Objectifs du projet

1. Comprendre le principe de diviseur de fréquence
2. Savoir calculer le nombre des bits du compteur et la valeur de chargement
3. Se familiariser avec un système multi-horloges
4. Utilisation d'un décodeur BCD to BCD 7 Segments
5. Autres astuces de programmation

Analyse de fonctionnement

Le circuit permet de générer une multitude des horloges allant de 6 MHz à 1 Hz, 8 horloges en total.

L'horloge de 1Hz (période d'une seconde) Contrôle l'afficheur 7 segments allant de 0 à 10 équivalents à une horloge de 10 secondes, un autre projet sera étudier prochainement pour créer une montre à 3 digits (9 :99).

Le circuit possède une entrée de réinitialisation RST et une entrée de validation CE (CE='1'), lorsque CE='0', le système mémorise l'état actuel, CE='1' le système continue le fonctionnement à partir de l'état précédente.

8 LED d'état sont liés avec les horloges. Les trois afficheurs sont actifs en permanent par les signaux, donc la même valeur s'affiche partout.

Les sorties du compteur et la fréquence équivalente

On considère un compteur sur 24 bits[0..23], le tableau ci-dessous montre la fréquence de chaque sortie avec F0=12MHz :

Sortie	Rapport de division	Fréquence (MHz)
0	2	6
1	1/4	3
2	1/8	1.5
3	1/16	0.75
4	1/32	0.375
5	1/64	0.1875
6	1/128	0.0938
7	1/256	0. 0469
23	1/16777216	0.7153 e-6

Calcul de la valeur du compteur et le nombre de bit en fonction de la fréquence désirée

La fréquence interne du kit kit de développement Elbert V2 (clk) est liée à la fréquence 12 Mhz (voir le guide):

F_clk= 12Mhz => T0 = 1/F_clk = 83.33 ns. Pour obtenir une fréquence de Fx=1/Tx , il faut compter Nx période de T0, donc :

Tx = Nx * T0 => Nx = Tx/T0 = Tx*F0= F0/(2*Fx) avec un rapport de deux pret.

Ex : Pour obtenir une horloge de 1Hz (Fx=1Hz) : Nx = F0/(2*Fx)=12e6/2 =6.000.000 période de 83.33 ns

Calculer le nombre des bits du compteur

Pour savoir le nombre des bits, il suffit de convertir la valeur décimal en héxadécimal, le

nombre des bits égal à la longeur du mot

Ex : 16e = 0x”5B8D80″ = b’101101110001101100000000′, donc le nombre des bits égal à 23 bits !

Site pour convertir un nombre décimal en Héxa ou binaire : click içi

Programme VHDL

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


entity ClkGen is

        GENERIC
   (
      N : positive :=24;
                M : positive :=8
   );

    Port ( clk_12M          : in  STD_LOGIC;
           CE                          : in  STD_LOGIC:='0';
           RST                 : in  STD_LOGIC:='0';

                           Clk_6M          : out  STD_LOGIC :='0'; --
6MHz
                           Clk_3M          : out  STD_LOGIC :='0'; --
3MHz
                           Clk_750k          : out  STD_LOGIC :='0'; --
750KHz
                           Clk_1P46k : out  STD_LOGIC :='0'; -- 1.46KHz
                           Clk_22P88 : out  STD_LOGIC :='0'; -- 22.88Hz
                           Clk_11P44 : out  STD_LOGIC :='0'; -- 11.44Hz
                           Clk_2P86         : out  STD_LOGIC :='0'; --
2.86Hz

                           Clk_1_HZ          : out STD_LOGIC :='0' ;
```

```
  Seg_value : out std_logic_vector(M-1 downto 0):=x"3F";
                          TransEN1        : out STD_LOGIC :='0' ;
                          TransEN2        : out STD_LOGIC :='0' ;
                          TransEN3        : out STD_LOGIC :='0'


                          );
end ClkGen;

architecture Beh_ClkGen of ClkGen is

SIGNAL         Count_tmp : std_logic_vector(N-1 downto 0):= x"000000";
SIGNAL
Count_1_sec : std_logic_vector(N-1 downto 0):= x"000000";
CONSTANT Value_1_sec : std_logic_vector(N-1 downto 0):= x"5B8D80"; --
 = F0/2 dec2hex(12e6/2)(matlab)
SIGNAL         Clk_1_HZ_tmp : std_logic:='0';


SIGNAL         sel : std_logic_vector(3 downto 0):=x"0";
SIGNAL         Seg_temp : std_logic_vector(M-1 downto 0):=x"FF";
TYPE
T_DATA is array (0 to 9) of std_logic_vector(M-1 downto 0);

-- Anode commune
CONSTANT SEG_7 : T_DATA :=

            (x"C0",  -- '0'
                                 x"F9",  -- '1'
             x"A4",  -- '2'
             x"B0",  -- '3'
             x"99",  -- '4'
```

```
                                    x"92",  -- '5'
                                    x"82",  -- '6'
                                    x"F8",  -- '7'
                                    x"80",  -- '8'
                                    x"90"); -- '9'


BEGIN

-- BCD TO 7 SEG CONVERTER

        PROCESS (Clk_1_HZ_tmp,sel,RST)
                BEGIN
                        IF RST = '1' THEN
                                Seg_temp <= x"00";
ELSIF (Clk_1_HZ_tmp'EVENT AND Clk_1_HZ_tmp='1') THEN
                                CASE sel IS
WHEN x"0" => Seg_temp<=SEG_7(0);
WHEN x"1" => Seg_temp<=SEG_7(1);
WHEN x"2" => Seg_temp<=SEG_7(2);
WHEN x"3" => Seg_temp<=SEG_7(3);
WHEN x"4" => Seg_temp<=SEG_7(4);
WHEN x"5" => Seg_temp<=SEG_7(5);
WHEN x"6" => Seg_temp<=SEG_7(6);
WHEN x"7" => Seg_temp<=SEG_7(7);
WHEN x"8" => Seg_temp<=SEG_7(8);
WHEN x"9" => Seg_temp<=SEG_7(9);
WHEN OTHERS => Seg_temp<=SEG_7(0);
                                END CASE ;
                    END IF;
        END PROCESS;
```

```
        Seg_value<=Seg_temp;
        TransEN1<= '0';
        TransEN2<= '0';
        TransEN3<= '0';

--
 Process de controle de l'afficheur 7 Segs : Incrémentation chaque séc
onde
-- de la valeur de sélection [Horloge 0-9 seconde]
        PROCESS (Clk_1_HZ_tmp, RST,CE)
                BEGIN
                 IF RST ='1' THEN
                        sel <= x"0";
                 ELSIF (Clk_1_HZ_tmp'EVENT AND Clk_1_HZ_tmp='1') THEN
                        IF CE ='1' THEN
                                sel<= sel + 1 ;
                                IF sel = x"9" THEN
                                        sel<= x"0";
                                END IF ;
                        ELSE
                                sel<=sel;
                        END IF;
                 END IF ;
        END PROCESS;


-- Génerateur des clocks de 6 MHh ----> 11Hz

        PROCESS (clk_12M, RST,CE)
                BEGIN
                 IF RST ='1' THEN
                        Count_tmp <= x"000000"; --(others =>'0')
```

```
                ELSIF (clk_12M'EVENT AND clk_12M='1') THEN
                        IF CE ='1' THEN
                                Count_tmp<= Count_tmp + 1 ;
                        ELSE
                                Count_tmp<=Count_tmp;
                        END IF;
                END IF ;
        END PROCESS;


        Clk_6M<= Count_tmp(0);
        Clk_3M<= Count_tmp(1);
        Clk_750k<= Count_tmp(3);
        Clk_1P46k<= Count_tmp(12); -- 1.46KHz
        Clk_22P88<= Count_tmp(18); -- 22.88Hz
        Clk_11P44<= Count_tmp(19); -- 11.44Hz
        Clk_2P86<= Count_tmp(N-2); -- 2.86Hz

-- Génerateur d'une horloge de 1 Hz : Horloge de l'afficheur 7 Seg
        SEC_1 : PROCESS (clk_12M, RST,CE)
                BEGIN
                 IF RST ='1' THEN
                        Count_1_sec <= x"000000";
                        Clk_1_HZ_tmp<='0';
                 ELSIF (clk_12M'EVENT AND clk_12M='1') THEN
                        IF CE ='1' THEN
                                Count_1_sec<= Count_1_sec + 1 ;
                                IF Count_1_sec = Value_1_sec THEN
                                        Count_1_sec <= x"000000";
Clk_1_HZ_tmp<= not(Clk_1_HZ_tmp);
                           END IF ;
                        ELSE
```

```
                                Count_1_sec<=Count_1_sec;
                        END IF;
                END IF ;
        END PROCESS;

        Clk_1_HZ <= Clk_1_HZ_tmp;

end Beh_ClkGen;
```

Fichier Pinout

```
CONFIG VCCAUX = "3.3" ;
# Clock 12 MHz
NET "clk_12M"                   LOC = P129  | IOSTANDARD = LVCMOS33 | P
ERIOD = 12MHz;
#NET "Clk"
LOC = P57   | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;


    NET "Clk_6M"
LOC = P46   | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 12;
    NET "Clk_3M"
LOC = P47   | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 12;
    NET "Clk_750k"
LOC = P48   | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 12;
    NET "Clk_1P46k"
LOC = P49   | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 12;
    NET "Clk_22P88"
LOC = P50   | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 12;
    NET "Clk_11P44"
LOC = P51   | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 12;
    NET "Clk_2P86"
LOC = P54   | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 12;

    NET "Clk_1_HZ"
LOC = P55   | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 12;


    # DP SWITCH
    NET "CE"
LOC = P70   | PULLUP  | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE =
12;
    NET "RST"
```

```
LOC = P69   | PULLUP  | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE =
12;


NET "Seg_value[0]"    LOC = P117  | IOSTANDARD = LVCMOS33 | SLEW = FAS
T | DRIVE = 12;
    NET "Seg_value[1]"    LOC = P116  | IOSTANDARD = LVCMOS33 | SLEW =
 FAST | DRIVE = 12;
    NET "Seg_value[2]"    LOC = P115  | IOSTANDARD = LVCMOS33 | SLEW =
 FAST | DRIVE = 12;
    NET "Seg_value[3]"    LOC = P113  | IOSTANDARD = LVCMOS33 | SLEW =
 FAST | DRIVE = 12;
    NET "Seg_value[4]"    LOC = P112  | IOSTANDARD = LVCMOS33 | SLEW =
 FAST | DRIVE = 12;
    NET "Seg_value[5]"    LOC = P111  | IOSTANDARD = LVCMOS33 | SLEW =
 FAST | DRIVE = 12;
    NET "Seg_value[6]"    LOC = P110  | IOSTANDARD = LVCMOS33 | SLEW =
 FAST | DRIVE = 12;
    NET "Seg_value[7]"    LOC = P114  | IOSTANDARD = LVCMOS33 | SLEW =
 FAST | DRIVE = 12;

NET "TransEN1"        LOC = P124  | IOSTANDARD = LVCMOS33 | SLEW = FAS
T | DRIVE = 12;
    NET "TransEN2"
LOC = P121  | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
    NET "TransEN3"        LOC = P120  | IOSTANDARD = LVCMOS33 | SLEW =
 SLOW | DRIVE = 12;
```

Programme de simulation (Testbanch)

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY ClkGen_Test IS
END ClkGen_Test;

ARCHITECTURE behavior OF ClkGen_Test IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT ClkGen
    PORT(
         clk_12M : IN  std_logic;
         CE : IN  std_logic;
         RST : IN  std_logic;
         Clk_6M : OUT  std_logic;
         Clk_3M : OUT  std_logic;
         Clk_750k : OUT  std_logic;
         Clk_1P46k : OUT  std_logic;
         Clk_22P88 : OUT  std_logic;
         Clk_11P44 : OUT  std_logic;
                        Clk_1_HZ : OUT  std_logic
        );
    END COMPONENT;


   --Inputs
   signal clk_12M : std_logic := '0';
   signal CE : std_logic := '0';
   signal RST : std_logic := '0';

         --Outputs
   signal Clk_6M : std_logic;
```

```
   signal Clk_3M : std_logic;
   signal Clk_750k : std_logic;
   signal Clk_1P46k : std_logic;
   signal Clk_22P88 : std_logic;
   signal Clk_11P44 : std_logic;
        signal Clk_1_HZ : std_logic;

   -- Clock period definitions
   constant clk_12M_period : time := 83.33333333333333333
ns;

BEGIN

        -- Instantiate the Unit Under Test (UUT)
   uut: ClkGen PORT MAP (
          clk_12M => clk_12M,
          CE => CE,
          RST => RST,
          Clk_6M => Clk_6M,
          Clk_3M => Clk_3M,
          Clk_750k => Clk_750k,
          Clk_1P46k => Clk_1P46k,
          Clk_22P88 => Clk_22P88,
          Clk_11P44 => Clk_11P44,
                         Clk_1_HZ => Clk_1_HZ
        );

   -- Clock process definitions
   clk_12M_process :process
   begin
                clk_12M <= '0';
                wait for clk_12M_period/2;
```

```
                clk_12M <= '1';
                wait for clk_12M_period/2;
   end process;

        RST<='0';
        CE <='1';

END;
```

Photos du projet

Un petit commentaire de vous, un Grand encouragement pour nous

– Bon Courage –

Dernières réalisations

- Projet électronique FPGA #2 : Gestion de l'afficheur 7 Segments
- Projet électronique : Fréquencemètre numérique à base du microcontrôleur PIC16F877A #V1
-  Description du projet (Codes + Montage) + Téléchargement du projet complet
- Projet électronique FPGA #1 : Détecteur d'une séquence parllèle
- Projet électronique FPGA 4 #2/3 : Capteur de distance ultrasonique à base du FPGA & Arduino
- Projet électronique FPGA #6 : Commande synchrone multicanaux d'un moteur à CC
- Projet électronique FPGA #7 : Calcul de Factorielle – n!
- Projet électronique : Gestion d'une matrice des LED avec Arduino
- Projet électronique FPGA #4 #1/3 : Capteur de distance ultrasonique à base du FPGA &

Arduino

Articles

- Analyse de Fourier d’un circuit électronique – Filtre passe-bas RC – Partie 1/2
- ISIS PROTEUS : Interface Interactive
- Projets Matlab & Microcontrôleur #7: Arduino Clap Clap
- Infrarouge IR: Variateur de vitesse à MLI #2/2
- Contrôleur PWM commandé en tension
- Que veut dire AOP ?
- Comment fabriquer un détecteur EMF (champ électromagnétique) avec Arduino ?