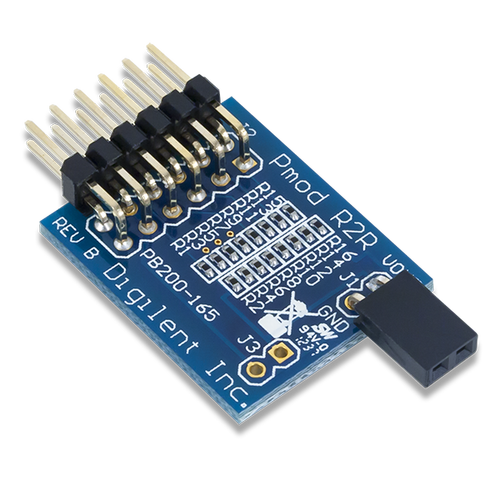
 UNIDAD PROFESIONAL INTERDISCIPLINARIA DE INGENERIA Y TECNOLOGIAS AVANZADAS

**“Generador de frecuencias de 0000 a 9999Hz  de 4 tipos de forma de onda”**

Alumno: ZARAZUA AGUILAR LUIS FERNANDO

GRUPO: 2mm9

PROFESOR: Rodríguez fuentes miguel Ángel

MATERIA: DISPOSITIVOS LÓGICOS PROGRAMABLES

**Planteamiento del Problema**

Esta práctica tiene como objetivo mostrar 4 señales distintas en el osciloscopio con frecuencias desde 0 Hz hasta 9999 Hz, para esto se usó un arreglo R2R de 16 bits que divide ponderadamente el voltaje en potencias de 2. En la resolución del problema se usaron dos memorias una que contenía las señales y otra que tenía los valores del contador para generar correctamente la frecuencia esto para reducir la labor matemática de calcular los valores. Para seleccionar la frecuencia adecuada se cuenta con un selector de 2 bits, un botón de incremento y otro de decremento, con el selector se puede seleccionar si se incrementa/decrementa de 1,10,100 o 1000.

**Código Principal**

**library** IEEE**;**

**use** IEEE**.**STD\_LOGIC\_1164**.ALL;**

**use** IEEE**.**NUMERIC\_STD**.ALL;**

**use** IEEE**.**STD\_LOGIC\_UNSIGNED**.ALL;**

-- Uncomment the following library declaration if using

-- arithmetic functions with Signed or Unsigned values

--use IEEE.NUMERIC\_STD.ALL;

-- Uncomment the following library declaration if instantiating

-- any Xilinx primitives in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

**entity** Principal **is**

**Port** **(** clk\_nexys **:** **in** STD\_LOGIC**;**

Sel\_Escala **:** **in** STD\_LOGIC\_VECTOR **(**1 **downto** 0**);**

Boton\_bajar **:** **in** STD\_LOGIC**;**

Boton\_subir **:** **in** STD\_LOGIC**;**

reset**:** **in** STD\_LOGIC**;**

Sel\_Senal **:** **in** STD\_LOGIC\_VECTOR **(**1 **downto** 0**);**

Vsal **:** **out** STD\_LOGIC\_VECTOR **(**15 **downto** 0**);**

Salidas\_7segc **:** **out** STD\_LOGIC\_VECTOR **(**7 **downto** 0**);**

Control\_Disp\_7segc**:** **out** STD\_LOGIC\_VECTOR **(**3 **downto** 0**));**

**end** Principal**;**

-- Modulos usados.

**architecture** Behavioral **of** Principal **is**

**component** debounce4 **is**

**Port** **(** clr **:** **in** STD\_LOGIC**;**

clk **:** **in** STD\_LOGIC**;**

inp **:** **in** STD\_LOGIC\_VECTOR**(**3 **downto** 0**);**

outp **:** **out** STD\_LOGIC\_VECTOR**(**3 **downto** 0**));**

**end** **component;**

**component** Selector\_de\_Frecuencia **is**

**Port** **(** clk\_in **:** **in** STD\_LOGIC**;**

Escala **:** **in** STD\_LOGIC\_VECTOR **(**1 **downto** 0**);**

Boton\_Subida **:** **in** STD\_LOGIC**;**

Boton\_Bajada **:** **in** STD\_LOGIC**;**

Frecuencia **:** **out** STD\_LOGIC\_VECTOR **(**13 **downto** 0**));**

**end** **component;**

**component** Generador\_Func **is**

**Port** **(** clk **:** **in** STD\_LOGIC**;**

rst **:** **in** STD\_LOGIC**;**

cuentamax **:** **in** STD\_LOGIC\_VECTOR **(**17 **downto** 0**);**

sel\_senal **:** **in** STD\_LOGIC\_VECTOR **(**1 **downto** 0**);**

Direccion\_Memoria\_Senal **:** **out** STD\_LOGIC\_VECTOR **(**9 **downto** 0**));**

**end** **component;**

**component** Mem\_Senales **IS**

**PORT** **(**

clka **:** **IN** STD\_LOGIC**;**

addra **:** **IN** STD\_LOGIC\_VECTOR**(**9 **DOWNTO** 0**);**

douta **:** **OUT** STD\_LOGIC\_VECTOR**(**15 **DOWNTO** 0**));**

**end** **component;**

**component** Mem\_Frecuencias **IS**

**PORT** **(**

clka **:** **IN** STD\_LOGIC**;**

addra **:** **IN** STD\_LOGIC\_VECTOR**(**13 **DOWNTO** 0**);**

douta **:** **OUT** STD\_LOGIC\_VECTOR**(**17 **DOWNTO** 0**));**

**end** **component;**

**component** Mem\_Digitos **IS**

**PORT** **(**

clka **:** **IN** STD\_LOGIC**;**

addra **:** **IN** STD\_LOGIC\_VECTOR**(**13 **DOWNTO** 0**);**

douta **:** **OUT** STD\_LOGIC\_VECTOR**(**15 **DOWNTO** 0**));**

**end** **component;**

**component** MUX\_Freq\_0 **is**

**Port** **(** Dato **:** **in** STD\_LOGIC\_VECTOR **(**15 **downto** 0**);**--Dato de la señal generada.

Freq **:** **in** STD\_LOGIC\_VECTOR **(**13 **downto** 0**);**--Frecuencia deseada.

Vout **:** **out** STD\_LOGIC\_VECTOR **(**15 **downto** 0**));**--Dato de salida.

**end** **component;**

**component** Decodififcador\_Digitos **is**

**Port** **(** clk **:** **in** STD\_LOGIC**;**

Dato **:** **in** STD\_LOGIC\_VECTOR **(**13 **downto** 0**);**

Digito\_4 **:** **out** STD\_LOGIC\_VECTOR **(**7 **downto** 0**);**

Digito\_3 **:** **out** STD\_LOGIC\_VECTOR **(**7 **downto** 0**);**

Digito\_2 **:** **out** STD\_LOGIC\_VECTOR **(**7 **downto** 0**);**

Digito\_1 **:** **out** STD\_LOGIC\_VECTOR **(**7 **downto** 0**));**

**end** **component;**

**component** Leds\_Display\_7 **is**

**Port** **(** clkin **:** **in** STD\_LOGIC**;**

Entrada\_Disp\_1 **:** **in** STD\_LOGIC\_VECTOR **(**7 **downto** 0**);**

Entrada\_Disp\_2 **:** **in** STD\_LOGIC\_VECTOR **(**7 **downto** 0**);**

Entrada\_Disp\_3 **:** **in** STD\_LOGIC\_VECTOR **(**7 **downto** 0**);**

Entrada\_Disp\_4 **:** **in** STD\_LOGIC\_VECTOR **(**7 **downto** 0**);**

Salidas\_7seg **:** **out** STD\_LOGIC\_VECTOR **(**7 **downto** 0**);**

Control\_Disp\_7seg **:** **out** STD\_LOGIC\_VECTOR **(**3 **downto** 0**));**

**end** **component;**

**signal** Salida\_ROM\_Frecuencias**:** STD\_LOGIC\_VECTOR**(**17 **DOWNTO** 0**);**

**signal** Direccion\_ROM\_Senales**:** STD\_LOGIC\_VECTOR**(**9 **DOWNTO** 0**);**

**signal** Salida\_ROM\_Senales**,**Digitos**:** STD\_LOGIC\_VECTOR**(**15 **DOWNTO** 0**);**

**signal** Frecuencia**:** STD\_LOGIC\_VECTOR**(**13 **DOWNTO** 0**);**

**signal** Btn\_Subir**:** STD\_LOGIC**;**

**signal** Btn\_Bajar**:** STD\_LOGIC**;**

**signal** Btn\_Reset**:** STD\_LOGIC**;**

**signal** Botones**:** STD\_LOGIC\_VECTOR**(**3 **DOWNTO** 0**);**

**signal** Salida\_BTN**:** STD\_LOGIC\_VECTOR**(**3 **DOWNTO** 0**);**

**signal** Display\_1c**,**Display\_2c**,**Display\_3c**,**Display\_4c**:** STD\_LOGIC\_VECTOR **(**7 **downto** 0**);**

**begin**

-- Display\_1c<=Frecuencia(7 downto 0);

-- Display\_2c<="00"&Frecuencia(13 downto 8);

-- Display\_3c<=Salida\_ROM\_Senales(15 downto 8);

-- Display\_4c<=Salida\_ROM\_Senales(7 downto 0);

Botones**<=(**Boton\_subir**&**Boton\_bajar**&**reset**&**reset**);**

U0**:** debounce4 **port** **map(** clr **=>** reset**,**

clk **=>** clk\_nexys**,**

inp **=>** Botones**,**

outp **=>** Salida\_BTN**);**

Btn\_Subir**<=**Salida\_BTN**(**3**);**

Btn\_Bajar**<=**Salida\_BTN**(**2**);**

Btn\_Reset**<=**Salida\_BTN**(**1**)** **and** Salida\_BTN**(**0**);**

U1**:** Selector\_de\_Frecuencia **port** **map(** clk\_in **=>** clk\_nexys**,**

Escala **=>** Sel\_Escala**,**

Boton\_Subida **=>** Btn\_Subir**,**

Boton\_Bajada **=>** Btn\_Bajar**,**

Frecuencia **=>** Frecuencia**);**

ROM1**:** Mem\_Frecuencias **port** **map(**clka **=>** clk\_nexys**,**

addra **=>** Frecuencia**,**

douta **=>** Salida\_ROM\_Frecuencias**);**

U2**:** Generador\_Func **port** **map(** clk **=>** clk\_nexys**,**

rst **=>** Btn\_Reset**,**

cuentamax **=>** Salida\_ROM\_Frecuencias**,**

sel\_senal **=>** Sel\_Senal**,**

Direccion\_Memoria\_Senal **=>** Direccion\_ROM\_Senales**);**

ROM2**:** Mem\_Senales **port** **map(**clka **=>** clk\_nexys**,**

addra **=>** Direccion\_ROM\_Senales**,**

douta **=>** Salida\_ROM\_Senales**);**

U3**:** MUX\_Freq\_0 **port** **map(** Dato **=>** Salida\_ROM\_Senales**,**

Freq **=>** Frecuencia**,**

Vout **=>** Vsal**);**

--ROM3: Mem\_Digitos port map(clka => clk\_nexys,

-- addra => Frecuencia,

-- douta => Digitos);

U4**:** Decodififcador\_Digitos **port** **map(** clk**=>**clk\_nexys**,**

Dato **=>** Frecuencia**,**

Digito\_4 **=>** Display\_4c**,**

Digito\_3 **=>** Display\_3c**,**

Digito\_2 **=>** Display\_2c**,**

Digito\_1 **=>** Display\_1c**);**

U5**:** Leds\_Display\_7 **port** **map(** clkin **=>** clk\_nexys**,**

Entrada\_Disp\_1 **=>** Display\_1c**,**

Entrada\_Disp\_2 **=>** Display\_2c**,**

Entrada\_Disp\_3 **=>** Display\_3c**,**

Entrada\_Disp\_4 **=>** Display\_4c**,**

Salidas\_7seg **=>** Salidas\_7segc**,**

Control\_Disp\_7seg **=>** Control\_Disp\_7segc**);**

**end** Behavioral**;**

Este código se encarga de unir los bloques funcionales del generador de funciones para que se pueda elegir la frecuencia y la señal deseada y posteriormente pueda ser mostrada en el osciloscopio, así como saber en qué frecuencia se está por medio de los displays. Este código contiene 9 componentes de los cuales 2 son memorias.

**Código Anti-rebotes**

**library** IEEE**;**

**use** IEEE**.**STD\_LOGIC\_1164**.ALL;**

**use** IEEE**.**STD\_LOGIC\_ARITH**.ALL;**

**use** IEEE**.**STD\_LOGIC\_UNSIGNED**.ALL;**

---- Uncomment the following library declaration if instantiating

---- any Xilinx primitives in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

**entity** debounce4 **is**

**Port** **(** clr **:** **in** STD\_LOGIC**;**

clk **:** **in** STD\_LOGIC**;**

inp **:** **in** STD\_LOGIC\_VECTOR**(**3 **downto** 0**);**

outp **:** **out** STD\_LOGIC\_VECTOR**(**3 **downto** 0**));**

**end** debounce4**;**

**architecture** Behavioral **of** debounce4 **is**

**signal** delay1**,**delay2**,**delay3**:** std\_logic\_vector**(**3 **downto** 0**);**

**signal** clkdiv**:** std\_logic\_vector**(**17 **downto** 0**);**

**begin**

**process(**clk**)**

**begin**

**if** **rising\_edge(**clk**)** **then**

clkdiv **<=** clkdiv **+**1**;**

**end** **if;**

**end** **process;**

**process(**clr**,**clkdiv**(**17**))**

**begin**

**if** clr **=** '1' **then**

delay1 **<=** "0000"**;**

delay2 **<=** "0000"**;**

delay3 **<=** "0000"**;**

**elsif** **rising\_edge(**clkdiv**(**17**))** **then**

delay1 **<=** inp**;**

delay2 **<=** delay1**;**

delay3 **<=** delay2**;**

**end** **if;**

**end** **process;**

outp **<=** delay1 and delay2 and delay3**;**

**end** Behavioral**;**

Este código se encarga de eliminar los posibles rebotes en la señal de mandar una señal limpia hacia la máquina de estados que ve si un botón ha sido presionado. Elimina los rebotes de 4 botones.

**Código Selector de Frecuencia**

**library** IEEE**;**

**use** IEEE**.**STD\_LOGIC\_1164**.ALL;**

**use** IEEE**.**numeric\_std**.all;**

**use** IEEE**.**std\_logic\_unsigned**.all;**

-- Uncomment the following library declaration if using

-- arithmetic functions with Signed or Unsigned values

--use IEEE.NUMERIC\_STD.ALL;

-- Uncomment the following library declaration if instantiating

-- any Xilinx primitives in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

**entity** Selector\_de\_Frecuencia **is**

**Port** **(** clk\_in **:** **in** STD\_LOGIC**;**

Escala **:** **in** STD\_LOGIC\_VECTOR **(**1 **downto** 0**);**

Boton\_Subida **:** **in** STD\_LOGIC**;**

Boton\_Bajada **:** **in** STD\_LOGIC**;**

Frecuencia **:** **inout** STD\_LOGIC\_VECTOR **(**13 **downto** 0**):=**"00000000000000"**);**

**end** Selector\_de\_Frecuencia**;**

**architecture** Behavioral **of** Selector\_de\_Frecuencia **is**

**signal** Contador**,**Contador2**:** SIGNED**(**14 **downto** 0**);**--14 Signo 13-0 Dato.

**signal** Cantidad**:** SIGNED**(**10 **downto** 0**);**--10 Signo 9-0 Dato.

**type** state\_type **is** **(**Inicio**,**Reposo**,**Verificacion\_1**,**Esperar\_1**,**Verificacion\_2**,**Esperar\_2**,**Cargar\_Frecuencia**);**

**signal** state**:** state\_type**;**

**signal** clkdiv**:** std\_logic\_vector**(**0 **downto** 0**);**

**begin**

--Divisor de Frecuencia

**process(**clk\_in**)**

**begin**

**if** **rising\_edge(**clk\_in**)** **then**

clkdiv **<=** clkdiv **+**1**;**

**end** **if;**

**end** **process;**

--Selector de Factor

**with** Escala **select**

Cantidad **<=** "00000000001" **when** "00"**,**--Declaración del factor X1.

"00000001010" **when** "01"**,**--Declaración del factor X10.

"00001100100" **when** "10"**,**--Declaración del factor X100.

"01111101000" **when** **others;**--Declaración del factor X1000.

--Màquina de Estados.

**process(**clkdiv**(**0**),**Boton\_Subida**,**Boton\_Bajada**,**Cantidad**)**

**begin**

**if** **rising\_edge(**clkdiv**(**0**))** **then**--Elige como reloj de la máquina de estados a clkdiv.

**case** state **is**

**when** Inicio **=>**--Estado de Inicio

Frecuencia**<=**"00001111101000"**;**--Frecuencia Inicial de 1000Hz.

Contador**<=**"000001111101000"**;**

state **<=**Reposo**;**

**when** Reposo **=>**--Estado de Inicio

**if** Boton\_Subida**=**'1' **then**

Contador2**<=**Contador**+**Cantidad**;**

state **<=**Verificacion\_1**;**

**elsif** Boton\_Bajada**=**'1' **then**

Contador2**<=**Contador**-**Cantidad**;**

state **<=**Verificacion\_2**;**

**else**

--Contador2<=Contador2;

state **<=**Reposo**;**

**end** **if;**

**when** Verificacion\_1 **=>**--Estado de Verificación

**if** Contador2**>**9999 **then**

Contador**<=**Contador**;**

**else**

Contador**<=**Contador2**;**

**end** **if;**

state**<=**Esperar\_1**;**

**when** Esperar\_1 **=>**--Estado de Espera\_1

**if** Boton\_Subida**=**'0' **then**

state**<=**Cargar\_Frecuencia**;**

**else**

state**<=**Esperar\_1**;**

**end** **if;**

**when** Verificacion\_2 **=>**--Estado de Inicio

**if** Contador2**<=**0 **then**

**if** Contador2**=**0 **then**

Contador**<=(others** **=>**'0'**);**

**else**

Contador**<=**Contador**;**

**end** **if;**

**else**

Contador**<=**Contador2**;**

**end** **if;**

state**<=**Esperar\_2**;**

**when** Esperar\_2 **=>**--Estado de Inicio

**if** Boton\_Bajada**=**'0' **then**

state**<=**Cargar\_Frecuencia**;**

**else**

state**<=**Esperar\_2**;**

**end** **if;**

**when** Cargar\_Frecuencia **=>**--Estado de Inicio

Frecuencia**<=**std\_logic\_vector**(**Contador**(**13 **downto** 0**));**

state **<=**Reposo**;**

**end** **case;**

**end** **if;**

**end** **process;**

**end** Behavioral**;**

Este código por medio de una máquina de estados recopila que botón fue presionado y en qué posición estaba el selector, luego verifica si el dato no sobrepasa los límites con la operación de decremento o incremento deseado y si no entonces modifica el registro de frecuencia para su posterior lectura en otros bloques. Esta salida de frecuencia es decodificada por una memoria que asigna el valor correcto para el contador en el generador de señal.

**Código para la Generación de la Señal**

**library** IEEE**;**

**use** IEEE**.**STD\_LOGIC\_1164**.ALL;**

**use** IEEE**.**NUMERIC\_STD**.ALL;**

**use** IEEE**.**STD\_LOGIC\_UNSIGNED**.ALL;**

-- Uncomment the following library declaration if using

-- arithmetic functions with Signed or Unsigned values

--use IEEE.NUMERIC\_STD.ALL;

-- Uncomment the following library declaration if instantiating

-- any Xilinx primitives in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

**entity** Generador\_Func **is**

**Port** **(** clk **:** **in** STD\_LOGIC**;**

rst **:** **in** STD\_LOGIC**;**

cuentamax **:** **in** STD\_LOGIC\_VECTOR **(**17 **downto** 0**);**

sel\_senal **:** **in** STD\_LOGIC\_VECTOR **(**1 **downto** 0**);**

Direccion\_Memoria\_Senal **:** **out** STD\_LOGIC\_VECTOR **(**9 **downto** 0**));**

**end** Generador\_Func**;**

**architecture** Behavioral **of** Generador\_Func **is**

**signal** contador**:** std\_logic\_vector**(**7 **downto** 0**);**--Contador de Muestras.

**signal** masterc**:** std\_logic\_vector**(**17 **downto** 0**);**--0-196078 para 1Hz Contador de Intervalo.

**begin**

**process(**clk**,**rst**)**

**begin**

**if** rst**=**'1' **then**

contador **<=**"00000000"**;**

**elsif** **rising\_edge(**clk**)** **then**

masterc**<=**masterc**+**1**;**

**if** masterc**>**cuentamax **then**--Indica hasta que número llegar antes de incrementar el contador.

masterc**<=(others** **=>**'0'**);**--Resetea el contador que controla la frecuencia.

contador**<=**contador**+**1**;**--Incrementa el contador para acceder a la señal.

**end** **if;**

**end** **if;**

**end** **process;**

Direccion\_Memoria\_Senal**<=**sel\_senal **&** contador**;**--Concatenar selector de señal y conteo.

**end** Behavioral**;**

Este código se encarga se generar un contador ascendente para poder acceder a la memoria correctamente, el tiempo del contador está determinado por la frecuencia ya decodificada en la memoria que nos da hasta que número llegar, este valor de contador es concatenado con el selector de señal para así obtener correctamente a que dirección de la memoria (que contiene los datos de las señales) se va a acceder. Una vez obtenido ese valor de dirección de memoria se accede a la memoria de señales y así se obtiene el dato correcto a mandar.

**Código Multiplexor para Frecuencia 0**

**library** IEEE**;**

**use** IEEE**.**STD\_LOGIC\_1164**.ALL;**

**use** IEEE**.**NUMERIC\_STD**.ALL;**

**use** IEEE**.**STD\_LOGIC\_UNSIGNED**.ALL;**

-- Uncomment the following library declaration if using

-- arithmetic functions with Signed or Unsigned values

--use IEEE.NUMERIC\_STD.ALL;

**entity** MUX\_Freq\_0 **is**

**Port** **(** Dato **:** **in** STD\_LOGIC\_VECTOR **(**15 **downto** 0**);**--Dato de la señal generada.

Freq **:** **in** STD\_LOGIC\_VECTOR **(**13 **downto** 0**);**--Frecuencia deseada.

Vout **:** **out** STD\_LOGIC\_VECTOR **(**15 **downto** 0**));**--Dato de salida.

**end** MUX\_Freq\_0**;**

**architecture** Behavioral **of** MUX\_Freq\_0 **is**

**begin**

**process(**Freq**,**Dato**)**

**begin**

**if** Freq**=**0 **then**

Vout**<=(others** **=>**'0'**);**--Si es 0 la frecuencia conecta a tierra la señal

**else**

Vout**<=**Dato**;**--Si la frecuencia es distinta de 0, pasa el dato dado por la memoria de señal.

**end** **if;**

**end** **process;**

**end** Behavioral**;**

Este código sirve para separar el caso en que la frecuencia es 0 y para evitar que se quede en un voltaje fijo aleatorio, para esto se conecta a tierra por medio de este multiplexor con el valor de 0, en caso de cualquier otra frecuencia deja pasar el dato generado por la memoria.

**Código para la decodificación de los dígitos**

**library** IEEE**;**

**use** IEEE**.**STD\_LOGIC\_1164**.ALL;**

**use** IEEE**.**NUMERIC\_STD**.ALL;**

**use** IEEE**.**STD\_LOGIC\_UNSIGNED**.ALL;**

-- Uncomment the following library declaration if using

-- arithmetic functions with Signed or Unsigned values

--use IEEE.NUMERIC\_STD.ALL;

**entity** Decodififcador\_Digitos **is**

**Port** **(** clk **:** **in** STD\_LOGIC**;**

Dato **:** **in** STD\_LOGIC\_VECTOR **(**13 **downto** 0**);**

Digito\_4 **:** **out** STD\_LOGIC\_VECTOR **(**7 **downto** 0**);**

Digito\_3 **:** **out** STD\_LOGIC\_VECTOR **(**7 **downto** 0**);**

Digito\_2 **:** **out** STD\_LOGIC\_VECTOR **(**7 **downto** 0**);**

Digito\_1 **:** **out** STD\_LOGIC\_VECTOR **(**7 **downto** 0**));**

**end** Decodififcador\_Digitos**;**

**architecture** Behavioral **of** Decodififcador\_Digitos **is**

**type** state\_type **is** **(**Inicio**,**Recopilar\_Datos**,**Decrementar\_Millares**,**Decrementar\_Centenas**,**Decrementar\_Decenas**,**Cargar\_Digitos**);**

**signal** state**:** state\_type**;**

**signal** Dato\_op**:** STD\_LOGIC\_VECTOR **(**13 **downto** 0**);**

**signal** clkdiv**:** STD\_LOGIC\_VECTOR **(**0 **downto** 0**);**

**signal** Dig4**,**Dig3**,**Dig2**,**Dig1**:** STD\_LOGIC\_VECTOR **(**3 **downto** 0**);**

**signal** Dig4b**,**Dig3b**,**Dig2b**,**Dig1b**:** STD\_LOGIC\_VECTOR **(**3 **downto** 0**);**

**begin**

**process(**clk**)**

**begin**

**if** **rising\_edge(**clk**)** **then**

clkdiv **<=** clkdiv **+**1**;**

**end** **if;**

**end** **process;**

--Màquina de Estados.

**process(**clkdiv**(**0**),**Dato**)**

**begin**

**if** **rising\_edge(**clkdiv**(**0**))** **then**--Elige como reloj de la máquina de estados a clkdiv.

**case** state **is**

**when** Inicio **=>**--Estado de Inicio

state **<=**Recopilar\_Datos**;**

**when** Recopilar\_Datos **=>**--Estado de Inicio

Dig1**<=**"0000"**;**

Dig2**<=**"0000"**;**

Dig3**<=**"0000"**;**

Dig4**<=**"0000"**;**

Dato\_op**<=**Dato**;**

state **<=**Decrementar\_Millares**;**

**when** Decrementar\_Millares **=>**--Estado de Decrementar Millares

**if** Dato\_op**>=**1000 **then**

Dato\_op**<=**Dato\_op**-**1000**;**

Dig4**<=**Dig4**+**1**;**

state**<=**Decrementar\_Millares**;**

**else**

Dig4**<=**Dig4**;**

state**<=**Decrementar\_Centenas**;**

**end** **if;**

**when** Decrementar\_Centenas **=>**--Estado de Decrementar\_Centenas

**if** Dato\_op**>=**100 **then**

Dato\_op**<=**Dato\_op**-**100**;**

Dig3**<=**Dig3**+**1**;**

state**<=**Decrementar\_Centenas**;**

**else**

Dig3**<=**Dig3**;**

state**<=**Decrementar\_Decenas**;**

**end** **if;**

**when** Decrementar\_Decenas **=>**--Estado de Decrementar\_Decenas

**if** Dato\_op**>=**10 **then**

Dato\_op**<=**Dato\_op**-**10**;**

Dig2**<=**Dig2**+**1**;**

state**<=**Decrementar\_Centenas**;**

**else**

Dig1**<=**Dato\_op**(**3 **downto** 0**);**

Dig2**<=**Dig2**;**

state**<=**Cargar\_Digitos**;**

**end** **if;**

**when** Cargar\_Digitos **=>**--Estado de Cargar\_Digitos

Dig1b**<=**Dig1**;**

Dig2b**<=**Dig2**;**

Dig3b**<=**Dig3**;**

Dig4b**<=**Dig4**;**

state **<=**Recopilar\_Datos**;**

**end** **case;**

**end** **if;**

**end** **process;**

**with** Dig4b **select**

Digito\_4 **<=** not**(**X"C0"**)** **when** X"0"**,**--Declaración del dígito "0" para display de 7 segmentos ánodo.

not**(**X"79"**)** **when** X"1"**,**--Declaración del dígito "1" para display de 7 segmentos ánodo.

not**(**X"A4"**)** **when** X"2"**,**--Declaración del dígito "2" para display de 7 segmentos ánodo.

not**(**X"30"**)** **when** X"3"**,**--Declaración del dígito "3" para display de 7 segmentos ánodo.

not**(**X"99"**)** **when** X"4"**,**--Declaración del dígito "4" para display de 7 segmentos ánodo.

not**(**X"12"**)** **when** X"5"**,**--Declaración del dígito "5" para display de 7 segmentos ánodo.

not**(**X"83"**)** **when** X"6"**,**--Declaración del dígito "6" para display de 7 segmentos ánodo.

not**(**X"78"**)** **when** X"7"**,**--Declaración del dígito "7" para display de 7 segmentos ánodo.

not**(**X"80"**)** **when** X"8"**,**--Declaración del dígito "8" para display de 7 segmentos ánodo.

not**(**X"18"**)** **when** X"9"**,**--Declaración del dígito "9" para display de 7 segmentos ánodo.

not**(**X"A7"**)** **when** X"A"**,**--Declaración del dígito "A" para display de 7 segmentos ánodo.

not**(**X"33"**)** **when** X"B"**,**--Declaración del dígito "B" para display de 7 segmentos ánodo.

not**(**X"9D"**)** **when** X"C"**,**--Declaración del dígito "C" para display de 7 segmentos ánodo.

not**(**X"16"**)** **when** X"D"**,**--Declaración del dígito "D" para display de 7 segmentos ánodo.

not**(**X"87"**)** **when** X"E"**,**--Declaración del dígito "E" para display de 7 segmentos ánodo.

not**(**X"7F"**)** **when** **others;**--Declaración del dígito "F" para display de 7 segmentos ánodo.

**with** Dig3b **select**

Digito\_3 **<=** not**(**X"C0"**)** **when** X"0"**,**--Declaración del dígito "0" para display de 7 segmentos ánodo.

not**(**X"79"**)** **when** X"1"**,**--Declaración del dígito "1" para display de 7 segmentos ánodo.

not**(**X"A4"**)** **when** X"2"**,**--Declaración del dígito "2" para display de 7 segmentos ánodo.

not**(**X"30"**)** **when** X"3"**,**--Declaración del dígito "3" para display de 7 segmentos ánodo.

not**(**X"99"**)** **when** X"4"**,**--Declaración del dígito "4" para display de 7 segmentos ánodo.

not**(**X"12"**)** **when** X"5"**,**--Declaración del dígito "5" para display de 7 segmentos ánodo.

not**(**X"83"**)** **when** X"6"**,**--Declaración del dígito "6" para display de 7 segmentos ánodo.

not**(**X"78"**)** **when** X"7"**,**--Declaración del dígito "7" para display de 7 segmentos ánodo.

not**(**X"80"**)** **when** X"8"**,**--Declaración del dígito "8" para display de 7 segmentos ánodo.

not**(**X"18"**)** **when** X"9"**,**--Declaración del dígito "9" para display de 7 segmentos ánodo.

not**(**X"A7"**)** **when** X"A"**,**--Declaración del dígito "A" para display de 7 segmentos ánodo.

not**(**X"33"**)** **when** X"B"**,**--Declaración del dígito "B" para display de 7 segmentos ánodo.

not**(**X"9D"**)** **when** X"C"**,**--Declaración del dígito "C" para display de 7 segmentos ánodo.

not**(**X"16"**)** **when** X"D"**,**--Declaración del dígito "D" para display de 7 segmentos ánodo.

not**(**X"87"**)** **when** X"E"**,**--Declaración del dígito "E" para display de 7 segmentos ánodo.

not**(**X"7F"**)** **when** **others;**--Declaración del dígito "F" para display de 7 segmentos ánodo.

**with** Dig2b **select**

Digito\_2 **<=** not**(**X"C0"**)** **when** X"0"**,**--Declaración del dígito "0" para display de 7 segmentos ánodo.

not**(**X"79"**)** **when** X"1"**,**--Declaración del dígito "1" para display de 7 segmentos ánodo.

not**(**X"A4"**)** **when** X"2"**,**--Declaración del dígito "2" para display de 7 segmentos ánodo.

not**(**X"30"**)** **when** X"3"**,**--Declaración del dígito "3" para display de 7 segmentos ánodo.

not**(**X"99"**)** **when** X"4"**,**--Declaración del dígito "4" para display de 7 segmentos ánodo.

not**(**X"12"**)** **when** X"5"**,**--Declaración del dígito "5" para display de 7 segmentos ánodo.

not**(**X"83"**)** **when** X"6"**,**--Declaración del dígito "6" para display de 7 segmentos ánodo.

not**(**X"78"**)** **when** X"7"**,**--Declaración del dígito "7" para display de 7 segmentos ánodo.

not**(**X"80"**)** **when** X"8"**,**--Declaración del dígito "8" para display de 7 segmentos ánodo.

not**(**X"18"**)** **when** X"9"**,**--Declaración del dígito "9" para display de 7 segmentos ánodo.

not**(**X"A7"**)** **when** X"A"**,**--Declaración del dígito "A" para display de 7 segmentos ánodo.

not**(**X"33"**)** **when** X"B"**,**--Declaración del dígito "B" para display de 7 segmentos ánodo.

not**(**X"9D"**)** **when** X"C"**,**--Declaración del dígito "C" para display de 7 segmentos ánodo.

not**(**X"16"**)** **when** X"D"**,**--Declaración del dígito "D" para display de 7 segmentos ánodo.

not**(**X"87"**)** **when** X"E"**,**--Declaración del dígito "E" para display de 7 segmentos ánodo.

not**(**X"7F"**)** **when** **others;**--Declaración del dígito "F" para display de 7 segmentos ánodo.

**with** Dig1b **select**

Digito\_1 **<=** not**(**X"C0"**)** **when** X"0"**,**--Declaración del dígito "0" para display de 7 segmentos ánodo.

not**(**X"79"**)** **when** X"1"**,**--Declaración del dígito "1" para display de 7 segmentos ánodo.

not**(**X"A4"**)** **when** X"2"**,**--Declaración del dígito "2" para display de 7 segmentos ánodo.

not**(**X"30"**)** **when** X"3"**,**--Declaración del dígito "3" para display de 7 segmentos ánodo.

not**(**X"99"**)** **when** X"4"**,**--Declaración del dígito "4" para display de 7 segmentos ánodo.

not**(**X"12"**)** **when** X"5"**,**--Declaración del dígito "5" para display de 7 segmentos ánodo.

not**(**X"83"**)** **when** X"6"**,**--Declaración del dígito "6" para display de 7 segmentos ánodo.

not**(**X"78"**)** **when** X"7"**,**--Declaración del dígito "7" para display de 7 segmentos ánodo.

not**(**X"80"**)** **when** X"8"**,**--Declaración del dígito "8" para display de 7 segmentos ánodo.

not**(**X"18"**)** **when** X"9"**,**--Declaración del dígito "9" para display de 7 segmentos ánodo.

not**(**X"A7"**)** **when** X"A"**,**--Declaración del dígito "A" para display de 7 segmentos ánodo.

not**(**X"33"**)** **when** X"B"**,**--Declaración del dígito "B" para display de 7 segmentos ánodo.

not**(**X"9D"**)** **when** X"C"**,**--Declaración del dígito "C" para display de 7 segmentos ánodo.

not**(**X"16"**)** **when** X"D"**,**--Declaración del dígito "D" para display de 7 segmentos ánodo.

not**(**X"87"**)** **when** X"E"**,**--Declaración del dígito "E" para display de 7 segmentos ánodo.

not**(**X"7F"**)** **when** **others;**--Declaración del dígito "F" para display de 7 segmentos ánodo.

**end** Behavioral**;**

Este código sirve para codificar la frecuencia en dígitos que puedan ser mostrados en los displays de 7 segmentos, para esto se realizó máquina de estados que separa en millares, centenas, decenas y unidades, posteriormente cada digito individual es decodificado en su valor para 7 segmentos y enviado al multiplexor que prende los displays.

**Código del Display**

**library** IEEE**;**

**use** ieee**.**std\_logic\_1164**.all;**

**use** ieee**.**std\_logic\_arith**.all;**

**use** ieee**.**std\_logic\_unsigned**.all;**

-- Uncomment the following library declaration if using

-- arithmetic functions with Signed or Unsigned values

--use IEEE.NUMERIC\_STD.ALL;

-- Uncomment the following library declaration if instantiating

-- any Xilinx primitives in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

**entity** Leds\_Display\_7 **is**

**Port** **(** clkin **:** **in** STD\_LOGIC**;**

Entrada\_Disp\_1 **:** **in** STD\_LOGIC\_VECTOR **(**7 **downto** 0**);**

Entrada\_Disp\_2 **:** **in** STD\_LOGIC\_VECTOR **(**7 **downto** 0**);**

Entrada\_Disp\_3 **:** **in** STD\_LOGIC\_VECTOR **(**7 **downto** 0**);**

Entrada\_Disp\_4 **:** **in** STD\_LOGIC\_VECTOR **(**7 **downto** 0**);**

Salidas\_7seg **:** **out** STD\_LOGIC\_VECTOR **(**7 **downto** 0**);**

Control\_Disp\_7seg **:** **out** STD\_LOGIC\_VECTOR **(**3 **downto** 0**));**

**end** Leds\_Display\_7**;**

**architecture** Behavioral **of** Leds\_Display\_7 **is**

**signal** clkdiv**:** STD\_LOGIC\_VECTOR **(**16 **downto** 0**);**

**signal** contador\_disp**:** STD\_LOGIC\_VECTOR **(**1 **downto** 0**):=**"00"**;**

**begin**

**process(**clkin**)**

**begin**

**if** **rising\_edge(**clkin**)** **then**

clkdiv **<=** clkdiv **+**1**;**

**end** **if;**

**end** **process;**

**process(**clkdiv**(**16**),**contador\_disp**)**

**begin**

**if** **rising\_edge(**clkdiv**(**16**))** **then**

contador\_disp**<=**contador\_disp**+**1**;**

**if** contador\_disp**=**0 **then**

Control\_Disp\_7seg**<=**"0111"**;**

Salidas\_7seg**<=**not**(**Entrada\_Disp\_1**);**

**elsif** contador\_disp**=**1 **then**

Control\_Disp\_7seg**<=**"1011"**;**

Salidas\_7seg**<=**not**(**Entrada\_Disp\_2**);**

**elsif** contador\_disp**=**2 **then**

Control\_Disp\_7seg**<=**"1101"**;**

Salidas\_7seg**<=**not**(**Entrada\_Disp\_3**);**

**else**

Control\_Disp\_7seg**<=**"1110"**;**

Salidas\_7seg**<=**not**(**Entrada\_Disp\_4**);**

**end** **if;**

**end** **if;**

**end** **process;**

**end** Behavioral**;**

Este código se implementa para mostrar los datos decodificados de los dígitos en el display de 7 segmentos.

**Código en Matlab para generar los valores del contador**

clc**,** clear**,** close all

nm**=**2**^**8**;**%%numero de muestras.

nm**=**nm**-**1**;**

freq\_1s**=**1**/**nm**;**

valor\_cont\_1s**=**50e6**\***freq\_1s**;**

**for** i**=**1**:**1**:**9999

Valores\_Freq**(**i**)=**round**(**valor\_cont\_1s**/**i**);**

**end**

Valores\_Freq**=[**Valores\_Freq**(**1**),**Valores\_Freq**];**

t**=**0**:**1**:**9999**;**

stem**(**t**,**Valores\_Freq**);**

Palabra**=**dec2hex**(**Valores\_Freq**,**4**);**

outfile**=**'Val\_Cont.coe'**;**

s **=** fopen**(**outfile**,**'w+'**);** %opens the output file

fprintf**(**s**,**'%s\n'**,**'; VGA Memory Map '**);**

fprintf**(**s**,**'%s\n'**,**'; .COE file with hex coefficients '**);**

fprintf**(**s**,**'%s\n'**,**'memory\_initialization\_radix=16;'**);**

fprintf**(**s**,**'%s\n'**,**'memory\_initialization\_vector='**);**

**for** i**=**1**:**length**(**Valores\_Freq**)-**1

fprintf**(**s**,**'%c'**,**Palabra**(**i**,:));**

fprintf**(**s**,**'%c'**,**','**);**

**end**

fprintf**(**s**,**'%c'**,**Palabra**(**i**+**1**,:));**

fprintf**(**s**,**'%c'**,**';'**);**

Este código se encarga de generar el archivo .coe con los valores del contador para cada frecuencia, para esto se considera de cuantas muestras se quiere la señal y en base a esto y la frecuencia del reloj de la Nexys se obtiene un valor para cada frecuencia, como el contador es entero hay pérdida en los datos generando para algunas frecuencias valores iguales en el contador.

**Código en Matlab para generar los valores de la señal**

clc**,** clear all**,** close all

%8 bits

bits**=**16**;**

Voltaje**=**3.3**;**

n**=**2**^**8**;**%Número de muestras

n**=**n**-**1**;**

t**=**0**:**360**/**n**:**360**;**

Vec\_l**=**0**:**1**:**n**;**

%%%Señal Rampa

y1**=(**t**/**360**\***Voltaje**);**

Valor1**(**length**(**t**))=**0**;**

**for** i**=**1**:**length**(**t**)**

x**=**y1**(**i**);**

**for** j**=**1**:**1**:**bits

**if(**x**>=**Voltaje**/(**2**^**j**))**

x**=**x**-**Voltaje**/(**2**^**j**);**

Valor1**(**i**)=**Valor1**(**i**)+**2**^(**bits**-**j**);**

**end**

**end**

**end**

%%%Señal Seno

y2**=(**Voltaje**+**sind**(**t**)\***Voltaje**)/**2**;**%Señal Seno

Valor2**(**length**(**t**))=**0**;**

**for** i**=**1**:**length**(**t**)**

x**=**y2**(**i**);**

**for** j**=**1**:**1**:**bits

**if(**x**>=**Voltaje**/(**2**^**j**))**

x**=**x**-**Voltaje**/(**2**^**j**);**

Valor2**(**i**)=**Valor2**(**i**)+**2**^(**bits**-**j**);**

**end**

**end**

**end**

%%Señal exponencial

**for** i**=**1**:**length**(**t**)** %Exponencial.

y3**(**i**)=**Voltaje**\***exp**((**t**(**i**))/**36**)/**22e3**;**

**end**

Valor3**(**length**(**t**))=**0**;**

**for** i**=**1**:**length**(**t**)**

x**=**y3**(**i**);**

**for** j**=**1**:**1**:**bits

**if(**x**>=**Voltaje**/(**2**^**j**))**

x**=**x**-**Voltaje**/(**2**^**j**);**

Valor3**(**i**)=**Valor3**(**i**)+**2**^(**bits**-**j**);**

**end**

**end**

**end**

%%Señal libre

**for** i**=**1**:**length**(**t**)** %Circulo

**if** i**<=**length**(**t**)/**2**;**

y4**(**i**)=**Voltaje**\***0.1**+**Voltaje**\***0.9**\*(**1**+**sqrt**(**90**^**2**-(**t**(**i**)-**90**)^**2**)/**90**)/**2**-(**Voltaje**+**sind**(**15**\***t**(**i**))\***Voltaje**)/**20**;**

**else**

y4**(**i**)=**Voltaje**\***0.1**+**Voltaje**\***0.9**\*(**1**-**sqrt**(**90**^**2**-(**t**(**i**)-**270**)^**2**)/**90**)/**2**-(**Voltaje**+**sind**(**30**\***t**(**i**))\***Voltaje**)/**20**;**

**end**

**end**

% for i=1:length(t) %Triangular-Seno

% if i<length(t)/2;

% y4(i)=Voltaje\*(t(i)/180);

% else

% y4(i)=Voltaje\*(1+sind(t(i)-90))/2;

% end

% end

Valor4**(**length**(**t**))=**0**;**

**for** i**=**1**:**length**(**t**)**

x**=**y4**(**i**);**

**for** j**=**1**:**1**:**bits

**if(**x**>=**Voltaje**/(**2**^**j**))**

x**=**x**-**Voltaje**/(**2**^**j**);**

Valor4**(**i**)=**Valor4**(**i**)+**2**^(**bits**-**j**);**

**end**

**end**

**end**

figure**(**1**)**

axis equal**,** subplot**(**2**,**2**,**1**),**plot**(**t**,**y1**)**

subplot**(**2**,**2**,**2**),**stem**(**t**,**Valor1**)**

subplot**(**2**,**2**,**3**),**plot**(**Vec\_l**,**y1**)**

subplot**(**2**,**2**,**4**),**stem**(**Vec\_l**,**Valor1**)**

figure**(**2**)**

axis equal**,** subplot**(**2**,**2**,**1**),**plot**(**t**,**y2**)**

subplot**(**2**,**2**,**2**),**stem**(**t**,**Valor2**)**

subplot**(**2**,**2**,**3**),**plot**(**Vec\_l**,**y2**)**

subplot**(**2**,**2**,**4**),**stem**(**Vec\_l**,**Valor2**)**

figure**(**3**)**

axis equal**,** subplot**(**2**,**2**,**1**),**plot**(**t**,**y3**)**

subplot**(**2**,**2**,**2**),**stem**(**t**,**Valor3**)**

subplot**(**2**,**2**,**3**),**plot**(**Vec\_l**,**y3**)**

subplot**(**2**,**2**,**4**),**stem**(**Vec\_l**,**Valor3**)**

figure**(**4**)**

axis equal**,** subplot**(**2**,**2**,**1**),**plot**(**t**,**y4**)**

subplot**(**2**,**2**,**2**),**stem**(**t**,**Valor4**)**

subplot**(**2**,**2**,**3**),**plot**(**Vec\_l**,**y4**)**

**for** i**=**1**:**1**:**length**(**t**)**

Valor4b**(**2**\***i**-**1**)=**Valor4**(**i**);**

Angulo**(**2**\***i**-**1**)=**t**(**i**);**

**if** i**<**length**(**t**)**

Angulo**(**2**\***i**)=**t**(**i**+**1**);**

Valor4b**(**2**\***i**)=**Valor4**(**i**);**

**end**

**end**

subplot**(**2**,**2**,**4**),**plot**(**Angulo**,**Valor4b**)**

%%Generacion del punto coe.

Senales**=[**Valor1**,**Valor2**,**Valor3**,**Valor4**];**

Palabra**=**dec2hex**(**Senales**);**

outfile**=**'Senales.coe'**;**

s **=** fopen**(**outfile**,**'w+'**);** %opens the output file

fprintf**(**s**,**'%s\n'**,**'; VGA Memory Map '**);**

fprintf**(**s**,**'%s\n'**,**'; .COE file with hex coefficients '**);**

fprintf**(**s**,**'%s\n'**,**'memory\_initialization\_radix=16;'**);**

fprintf**(**s**,**'%s\n'**,**'memory\_initialization\_vector='**);**

**for** i**=**1**:**length**(**Senales**)-**1

fprintf**(**s**,**'%c'**,**Palabra**(**i**,:));**

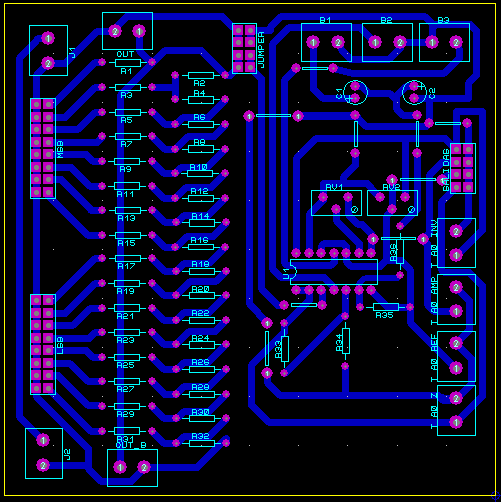
fprintf**(**s**,**'%c'**,**','**);**

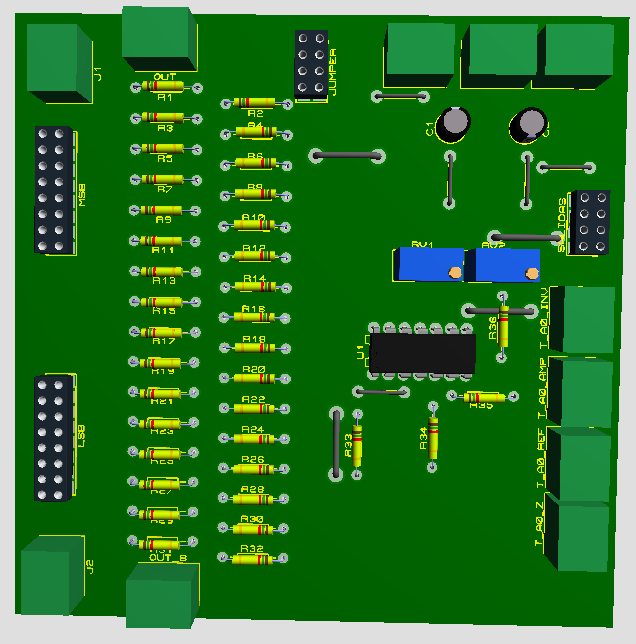
**end**

fprintf**(**s**,**'%c'**,**Palabra**(**i**+**1**,:));**

fprintf**(**s**,**'%c'**,**';'**);**

En este código se genera el archivo para las 4 señales distintas a las cuales se les asigna una función en el tiempo para que se puedan representar más fácilmente, posteriormente se encuentra el valor correcto por medio de un comparador y un ciclo while que decide qué valor digital tiene la señal y este se convierte a hexadecimal, posteriormente se concatenan las 4 señales en una arreglo y se escribe el archivo .coe de la memoria de señal.

**Arreglo R2R**



**Señales Mostradas**





