

# Projet de VHDL

## Réalisation d'un Fréquencemètre

JÉRÉMIE FOURMANN (Promo 2013 - Électronique)

MAXIME MORIN (Promo 2013 - Électronique)

29 mai 2012



## Plan

<b>1</b>	<b>Objectifs</b>	<b>2</b>
1.1	Rappel du cahier des charges . . . . .	2
1.2	Nos choix . . . . .	2
1.2.1	Choix de la validité des méthodes de mesures . . . . .	2
1.2.2	Choix de la conception en machine d'état . . . . .	3
<b>2</b>	<b>Conception du système</b>	<b>4</b>
2.1	Présentation du système . . . . .	4
2.2	Présentation des sous Modules . . . . .	4
2.2.1	Schéma bloc . . . . .	4
2.2.2	Machine d'état . . . . .	4
2.2.3	Résultat de simulation . . . . .	4
2.3	Performance du système . . . . .	4
<b>3</b>	<b>Bilan</b>	<b>5</b>
<b>A</b>	<b>Manuel d'utilisateur</b>	<b>6</b>
A.1	Introduction . . . . .	6
A.2	Component VHDL . . . . .	6
A.3	Implémentation sur Nexys 2 . . . . .	6
A.4	Fichier contrainte . . . . .	6
<b>B</b>	<b>Extrait de code VHDL</b>	<b>8</b>

# 1 Objectifs

Le but de ce projet est de mesurer la fréquence d'un signal, le résultat de cette mesure sera affichée sur quatre 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).

**Rafraîchissement 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 échantillonnage.

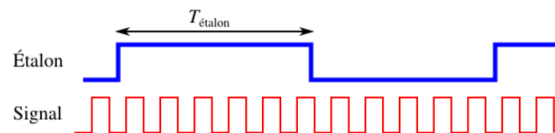


FIGURE 1 – Méthode étalon



FIGURE 2 – Méthode échantillonnage

Ces deux méthodes ne sont pas meilleures l'une que l'autre, elles sont complémentaires. En effet pour un signal de fréquence élevé nous utiliserons la méthode étalon, elle permet d'avoir le maximum de précision.

En revanche à basse fréquence cette méthode ne peut s'appliquer convenablement, nous utilisons alors la méthode échantillonnage.

Il faut donc trouver la valeur pour laquelle on change de méthode de mesure. Nous allons tracer la précision en fonction de la fréquence pour les deux méthodes

On a :

$$Precision_{BF} = \frac{F_{clk}}{F_{signal}}$$

$$Precision_{HF} = \frac{F_{signal}}{F_{talon}}$$

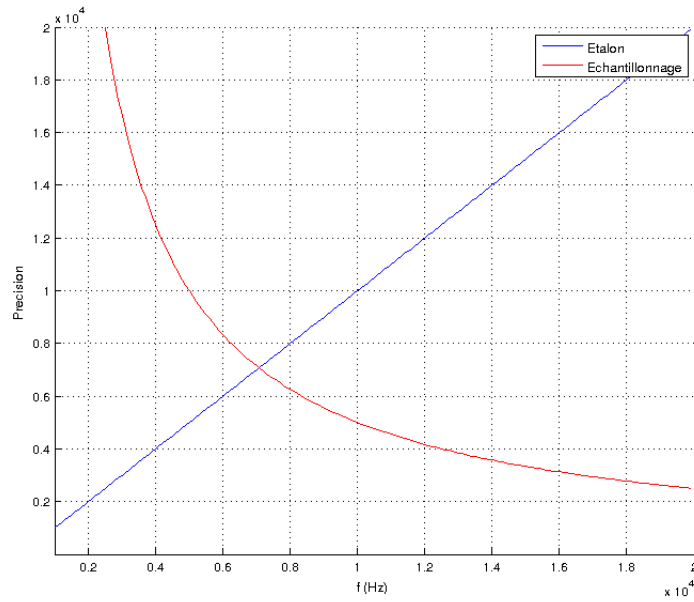


FIGURE 3 – Précision des deux méthodes

On s'aperçoit que pour avoir le maximum de précision il faut changer de méthode aux alentours de 7 KHz.

### 1.2.2 Choix de la conception en machine d'état

Notre conception étant entièrement synchrone, chaque sous module de notre système 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 fonctionnement du FPGA. Par ailleurs nous avons aussi une meilleur lisibilité de notre programme et une meilleur facon de débbugger car nous avons accès à l'état de notre machine lors des simulations.

## **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 Résultat de simulation**

### **2.3 Performance du système**

### 3 Bilan

## A Manuel d'utilisateur

### A.1 Introduction

L'utilisateur pourra utiliser le projet soit en utilisant le composant Fréquencemètre pour l'inclure dans un autre projet soit l'utiliser directement sur la Nexys 2.

### A.2 Composant VHDL

Notre projet peut se résumer en un seul composant suivant :

### A.3 Implémentation sur Nexys 2

En chargeant le .bit, l'utilisateur pourra utiliser le fréquencemètre.

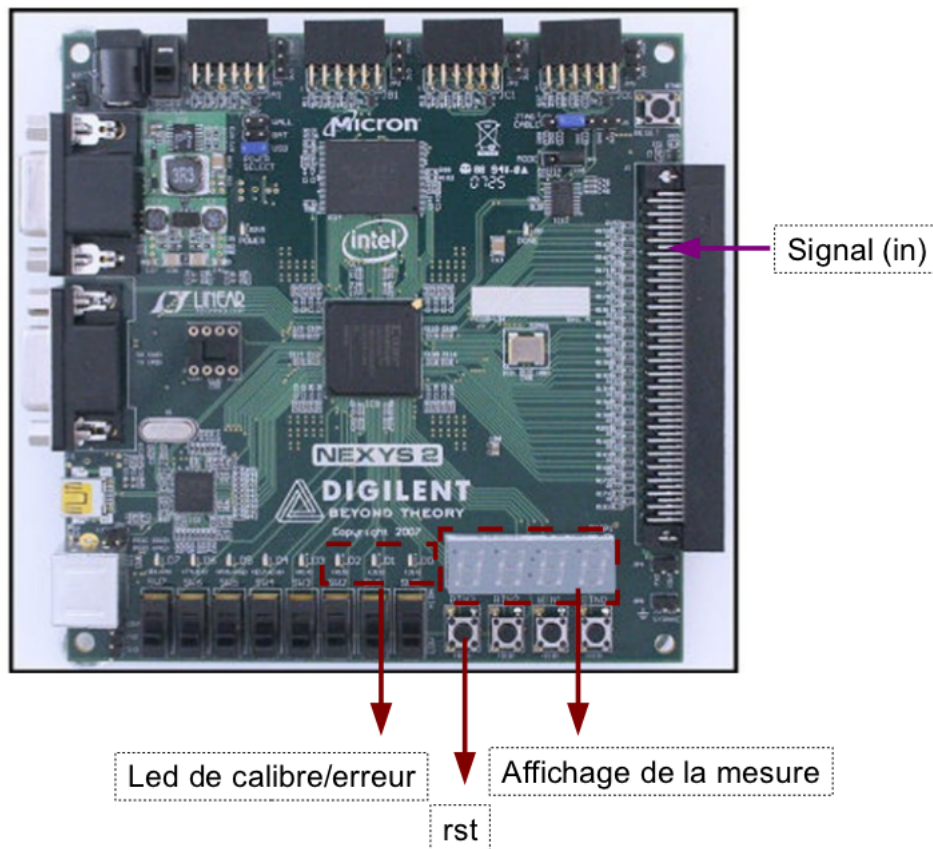


FIGURE 4 – Implémentation sur carte Nexys 2

### A.4 Fichier contrainte

L'utilisateur pourra éditer le fichier de contraintes selon ses besoins et ressources disponible.

```
1 net "rst" loc = "R17";  
  net "clk" loc = "B8";  
3 net "mhz" loc = "R4";  
  
5  
  
7 net "segments[0]" loc = "L18";  
  net "segments[1]" loc = "F18";  
9 net "segments[2]" loc = "D17";  
  net "segments[3]" loc = "D16";  
11 net "segments[4]" loc = "G14";  
   net "segments[5]" loc = "J17";
```

```
13 net "segments[6]" loc = "H14";  
   net "point" loc = "C17";  
15  
   net "selecteur[0]" loc = "F15";  
17 net "selecteur[1]" loc = "C18";  
   net "selecteur[2]" loc = "H17";  
19 net "selecteur[3]" loc = "F17";  
  
21 net "calibre[0]" loc = "J14";  
   net "calibre[1]" loc = "J15";  
23 net "calibre[2]" loc = "K15";  
  
25 net "entree" loc = "B4";
```

## B Extrait de code VHDL

```
-----
2  -- Company:
3  -- Engineer:
4  --
5  -- Create Date:      18:18:06 05/06/2012
6  -- Design Name:
7  -- Module Name:      nombre_24bits_generateur - Behavioral
8  -- Project Name:
9  -- Target Devices:
10 -- Tool versions:
11 -- Description:
12 --
13 -- Poue le test de notre module d'affichage
14 --
15 -- affiche une puissance de deux par seconde
16 --
-----
18 library IEEE;
19 use IEEE.STD_LOGIC_1164.ALL;
20
21 -- Uncomment the following library declaration if using
22 -- arithmetic functions with Signed or Unsigned values
23 --use IEEE.NUMERIC_STD.ALL;
24
25 -- Uncomment the following library declaration if instantiating
26 -- any Xilinx primitives in this code.
27 --library UNISIM;
28 --use UNISIM.VComponents.all;
29
30 entity nombre_24bits_generateur is
31     Port ( rst : in  STD_LOGIC;
32           clk : in  STD_LOGIC;
33           nombre : out STD_LOGIC_VECTOR (23 downto 0));
34 end nombre_24bits_generateur;
35
36 architecture Behavioral of nombre_24bits_generateur is
37     type etat is (init, calc_nombre, attente);
38
39     signal etatf : etat; --etat futur
40     signal etatp : etat; --etat present
41
42     signal timer : STD_LOGIC_VECTOR (23 downto 0);
43
44 begin
45
46     --Bloc F
47     process(etatp, rst)
48     begin
49         if rst='0' then etatf <=init ;
50         else
51             case etatp is
52             when init =>
53
54                 when others => etatf <= init;
55
56             end case;
57         end if;
58     end process;
59
60     --Bloc M
61     process(clk, rst)
62     begin
63         if (clk'event and clk = '1') then etatp <= etatf ;
64         end if;
```



```

        end process;
66
        --Bloc G
68        process(clk)
        begin
70            if(clk'event and clk='1') then

72                case etatp is
                    when init =>
74
                    when others =>
76
                    end case;
78
            end if;
80        end process;

82        --autres process
        process(step)
84        begin

86        end process;

88        end Behavioral;

```