END PROCESS;
50
```

```
51    speed <= "10" WHEN etat_actuel = initialisation ELSE
52             "11" WHEN etat_actuel = traitement ELSE
53             "01" WHEN etat_actuel = final ELSE
54             "00";
55
56    verrou <= '0' WHEN etat_actuel = attente ELSE '1';
57
58  END archi;
```

1. Quel est l'intérêt des affectations réalisées aux lignes 19 à 22 ? /0,5

2. Dessiner le diagramme états-transitions de la machine. /2,5