

MI41 : Compte-Rendu de Travaux Pratiques N°3

Gestion des éléments de la platine DLP

Pierre Mauduit
Cedric Le-Breton

14 décembre 2005

1 Objectifs

Les objectifs de cette troisième séance de travaux pratiques sont dans un premier temps de découvrir le langage de description VHDL vu en cours en réalisant des composants simples, et dans un deuxième temps, de tester ces composants en les associant graphiquement puis en les envoyant dans le composant microprogrammable FLEX10K à disposition.

2 1. Gestion de l'afficheur : transcodeur binaire 7 segments

Nous allons donc réaliser dans cette première partie l'afficheur ; il transcodera une valeur binaire sur 4 entrées, vers les 7 segments de l'afficheur. Le code VHDL d'un tel composant est donné ci dessous :

```
library ieee;
use ieee.std_logic_1164.all;

entity transcode is
  port(
    val_bin: in std_logic_vector(3 downto 0);
    segt: out std_logic_vector(6 downto 0)
  );
end transcode;

architecture a of transcode is
  signal nsegt : std_logic_vector(6 downto 0);
  begin
    with val_bin select
      nsegt <= "1111110" when "0000",
              "0110000" when "0001",
              "1101101" when "0010",
              "1111001" when "0011",
              "0110011" when "0100",
              "1011011" when "0101",
              "1011111" when "0110",
              "1110000" when "0111",
              "1111111" when "1000",
              "1111011" when "1001",
              "1110111" when "1010",
              "0011111" when "1011",
              "1001110" when "1100",
```

```

        "0111101" when "1101",
        "1001111" when "1110",
        "1000111" when "1111",
        "1001000" when others;

    segt <= not nsegt;
end a;

```

3 2. Gestion de l'horloge

La platine DLP dispose d'une horloge de 25,175 MHz. A partir de celle-ci, nous souhaitons en obtenir une de 100 Hz. Pour cela, nous devons diviser la fréquence par 251 750. Le travail de notre nouveau composant sera donc ici de fournir un signal en sortie tous les 251 750 signaux reçus en entrée, à l'aide d'un compteur.

Le code VHDL est donné ci-dessous :

```

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity clk_100 is
port(
    clk_in: in std_logic;
    clk_out: out std_logic
);
end clk_100;

architecture a of clk_100 is
begin
process(clk_in)
variable cpt: integer range 1 to 251750;
begin
if (clk_in'event and clk_in = '1') then
    if (cpt = 251750) then
        cpt := 1;
        clk_out <= '1';
    else
        -- sinon incremente le compteur

```

```

        cpt := cpt + 1;
        clk_out <= '0';
    end if;
end if;
end process;
end a;

```

A partir de cette horloge de 100 Hz et du transcodeur de la première partie, le but est créer un chronomètre binaire. Pour cela nous aurons besoin d'un compteur par 16, dont le code VHDL est donné ci-dessous :

```

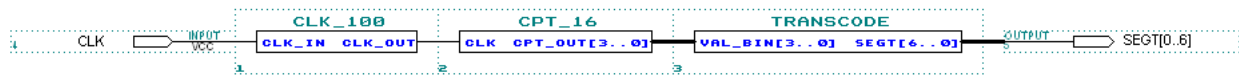
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity cpt_16 is
port(
    clk: in std_logic;
    cpt_out: out std_logic_vector(3 downto 0)
);
end cpt_16;

architecture a of cpt_16 is
-- compteur par 16
begin
process(clk)
variable m: std_logic_vector(3 downto 0);
begin
    if (clk'event and clk = '1') then
        m := m + "0001";
        cpt_out <= m;
    end if;
end process;
end a;

```

Une fois les trois composants réalisés, nous pouvons les relier les uns aux autres en mode graphique :



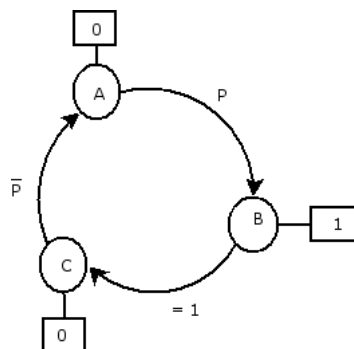
4 3. Gestion des boutons poussoir

Nous avons besoin de générer un composant permettant de transformer l'appui d'une durée quelconque sur un bouton en une commande active sur un seul cycle d'horloge. Avant de se lancer dans la rédaction du code VHDL d'un tel composant, il est nécessaire de réaliser un graphe à état.

Nous distinguons ici 3 états distincts :

- L'attente d'appui sur le bouton (A)
- L'appui sur le bouton, le composant envoie alors un signal en sortie (B)
- L'attente de la relache le bouton (C)

Voici un graphe à état, l'appui sur le bouton étant représenté par (P)



L'appui sur le bouton (P) provoque le passage dans l'état (B) avec l'émission d'un signal de sortie à 1, et une fois cela fait, on passe dans l'état (C) avec émission du signal 0.

Afin de limiter les effets de rebonds mécaniques, nous avons utilisé deux méthodes : l'échantillonnage à l'aide de l'horloge et l'utilisation d'un moyenneur, permettant de renvoyer la valeur présente en majorité sur 3 échantillons. Ainsi lorsque l'état du bouton sur 2 ou 3 échantillons est à 1, on considèrera le bouton comme actif.

Nous sommes maintenant en mesure de passer au codage VHDL du composant demandé :

```
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
```

```

entity btn_manager is
  port (
    CLK, I: in std_logic;
    O: out std_logic
  );
end btn_manager;

architecture a of btn_manager is
begin
  process(CLK)
    type t_st is (A, B, C);
    variable state: t_st;
    variable Ep: std_logic_vector(1 downto 0);
    variable moy: integer range 0 to 3;
  begin
    if (CLK'event and CLK = '1') then
      moy := CONV_INTEGER(Ep(1)) +
        CONV_INTEGER(Ep(0)) + CONV_INTEGER(I);
      Ep := Ep(0) & I;

      if (state = A and moy > 1) then
        state := B;
        O <= '1';
      elsif (state = B) then
        state := C;
        O <= '0';
      elsif (state = C and moy < 2) then
        state := A;
      end if;
    end if;
  end process;
end a;

```

Nous allons dorénavant associer tous les composants précédemment développés au cours de cette séance de TP, afin de gérer un chronomètre avec la fonctionnalité d'arrêt et de redémarrage par appui sur un bouton poussoir. Nous utiliserons pour cela le mode graphique de Max+PlusII. Il est nécessaire de recoder en VHDL le composant du compteur par 16, afin de lui rajouter cette commande d'activation.

```

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity cpt_16_act is
    port(
        CLK, I: in std_logic;
        O: out std_logic_vector(3 downto 0)
    );
end cpt_16_act;

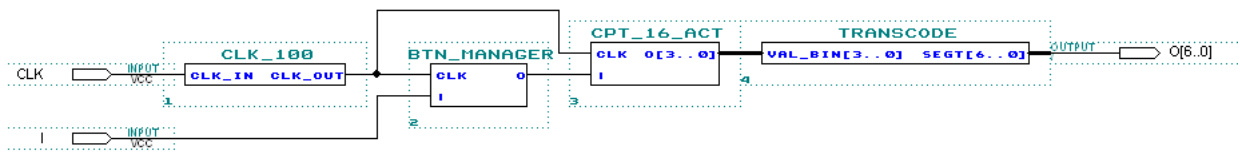
architecture a of cpt_16_act is
    begin
        process(CLK)
            variable m: std_logic_vector(3 downto 0);
            variable mode: std_logic;
            begin

                if (CLK'event and CLK = '1') then
                    if(I = '1') then
                        mode := not mode;
                    end if;

                    if(mode = '1') then
                        m := m + "0001";
                        O <= m;
                    end if;
                end if;
            end process;
        end a;

```

Associons maintenant en mode graphique les différents composants. Le schéma obtenu est donné ci-dessous :



Une rapide simulation nous permet de nous assurer que le résultat obtenu est bien le résultat attendu.

5 Conclusion

Cette séance de travaux pratiques a permis de se familiariser avec les bases de la programmation de composants en VHDL, en utilisant les éléments de platine DLP.