Examen Spécifications des Circuits Intégrés Numériques - 2h - (tous documents autorisés)

Pour cet examen vous devrez rendre, sur `Arche`, une archive à votre nom contenant les différents codes source VHDL (éventuellement commentés pour répondre aux questions), ainsi que des copies d'écran de simulation si nécessaire. Attention l'échéance de remise des archives est fixée à 16h.

1 Accumulateur

Le fichier *ex1_exam.qar* contient une archive de projet quartus. Pour restaurer le projet correspondant copiez le fichier *micro1_exam.qar* dans votre dossier de travail, puis dans `quartus` choisissez *Project => Restore Archived Project* ce qui recréera un projet incluant le fichier VHDL décrivant un système. Cette description comporte des erreurs.

1. Corrigez les erreurs du système décrit.
2. Écrivez et simulez un testbench pour vérifier le bon fonctionnement de l'accumulateur. (rendre une copie d'écran de la fenêtre wave de modelsim qui permet de voir le bon fonctionnement)
3. Modifiez le code pour que la taille des données de l'accumulateur soit générique.

2 Timer

Le code VHDL ci-dessous décrit le fonctionnement d'un timer.

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4  entity timer is
5     port(
6        clk, reset: in std_logic;
7        sec,min: out std_logic_vector(5 downto 0)
8     );
9  end timer;
10
11 architecture single_clock_arch of timer is
12    signal r_reg: unsigned(19 downto 0);
13    signal r_next: unsigned(19 downto 0);
14    signal s_reg, m_reg: unsigned(5 downto 0);
15    signal s_next, m_next: unsigned(5 downto 0);
16    signal s_en, m_en: std_logic;
17 begin
18    -- register
19    process(clk,reset)
20    begin
21       if (reset='1') then
22          r_reg <= (others=>'0');
23          s_reg <= (others=>'0');
24          m_reg <= (others=>'0');
25       elsif (clk'event and clk='1') then
26          r_reg <= r_next;
27          s_reg <= s_next;
28          m_reg <= m_next;
29       end if;
30    end process;
31    -- logique de l'etat futur/ de sortie pour le compteur modulo 1000
32    r_next <= (others=>'0') when r_reg=999 else
33              r_reg + 1;
34    s_en <= '1' when r_reg = 500 else '0';
35    -- logique de l'etat futur pour les secondes
36    s_next <= (others=>'0') when (s_reg=59 and s_en='1') else
37              s_reg + 1     when s_en='1' else
38              s_reg;
39    m_en <= '1' when s_reg=59 and s_en='1' else '0';
40    -- logique de l'etat futur pour les minutes
41    m_next <= (others=>'0') when (m_reg=59 and m_en='1') else
42              m_reg + 1     when m_en='1' else
43              m_reg;
```

```
44    -- logique de sortie
45    sec <= std_logic_vector(s_reg);
46    min <= std_logic_vector(m_reg);
47  end single_clock_arch;
```

1. Remplacez tous les signaux `unsigned` par `std_logic_vector` et apportez les modifications nécessaires pour que le code soit fonctionnel.
2. Réalisez les parties concurrentes en utilisant les instructions séquentielles. Pour ce faire, utilisez un `process` pour chaque partie (au total trois).
3. Si l'on souhaite compter les heures avec ce timer, rajoutez la sortie pour les heures `hour` et complétez le code initial permettant de la gérer également. Utilisez le format 0-24h où les heures sont remises à zéro après la valeur 23.

3 Générateur à modulation de largeur d'impulsions (MLI) ou Pulse Width Modulation (PWM)

La figure 1 représente le schéma d'un générateur de signal à modulation de largeur d'impulsions. L'entrée Div permet de fixer la valeur de la période T, alors que l'entrée RC permet de choisir le rapport cyclique du signal MLI.

1. Donnez le code VHDL de ce système (celui de la figure 1).
2. Donnez le code d'un testbench permettant de vérifier le fonctionnement de ce système.
3. Modifiez le code VHDL de ce système afin que la taille des entrées (Div et RC) soit générique.

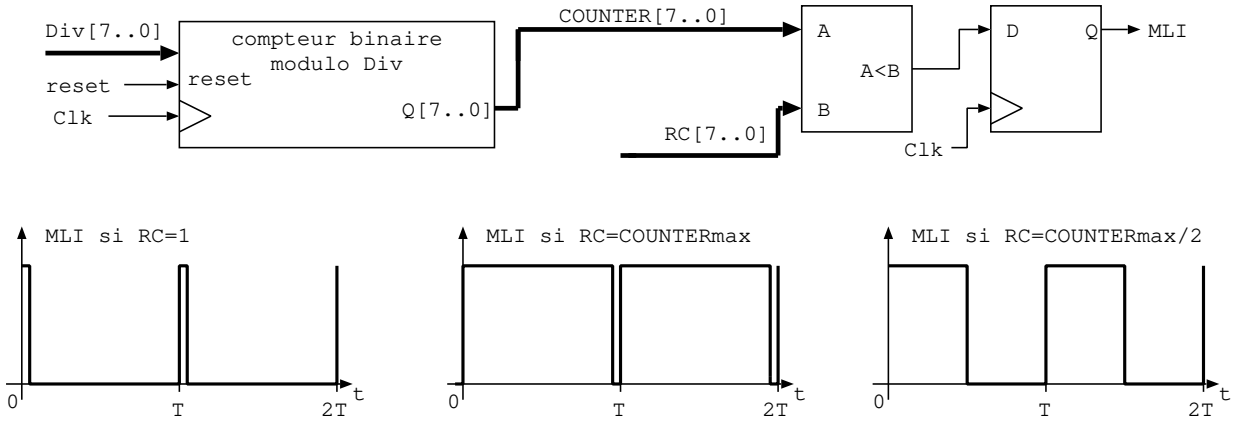

FIGURE 1 – Schéma de principe d'un générateur MLI

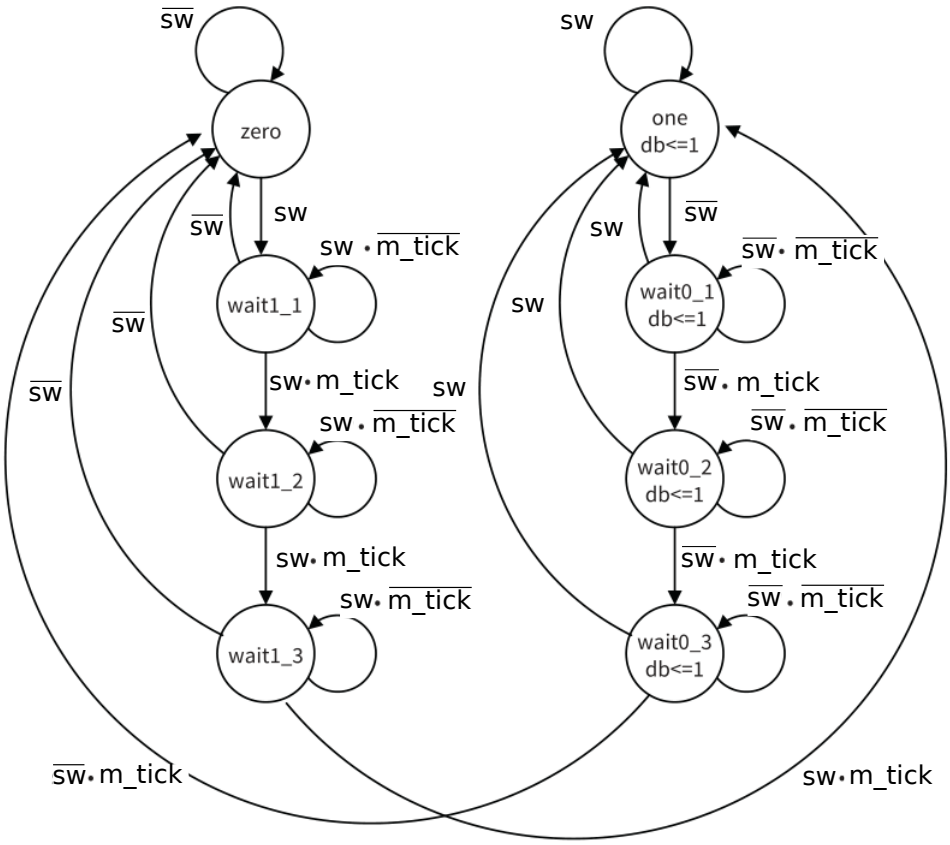

FIGURE 2 – graphe d'états du dispositif de comptage d'objets

4 Dispositif de filtrage des impulsions indésirables

On souhaite réaliser un dispositif de filtrage des impulsions indésirables dont la durée est inférieure à 30 ms. Pour ce faire, le système est composé d'un timer dont la période de comptage est de 10 ms, d'une machine à état finis dont le graphe d'états est donné figure 2 et d'un détecteur de front montant. Le signal `sw` représente le signal à filtrer tandis que le signal `m_tick` représente le signal indiquant les fronts montants du signal périodique du timer. Pour le timer, utilisez un compteur binaire cadencé par une horloge à 50MHz. La même fréquence d'horloge est utilisée pour la machine à états finis.

1. Codez en VHDL la machine à état fini permettant de filtrer les impulsions indésirables inférieurs à 30 ms.
2. Codez le testbench permettant de vérifier le bon fonctionnement de cette machine à état fini.
3. Donnez le code VHDL du système complet.
4. Donnez le code VHDL du testbench permettant de valider le fonctionnement du système complet.

5 Processeur sur FPGA

L'archive *micro_exam_SCIN.qar* décrit un processeur 4 bits fonctionnel et correspondant à celui que vous avez vu en TD. Pour restaurer le projet correspondant copiez le fichier *micro1_exam.qar* dans votre dossier de travail, puis dans `quartus` choisissez *Project => Restore Archived Project*. Vous pouvez ajouter l'état du décodeur d'instructions à la fenêtre de simulation avec la commande `modelsim` suivante :
`add wave sim:/micro1_vhd_tst/i1/upDECODEUR/state_reg`
Si vous souhaitez copier-coller cette commande faites le depuis le fichier `Microprocesseur1_tb.vhd` où il figure en commentaire, car le pdf copie parfois mal les "_".
Cela permet d'ajouter des signaux particuliers (`state_reg` dans l'exemple) et non pas des dizaines de signaux comme le fait le menu contextuel *Add to => Wave => All items in region and below*

1. Que devrait réaliser le programme contenu dans la *ROM* de ce processeur ?

2. Le testbench fourni devrait arrêter automatiquement la simulation après la dernière instruction. Vous pouvez constater que ce n'est pas le cas en raison d'erreurs dans le décodeur d'instructions. Corrigez ces erreurs.
3. Vous pourrez alors voir par simulation que ce processeur ne fonctionne pas correctement : les résultats dans la *RAM* ne sont pas corrects. Le problème vient d'un signal de contrôle mal affecté (localisation par simulation) et d'un problème de "câblage" des éléments du processeur (localisation dans la vue *RTL*). Corrigez ces problèmes.