Physique et Ingénierie

Implémentations de systèmes numériques sur FPGA à l'aide du langage VHDL

Mathieu ALLARD

15 mai 2014

Objectifs

On veut réaliser un compteur de fibonacci avec affichages sur blocs 7 segments, avec détection d'overflow et communication sur port série de l'état du système. (valeur bonne ou erronnée) La description des composants, comme l'assemblage sera réalisé entièrement en VHDL, à implanter sur une carte FPGA Altera DE2. On utilisera différents composants de cette dernière pour le système. (switchs, leds, afficheurs, Clock, transmission série)

1 Tutoriel et explications

Suivi du tutoriel pour la prise en main des outils et de la carte altera DE2 Tout du long, nous allons suivre la méthode suivante pour chaque partie :

— réalisation du code vhdl
— réalisation d'un testbench
— simulation
— implantation sur la carte et test

Dans les faits, les simulations n'ont pas toujours été réalisés, certains éléments étant relativement simples à faire fonctionner directement. Je n'ai pas pris la peine de joindre les testbenchs. Ceux-ci ont été réalisés graphiquement avec un outil de Quartus.

2 Décodeur 7 segments

En vue d'afficher la suite, il faut convertir les 4*4 bits du compteur en symboles pour les afficheurs 7 segments.(donc 4*7 bits) Ces derniers fonctionnant en logique négative, un '0' allume un segment et un '1' l'éteint. Par exemple "0011" (3) deviendra "0000110" Exemple de code :

```
PROCESS (in4)
        variable seg : std_logic_vector(0 to 6);
        BEGIN

IF (in4=b"0000") THEN
seg:=b"0000001" ;
ELSIF(in4=b"0001") THEN
seg:=b"1001111";
ELSIF(in4=b"0010") THEN
seg:=b"0010010";
```

3 diviseur d'horloge

On veut afficher les termes de la suite à une cadence d'environ 1 terme/sec. Il faut donc utiliser un des timer présents sur la carte et en diviser la fréquence. La particularité ici est que l'on ne veut pas d'une horloge à 1Hz, mais d'un timer qui ne donne qu'une brève impulsion chaque seconde. On utilise le timer 50MHz On va donc, comme pour un diviseur d'horloge "classique", incrémenter un compteur à chaque impulsion, de 0 à 49 999 999 pour avoir en sortie une impulsion à 1Hz. Seulement cette fois, on va remettre à zéro le signal en sortie à la clock suivante.

```
elsif rising_edge(CLK) then
                        if (counter = 49999999) then
                                        tmp <= '1';
                                        counter <= counter + 1;
                        elsif (counter = 50000000) then     --on veut une
                        --impulsion de courte durée, on repasse à 0 au front
                        --d'horloge suivant.
                        tmp <= '0';
                        counter <= 0;
                        else
                counter <= counter + 1;
            end if;
```

4 Générateur de la suite de Fibonacci

Le code est disponible en annexe. J'ai pris comme base le code réalisé au premier semestre en TP VHDL. On veut cette fois avoir une sortie sur 16 bits.

Principe de fonctionnement Soient T_n les termes de la suite, on a : $T_n = T_{n-1} + T_{n-2}$ Afin de pouvoir afficher les premiers termes et prévoir l'overflow au bon moment, il nous faut 3 variables : $a = T_n$, $b = T_{n-1}$, $c = T_{n-2}$

```
1 c<=b;
2 b<=a;
3 a<=a+b;
```

a et $b$sont initialisées à X"0001", c à X"0000". c permet de prévoir l'overflow un coup d'horloge à l'avance afin de pouvoir lever le flag au bon moment. On affiche b afin d'avoir les bons premiers termes de la suite (les deux premiers termes étant "1")

Ensuite, N, réglable sur des switchs, permet de choisir à partir de combien de termes on s'arrête de ceompter et on revient à zéro. Le reset est synchrone : le process n'est sensible qu'à CLK.

Enfin, on ajoute une sortie DataIN à transmettre à l'USART : les caractères E ou B à envoyer indiquant si la valeur est bonne ou non. Il faut encoder ces caractères en ascii sur 8 bits :
B=dec"66"=bin"01000010" E=dec"69"=bin"01000101"

5 Réalisation de l'USART

Les codes de l'usart et du baudgen sont disponibles en annexe

5.1 Baudgen

L'USART en soi ne définissant pas la vitesse de transmission, il faut un autre bloc pour cadencer celà à une vitesse respectant le protocole RS232. On veut pouvoir choisir entre 9600 et 19200 bauds, il faudra donc 2 diviseurs d'horloge délivrant un signal à ces fréquences que l'on enverra à l'usart. Attention, le système ne fonctionne qu'avec une seule véritable horloge : celle du système à 50Mhz, l'USART ne dérogera pas à la règle et le signal délivré par le baudgen ne servira pas d'horloge.

5.2 USART

L'USART reçoit un signal sur 8bits sur DataIN et l'envoie sur le port série bit à bit avec un start bit et un stop bit au début et à la fin de la transmission. La transmission se fait du LSB vers le MSB.
On crée un compteur qui va permettre de cadencer l'envoi du signal avec le baudgen en emettant un start bit à 0, le message bit à bit sur 8bits puis le stop bit.

La variable start est mise à zéro à la fin de l'envoi, et reçoit starttr (implémenté sur un switch) au début du programme, celà permet d'éviter l'envoi de trop de caractères d'un coup. Il aurait fallu utiliser une détection sur front de l'interrupteur pour que ce système fonctionne, ici trop de caractères sont envoyés à chaque transmission.

6 Assemblage et réalisation du séquenceur contrôlant les différents blocs

code disponible en annexe

On crée un fichier top.vhd dans lequel on va instancier tous les composants précédents. (voir annexe) Dans l'entité on déclare les ports extérieurs à assigner à des pins, afficheurs, interrupteurs... Puis dans l'architecture, on déclare des signaux correspondant aux signaux internes entre les composants.

Chaque composant est décrit par ses ports d'entrées/sorties. Enfin, on termine par la description du composant "général" en assignant les signaux entre les composants avec PORT MAP.

7 Problèmes rencontrés

Le tutoriel était plutôt facile à suivre et la prise en main du logiciel Quartus n'a pas posé de problème particulier.
Par la suite, certaines précautions ont été necessaires : la remise à zéro doit être synchrone, à vérifier dans les programmes.
Si on veut utiliser un bouton poussoir il est préférable de faire une détection sur front que sur état. J'ai choisi d'utiliser uniquement des switchs.

La partie la plus complexe à réaliser est l'USART, je n'ai pas réussi à le rendre tout à fait fonctionnel. Comme énoncé précédemment, le système d'envoi d'un seul caractère ne fonctionne pas, j'en ai compris la raison bien trop tard pour avoir le temps de le corriger. Concernant l'affichage des caractères il faut faire attention au sens des vecteurs de bits à envoyer sur les afficheurs 7 segments, dans mon cas les caractères envoyés ne sont pas les bons, je n'ai pas eu le temps de trouver l'erreur à corriger.

annexe : code, Fibonacci

```
--    __ _ _                _        _
--   / _(_) |__   _____   _| |__   __| |
--  | |_| | '_ \ / _ \ \ / / '_ \ / _` |
--  |  _| | |_) | (_) \ V /| | | | (_| |
--  |_| |_|_.__/ \___(_)_/ |_| |_|\__,_|

LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
USE IEEE.std_logic_unsigned.all;

ENTITY fibo IS
  PORT(
     N : IN std_logic_vector (7 downto 0) :="00011110";  --nbre termes =30 ici
     RAZ,CLK : IN std_logic:='0';
     EN : IN std_logic :='1';
      suitefibo : OUT std_logic_vector (15 downto 0):=X"0000"; --sortie
      Overflow : OUT std_logic:='0'; --carry
      caract : OUT std_logic_vector (0 to 7) :=X"00");
              --caractère à envoyer à l'USART : B ou E
END fibo;

ARCHITECTURE behav OF fibo IS
       SIGNAL c :std_logic_vector (15 downto 0):=X"0000";
       SIGNAL b : std_logic_vector (15 downto 0):=X"0001" ;
       SIGNAL a : std_logic_vector (15 downto 0) :=X"0001" ;
  BEGIN
    PROCESS(CLK)
       VARIABLE cnt : std_logic_vector (7 downto 0):=X"00"; --nbre de termes
    BEGIN


    IF(CLK'EVENT and CLK ='1') THEN

        IF (RAZ='1') THEN
      --raz synchrone
          c<=(others=>'0'); --n-2
          a<=X"0001"; --n
          b<=X"0001";  --n-1
          cnt:=(others=>'0');
          Overflow <='0';
        END IF;

        IF (c > b) THEN --c ne peut normalement être > à b
                      Overflow <='1';
                 --on encode le caractère à envoyer à l'USART
                     caract <= "01000101"; --E

        ELSIF (b > c) THEN caract <= "01000010"; --pas d'overflow, caractère B
        END IF;

                IF (EN='1') THEN
              --comportement normal
                c<=b;
                b<=a;
                a<=a+b;

                cnt:=cnt+1; --nbre de termes calculés

```

```
                            IF (cnt=N) THEN --reinitialisation
                            a<=X"0001";
                            b<=X"0001";
                            c<=(others=>'0');
                            cnt:=(others=>'0');
                            Overflow <='0';
                            END IF;
                        END IF;
        END IF;
      END PROCESS;

      suitefibo<=b;

END behav;
```

annexe : code, Baudgen et USART

```
--                                   _          _         _
--          _   _ ___  __ _ _ __/ /___    _/ /__    __/ /
--         / / / / __/ _` / '__/ __\ \ / / '_ \ / _` /
--         / /_/ \__ \ (_/ / /   / /_ \ V // / / / (_/ /
--         \__,_/___/\__,_/_/    \__(_)_/ /_/ /_/\__,_/

LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
USE IEEE.std_logic_unsigned.all;

ENTITY USART IS
port(dataIN : IN std_logic_vector(0 to 7);
clk : IN std_logic;
raz : IN std_logic;
starttr : IN std_logic :='1';
baudclk : IN std_logic;
TxD : OUT std_logic);
END ENTITY;
--on ne s'occupe que de la partie transmission
ARCHITECTURE USARTarch of USART is

signal tx_reg    :std_logic_vector (7 downto 0);
signal tx_full   :std_logic;
signal tx_out    :std_logic;
signal tx_count  :integer range 0 to 16;
signal start     :std_logic;
BEGIN
PROCESS(clk) BEGIN


if ( clk'EVENT AND clk='1') then      --attention, horloge 50Mhz
        if (raz = '0') then


        if (baudclk = '1') then      --cadence choisie
        start <= starttr;
            if (tx_full = '0' AND start = '1')  then
                tx_count <= tx_count +1;

                if (tx_count = 0) then            --start bit
                tx_out <= '0';
                end if;
                if (tx_count >=1 and tx_count <=8) then --données sur 8 bits
                tx_out <= dataIN(tx_count - 1);
```

```
                end if;
                if (tx_count =9) then           -- stop bit
                tx_out <= '1';
                tx_count <= 0;
                tx_full <= '0';
                end if;
            end if;
                end if;

        else
                start <= '0';
        tx_full <= '0';
        tx_count <= 0;
        tx_out <= '0';
    end if;
end if;
END PROCESS;
TxD <= tx_out;
END ARCHITECTURE;
```

```
--        _                  _                  _      _
--       / /__   __ _ _   _  __/ / __ _  ___ _ __ __   _/ /__   __/ /
--      /  '_ \ / _` / / / // _` // _` // _ \ '_ \\ \ / / '_ \ / _` /
--     / /_) / (_/ / /_/ / (_/ / (_/ /  __/ / / /\ V // / / / (_/ /
--    /_.__/ \__,_/\__,_/\__,_/\__, /\___/_/ /_(_)_/ /_/ /_/\__,_/
--                            /___/

LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
USE IEEE.std_logic_unsigned.all;

ENTITY baudgen IS
PORT( vit : IN STD_LOGIC :='1';
CLK : IN STD_LOGIC;
  baudclk : OUT STD_LOGIC);
END baudgen;

ARCHITECTURE archbaud OF baudgen IS
SIGNAL tmp : STD_LOGIC;
SIGNAL counter : INTEGER range 0 to 50000000 :=0;

BEGIN
PROCESS (CLK)
BEGIN
IF (CLK'EVENT AND CLK='1') THEN

        IF vit='0' THEN   --9600 bauds

        if rising_edge(CLK) then
                    if (counter = 5208) then
                                tmp <= NOT(tmp);
                                counter <= 0;
                    else
                counter <= counter + 1;
                    end if;
        end if;
    --     baudclk <= tmp;

        ELSIF vit='1' THEN  --19200 bauds

                if rising_edge(CLK) then
```

```
                    if (counter = 2604) then
                                    tmp <= NOT(tmp);
                                    counter <= 0;
                    else
              counter <= counter + 1;
                    end if;
        end if;
 --         baudclk <= tmp;

        END IF;
END IF;
END PROCESS;
baudclk <= tmp;
END archbaud;
```

annexe : code, assemblage

```
--     _                   _       _
--    /  /_ ___ _ __     __ _/  /_   __/  /
--   /   _/ _  \  '_  \  _   \ V /  '  \/ _`  /
--   \__ \___/ .__/ (_)    \_/ /_ //_ \__,_ /
--            /_ /

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use WORK.ALL;

ENTITY top IS
PORT(RAZ, CLK, vitesse, starttr :        IN STD_LOGIC;
        N:                               IN STD_LOGIC_VECTOR(7 downto 0);
        unite, dizaine, centaine, millier :     OUT    STD_LOGIC_VECTOR(0 to 6);
        OEF, Trans:                 OUT STD_LOGIC);
END top;

ARCHITECTURE struct OF top IS
    SIGNAL data : STD_LOGIC_VECTOR(15 downto 0);
    SIGNAL clk_en, baudclk : STD_LOGIC;
    SIGNAL message : STD_LOGIC_VECTOR(7 downto 0);

COMPONENT fibo
    port(CLK, RAZ, EN : IN STD_LOGIC;
        N : IN STD_LOGIC_VECTOR (7 downto 0);
    suitefibo : OUT STD_LOGIC_VECTOR(15 downto 0);
    caract : OUT STD_LOGIC_VECTOR(7 downto 0);
        Overflow : OUT STD_LOGIC);
END COMPONENT;

COMPONENT seven_seg
        PORT(
        in4 : IN STD_LOGIC_VECTOR(3 downto 0);
        seg7 : OUT STD_LOGIC_VECTOR(0 to 6));
END COMPONENT;

COMPONENT clkdiv
    PORT(CLK, RAZ :      IN STD_LOGIC;
    clk_out :    OUT STD_LOGIC);
END COMPONENT;

COMPONENT baudgen
    PORT( vit, CLK : IN STD_LOGIC;
    baudclk : OUT STD_LOGIC);
```

```
45 END COMPONENT;
46
47 COMPONENT usart
48         PORT( DataIN : IN STD_LOGIC_VECTOR(7 downto 0);
49     CLK, RAZ, starttr, baudclk : IN STD_LOGIC;
50     TxD : OUT std_logic);
51 END COMPONENT;
52
53 BEGIN --description
54 clk1: clkdiv PORT MAP(CLK => CLK,
55                       RAZ => RAZ,
56                       clk_out => clk_en);
57 baud1: baudgen PORT MAP (vit => vitesse, CLK => CLK, baudclk => baudclk);
58 fib1: fibo PORT MAP(CLK => CLK,RAZ => RAZ, EN => clk_en,
59                       N => N, suitefibo => data, caract => message,
60                       Overflow => OEF);
61 dec_0: seven_seg PORT MAP(data(3 downto 0),unite);
62 dec_1: seven_seg PORT MAP(data(7 downto 4),dizaine);
63 dec_2: seven_seg PORT MAP(data(11 downto 8),centaine);
64 dec_3: seven_seg PORT MAP(data(15 downto 12),millier);
65 usart1: usart PORT MAP(TxD => Trans, CLK => CLK,
66                         RAZ => RAZ, starttr => starttr,
67                         baudclk => baudclk,
68                         dataIN => message);
69
70 END struct;
```