**Universidad Nacional Autónoma de México**

**Facultad de Ingeniería**

**Diseño de Sistemas Digitales**

Aguilar Enriquez Paul Sebastian – 415028130

Cabrera López Oscar Emilio - 312333261

2016/10/13

**Practica 2: “Señales”**

**Descripción del problema**

Diseñar y montar un circuito que muestre mediante un osciloscopio las siguientes señales.

**Modelado de las variables internas**

Las señales se graficaron utilizando Excel y exportando los valores directamente a hexadecimal.

Los valores se encuentran en los módulos de memorias ROM implementadas en el código, en caso de querer revisar los valores, dirigirse al código en el repositorio de GitHub (el link esta al final del reporte), esto con objetivo de no imprimir todo el código que nos daba como resultado un uso de 25 hojas (por los valores hexadecimales).

Las señales que se realizaron son:

* Seno
* Triangulo
* Sierra
* Cuadrada
* Vicflo Señal

**Método de solución**

Para el desarrollo de esta práctica se decidió utilizar el FPGA Cyclone II.

El problema se abordó dividiéndolo en “módulos”.

Los módulos que se diseñaron e implementaron fueron:

* Modulo principal selector de señales
* Contador de 11 bits para ir avanzando sobre la memoria donde se almacena la señal
* Memoria ROM con la señal seno
* Memoria ROM con la señal triangulo
* Memoria ROM con la señal sierra
* Memoria ROM con la señal cuadrada
* Memoria ROM con la Vicflo señal

En cuanto a implementación física se realizó lo siguiente:

* Temporizador con el integrado 555 que nos permite cambiar la frecuencia de las señales mediante un potenciómetro.
* Un amplificador operacional para modificar la amplitud de las señales mediante un potenciómetro.
* Un convertidor Digital-Analógico (DAC) R2R de 12 bits, con R = 1000K y 2R = 2K utilizando el principio de superposición y el de Thevenin.
* Un dipswitch para seleccionar las señales.

**Código**

-- Copyright (c) 2016

-- Emilio Cabrera <emilio1625@gmail.com>

-- Sebastian Aguilar <penserbjorne>

--

-- GNU GENERAL PUBLIC LICENSE

-- Version 3, 29 June 2007

--

-- This program is free software: you can redistribute it and/or modify

-- it under the terms of the GNU General Public License as published by

-- the Free Software Foundation, either version 3 of the License, or

-- (at your option) any later version.

--

-- This program is distributed in the hope that it will be useful,

-- but WITHOUT ANY WARRANTY; without even the implied warranty of

-- MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

-- GNU General Public License for more details.

--

-- You should have received a copy of the GNU General Public License

-- along with this program. If not, see <http://www.gnu.org/licenses/>.

*Archivo Principal “practica2\_seniales.vhd”*

library ieee;

use ieee.std\_logic\_1164.all;

entity practica2\_seniales is

port (

clk: in std\_logic;

sel: in std\_logic\_vector(2 downto 0);

data: out std\_logic\_vector(11 downto 0)

);

end entity;

architecture arch of practica2\_seniales is

signal baddr: std\_logic\_vector(10 downto 0);

signal data1, data2, data3, data4, data5: std\_logic\_vector(11 downto 0);

begin

U1: entity work.count port map(

clk => clk,

conta => baddr

);

U2: entity work.rom\_seno port map(

clk => clk,

addr => baddr,

data => data1

);

U3: entity work.rom\_triangulo port map(

clk => clk,

addr => baddr,

data => data2

);

U4: entity work.rom\_sierra port map(

clk => clk,

addr => baddr,

data => data3

);

U5: entity work.rom\_cuadrada port map(

clk => clk,

addr => baddr,

data => data4

);

U6: entity work.rom\_vicflo port map(

clk => clk,

addr => baddr,

data => data5

);

data <= data2 when sel = "001" else

data3 when sel = "010" else

data4 when sel = "011" else

data5 when sel = "100" else

data1;

end architecture;

*Archivo Principal “count.vhd”*

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_arith.all;

use ieee.std\_logic\_unsigned.all;

entity count is

port (

clk: in std\_logic;

conta: buffer std\_logic\_vector(10 downto 0)

);

end entity;

architecture arch of count is

begin

process (clk) begin

if (clk'event and clk= '1') then

conta <= conta + 1;

end if;

end process;

end architecture;

*Archivo Principal “rom\_seno.vhd”*

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

entity rom\_seno is

port (

clk : in std\_logic;

en : in std\_logic := '1';

addr : in std\_logic\_vector(10 downto 0);

data : out std\_logic\_vector(11 downto 0)

);

end rom\_seno;

architecture arch of rom\_seno is

type memoria\_rom is array (0 to 2047) of std\_logic\_vector (11 downto 0);

constant ROM : memoria\_rom := (

x"800", x"806", x"80c", x"812", x"819", x"81f", x"825", x"82b",

x"832", x"838", x"83e", x"845", x"84b", x"851", x"857", x"85e",

x"864", x"86a", x"871", x"877", x"87d", x"883", x"88a", x"890",

...

x"76f", x"775", x"77c", x"782", x"788", x"78e", x"795", x"79b",

x"7a1", x"7a8", x"7ae", x"7b4", x"7ba", x"7c1", x"7c7", x"7cd",

x"7d4", x"7da", x"7e0", x"7e6", x"7ed", x"7f3", x"7f9", x"800"

);

signal tmp: std\_logic\_vector(11 downto 0);

begin

ret: process (addr,clk) begin

if (clk' event and clk = '1') then

tmp <= ROM(conv\_integer(addr));

end if;

end process;

buff: process (tmp,en) begin

if(en = '1') then

data <= tmp;

else

data <= (others => 'Z');

end if;

end process buff;

end arch;

*Archivo Principal “rom\_triangulo.vhd”*

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

entity rom\_triangulo is

port (

clk : in std\_logic;

en : in std\_logic := '1';

addr : in std\_logic\_vector(10 downto 0);

data : out std\_logic\_vector(11 downto 0)

);

end rom\_triangulo;

architecture arch of rom\_triangulo is

type memoria\_rom is array (0 to 2047) of std\_logic\_vector (11 downto 0);

constant ROM : memoria\_rom := (

x"004", x"008", x"00c", x"010", x"014", x"018", x"01c", x"020",

x"024", x"028", x"02c", x"030", x"034", x"038", x"03c", x"040",

x"044", x"048", x"04c", x"050", x"054", x"058", x"05c", x"060",

. . .

x"05c", x"058", x"054", x"050", x"04c", x"048", x"044", x"040",

x"03c", x"038", x"034", x"030", x"02c", x"028", x"024", x"020",

x"01c", x"018", x"014", x"010", x"00c", x"008", x"004", x"000"

);

signal tmp: std\_logic\_vector(11 downto 0);

begin

ret: process (addr, clk) begin

if (clk' event and clk = '1') then

tmp <= ROM(conv\_integer(addr));

end if;

end process;

buff: process (tmp, en) begin

if(en = '1') then

data <= tmp;

else

data <= (others => 'Z');

end if;

end process buff;

end arch;

*Archivo Principal “rom\_sierra.vhd”*

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

entity rom\_sierra is

port (

clk : in std\_logic;

en : in std\_logic := '1';

addr : in std\_logic\_vector(10 downto 0);

data : out std\_logic\_vector(11 downto 0)

);

end rom\_sierra;

architecture arch of rom\_sierra is

type memoria\_rom is array (0 to 2047) of std\_logic\_vector (11 downto 0);

constant ROM : memoria\_rom := (

x"002", x"004", x"006", x"008", x"00a", x"00c", x"00e", x"010",

x"012", x"014", x"016", x"018", x"01a", x"01c", x"01e", x"020",

x"022", x"024", x"026", x"028", x"02a", x"02c", x"02e", x"030",

. . .

x"fd2", x"fd4", x"fd6", x"fd8", x"fda", x"fdc", x"fde", x"fe0",

x"fe2", x"fe4", x"fe6", x"fe8", x"fea", x"fec", x"fee", x"ff0",

x"ff2", x"ff4", x"ff6", x"ff8", x"ffa", x"ffc", x"ffe", x"fff"

);

signal tmp: std\_logic\_vector(11 downto 0);

begin

ret: process (addr,clk) begin

if (clk' event and clk = '1') then

tmp <= ROM(conv\_integer(addr));

end if;

end process;

buff: process (tmp,en) begin

if(en = '1') then

data <= tmp;

else

data <= (others => 'Z');

end if;

end process buff;

end arch;

*Archivo Principal “rom\_cuadrada.vhd”*

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

entity rom\_cuadrada is

port (

clk : in std\_logic;

en : in std\_logic := '1';

addr : in std\_logic\_vector(10 downto 0);

data : out std\_logic\_vector(11 downto 0)

);

end rom\_cuadrada;

architecture arch of rom\_cuadrada is

type memoria\_rom is array (0 to 2047) of std\_logic\_vector (11 downto 0);

constant ROM : memoria\_rom := (

x"000", x"000", x"000", x"000", x"000", x"000", x"000", x"000",

x"000", x"000", x"000", x"000", x"000", x"000", x"000", x"000",

x"000", x"000", x"000", x"000", x"000", x"000", x"000", x"000",

. . .

x"fff", x"fff", x"fff", x"fff", x"fff", x"fff", x"fff", x"fff",

x"fff", x"fff", x"fff", x"fff", x"fff", x"fff", x"fff", x"fff",

x"fff", x"fff", x"fff", x"fff", x"fff", x"fff", x"fff", x"fff"

);

signal tmp: std\_logic\_vector(11 downto 0);

begin

ret: process (addr,clk) begin

if (clk' event and clk = '1') then

tmp <= ROM(conv\_integer(addr));

end if;

end process;

buff: process (tmp,en) begin

if(en = '1') then

data <= tmp;

else

data <= (others => 'Z');

end if;

end process buff;

end arch;

*Archivo Principal “rom\_vicflo.vhd”*

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

entity rom\_vicflo is

port (

clk : in std\_logic;

en : in std\_logic := '1';

addr : in std\_logic\_vector(10 downto 0);

data : out std\_logic\_vector(11 downto 0)

);

end rom\_vicflo;

architecture arch of rom\_vicflo is

type memoria\_rom is array (0 to 2047) of std\_logic\_vector (11 downto 0);

constant ROM : memoria\_rom := (

x"7f9", x"7ed", x"7e0", x"7d3", x"7c7", x"7ba", x"7ae", x"7a1",

x"795", x"788", x"77c", x"76f", x"762", x"756", x"749", x"73d",

x"730", x"724", x"717", x"70b", x"6fe", x"6f2", x"6e5", x"6d9",

. . .

x"800", x"800", x"800", x"800", x"800", x"800", x"800", x"800",

x"800", x"800", x"800", x"800", x"800", x"800", x"800", x"800",

x"800", x"800", x"800", x"800", x"800", x"800", x"800", x"800"

);

signal tmp: std\_logic\_vector(11 downto 0);

begin

ret: process (addr,clk) begin

if (clk' event and clk = '1') then

tmp <= ROM(conv\_integer(addr));

end if;

end process;

buff: process (tmp,en) begin

if(en = '1') then

data <= tmp;

else

data <= (others => 'Z');

end if;

end process buff;

end arch;

**Construcción virtual**

Modulo top

Selector Señal

Potenciómetro Amplitud

Potenciómetro Frecuencia

Reloj  
50 Mhz

ROM  
Seno

ROM  
Triangulo

ROM  
Sierra

ROM  
Cuadrada

ROM  
Vicflo Señal

DAC

Osciloscopio

**Conclusiones**

* Aguilar Enriquez Paul Sebastian: El diseño e implementación de esta práctica fue sencillo, a pesar de tener que investigar sobre los componentes necesarios no hubo dificultad alguna en implementarlos, tal vez se deba a que el lenguaje VHDL ayuda bastante en la abstracción de los componentes y su implementación, sin embargo considero que la práctica es sencilla por sí misma, incluso la realización del DAC y la modificación de las señales fue sencillo de realizar.
* Cabrera López Oscar Emilio: La práctica fue sencilla de diseñar e implementar, tuvimos que investigar sobre algunos componentes como el DAC para realizar la conversión así que volver interactivo el uso del temporizador del 555, a pesar de esto el circuito es simple y su implementación en VHDL aún más.

**Repositorio en GitHub** <https://github.com/penserbjorne/clase-diseniosistemasdigitales-2017-1>