Réalisation partielle d'un réveil

Règles de déroulement de l'épreuve

Il n'est pas interdit de communiquer verbalement entre binômes tant que cela ne perturbe pas le déroulement correct de cette séance.
Il vous est absolument interdit, par contre, de vous déplacer (sauf urgence), d'échanger des feuilles de brouillons, des notes ainsi que des fichiers et des clés USB.
Les téléphones portables sont interdits pendant toute la durée de l'épreuve.

Les fichiers nécessaires à la réalisation complète de cet examen médian sont comme d'habitude disponibles dans le répertoire median2017 du fichier ressource2015.zip disponible sur le site personnel de Serge Moutou

I) Introduction

Notre objectif est de réaliser un réveil heures/minutes sur l'écran LCD. Dans un premier temps on s'occupera de l'heure courante celle qui défile constamment (pas à vitesse réelle pour que vos temps d'essais ne soient pas interminables). Trois exercices très simples vous sont proposés pour vous familiariser avec l'affichage LCD et les compteurs BCD.

Dans un deuxième temps on réalisera un compteur Heure Minutes de l'heure courante et son affichage que l'on notera HH:MM.

Dans un troisième temps, on s'occupera du réglage de l'heure de réveil à l'aide d'un bouton rotatif à codeur incrémental.

Dans un quatrième temps on essaiera de réaliser un mécanisme de sonnerie.

II) Première Partie

On vous donne complètement le module qui affiche du texte sur l'afficheurLCD. Pour cela, nous allons essayer dans cette section, d'utiliser un module VHDL touvé chez Opencores. Il se trouve dans [http://opencores.org/project,16x2_lcd_controller Projets -> Other -> 16x2 LCD controller] et a été réalisé par **Daniel Drescher**. Pour éviter d'aller les chercher sur Internet, ces fichiers vous sont fournis dans ressources2015.zip avec trois fichiers spécifiés dans l'énoncé de l'exercice 1.

Exercice 1

1°) On vous demande de réaliser un projet à l'aide des trois fichiers de départ pour montrer votre capacité à réaliser un projet. Il n'y a rien à changer dans les fichiers à ce stade.

2°) Modifier le texte affiché et **faire constater par l'enseignant**.

Le projet utilise les trois fichiers suivants

- lcd16x2_ctrl.vhd

- lcd16x2_ctrl_demo.vhd
- lcd16x2_ctrl_demo.ucf

Le fichier lcd16x2_ctrl.vhd gère l'interface physique entre le FPGA et l'afficheur LCD et **ne sera jamais modifié** ! Seuls les deux autres le seront au fur et à mesure du déroulement de l'épreuve.

Exercice 2

Notre objectif étant d'afficher des heures minutes, nous allons spécialiser un peu cet afficheur. Dans un premier temps, on le destine à afficher les 4 digits d'un compteur binaire. En clair il doit afficher entre 0000 et FFFF. Mais l'exercice précédent vous a montré que l'afficheur lcd ne connaît que les codes ASCII. Vous allez réaliser un circuit combinatoire qui réalise le transcodage nécessaire. Comme il nous faut garder le gestionnaire de l'écran lcd, nous vous proposons un fichier lcd16x2_ctrl_demo_exo2.vhd modifié dans lequel on vous demande de modifier la partie combinatoire en fin de fichier. Il s'agit de réaliser un transcodeur : c'est donc du combinatoire et peut donc se réaliser avec un « with select when ».

Le projet utilise les trois fichiers suivants

- lcd16x2_ctrl.vhd
- lcd16x2_ctrl_demo_exo2.vhd
- lcd16x2_ctrl_demo.ucf

Faire constater à l'enseignant.

Exercice 3

Le problème du transcodeur réalisé dans l'exercice 2 de l'épreuve peut être facilement résolu si l'on remplace les compteurs hexadécimaux par des compteurs BCD (qui ne comptent que de 0 à 9). C'est ce que fait le nouveau code de lcd16x2_ctrl_demo_exo3.vhd. Vous y remarquerez la disparition du transcodeur de l'exercice 2. Essayez de comprendre pourquoi.

Dans cet exercice, on vous demande simplement d'insérer un ":" entre les deux premiers digits et les deux derniers. On vous rappelle en effet que l'on finira par afficher les heures et minutes (qui en général sont séparées par ce ":" avec HH:MM) d'où le titre de cette épreuve.

Le projet utilise les quatre fichiers suivants

- lcd16x2_ctrl.vhd
- lcd16x2_ctrl_demo_exo3.vhd
- counterBCD.vhd
- lcd16x2_ctrl_demo.ucf

Faire constater à l'enseignant.

III) Deuxième partie : réalisation des heures minutes

L'arithmétique des heures et minutes n'est pas complètement du BCD.

Exercice 4

Le comptage des minutes se fait de 00 à 59. Les unités sont donc du BCD.

Pour compter les dizaines des minutes, il nous faut un compteur modulo 6 qui soit cascadable. Modifier le compteur BCD de l'exercice 3 pour qu'il compte de 0 à 5 et génère un signal ENO quand il est à 5 et quand son entrée EN est à 1.

Vous aurez deux compteurs à ce stade : un compteur BCD et un compteur modulo 6 et **les deux seront utilisés**.

Le projet utilise les cinq fichiers suivants

- lcd16x2_ctrl.vhd
- lcd16x2_ctrl_demo_exo3.vhd
- counterBCD.vhd
- counterModulo6.vhd
- lcd16x2_ctrl_demo.ucf

Faire constater à l'enseignant.

Exercice 5

Pour éviter la prolifération des fichiers dans la suite de cette épreuve, on vous demande de dépalcer les fichiers :

- counterBCD.vhd
- counterModulo6.vhd

à la fin du fichier « lcd16x2_ctrl_demo_exo3.vhd »

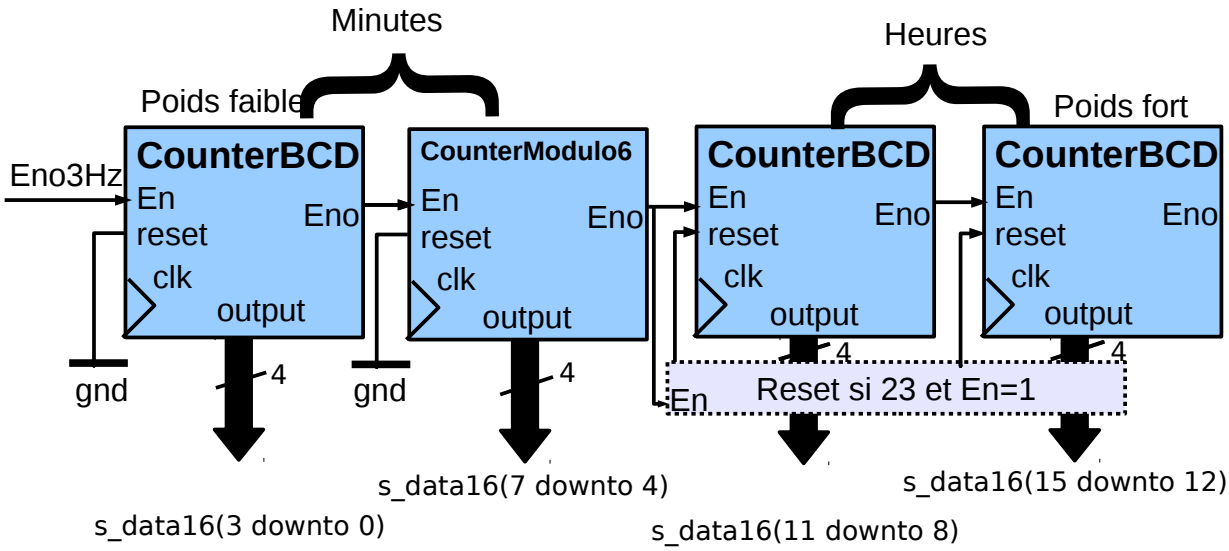
Nous allons ajouter un comptage correct des heures au compteur de l'exercice 4 qui compte normalement la partie minutes.
Le comptage des heures est lui aussi très particulier. Les unités nécessitent un compteur décimal mais qui doit être interrompu lorsque les dizaines passent à 2 ! On ne doit pas alors dépasser 3. Nous allons utiliser un montage bouclé pour réaliser cela : lorsque l'on est à 23, un EN='1' fait repasser les heures à 0. Ce n'est pas la peine de le faire pour les minutes, cela sera automatique.

Indications : la gestion du bouclage peut être réalisé par quelque chose du genre :

```
HeurUnit: CounterBCD port map(
        EN => s_eno2,
        Clock => clk,
        Reset => s_reset,
        ENO => s_eno3,
        Output => s_data16(11 downto 8));
 HeurDiz: CounterBCD port map(
        EN => s_eno3,
        Clock => clk,
        Reset => s_reset,
        ENO => open,
        Output => s_data16(15 downto 12));

 -- reset quand 23 est detecte       par rebouclage
   s_eno2_heure <= s_eno2 & s_data16(15 downto 8);
   with s_eno2_heure select
         s_reset <= '1' when "100100011",
             '0' when others;
```

Voici le schéma correspondant :



Faire constater à l'enseignant.

IV) Troisième partie : Utilisation d'un codeur incrémental

On trouve chez Xilinx un exemple de gestion du codeur incrémental. Celui-ci déplace une led allumée parmi les 8 leds éteintes et une led éteinte parmi les huit allumées si l'on appuie sur le codeur.

Exercice 6

On vous demande de compiler cet exemple et de le faire fonctionner. Tout est fourni, fichier VHDL et fichier UCF (left_right_leds.vhd et left_right_leds.ucf dans les ressources).

Faire constater à l'enseignant.

Le projet utilise les deux fichiers suivants

- left_right_leds.vhd
- left_right_leds.ucf

Soyez curieux et lisez attentivement le fichier UCF. La façon de déclarer les entrées correspondantes du décodeur comme PULLUP ou PULLDOWN est très importante sous peine de non fonctionnement. **On l'utilisera dans l'exercice suivant**

Exercice 7

A ce stade vous disposez de deux fonctions séparées :

- compteur heures minutes de l'exercice 5
- codeur incrémental de l'exercice 6

Pour utiliser le codeur incrémental pour régler la nouvelle heure de réveil, il vous faut d'abord réaliser un nouveau compteur/décompteur HH:MM. Pour vous éviter ce long travail, un tel compteur décompteur vous est fourni en ressource dans ReveilHHMMUp_Down.vhd.

1°) Insérer le composant ReveilHHMMUp_Down du fichier ReveilHHMMUp_Down.vhd dans votre projet pour que l'heure réveil soit affichée juste sous l'heure courante (en 2° ligne). Réaliser l'entrée ud (up/down) sur un switch pour vérifier le bon fonctionnement du comptage décomptage des heures/minutes. L'entrée « en » sera reliée au signal « eno10Hz » comme pour l'heure courante (pour le moment).

Le projet utilise les quatre fichiers suivants

- lcd16x2_ctrl.vhd
- lcd16x2_ctrl_demo_exo5.vhd (le votre)
- ReveilHHMMUp_Down.vhd
- lcd16x2_ctrl_demo.ucf

Faire constater à l'enseignant.

2°) Il est possible d'utiliser directement le codeur incremental de l'exercice 6 pour régler le compteur des heures minutes de l'heure de réveil. Si on tourne dans un sens on incrémente le compteur HH:MM, si on tourne dans l'autre sens on décrémente. Il vous faut donc essayer de produire un signal « en » et « du » en sortie à l'aide du fichier « left_right_leds.vhd » que vous devez modifier et insérer dans

Le projet utilise les cinq fichiers suivants

- lcd16x2_ctrl.vhd
- lcd16x2_ctrl_demo_exo5.vhd (le votre)

- ReveilHHMMUp_Down.vhd
- left_right_leds.vhd (modifié)
- lcd16x2_ctrl_demo.ucf **mélangé avec left_right_leds.ucf**

Indications :

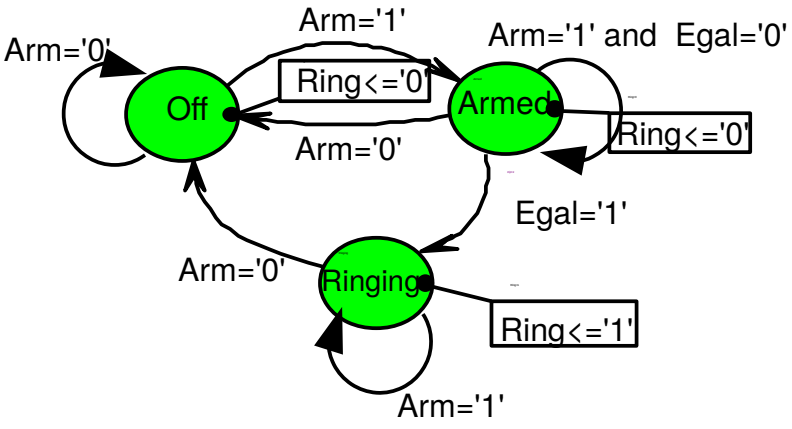
- Pour savoir comment produire les deux sorties « en » et « du », il est bon de regarder le process « led_display » en fin de « left_right_leds.vhd ». On voit tout de suite que ce process utilise « rotary_event » et « rotary_left » qui vont produire « en » et « du »
- Si vous voulez retirer la sortie des leds, vous pouvez retirer ce process « led_display ». On vous laisse le choix, et pour info la sortie sur led n'a pas besoin d'être utilisée.
- L'entité (et le component) auront la forme :

```
entity left_right_leds is Port (
    DU,EN  : out std_logic;
    led  : out std_logic_vector(7 downto 0);
    rotary_a : in std_logic;
    rotary_b : in std_logic;
    rotary_press : in std_logic;
    clk : in std_logic);
end left_right_leds;
```

Faire constater à l'enseignant.

V) Quatrième partie : réalisation de la sonnerie de réveil

Le mécanisme de sonnerie d'un réveil peut être décrit par le graphe d'évolution suivant :



Hormis l'horloge, ce mécanisme possède deux entrées :

- « Arm » qui est une touche d'armement de la sonnerie du réveil

- « Egal » qui détecte l'égalité entre l'heure courante et l'heure de réveil

La sortie est Ring. Quand elle est à un le réveil sonne.

Exercice 8

1°) Écrire une entité/architecture capable de réaliser le graphe d'évolution de l'automatisme de sonnerie. Il sera testé avec des boutons comme entrées (Arm et Trip). Et évidemment inséré dans « lcd16x2_ctrl_demo.vhd » avec un port map.

Faire constater à l'enseignant.

2°) Remplacer le Trip de la question précédente par un signal s_egalite que vous devez réaliser avec les signaux internes « s_data16 » (qui est l'heure courante) et « s_HHMMR » (qui est l'heure de réveil).

Faire constater à l'enseignant.

Exercice 9

On vous demande d'améliorer l'ensemble réalisé en exercice 8 :

- possibilité de régler l'heure courante (en plus de la modification de l'heure réveil)
- modifier l'automatisme de sonnerie pour que le réveil sonne pendant une heure s'il n'est pas coupé par « Arm »
- Amélioration de votre choix

Faire constater à l'enseignant.