M1 Info – ARC - LAB2 Extension

Olivier HUREAU - Groupe 3 18/03/2020

1 Testbench: check frequency.

Soit une horloge avec une fréquence de 10MHZ. Alors il faut que l'horloge reviennent à 1 toute les $\frac{1}{10MHz}s$ Soit $\frac{1}{10^7Hz}s=10^{-7}=100nanosecondes$

La bonne implémentation est alors :

```
begin
clk <= '0';
wait for 50 ns;
clk <= '1';
wait for 50 ns;
end process;</pre>
```

2 Counter: the form of the description is not correct.

En reprennant la slide 125 il faut donc trois process pour modéliser le counter.

2.1 Calcul de l'état suivant

D'après le slide, ce process doit calculer le prochain état.

2.1.1 Le code

Pour rappel le code est :

```
process (state, start, c)
begin
    case state is
    when IDLE =>
        if start = '1' then
            nextstate <= COUNTING;
    elsif start = '0' then
            nextstate <= IDLE;
    end if;
    when COUNTING =>
        if c < threshold then
            nextstate <= COUNTING;
    else
        nextstate <= IDLE;
    end if;
end case;
end process;</pre>
```

Figure 1: Process #1

2.1.2 Vérification du code

Sensitity list Tout les éléments pris en compte pour le calcul de l'état suivant sont : State, start et c. Alors la liste dois être (state, start,c).

Calcul des états Si l'on regarde le dessin de l'automate fournis :

- Dans l'état IDLE on reste dans cet état si start vaut '0', sinon si start vaux '1' on vas dans l'état COUNTING. Ce qui est exprimé dans ce code
- Dans l'état COUNTING on reste dans cet état si c est ; à treshold, sinon on vas dans l'état IDLE. Ce qui est aussi exprimé dans ce code.

2.2 Mise à jour de l'état

D'après les slide, ce process exprime la mise à jour de la valeur du registre courrant en synchronisation avec l'horloge. Ce process peux aussi prendre en considération le signal "reset".

2.2.1 Le code

Après avoir modifier le code tel que :

```
-- MISE A JOUR DU REGISTRE D'ETAT
                                                                                                             -- MISE A JOUR DU REGISTRE D'ETAT
process(reset, clk, start)
                                                                                                             process(reset, clk, start)
         - RESET : asynchrone haut
                                                                                                                      -- RESET : asynchrone haut
       if reset = '1' then
                                                                                                                     if reset = '1' then
              state <= IDLE:
                                                                                                                            state <= IDLE:
        -- HORLOGE : front montant
                                                                                                                      -- HORLOGE : front montant
       elsif (clk'event and clk = '1') then
                                                                                                                     elsif (clk'event and clk = '1') then
               state <= nextstate;
                                                                                                                             state <= nextstate;
               if ( (start = '1') and (state = IDLE) )then
                                                                                                                      -- Detecter un pic sur start
                                                                                                                     elsif ( (start = '1') and (state = IDLE) )then
    end if;
       end if;
                                                                                                                     end if;
end process;
                                                                                                              end process;
```

Figure 2: Process #2 Avant/après

2.2.2 Vérification du code

Sensitity list Tout les éléments pris en compte pour le calcul de l'état suivant sont : reset clock et start. Je prend en compte ici start car je veux que la moindre impulsion sur le signal start puisse déclancher le compte à rebours et que pas uniquement lorsque l'horloge est sur un front montant. Je suis donc obligé de l'écrire ici. Néanmoins, j'avais fais une erreur d'imbrication de if/elsif précédement.

Reset Il y a bien une mise à de l'état courrant si le reset est à sur un 1 (asynchronous reset HIGH)

Front montant L'état ce met bien à jour si l'on se trouve sur un front montant de l'horloge.

2.3 Mise à jour de du compteur

Il n'y as pas d'indication dans le cours pour implémenter le counter mais d'après moi il faut un process qui s'en occupe.

2.3.1 Le code

Après avoir modifier le code tel que :

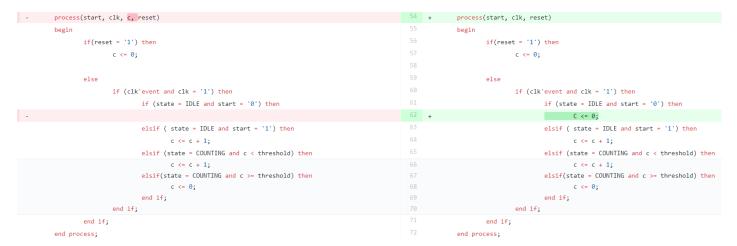


Figure 3: Process #3 Avant/après

J'avais "juste" oublié une ligne de code et un signal en trop dans la liste de senssibilité. C'est comme ça qu'une fusée peux exploser...

2.3.2 Vérification du code

Sensitity list La valeur de c peux changer si et seulement si soit start, clk et reset d'ou la list (start, clk, reset).

Transitions J'ai retranscris les transitions de l'automate tel que les calculs de c ne se fasse uniquement une fois par cycle.

2.4 Exprime les fonctions de sorties

D'après les slides il faut un process qui exprime les fonctions de sorties (ou un ensemble de concurent statement)

2.4.1 Le code

J'avais exprimé cela sous la forme :

```
aboveth <= '0' when c < threshold else '1';
```

car je n'ai pas réussi à trouver la comparaison bouléenne exprimant \geq Pour être sur de moi, j'ai transformé le tout en un process :

Figure 4: Process #4 Avant/après

2.4.2 Vérification du code

Je pense pas qu'il n'y ai vraiment grand chose à commenter.

3 Improve testbench for the system.

Il faut déjà changé le testbench avec la vrai valeur de l'horloge, je n'ai pas vraiment le temps de le faire...