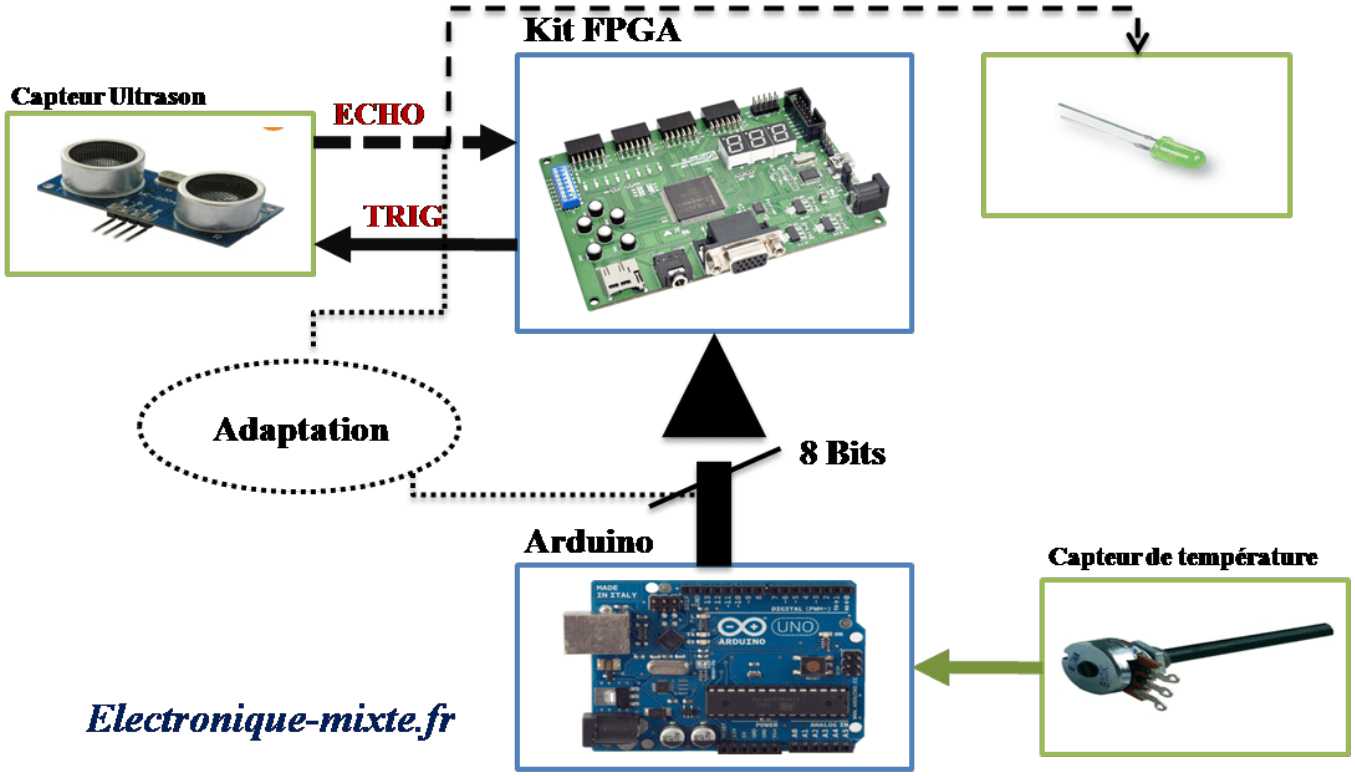
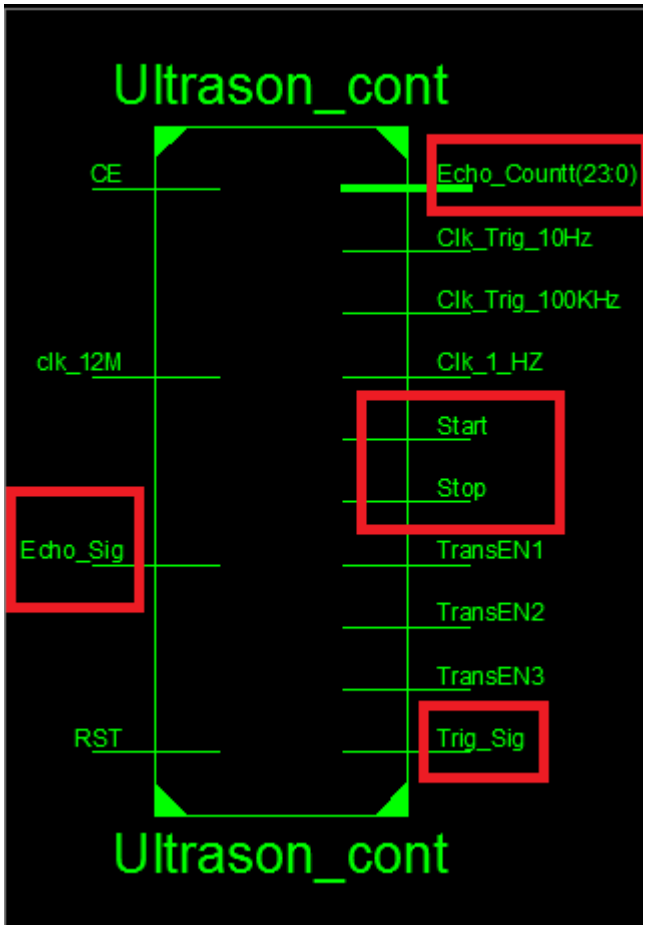

Kit FPGA
Capteur Ultrason
ECHO
TRIG
Adaptation
8 Bits
Arduino
Capteur de température
Electronique-mixte.fr



Sommaire

Objectifs du projet global :

1. Comprendre le principe du fonctionnement du capteur ultrasonique
2. Mise en ouvre de deux machines à état de Moore pour le calcul et la génération des signaux
3. Savoir comment générer un signal avec une largeur et fréquence fixent
4. Savoir comment adapter la logique Arduino (5V) et FPGA (3.3V)
5. Implémentation en virgule d'une fonction mathématique (fonction de la distance en fonction de la température)
6. Savoir comment détecter un front mentant & descendant d'un signal
7. Autres astuces de programmation

Caractéristique du capteur :

1. Fréquence maximale du système 12 MHz
2. Résolution de mesure 83 ns
3. Mesure de la distance par principe des ondes ultrasoniques

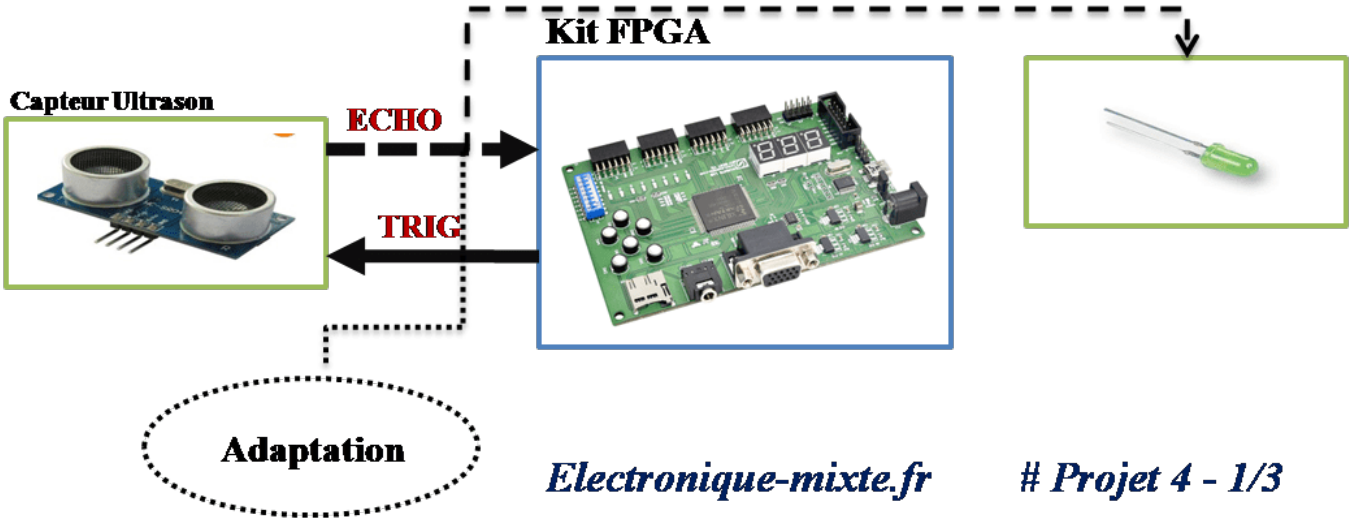
4. Capteur de température intégré
5. Résolution binaire de 8 bits du capteur de la température
6. Affichage sur 8 bits de la température (affichage par les LED)

Analyse de fonctionnement :

Le projet consiste la mesure et l'affichage de la distance par un capteur ultrasonique en onction de la température ambiante par FPGA et Arduino. Le principe de mesure utilisé est la mesure de la largeur d'impulsion par un compteur haute fréquence (12 MHz).

La première partie du projet sera consacrée à :

- La génération d'un signal de déclanchement du capteur ultrasonique(TRIG)
- L'acquisition et la mesure de la largeur de l'impulsion du signal du retour (ECHO)
- Affichage de la valeur du compteur (MSB de la valeur sur 8 bits)



Dans cette partie la température est supposée constante. Une cellule d'adaptation est

utilisée pour passer du niveau 5V au niveau 3.3V entre l'Arduino et l'FPGA.
La LED permet de visualiser la variation la largeur de l'impulsion (durée de clignotement) lorsque la distance change.

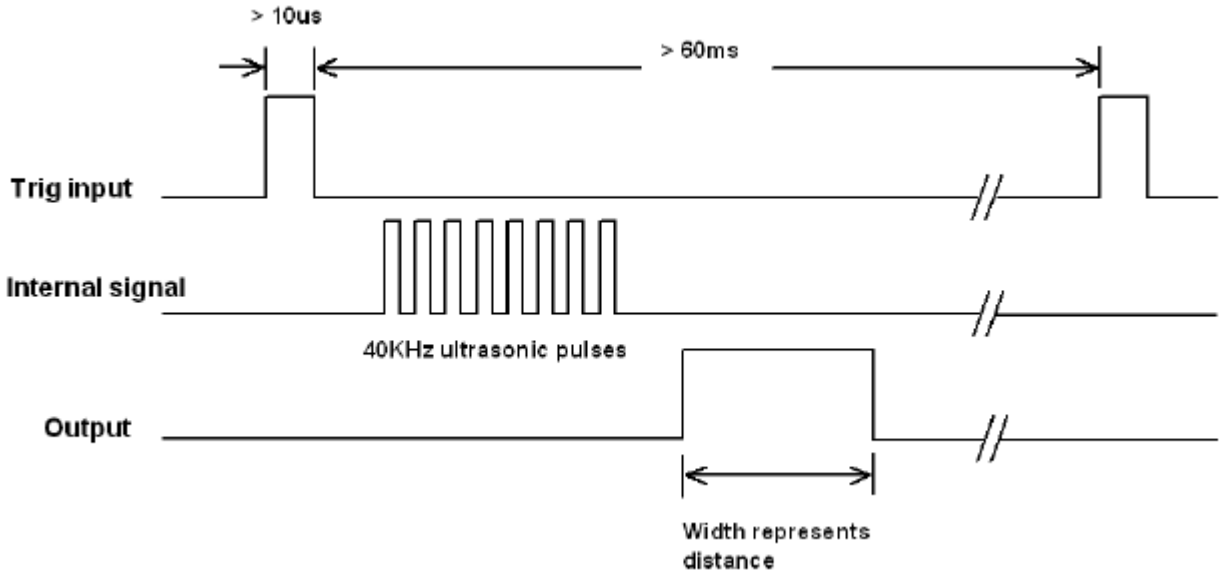
Fonctionnement du capteur ultrasonique

Les caractéristiques techniques du capteur – HC-SR04

- Alimentation : 5V
- Consommation : 15 mA.

- Portée: 2 cm à 5 m.
- Résolution : 0.3 cm.
- Angle de mesure : < 15°

Chronogramme de fonctionnement

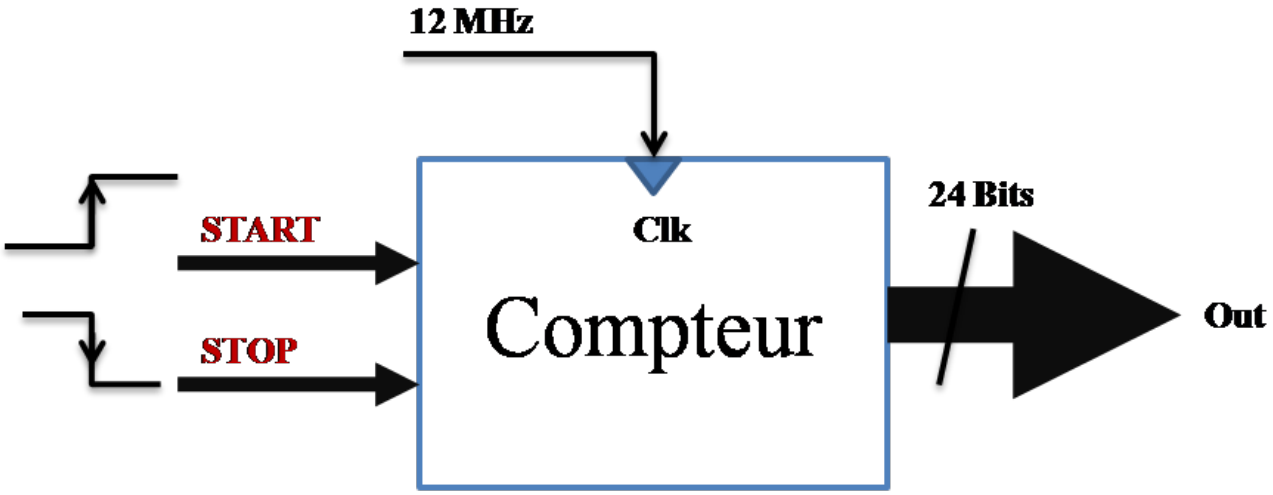


Le capteur contient deux signaux TRIG et ECHO, et une broche d'alimentation à 5V et une masse.

Le signal TRIG (une entrée) est une impulsion de 10 us qui permet de déclencher le circuit à l'intérieur du capteur pour activer l'émetteur ultrason (signal interne) et attendre l'arrivé de l'onde. Le capteur génère une sortie ECHO sous forme d'une impulsion électrique avec une largeur proportionnelle à la distance Aller/Retour.

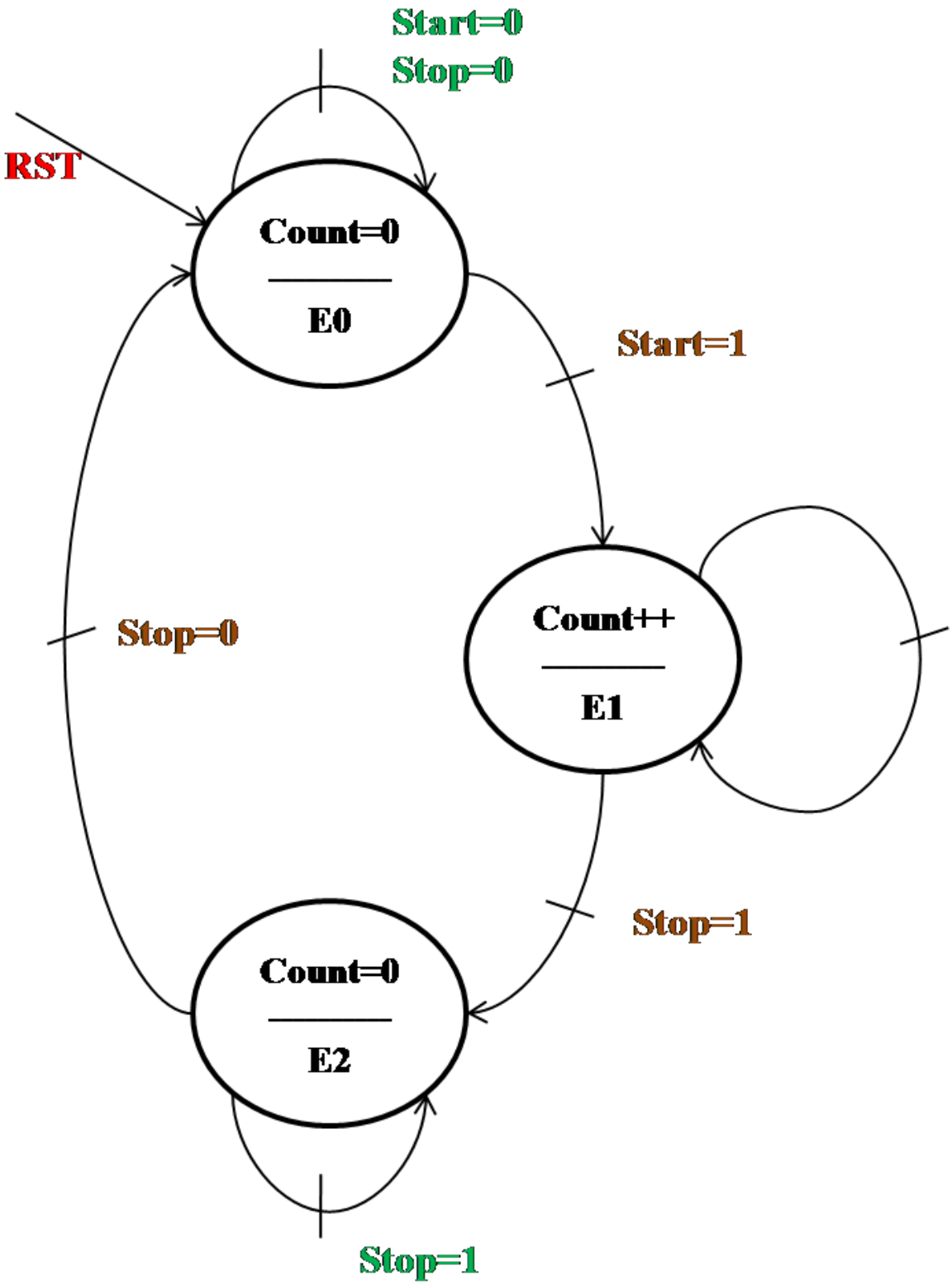
Circuit de mesure de la largeur de

l’impulsion

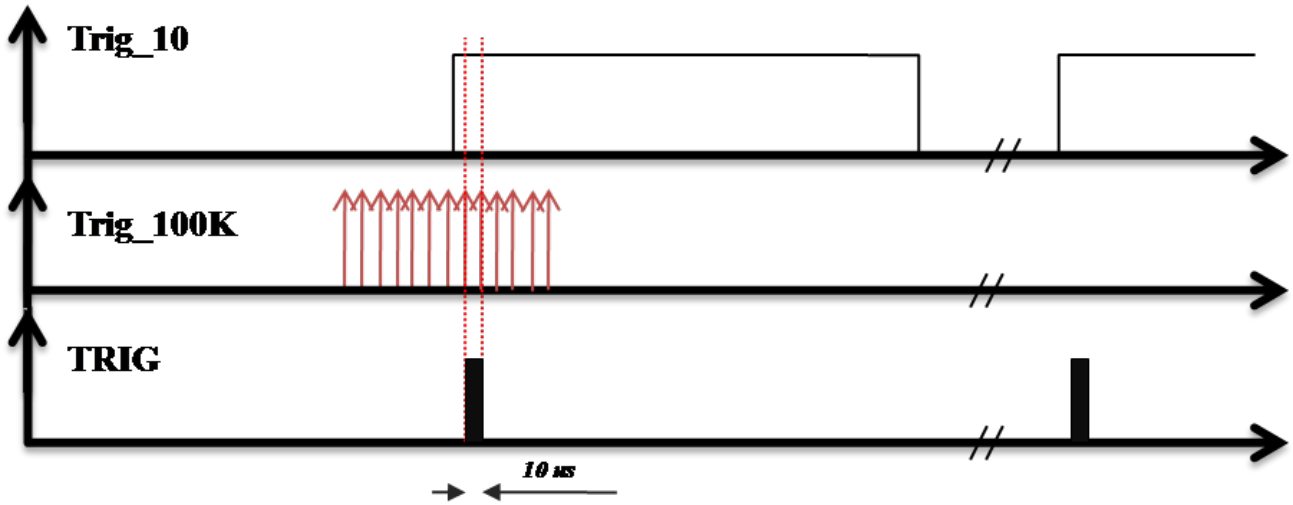


Le noya du système est basé par un compteur sur 24 bits (voir projet x concernant le choix des nombre des bits) avec une horloge de 12 MHz (83.33 ns).Le compteur commence par le front montant du signal ECHO (START) reçue du capteur ultrasonique puis s’arrête au front descendant du même signal (STOP). Deux processus sont utilisés pour la détection du front montant (START) et le font descendant (STOP) en basant sur le temps de la propagation d’une porte logique. En absence du signal ECHO le compteur mémorise la valeur précédente.

Machine du Moore du compteur (mesure de la largeur du signal ECHO)


Start=0
Stop=0
RST
Count=0
E0
Start=1
Count++
E1
Stop=0
Stop=1
Count=0
E2
Stop=1

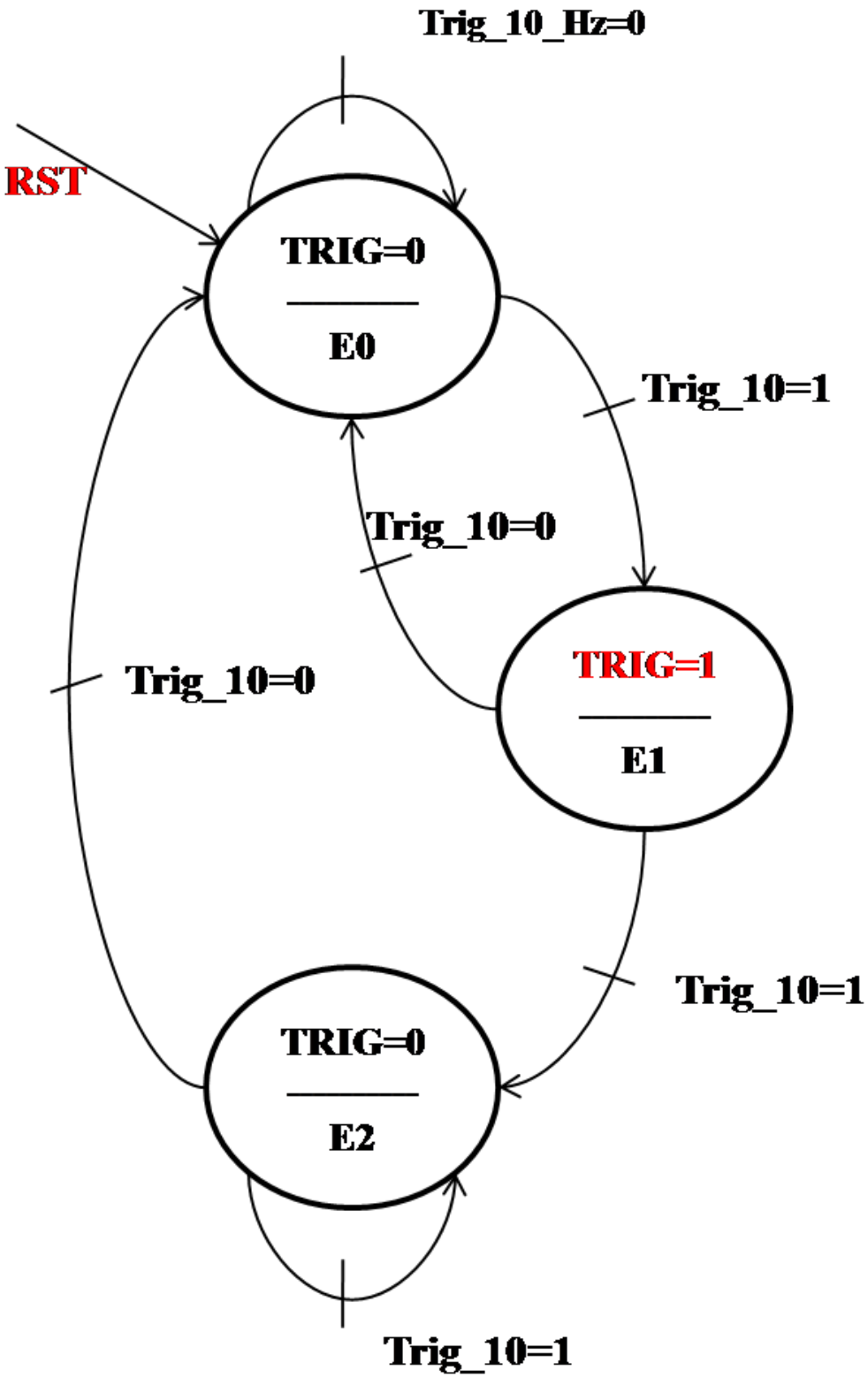
Circuit pour la génération du signal TRIG



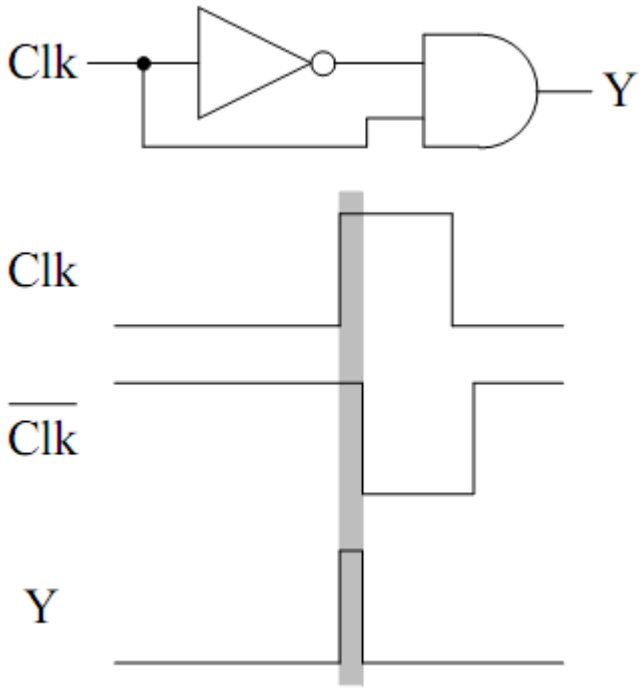
Le circuit permet de générer le signal TRIG pour le capteur ultrasonique. Le signal TRIG à une durée de 10 us (100 KHz) et une période de 100 ms (10 Hz). Le système permet de générer deux horloges différences une de 10 Hz (10us) une autre de 100Khz (100ms). La machine à état utilise l'horloge de 100Khz pour la transition en les états et le signal d'horloge de 10 Hz comme entrée.

Le signal TRIG=1 lorsque on détecte un front montant du signal long (10 Hz) pendant une période d'horloge de 100Khz (10us) puis retourne à zéro quelque soit l'état de l'entrée 10 Hz. Ci dessus la machine à état qui permet de générer une impulsion de 10 us.

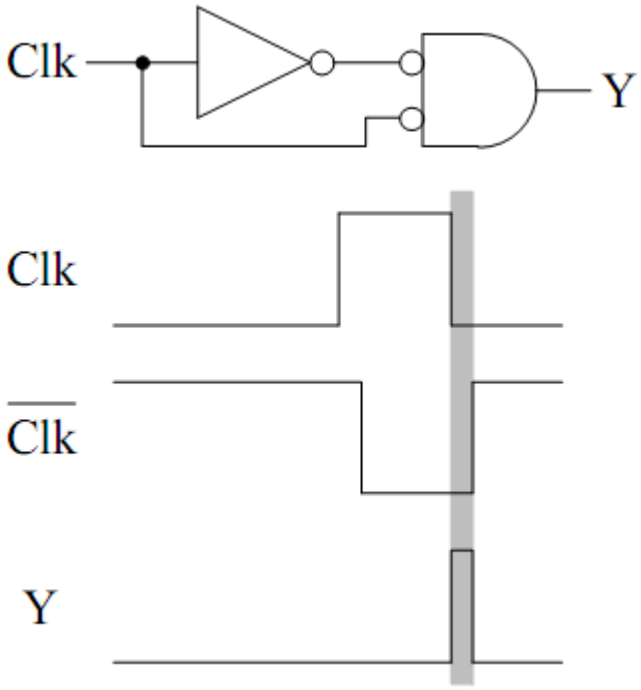
Machine du Moore pour la génération du signal TRIG


Trig_10_Hz=0
RST
TRIG=0
E0
Trig_10=1
Trig_10=0
TRIG=1
E1
Trig_10=0
Trig_10=1
TRIG=0
E2
Trig_10=1

Détection du front montant



La figure ci-dessus illustre le principe de détection d'un front montant en exploitant le temps de propagation du porte Not. On peut utiliser plusieurs inverseur en cascade pour augmenter la largeur de l'impulsion.



Code VHDL pour la détection d'un front montant et le front descendant d'un signal

```
        Rising : PROCESS (clk_12M,Echo_Sig)
        BEGIN
                IF (clk_12M'EVENT AND clk_12M='1') THEN
                        Fron_mon<=Echo_Sig;
                END IF ;
        END PROCESS;
        Start_count<=Echo_Sig and (not(Fron_mon));

        Falling : PROCESS (clk_12M,Echo_Sig)
        BEGIN
                IF (clk_12M'EVENT AND clk_12M='1') THEN
                        Fron_des<=Echo_Sig;
                END IF ;
```

```
        END PROCESS;
        Stop_count<= Fron_des and (not(Echo_Sig));

        Start<= Start_count;
        Stop<= Stop_count;
```

Cours synthèse des machines à état en VHDL

- Cours 1 : Synthèse en 1, 2 ou 3 Process
- Cours 2 : Machines à états finie

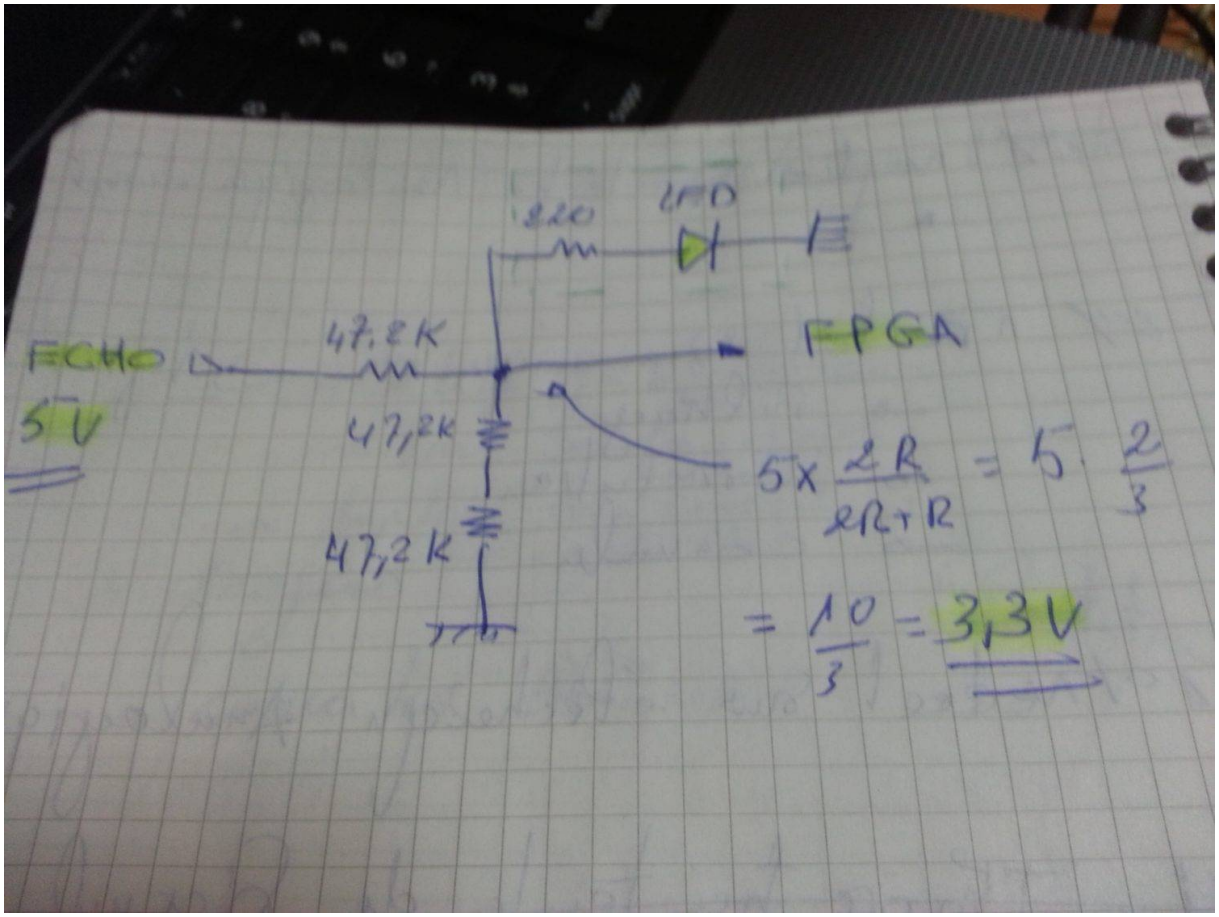
> *La méthode de synthèse utilisée dans ce projet est celle des 3 process (voirs cours 2)*

Circuit d'adaptation

Le circuit de l'adaptation est basé sur un diviseur de tension R/2R simple à mètre en œuvre intercalé entre le capteur (ou Arduino) et FPGA.

La LED permet de visualiser le signal ECHO. Le courant au borne de la LED est égal à 3.3/220 = 15mA.

On verra dans la suite du projet une astuce pratique pour convertir la logique 5V en logique 3.3V sans composants externes !



Code VHDL du projet

```
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 Ultrason_cont is

        GENERIC
   (
      N : positive :=24;
```

```
               M : positive :=8
   );

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

                          Trig_Sig         : out  STD_LOGIC :='0';

                          Clk_1_HZ         : out STD_LOGIC :='0' ;
                          Clk_Trig_100KHz        : out STD_LOGIC :='0' ;
                          Clk_Trig_10Hz        : out STD_LOGIC :='0';

                          Start
: out STD_LOGIC :='0';
                          Stop                              : out STD_LOGIC :='0';
                          TransEN1                       : out STD_LOGIC :='1';
                          TransEN2                       : out STD_LOGIC :='1';
                          TransEN3                       : out STD_LOGIC :='1';

                          Echo_Countt
: out std_logic_vector(N-1 downto 0):= x"000000"

                          );
end Ultrason_cont;

architecture Behavioral of Ultrason_cont is

SIGNAL                 Count_1_sec
: std_logic_vector(N-1 downto 0):= x"000000";
SIGNAL                 Count_10_usec
: std_logic_vector(M-1 downto 0):= x"00";
SIGNAL                 Count_100_msec : std_logic_vector(N-5 downto 0):= x"00000";

SIGNAL                 Clk_1_HZ_tmp         : std_logic:='0';
SIGNAL                 Clk_Trig_100K         : std_logic:='0';         -- 10us
SIGNAL                 Clk_Trig_10         : std_logic:='0';    -- 100ms


-- FSM Trig signal genertor
TYPE                         Etat is (E0, E1, E2);
SIGNAL                Etat_present, Etat_utur : Etat :=E0;
```

```
-- FSM Echo signal Reader
TYPE                        Etatt is (E00, E11);
SIGNAL                Etat_presentt, Etat_uturr : Etatt :=E00;

SIGNAL                 Fron_mon                 : std_logic:='0';
SIGNAL                 Fron_des                 : std_logic:='0';

SIGNAL                 Start_count         : std_logic:='0';
SIGNAL                 Stop_count         : std_logic:='0';
SIGNAL                 Echo_count
: std_logic_vector(N-1 downto 0):= x"000000";

BEGIN

-- Génerateur des clocks 12MHh----> 1Hz

        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 = x"5B8D80" 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;

-- Génerateur des clocks 12MHh----> 100KHz 10u

        uSEC_10 : PROCESS (clk_12M, RST,CE)
        BEGIN
                 IF RST ='1' THEN
                        Count_10_usec <= x"00";
                        Clk_Trig_100K<='0';
                 ELSIF (clk_12M'EVENT AND clk_12M='1') THEN
```

```
                        IF CE ='1' THEN
                                Count_10_usec<= Count_10_usec + 1 ;
                                IF Count_10_usec = x"3C" THEN
                                        Count_10_usec <= x"00";
                                        Clk_Trig_100K<= not(Clk_Trig_100K);
                           END IF ;
                        ELSE
                                Count_10_usec<=Count_10_usec;
                        END IF;
                END IF ;
        END PROCESS;

        Clk_Trig_100KHz<= Clk_Trig_100K;


-- Génerateur des clocks 12MHh----> 10Hz

        mSEC_100 : PROCESS (clk_12M, RST,CE)
        BEGIN
                IF RST ='1' THEN
                        Count_100_msec <= x"00000";
                        Clk_Trig_10<='0';
                ELSIF (clk_12M'EVENT AND clk_12M='1') THEN
                        IF CE ='1' THEN
                                Count_100_msec<= Count_100_msec + 1 ;
                                IF Count_100_msec = x"927C0" THEN
                                        Count_100_msec <= x"00000";
                                        Clk_Trig_10<= not(Clk_Trig_10);
                           END IF ;
                        ELSE
                                Count_100_msec<=Count_100_msec;
                        END IF;
                END IF ;
        END PROCESS;

        Clk_Trig_10Hz<= Clk_Trig_10;

-- FSM Génertion du signal TRIG (une impultion de 10 us de fréquence 10 Hz)

        Mem : PROCESS(RST,Clk_Trig_100K)
        BEGIN
                IF RST ='1' THEN
                        Etat_present <= E0;
                 ELSIF (Clk_Trig_100K'EVENT AND Clk_Trig_100K='1') THEN
```

```
                        Etat_present<=Etat_utur;
                END IF ;
        END PROCESS;

        Combin : PROCESS(Etat_present,Clk_Trig_10)
        BEGIN
                CASE Etat_present IS
                        WHEN E0 =>          IF Clk_Trig_10 ='1' THEN
                                                        Etat_utur<=E1;
                                                ELSE
                                                        Etat_utur<=E0;
                                                END IF;

                        WHEN E1 =>          IF Clk_Trig_10 ='1' THEN
                                                        Etat_utur<=E2;
                                                ELSE
                                                        Etat_utur<=E0;
                                                END IF;

                        WHEN E2 =>          IF Clk_Trig_10 ='1' THEN
                                                        Etat_utur<=E2;
                                                ELSE
                                                        Etat_utur<=E0;
                                                END IF;
                END CASE;
        END PROCESS;

        Sortie : PROCESS(Etat_present)
        BEGIN
                IF Etat_present = E1 THEN
                        Trig_Sig<='1';
                ELSE
                        Trig_Sig<='0';
                END IF;
        END PROCESS;

-- FSM Mesure de la largeur du signal ECHO avec une précision de 83.33ns

        Memm : PROCESS(RST,clk_12M)
        BEGIN
                IF RST ='1' THEN
                        Etat_presentt <= E00;
                 ELSIF (clk_12M'EVENT AND clk_12M='1') THEN
                        Etat_presentt<=Etat_uturr;
```

```
                 END IF ;
        END PROCESS;

        Combinn : PROCESS(Etat_presentt,Stop_count,Start_count)
        BEGIN
                CASE Etat_presentt IS
                        WHEN E00 => IF Start_count ='1' THEN
                                                                Etat_uturr<=E11;
                                                        ELSE
                                                                Etat_uturr<=E00;
--
Echo_Countt<=x"000000";
                                                        END IF;

                        WHEN E11 => IF Stop_count ='1' THEN
                                                                Etat_uturr<=E00;
Echo_Countt<=Echo_count;
                                                        ELSE
                                                                Etat_uturr<=E11;
                                                        END IF;
                END CASE;
        END PROCESS;

        Sortiee : PROCESS(Etat_presentt,clk_12M )
        BEGIN
                IF (clk_12M'EVENT AND clk_12M='1') THEN
                        IF Etat_presentt = E00 THEN
                                Echo_count<=x"000000";
                        ELSE
                                Echo_count<=Echo_count+1;
                        END IF;
                END IF;
        END PROCESS;


--        Process pour la détection du ront montnt et le front descendant

        Rising : PROCESS (clk_12M,Echo_Sig)
        BEGIN
                IF (clk_12M'EVENT AND clk_12M='1') THEN
                        Fron_mon<=Echo_Sig;
                END IF ;
        END PROCESS;
        Start_count<=Echo_Sig and (not(Fron_mon));
```

```
        Falling : PROCESS (clk_12M,Echo_Sig)
        BEGIN
                IF (clk_12M'EVENT AND clk_12M='1') THEN
                        Fron_des<=Echo_Sig;
                END IF ;
        END PROCESS;
        Stop_count<= Fron_des and (not(Echo_Sig));

        Start<= Start_count;
        Stop<= Stop_count;

        TransEN1<='1';
        TransEN2<='1';
        TransEN3<='1';

--
 Calcul & Afficher de la distance  d = 340*t/2 = N*T0*170 = 170*83.33ns*N = 14.16*N
(us)
-- N : Valeur du compteur Echo_Countt
-- Next project ...

End Behavioral;
```

Fichier VHDL de simulation

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--USE ieee.numeric_std.ALL;

ENTITY Test_US IS
END Test_US;

ARCHITECTURE behavior OF Test_US IS

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

```
    COMPONENT Ultrason_cont
    PORT(
         clk_12M : IN  std_logic;
         CE : IN  std_logic;
         RST : IN  std_logic;
         Echo_Sig : IN  std_logic;
         Trig_Sig : OUT  std_logic;
         Clk_1_HZ : OUT  std_logic;
         Clk_Trig_100KHz : OUT  std_logic;
         Clk_Trig_10Hz : OUT  std_logic;
         Echo_Countt : OUT  std_logic_vector(23 downto 0);

                        Start                                    : out STD_LOGIC ;
                        Stop                                   : out STD_LOGIC

        );
    END COMPONENT;


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

          --Outputs
   signal Trig_Sig : std_logic;
   signal Clk_1_HZ : std_logic;
   signal Clk_Trig_100KHz : std_logic;
   signal Clk_Trig_10Hz : std_logic;
   signal Echo_Countt : std_logic_vector(23 downto 0);

         signal Start                                    :  STD_LOGIC ;
         signal Stop                                   :  STD_LOGIC;

   -- Clock period definitions
   constant clk_12M_period : time := 83.333333333 ns;
--   constant Clk_1_HZ_period : time := 10 ns;
--   constant Clk_Trig_100KHz_period : time := 10 ns;
--   constant Clk_Trig_10Hz_period : time := 10 ns;

BEGIN

        -- Instantiate the Unit Under Test (UUT)
```

```
   uut: Ultrason_cont PORT MAP (
          clk_12M => clk_12M,
          CE => CE,
          RST => RST,
          Echo_Sig => Echo_Sig,
          Trig_Sig => Trig_Sig,
          Clk_1_HZ => Clk_1_HZ,
          Clk_Trig_100KHz => Clk_Trig_100KHz,
          Clk_Trig_10Hz => Clk_Trig_10Hz,
          Echo_Countt => Echo_Countt,

                         Start=>Start,
                         Stop=>Stop
        );

   -- 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';

stim_proc: process
   begin

        Echo_Sig<='0';
        wait for 10 ms ;

        Echo_Sig<='1';
        wait for 10 ms ;
        Echo_Sig<='0';

        wait for 10 ms ;

        Echo_Sig<='1';
        wait for 20 ms ;
        Echo_Sig<='0';

        wait for 10 ms ;
```

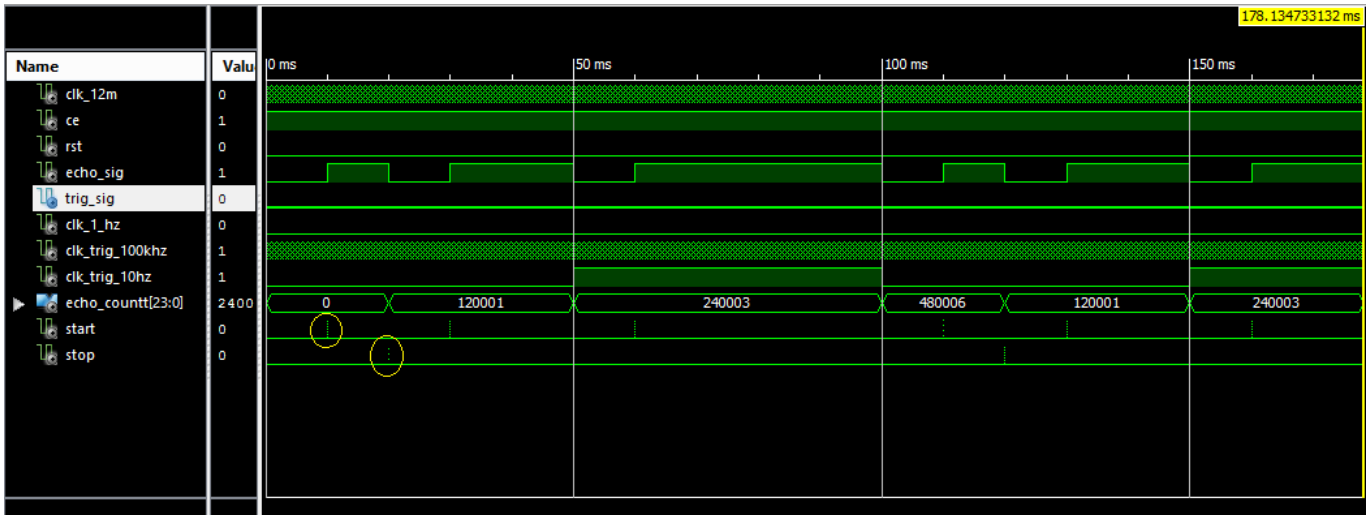
```
        Echo_Sig<='1';
        wait for 40 ms ;
        Echo_Sig<='0';

        END PROCESS;

END;
```

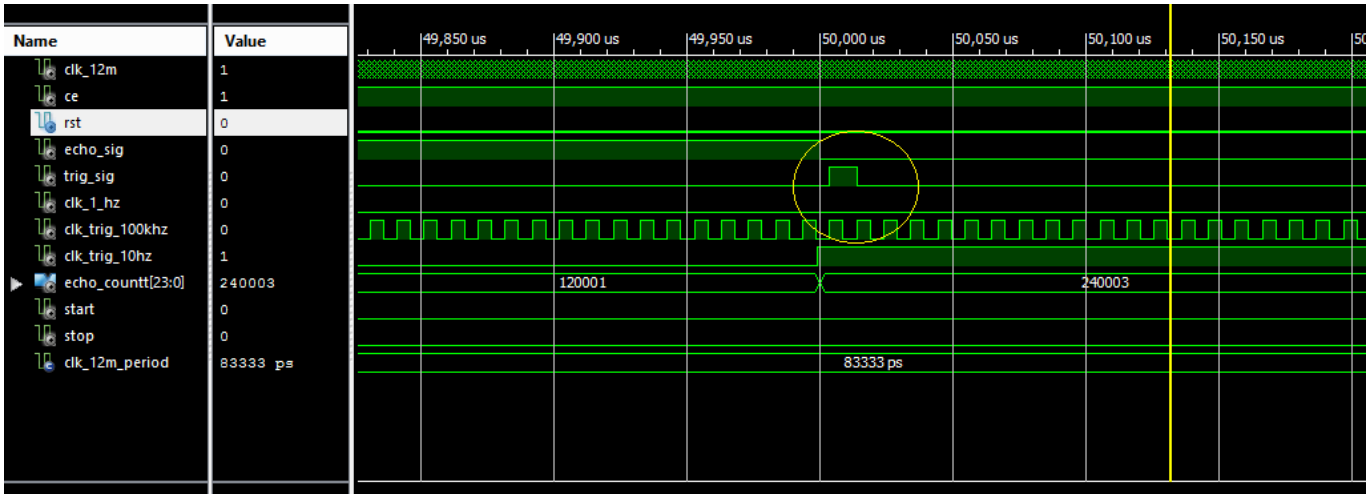
Fichier de contrainte (Pinout) du FPGA :

```
CONFIG VCCAUX = "3.3" ;

    # Clock 12 MHz
NET "clk_12M"                  LOC = P129  | IOSTANDARD = LVCMOS33 | PERIOD = 12MHz;


NET "Echo_Countt[10]"             LOC = P46   | IOSTANDARD = LVCMOS33 | SLEW = SLOW
| DRIVE = 12;
    NET "Echo_Countt[11]"             LOC = P47   | IOSTANDARD = LVCMOS33 | SLEW = S
LOW | DRIVE = 12;
    NET "Echo_Countt[12]"             LOC = P48   | IOSTANDARD = LVCMOS33 | SLEW = S
LOW | DRIVE = 12;
    NET "Echo_Countt[13]"             LOC = P49   | IOSTANDARD = LVCMOS33 | SLEW = S
LOW | DRIVE = 12;
    NET "Echo_Countt[14]"             LOC = P50   | IOSTANDARD = LVCMOS33 | SLEW = S
LOW | DRIVE = 12;
    NET "Echo_Countt[15]"             LOC = P51   | IOSTANDARD = LVCMOS33 | SLEW = S
LOW | DRIVE = 12;
    NET "Echo_Countt[16]"             LOC = P54   | IOSTANDARD = LVCMOS33 | SLEW = S
LOW | DRIVE = 12;
    NET "Echo_Countt[17]"             LOC = P55   | IOSTANDARD = LVCMOS33 | SLEW = S
LOW | DRIVE = 12;

    NET "CE"
LOC = P70   | PULLUP  | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
    NET "RST"
LOC = P69   | PULLUP  | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;

    NET "Trig_Sig"
LOC = P31   | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;  #0
```

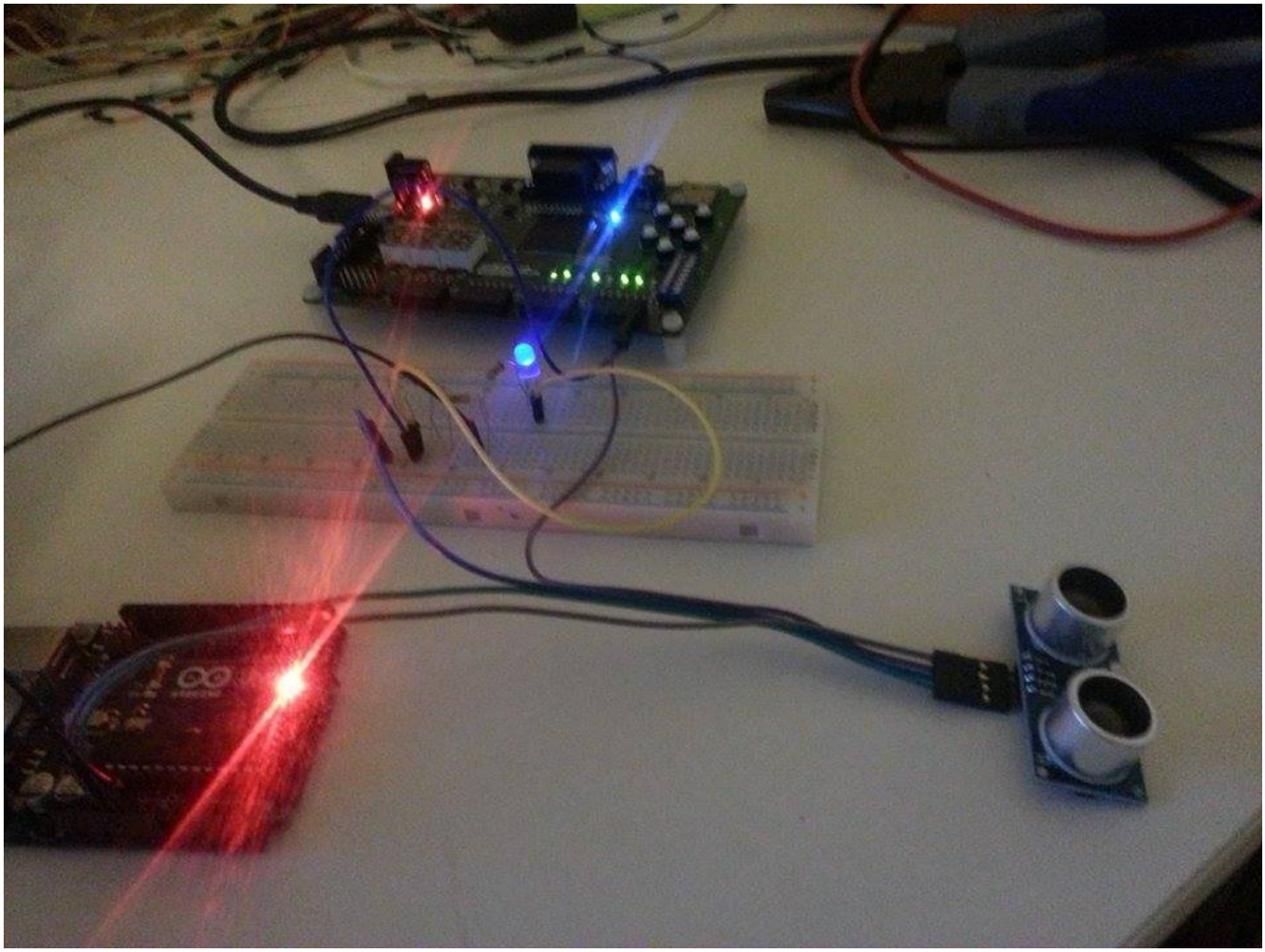
```
    NET "Echo_Sig"
LOC = P32   | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;  # 1

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

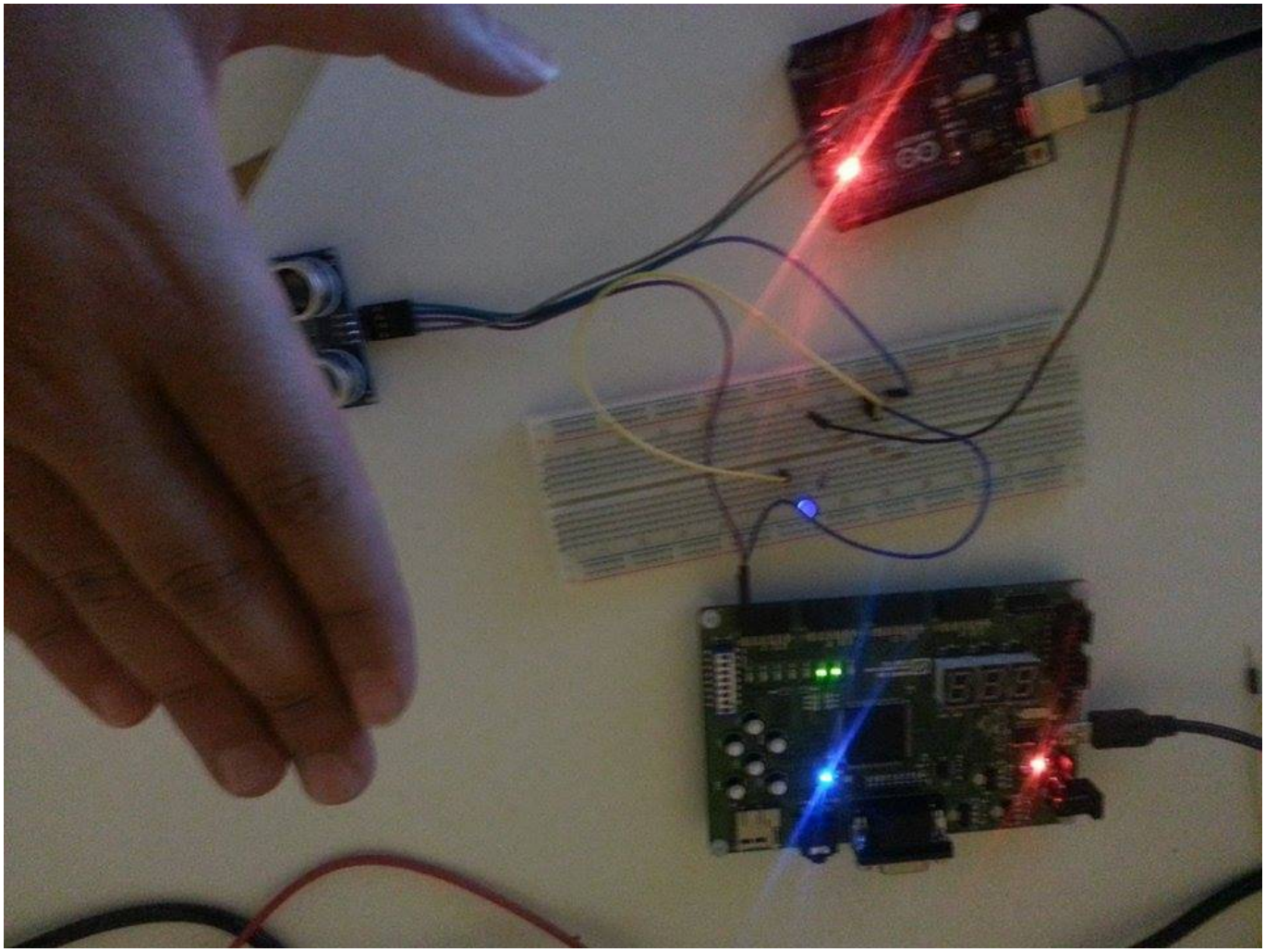
Résultats de simulation





Photos du projet





Un petit commentaire de vous, un Grand encouragement pour nous

– Bon Courage –