Projet de VHDL

Réalisation d'un Fréquencemètre

Jérémie Fourmann (Promo 2013 - Eléctronique)
Maxime Morin (Promo 2013 - Eléctronique)

29 mai 2012

Plan

1 Objectifs

Le but de ce projet est de meusurer la fréquence d'un signal, le résultat de cette mesure sera affichée sur quatres afficheurs 7 segments.

1.1 Rappel du cahier des charges

Notre projet doit répondre aux contraintes suivantes :

Changement de gamme automatique : Pour garder le plus de précision possible, il faudra changer de méthode de mesure quand la fréquence dépassera une certaine valeur (voir partie nos choix).

Rafraichissement de la mesure de 1s : Permet de garder une bonne fluidité lorsque la fréquence est amenée à changer.

Affichage sur 4 digits : Permet d'avoir une précision correcte

Plage de fréquence de 1Hz à 10 MHz

Affichage des calibres

RQ : Par ailleurs notre système affichera un message d'alerte quand le signal mesuré dépasse la plage de fréquence admise.

1.2 Nos choix

1.2.1 Choix de la validité des méthodes de mesures

Nous avons deux méthodes de mesures à notre disposition : méthode étalon et méthode échantillonage.

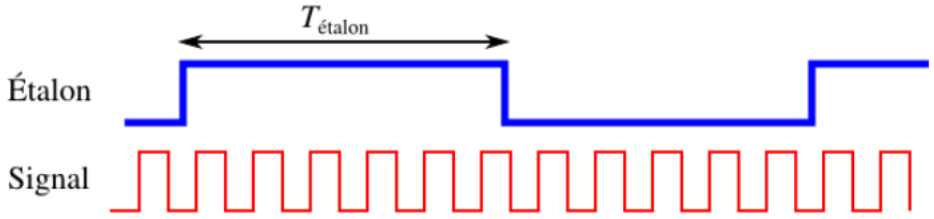

FIGURE 1 – Métode étalon

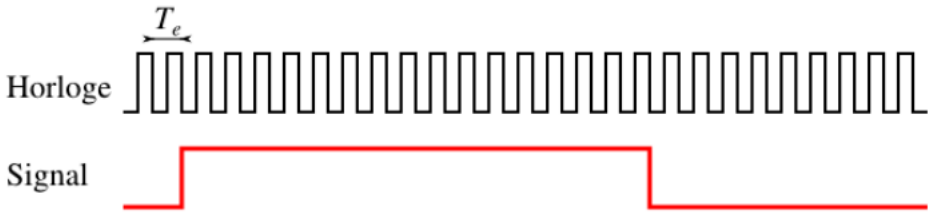

FIGURE 2 – Méthode échantillonage

Ces deux méthodes sont complémentaires.
En effet pour un signal de fréquence élevé nous utiliseron la méthode étalon, elle permet d'avoir le maximum de précision.
En revanche à basse fréquence cette méthode ne peut s'apliquer convenablement, nous utilisons alors la méthode échantillonage.

Il faut donc trouver la valeur pour laquelle on change de méthode de meusure. Nous allons tracer la précision en fonction de la fréquence pour les deux méthodes, à partir de cette courbe nous choisirons la valeur de basculement.

1.2.2 Choix de la conception en machine d'état

Notre conception étant entièrement synchrone, chaque sous module de notre sytème sera une machine d'état avec le bloc M et G synchronisé sur la clock du FPGA.
Cette conception en machine d'état nous assure une implémentation adapté aux fonctionement du FPGA.
Par ailleurs nous avons aussi une meilleur lisibilité de notre programme et une meilleur facon de débugger car nous avons accès à l'état de notre machine lors des simulation.

2 Conception du système

2.1 Présentation du système

2.2 Présentation des sous Modules

2.2.1 Schéma bloc

2.2.2 Machine d'état

2.2.3 Résulat de simulation

2.3 Performance du système

3 Bilan

A Manuel d'utilisateur

A Extrait de code VHDL

```
1  ----------------------------------------------------------------------------------
   -- Company:
3  -- Engineer:
   --
5  -- Create Date:    18:18:06 05/06/2012
   -- Design Name:
7  -- Module Name:    nombre_24bits_generateur - Behavioral
   -- Project Name:
9  -- Target Devices:
   -- Tool versions:
11 -- Description:
   --
13 -- Poue le test de notre module d'affichage
   --
15 -- affiche une puissance de deux par seconde
   --
17 ----------------------------------------------------------------------------------
   library IEEE;
19 use IEEE.STD_LOGIC_1164.ALL;

21 -- Uncomment the following library declaration if using
   -- arithmetic functions with Signed or Unsigned values
23 --use IEEE.NUMERIC_STD.ALL;

25 -- Uncomment the following library declaration if instantiating
   -- any Xilinx primitives in this code.
27 --library UNISIM;
   --use UNISIM.VComponents.all;
29
   entity nombre_24bits_generateur is
31     Port ( rst : in  STD_LOGIC;
              clk : in  STD_LOGIC;
33            nombre : out  STD_LOGIC_VECTOR (23 downto 0));
   end nombre_24bits_generateur;
35
   architecture Behavioral of nombre_24bits_generateur is
37 type etat is (init, calc_nombre, attente);

39 signal etatf : etat; --etat futur
   signal etatp : etat; --etat present
41
   signal timer : STD_LOGIC_VECTOR (23 downto 0);
43
   begin
45
   --Bloc F
47 process(etatp, rst)
   begin
49     if rst='0' then etatf <=init ;
       else
51         case etatp is
           when init =>
53
           when others => etatf <= init;
55
           end case;
57     end if;
   end process;
59
   --Bloc M
61 process(clk, rst)
   begin
63     if (clk'event and clk = '1') then etatp <= etatf ;
       end if;
```

```
65  end process;

67  --Bloc G
    process(clk)
69  begin
    if(clk'event and clk='1') then
71
        case etatp is
73      when init =>

75      when others =>

77      end case;

79  end if;
    end process;
81
    --autres process
83  process(step)
    begin
85
    end process;
87

89  end Behavioral;
```