

Práctica 7.

DISEÑO DEL CONTROL DE SENSORES ULTRASÓNICO

OBJETIVO:

El alumno aprenderá a diseñar mediante la utilización de atributos a señales ('HIGH') y tipos de variables (UNSIGNED) el control de un sensor ultrasónico (HC-SR04).

ESPECIFICACIONES:

Diseñar un circuito utilizando FPGA que se encargue de calcular la distancia de un obstáculo por medio de un sensor ultrasónico (HC-SR04), y observar los resultados de distancia por medio de 2 displays de 7 segmentos. La figura 7.1 muestra el diagrama a bloques del sistema.

DIAGRAMA DE BLOQUES:

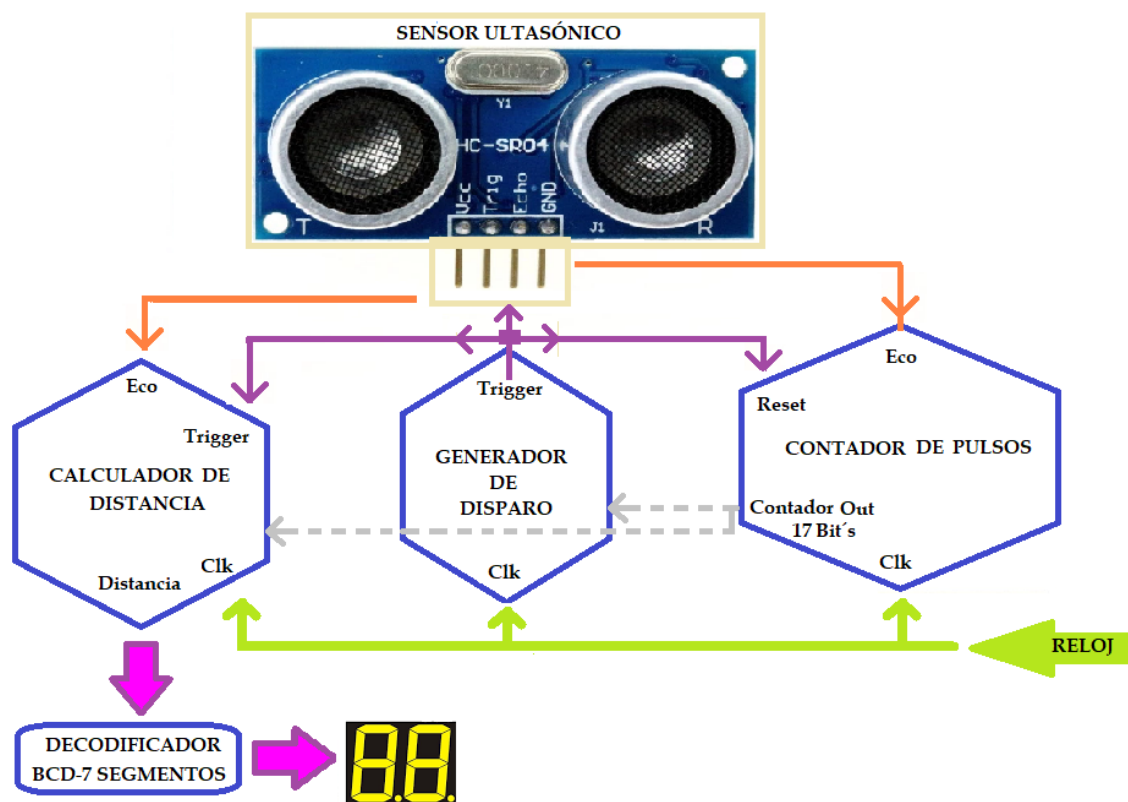


Figura 7.1. Diagrama a bloques del control para el sensor ultrasónico

Las siguientes figuras muestran el código del control para el sensor ultrasónico, que estará contenido en el archivo **sonicos.vhd**. El código fue separado, para su mejor comprensión, de acuerdo a como está mostrado en el diagrama a bloques.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity sonicos is
    Port (clk: in STD_LOGIC;
          sensor_disp: out STD_LOGIC;
          sensor_eco: in STD_LOGIC;
          anodos: out STD_LOGIC_VECTOR (3 downto 0);
          segmentos: out STD_LOGIC_VECTOR (7 downto 0));
end sonicos;

architecture Behavioral of sonicos is
    signal cuenta: unsigned(16 downto 0) := (others => '0');
    signal centimetros: unsigned(15 downto 0) := (others => '0');
    signal centimetros_unid: unsigned(3 downto 0) := (others => '0');
    signal centimetros_dece: unsigned(3 downto 0) := (others => '0');
    signal sal_unid: unsigned(3 downto 0) := (others => '0');
    signal sal_dece: unsigned(3 downto 0) := (others => '0');
    signal digito: unsigned(3 downto 0) := (others => '0');
    signal eco_pasado: std_logic := '0';
    signal eco_sinc: std_logic := '0';
    signal eco_nsinc: std_logic := '0';
    signal espera: std_logic := '0';
    signal siete_seg_cuenta: unsigned(15 downto 0) := (others => '0');
begin
    anodos(1 downto 0) <= "11";

    siete_seg: process(clk)
    begin
        if rising_edge(clk) then
            if siete_seg_cuenta(siete_seg_cuenta'high) = '1' then
                digito <= sal_unid;
                anodos(3 downto 2) <= "01";
            else
                digito <= sal_dece;
                anodos(3 downto 2) <= "10";
            end if;
            siete_seg_cuenta <= siete_seg_cuenta + 1;
        end if;
    end process;
end Behavioral;
```

Figura 7.2. Código para la entidad y arquitectura de sonicos.vhd

La Figura 7.3 se observa el código de la señal Trigger.

```

Trigger:process(clk)
begin
  if rising_edge(clk) then
    if espera = '0' then
      if cuenta = 500 then
        sensor_disp <= '0';
        espera <= '1';
        cuenta <= (others => '0');
      else
        sensor_disp <= '1';
        cuenta <= cuenta+1;
      end if;
    end if;
  end if;
end process;

```

Figura 7.3. Código del bloque generador de disparo (Trigger)

La Figura 7.4 se observa el código de los bloques calculador de distancia y contador de pulsos.

```

elseif eco_pasado = '0' and eco_sinc = '1' then
  cuenta <= (others => '0');
  centimetros <= (others => '0');
  centimetros_unid <= (others => '0');
  centimetros_dece <= (others => '0');
elseif eco_pasado = '1' and eco_sinc = '0' then
  sal_unid <= centimetros_unid;
  sal_dece <= centimetros_dece;
elseif cuenta = 2900-1 then
  if centimetros_unid = 9 then
    centimetros_unid <= (others => '0');
    centimetros_dece <= centimetros_dece + 1;
  else
    centimetros_unid <= centimetros_unid + 1;
  end if;
  centimetros <= centimetros + 1;
  cuenta <= (others => '0');
  if centimetros = 3448 then
    espera <= '0';
  end if;
else
  cuenta <= cuenta + 1;
end if;
eco_pasado <= eco_sinc;
eco_sinc <= eco_nsinc;
eco_nsinc <= sensor_eco;
end if;
end process;

```

Figura 7.4 Código del bloque calculador de distancia y del bloque contador de pulsos

La figura 7.5 muestra el código para la decodificación de datos a dos displays de siete segmentos; este código está diseñado para utilizar displays de ánodo común.

```
Decodificador: process (digito)
begin
    if      digito=X"0" then segmentos <= X"81"; -- G F E D C B A dp
    elsif  digito=X"1" then segmentos <= X"F3";
    elsif  digito=X"2" then segmentos <= X"49";
    elsif  digito=X"3" then segmentos <= X"61";
    elsif  digito=X"4" then segmentos <= X"33";
    elsif  digito=X"5" then segmentos <= X"25";
    elsif  digito=X"6" then segmentos <= X"05";
    elsif  digito=X"7" then segmentos <= X"F1";
    elsif  digito=X"8" then segmentos <= X"01";
    elsif  digito=X"9" then segmentos <= X"21";
    elsif  digito=X"a" then segmentos <= X"11";
    elsif  digito=X"b" then segmentos <= X"07";
    elsif  digito=X"c" then segmentos <= X"8D";
    elsif  digito=X"d" then segmentos <= X"43";
    elsif  digito=X"e" then segmentos <= X"0D";
    else
        segmentos<= X"1D";
    end if;
end process;
end Behavioral;
```

Figura 7.5 Código del Bloque Decodificador BCD-7 Segmentos.

ACTIVIDADES COMPLEMENTARIAS:

1.- A partir del código presentado, el alumno desarrollará un bloque funcional genérico para utilizar un medidor de distancia ultrasónico, el cual reciba una señal que indique el inicio del proceso de medición, y mostrará como salida la distancia medida en dos formatos, la distancia total en centímetros representada como un valor en binario, y por otro lado, el mismo valor representado a través de 3 dígitos en BCD, cuyo diagrama de bloques se muestra en la figura 7.6, y que pasará a ser parte de la biblioteca de módulos funcionales del alumno.



Figura 7.6. Diagrama de bloque del módulo para sensor ultrasónico

2.- Las demás actividades complementarias serán presentadas el día de la práctica.